Specification and Model Checking of Time Petri Nets and Timed Automata

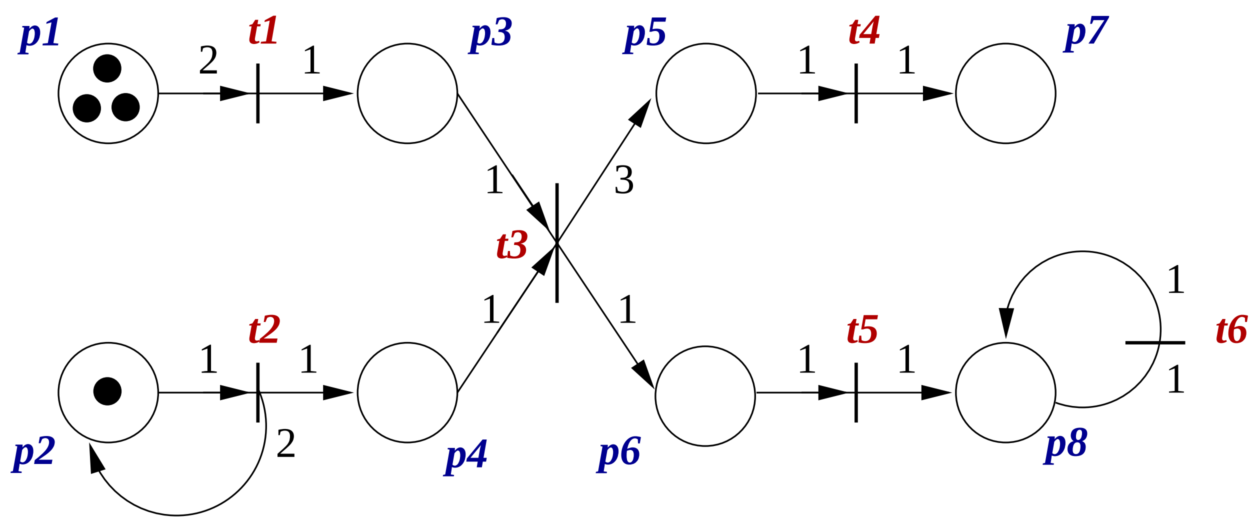
WOJCIECH PENCZEK

ICS PAS, Warsaw, Poland

Outline

- Petri nets (PNs)
- Time Petri nets (TPNs)
- Timed automata (TA)
- Timed temporal logics: TCTL
- Verification methods for TPNs: state class approaches
- From TPNs to TA
- Verification methods for TA: partitioning and SAT-based approaches
- Experimental results for verifying TPNs directly and TPNs via TA

Petri Nets



Petri nets are directed weighted graphs of two types of nodes: places (representing conditions) and transitions (representing events). The arcs are assigned positive weights.

Definition

A Petri net is a four-element tuple $\mathcal{P} = (P, T, F, m^0)$, where

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

Some history

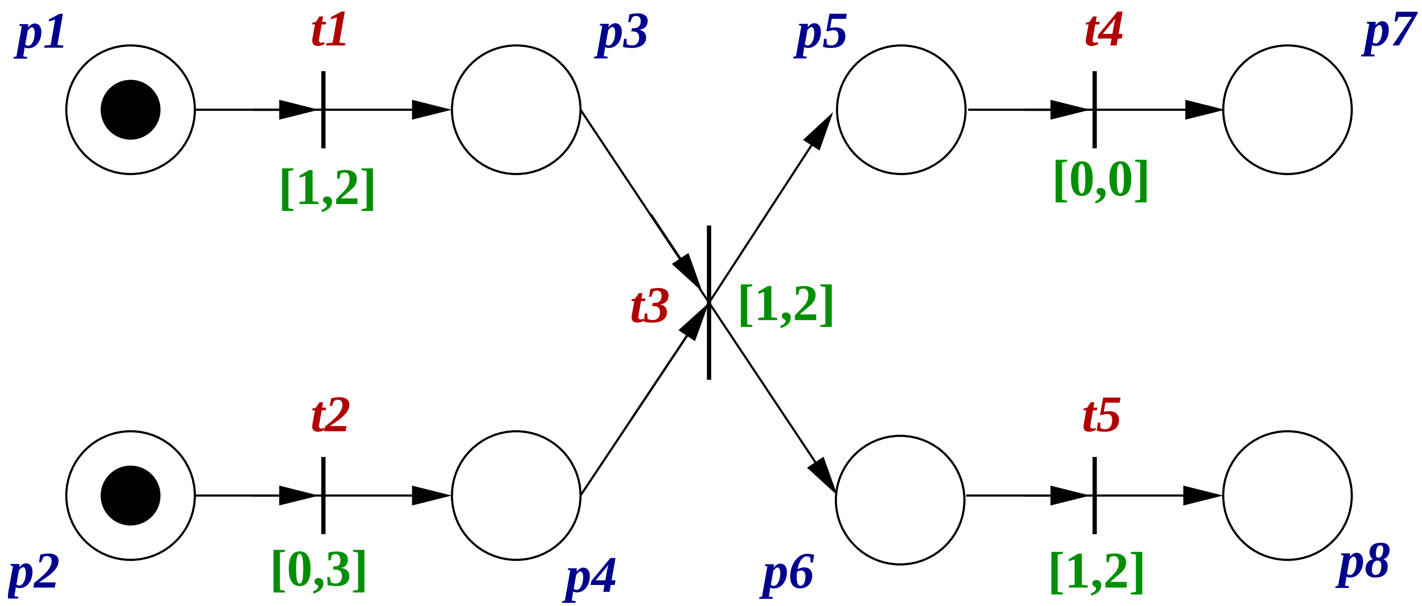
Timed extensions of Petri nets:

- Timed Petri nets [*Ramchandani'74*]
- Time Petri nets [*Merlin, Farber'76*]

Timed extensions of automata theory:

- Timed automata [*Alur,Dill'90*]
- Hybrid automata
  [*Alur, Courcoubetis, Henzinger, Ho'93; Nicollin, Olivero, Sifakis, Yovine'93*]

Time Petri nets - an example



Time Petri nets - definition

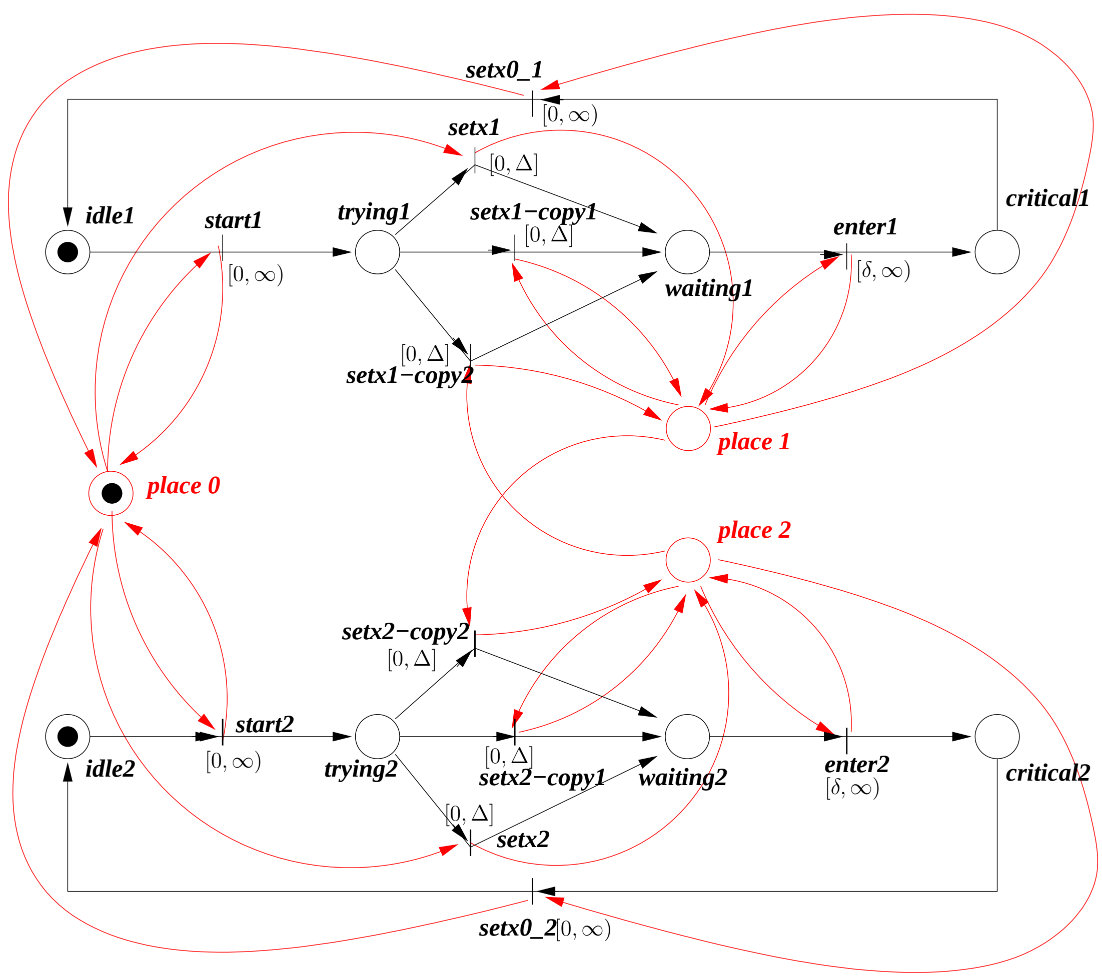
A *time Petri net* (TPN): $\mathcal{N} = (P, T, FR, Eft, Lft, m_0)$, where

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

TPNs - some definitions

- $\bullet t = \{p \in P \mid (p,t) \in FR\}$ - the *preset* of $t \in T$,
- $t\bullet = \{p \in P \mid (t,p) \in FR\}$ - the *postset* of $t \in T$,
- a *marking* of $\mathcal{N}$ - any subset $m \subseteq P$,
- a transition $t \in T$ is *enabled* at m ($m[t\rangle$ for short) if $\bullet t \subseteq m$ and $t \bullet \cap (m \setminus \bullet t) = \emptyset$,
- $en(m) = \{t \in T \mid m[t\rangle\}$.

TPN: Mutual Exclusion Protocol



Concrete states of TPNs: clock approach

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

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

Concrete states of TPNs: clock approach

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

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

Clocks can be associated with:

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

Concrete states of TPNs: clock approach

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

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

Clocks can be associated with:

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

Concrete states change because of:

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

Concrete states of TPNs: clock approach

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

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

Clocks can be associated with:

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

Concrete states change because of:

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

Discrete transition relation: $\sigma \xrightarrow{t}_d \sigma'$ iff $\sigma \xrightarrow{\tau^*}_c \xrightarrow{t}_c \xrightarrow{\tau^*}_c \sigma'$, $t \in T$

Concrete states of TPNs: firing interval approach

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

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

Concrete states of TPNs: firing interval approach

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

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

Concrete states change because of:

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

Concrete models for TPNs

Σ - a set of all the concrete states of $\mathcal{N}$

$PV = \{\wp_p \mid p \in P\}$ - a set of propositional variables

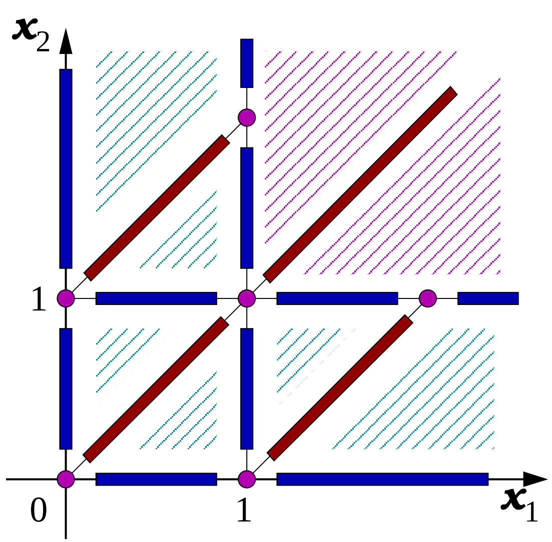
$V_c : \Sigma \to PV$ - a *valuation function* s.t.
$V_c((m, \cdot)) = \{\wp_p \mid p \in m\}$

$M_c(\mathcal{N}) = ((\Sigma, \sigma^0, \to), V_c)$, where $\to \in \{\to_c, \to_d\}$
- a *concrete model* of $\mathcal{N}$ (**usually infinite**)

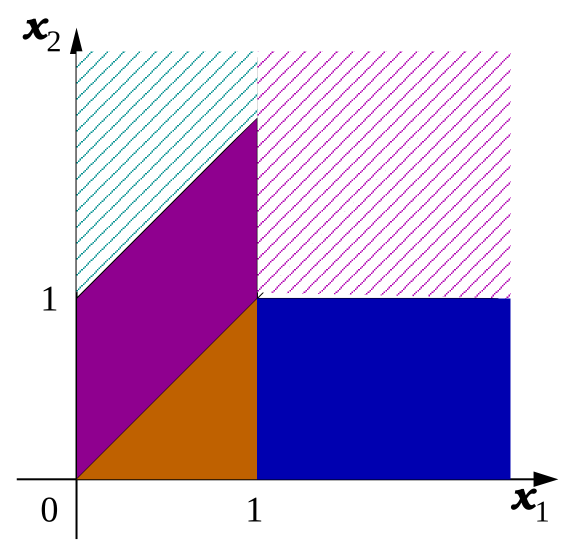
$\mathcal{X} = \{x_1, \ldots, x_n\}$ - a set of variables (*clocks*).

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

$Z(n)$ - the set of all the zones in $\mathbb{R}^n$

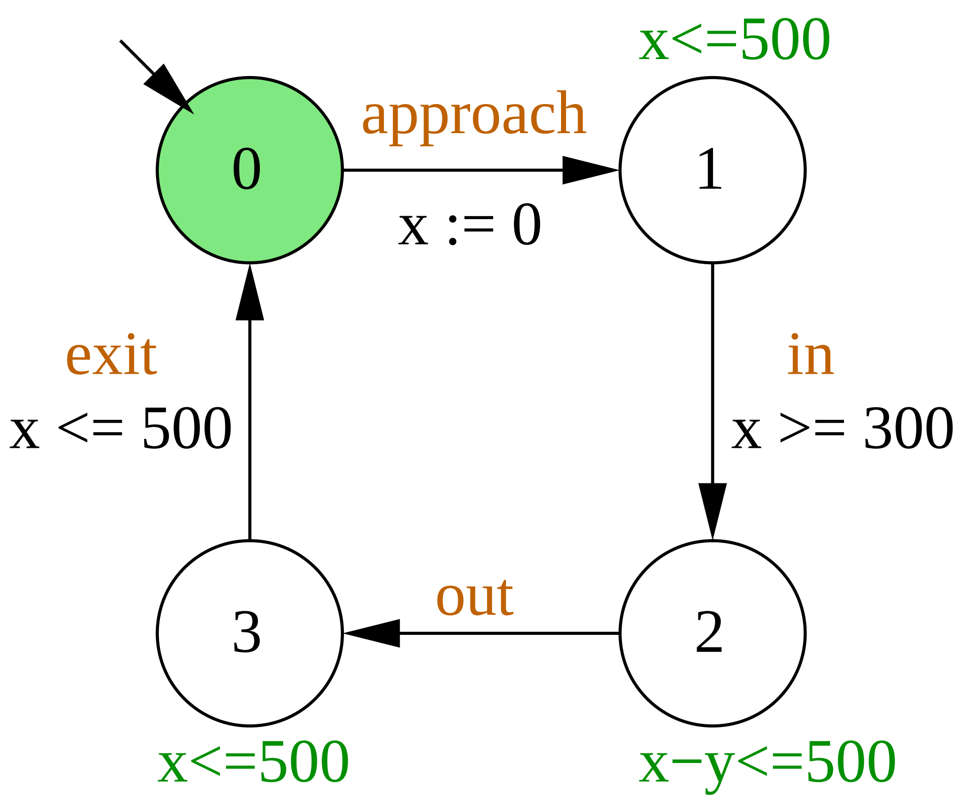


detailed zones



non-detailed zones

Timed automata - an example



Timed automata - definition

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

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

Timed automata - definition

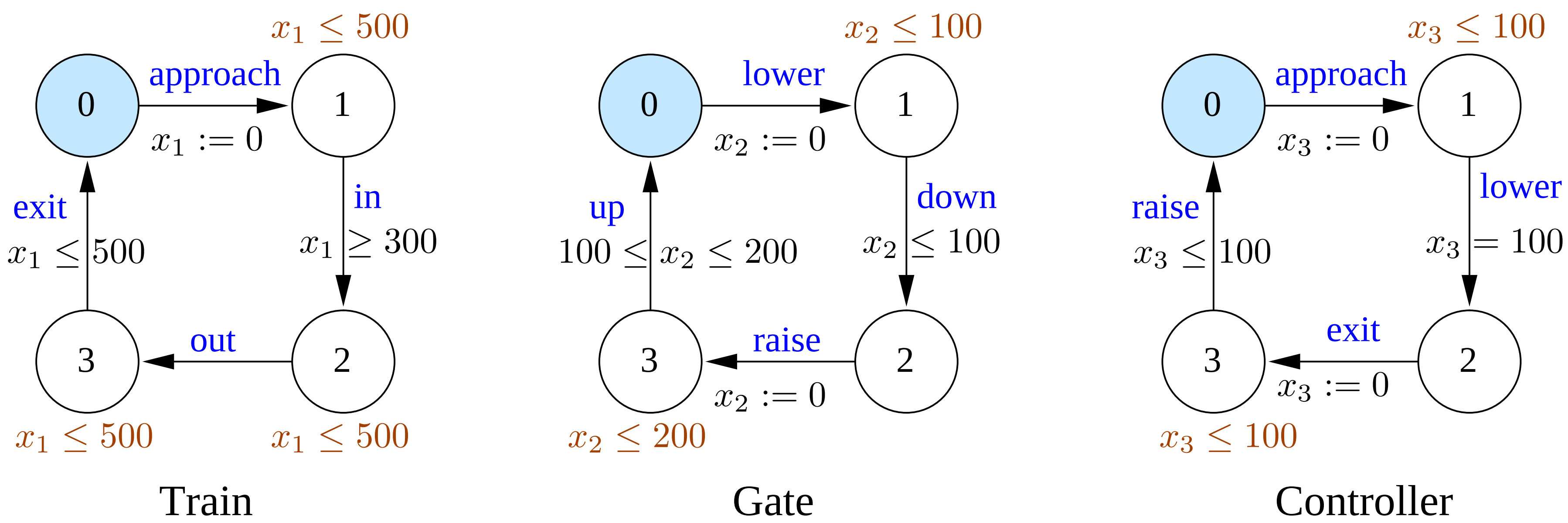
A *timed automaton* $\mathcal{A}$ is a tuple $(A, L, \mathcal{X}, l^0, E, \mathcal{I})$, where:

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

To reason about properties:

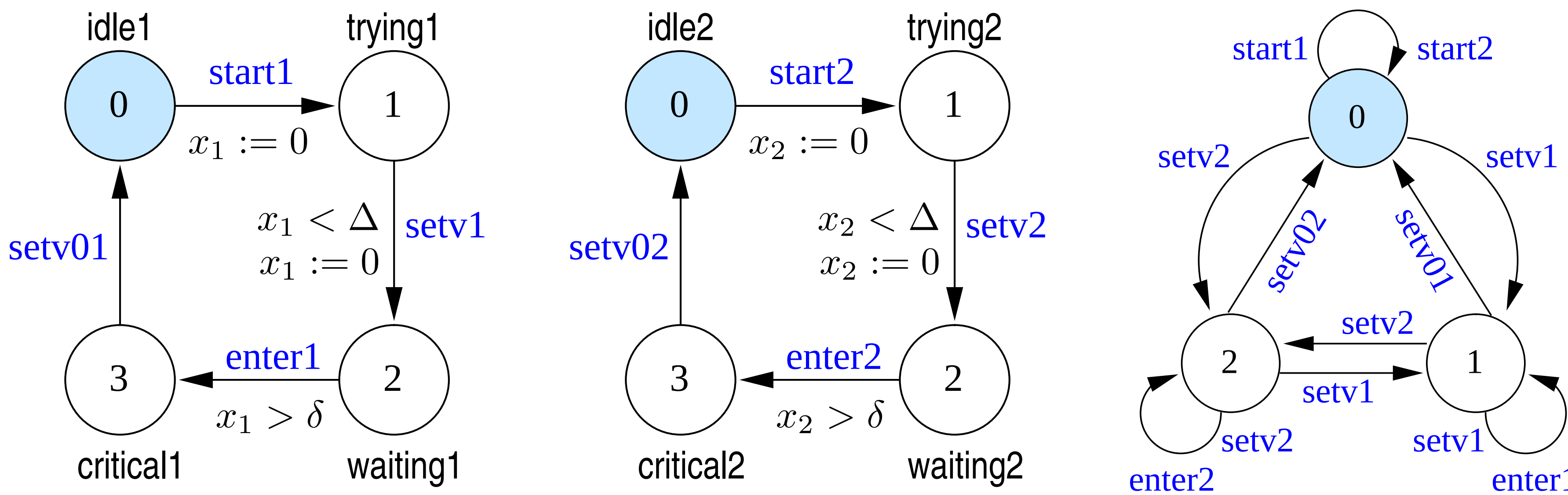
- $V_{\mathcal{A}} : L \to 2^{PV}$ - a *valuation function*
  for a set of propositional variables PV.

TA: TGC Protocol



Train–Gate–Controller example

TA: Mutual Exclusion Protocol



Process 1 | Process 2 | Variable V

Fischer’s mutual exclusion protocol for two processes

Concrete states of TA

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

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

Concrete states of TA

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

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

Concrete states can change because of:

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

Concrete states of TA

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

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

Concrete states can change because of:

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

Discrete transition relation:

$$q \xrightarrow{e}_d q' \text{ iff } q \xrightarrow{\tau^*}_c \xrightarrow{e}_c \xrightarrow{\tau^*}_c q',\ e \in E$$

Concrete models for TA

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

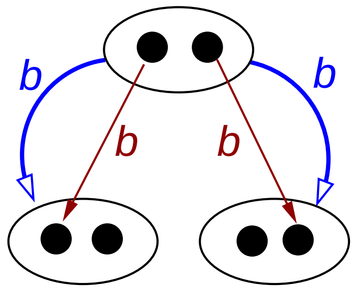
PV - a set of propositional variables

$V_c : Q \rightarrow PV$ - a *valuation function* which extends $V_{\mathcal{A}}$
$V_c((l,\cdot)) = V_{\mathcal{A}}(l)$ (assigns the same propositions to the states with the same locations)

$M_c(\mathcal{A}) = ((Q, q^0, \rightarrow), V_c)$, where $\rightarrow \in \{\rightarrow_c, \rightarrow_d\}$
- a *concrete model* of $\mathcal{N}$ (**usually infinite**)

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

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

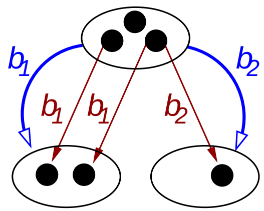


Other conditions depend on the properties to be preserved.

Examples of abstract models

- *Surjective models*:

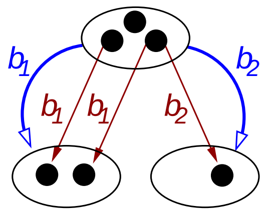
EA) $w_1 \xrightarrow{b} w_2$ iff $(\forall s_2 \in w_2)\ (\exists s_1 \in w_1)\ s_1 \xrightarrow{b} s_2$.



Examples of abstract models

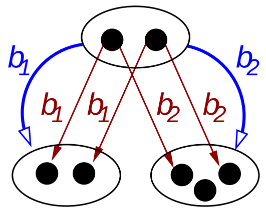
- *Surjective models*:

EA) $w_1 \xrightarrow{b} w_2$ iff $(\forall s_2 \in w_2)\ (\exists s_1 \in w_1)\ s_1 \xrightarrow{b} s_2$.



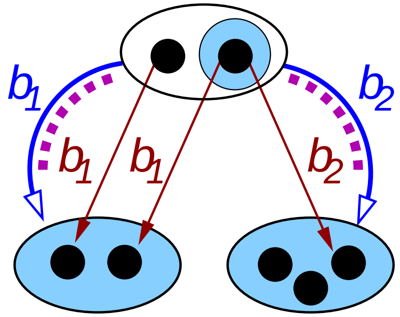
- *Bisimulating (b-) models*:

AE) $w_1 \xrightarrow{b} w_2$ iff $(\forall s_1 \in w_1)\ (\exists s_2 \in w_2)\ s_1 \xrightarrow{b} s_2$.



Examples of abstract models - cont'd

- Simulating (s-) models:
  for each $w \in W$ there is a nonempty $w^{cor} \subseteq w$ s.t.
  - $s^0 \in (w^0)^{cor}$, and
  - U) $w_1 \xrightarrow{b} w_2$ iff $(\forall s_1 \in w_1^{cor})\ (\exists s_2 \in w_2^{cor})\ s_1 \xrightarrow{b} s_2$.



*Temporal logics: CTL**

$PV = \{\wp_1, \wp_2 \ldots\}$ - a set of propositional variables.

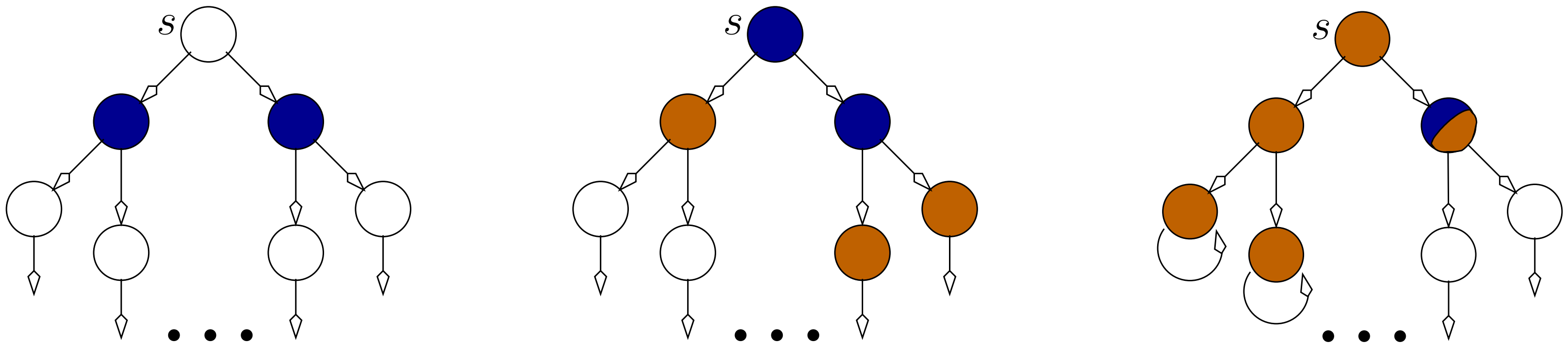
Syntax of CTL^*:
the state formulas φ_s, defined using path formulas φ_p:

$$\varphi_s := \wp \mid \neg\wp \mid \varphi_s \wedge \varphi_s \mid \varphi_s \vee \varphi_s \mid \mathrm{A}\varphi_p \mid \mathrm{E}\varphi_p$$
$$\varphi_p := \varphi_s \mid \varphi_p \wedge \varphi_p \mid \varphi_p \vee \varphi_p \mid \mathrm{X}\varphi_p \mid \varphi_p \mathrm{U} \varphi_p \mid \varphi_p \mathrm{R} \varphi_p$$

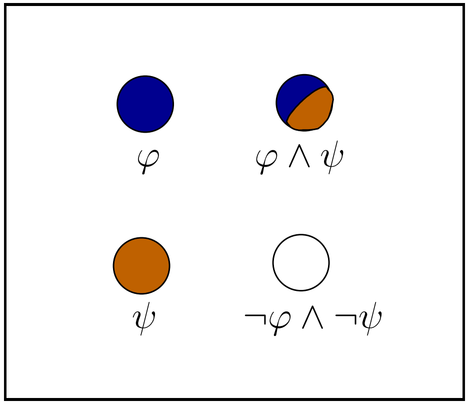
A ('for all paths') and E ('there exists a path') are *path quantifiers*,

X ('neXt'), U ('Until'), and R ('Release') are *state operators*.

Temporal operators of CTL



$s \models \mathsf{AX}\varphi$ $\quad$ $s \models \mathsf{A}(\varphi \mathsf{U} \psi)$ $\quad$ $s \models \mathsf{A}(\varphi \mathsf{R} \psi)$



Temporal logics: TCTL

$PV = \{\wp_1, \wp_2, \ldots\}$ - a set of propositional variables.

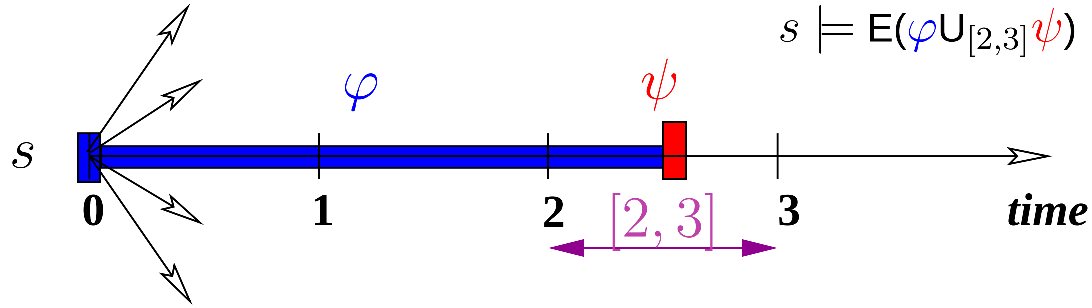
Syntax of TCTL:
the formulas defined by the grammar:

$$\varphi := \wp \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathrm{E}(\varphi \mathrm{U}_{\mathbf{I}} \varphi) \mid \mathrm{E}(\varphi \mathrm{R}_{\mathbf{I}} \varphi),$$

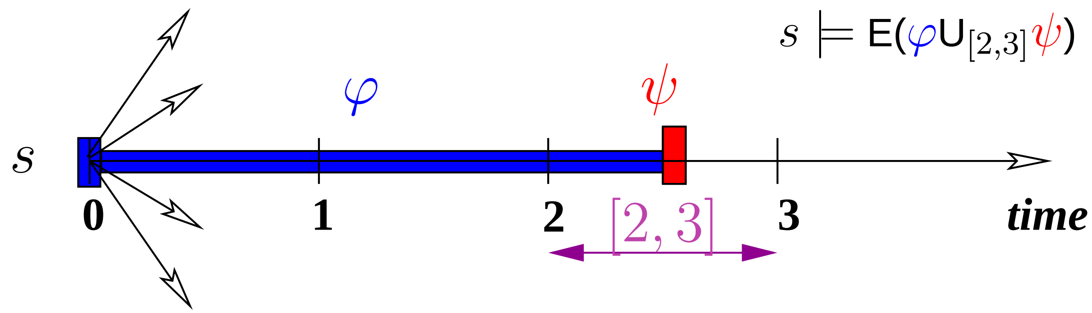
where $\wp \in PV$ and $\mathbf{I}$ is an interval in $\mathbb{N}$.

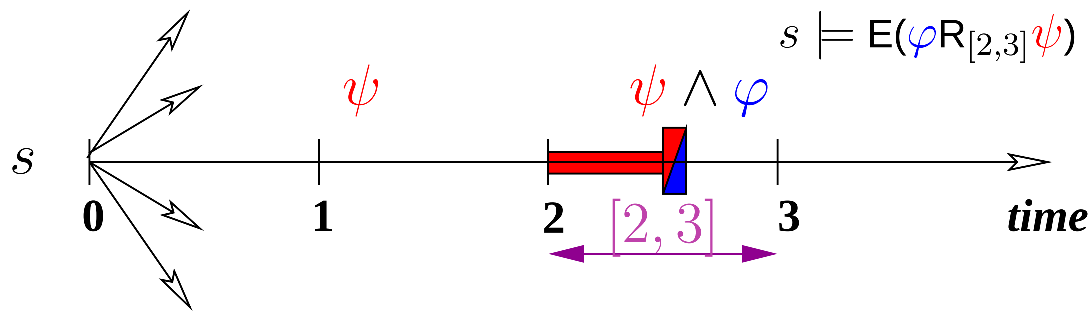
Temporal operators of TCTL

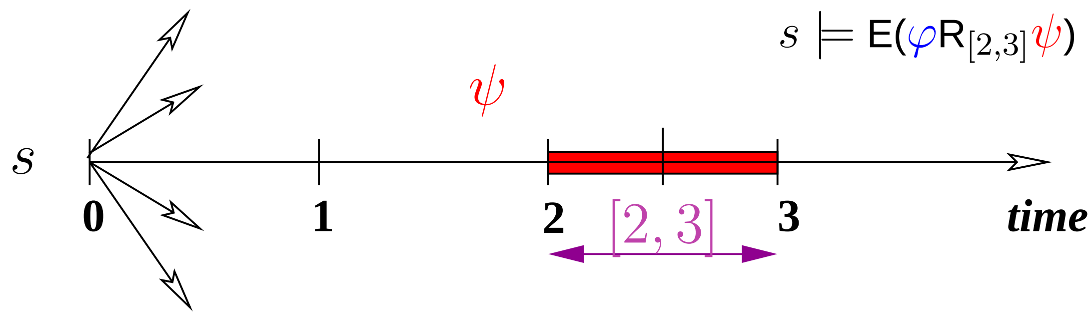
$s \models \mathsf{E}(\varphi \mathsf{U}_{[2,3]} \psi)$



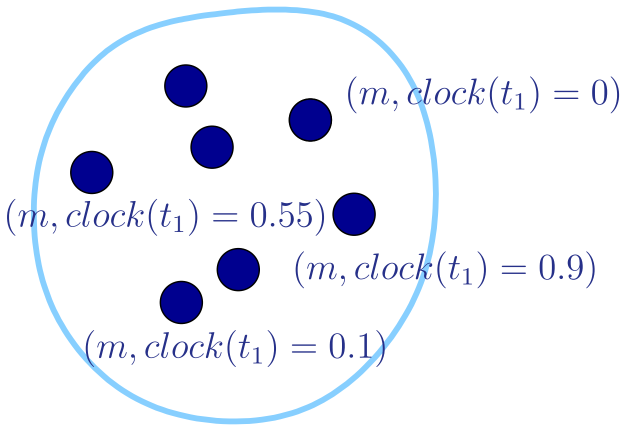
Temporal operators of TCTL



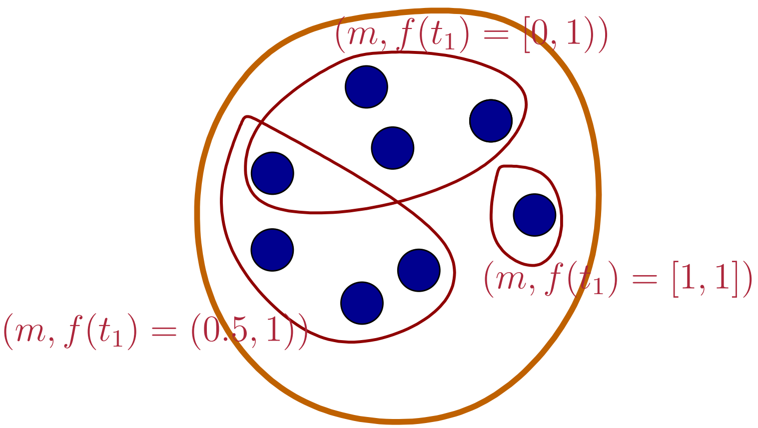




A *state class* of a TPN is a pair $C = (m, I)$, where m is a marking, and I is a set of inequalities built over variables corresponding to transitions.



$C = (m, \{0 \leq t_1 \leq 2\})$



$C = (m, \{0 \leq t_1 \leq 2\})$

State classes can be defined for both the clock and firing intervals approach.

Abstract models of timed systems

	TPN	TA
surjective models (LTL, reachability)	✓	✓

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

TA: Bouajjani, Tripakis, Yovine (RTSS'97)

Abstract models of timed systems

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

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

TA: *$\approx$ Alur, Courcoubetis, Dill, Halbwachs, Wong-Toi (CONCUR'92),*

Dembiński, Penczek, Półrola (Fundamenta Informaticae 2002)

Abstract models of timed systems

	TPN	TA
surjective models (LTL, reachability)	✓	✓
b-models, discrete semantics (CTL*)	✓	✓
b-models, dense semantics (CTL$^*_{-X}$, TCTL)	✓	✓

TPN: *Boucheneb, Gardey, Roux (J. Log. Comput. 2009)*

TA: *Alur, Courcoubetis, Dill, Halbwachs, Wong-Toi (RTSS'92); Yannakakis, Lee (CAV'93); Tripakis, Yovine (CAV'96)*

Abstract models of timed systems

	TPN	TA
surjective models (LTL, reachability)	✓	✓
b-models, discrete semantics (CTL*)	✓	✓
b-models, dense semantics (CTL$^*_{-X}$, TCTL)	✓	✓
s-models, discrete semantics (ACTL*)	✓	✓

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

TA: *Dembiński, Penczek, Półrola (Fundamenta Informaticae, 2002)*

Abstract models of timed systems

	TPN	TA
surjective models (LTL, reachability)	✓	✓
b-models, discrete semantics (CTL*)	✓	✓
b-models, dense semantics (CTL^*_{-X}, TCTL)	✓	✓
s-models, discrete semantics (ACTL*)	✓	✓
s-models, dense semantics ($ACTL^*_{-X}$, TACTL)	–	✓

TPN:

TA: Dembiński, Penczek, Półrola (Fundamenta Informaticae 2002)

Abstract models of timed systems

	TPN	TA
surjective models (LTL, reachability)	✓	✓
b-models, discrete semantics (CTL*)	✓	✓
b-models, dense semantics (CTL^*_{-X}, TCTL)	✓	✓
s-models, discrete semantics (ACTL*)	✓	✓
s-models, dense semantics ($ACTL^*_{-X}$, TACTL)	–	✓
pb-models (reachability)	–	✓

TPN:

TA: Półrola, Penczek, Szreter (Fundamenta Informaticae 2002)

Abstract models of timed systems

	TPN	TA
surjective models (LTL, reachability)	✓	✓
b-models, discrete semantics (CTL*)	✓	✓
b-models, dense semantics (CTL^*_{-X}, TCTL)	✓	✓
s-models, discrete semantics (ACTL*)	✓	✓
s-models, dense semantics (ACTL^*_{-X}, TACTL)	–	✓
pb-models (reachability)	–	✓
ps-models (reachability)	–	✓

TPN:

TA: Półrola, Penczek, Szreter (FORMATS'03)

Abstract models of timed systems

	TPN	TA
surjective models (LTL, reachability)	✓	✓
b-models, discrete semantics (CTL*)	✓	✓
b-models, dense semantics (CTL^*_{-X}, TCTL)	✓	✓
s-models, discrete semantics (ACTL*)	✓	✓
s-models, dense semantics ($ACTL^*_{-X}$, TACTL)	–	✓
pb-models (reachability)	–	✓
ps-models (reachability)	–	✓
detailed region graph (CTL*$_{-X}$, TCTL)	✓	✓

TPN: Okawa, Yoneda 1997, Virbitskaite, Pokozy 1999

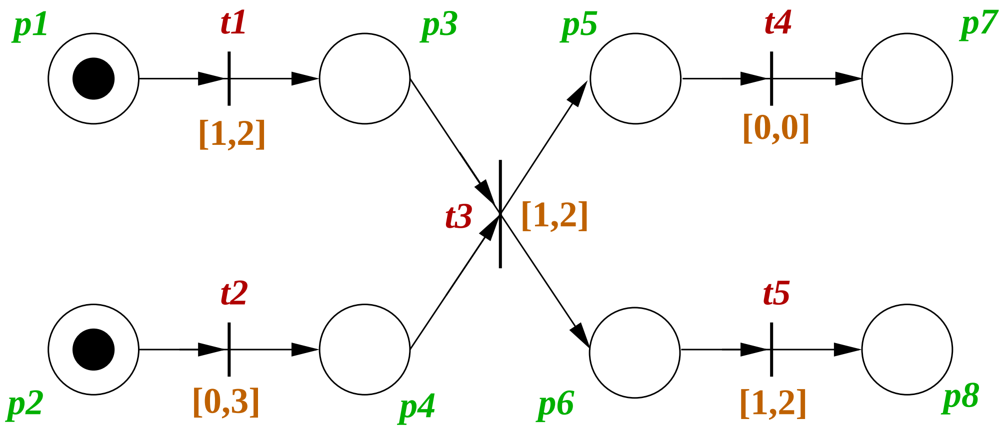
TA: Alur, Courcoubetis, Dill (LISC'90)

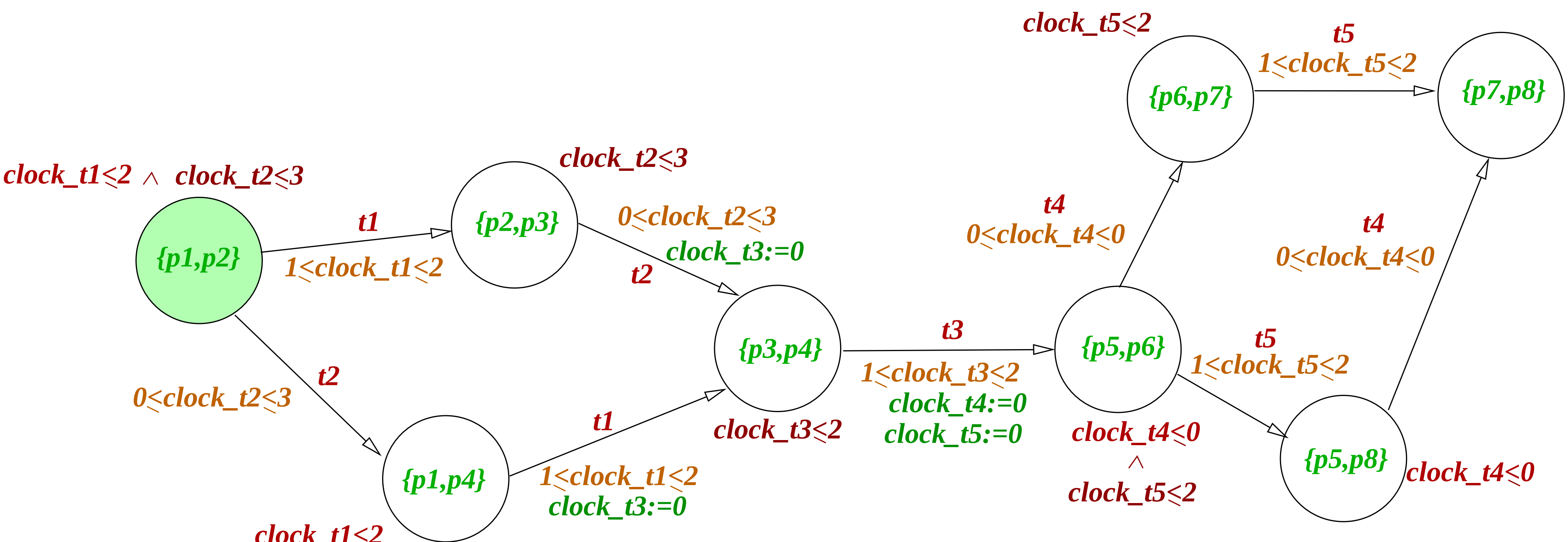
Verifying TPNs via a translation to TA

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

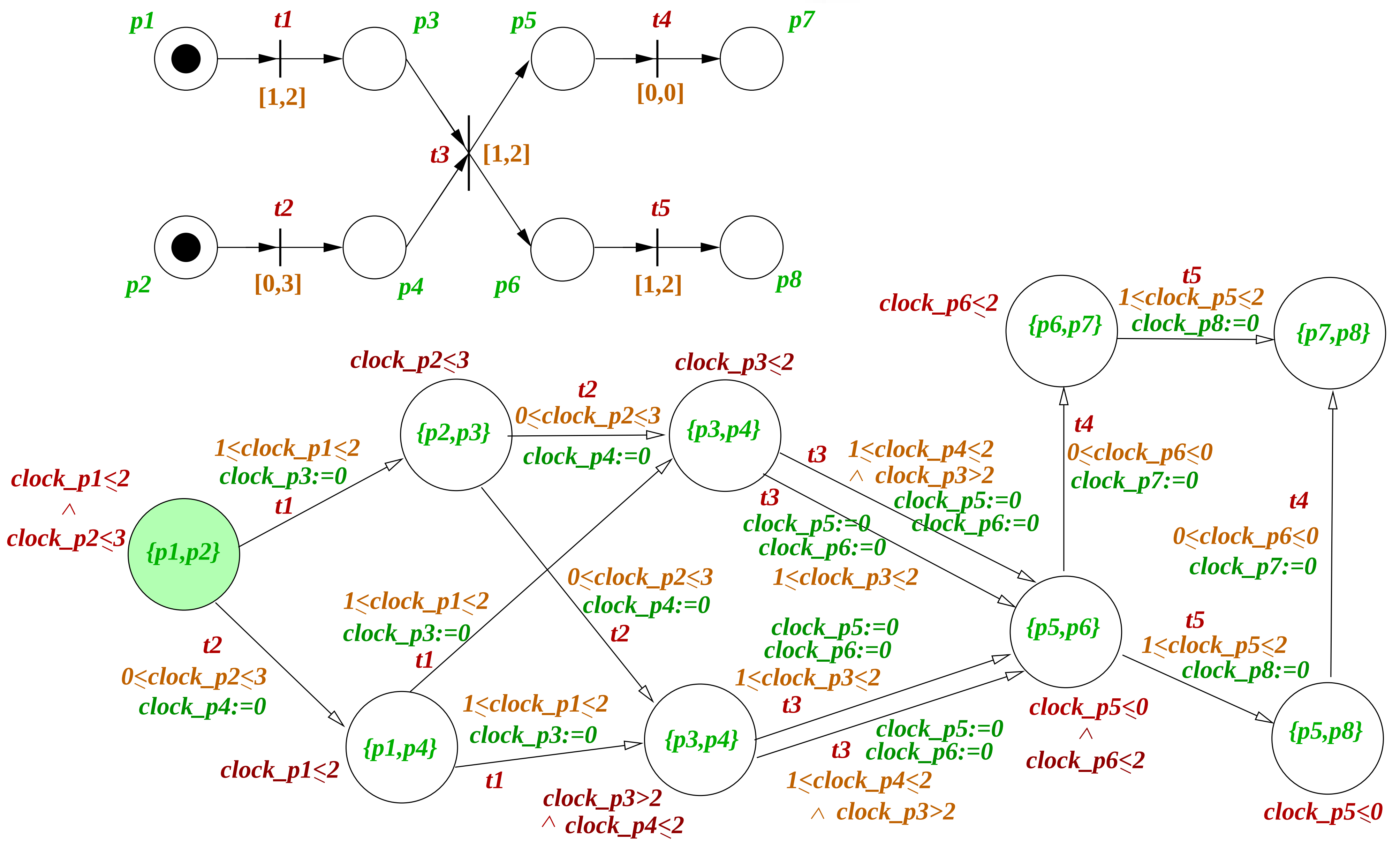
- clocks
- locations and invariants
- guards and resets.

“Transitions–as–clocks” approach





"Places–as–clocks" approach



Other translations TPN $\rightarrow$ TA

- **Sifakis, Yovine** (STACS'96)
  translating time stream Petri nets (TSPNs) to TA with disjunctions of clock constraints. TPNs are a subclass of TSPNs
- **Cortés, Eles, Peng** (RTCSA'02)
  translating extended TPNs (called PRES+ models) to a network of (extended) TA, exploiting "clusters" (sets of sequentially enabling transitions)
- **Lime, Roux** (PNPM'03)
  translation based on building SCG for the net ("state class automaton")
- **Gu, Shin** (DIPES'02), **Cassez, Roux** (MSR'03)
  translations to TA with shared variables and urgency modelling

Partitioning algorithms

$\Pi \subseteq 2^S$ - a *partition* of the state space S into *classes*

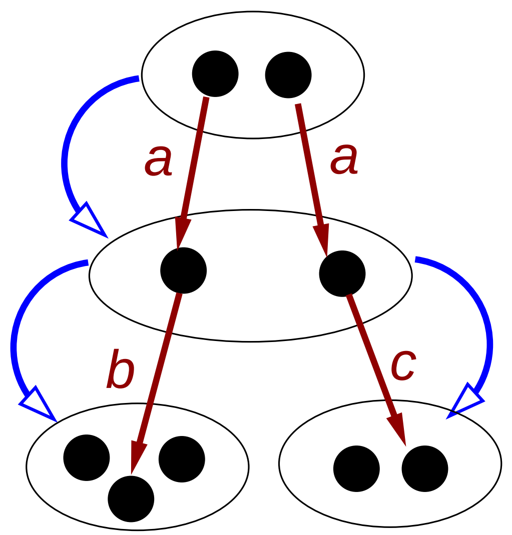
for TA, classes are represented by $(location, zone, \cdots)$

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

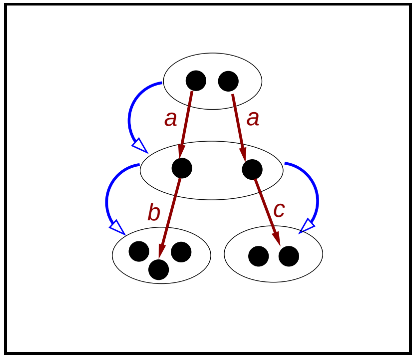
The partitioning (minimization) algorithms generate models whose states are classes of a partition:

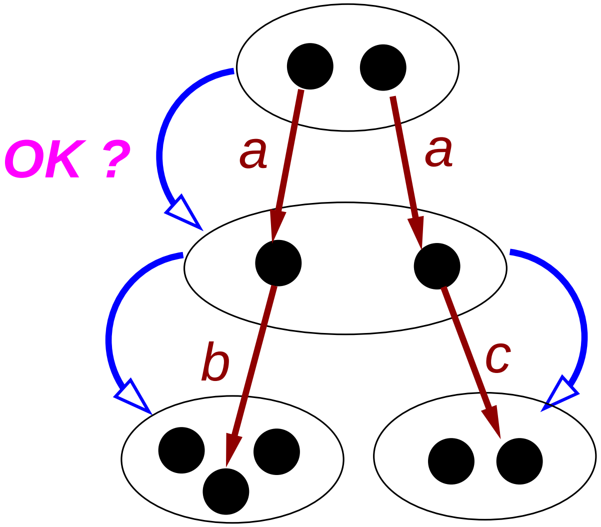
- start from *an initial partition* Π_0 of the state space,
- successively refine the partition until all the classes of Π satisfy an appropriate condition (AE, EA, U, ...).

Partitioning algorithm: how it works (an example for b-models)

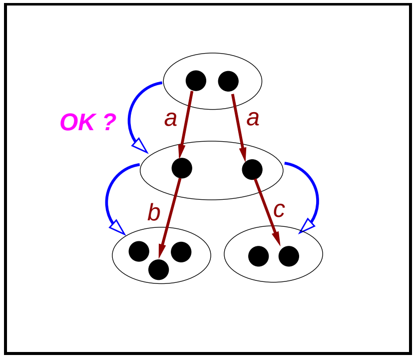


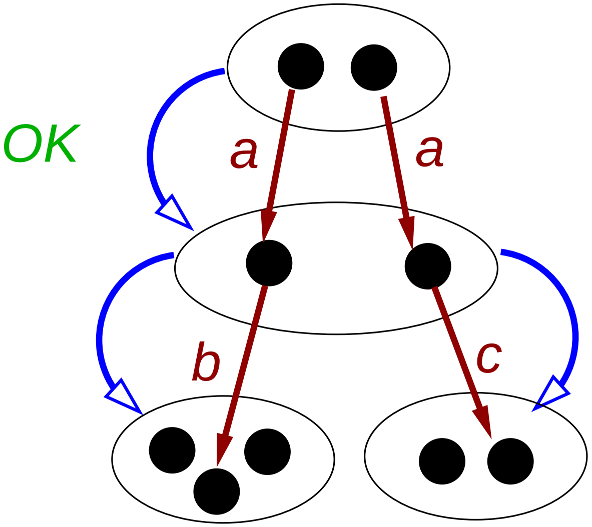
Partitioning algorithm: how it works (an example for b-models)



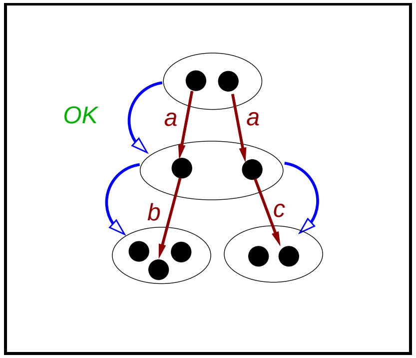


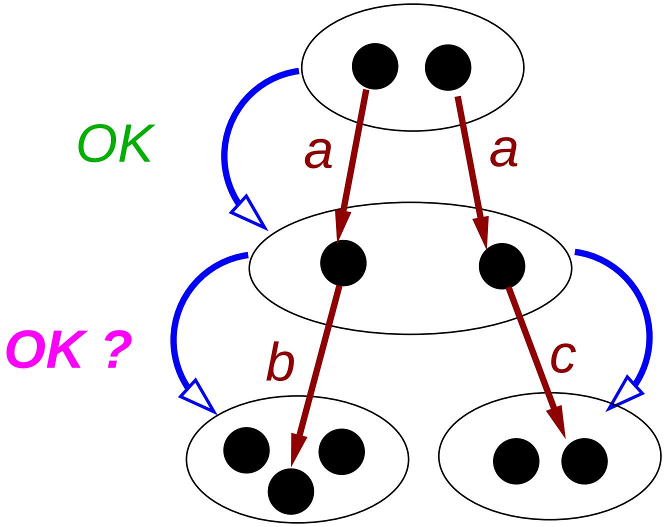
Partitioning algorithm: how it works (an example for b-models)



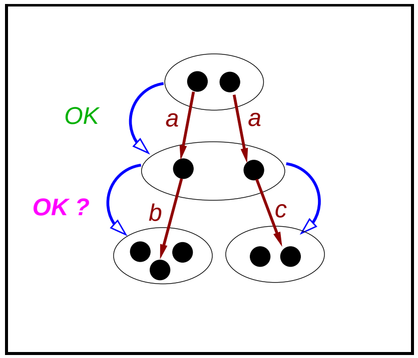


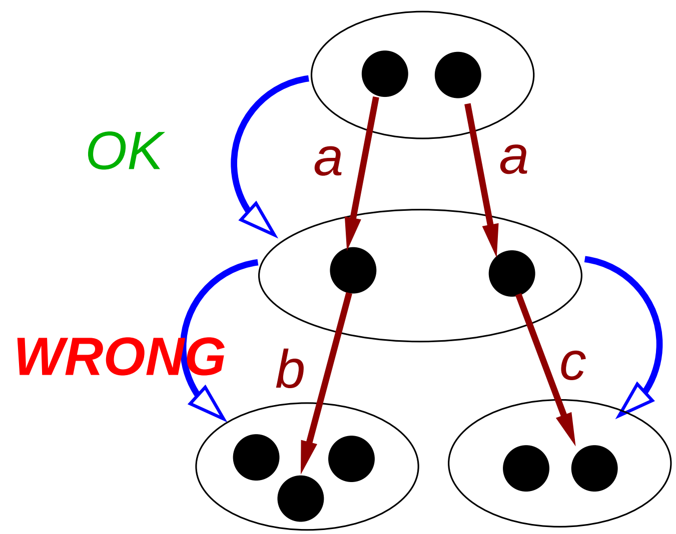
Partitioning algorithm: how it works (an example for b-models)



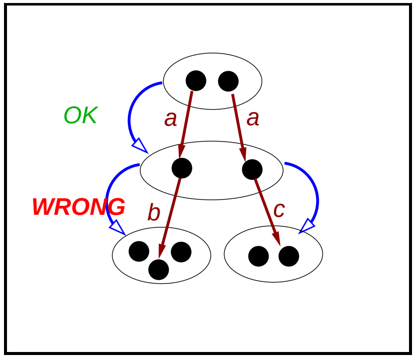


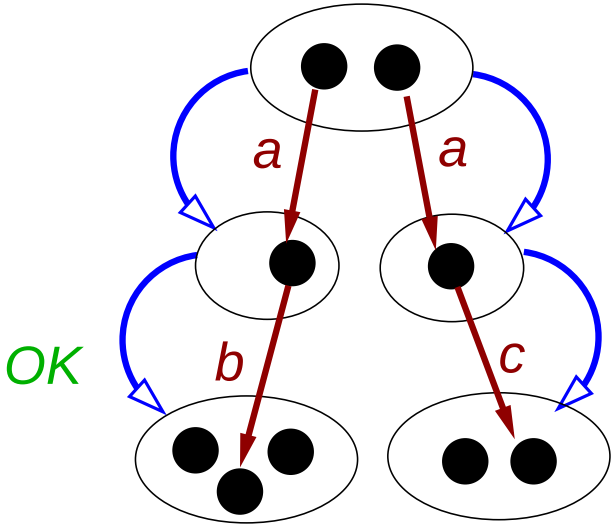
Partitioning algorithm: how it works (an example for b-models)



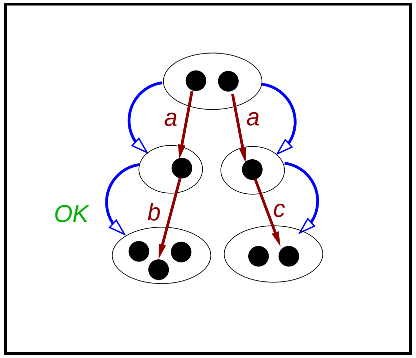


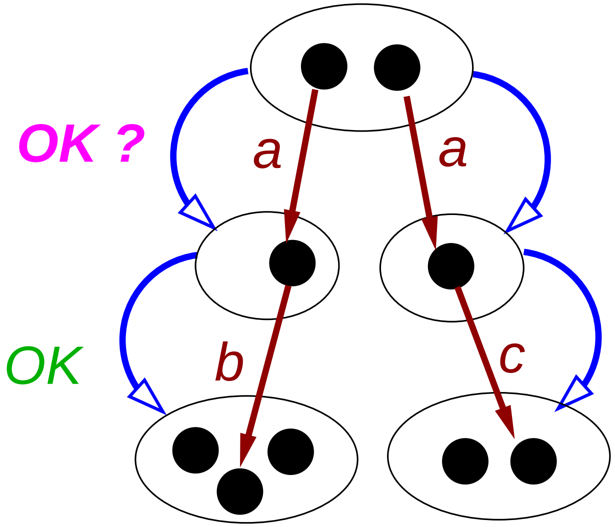
Partitioning algorithm: how it works (an example for b-models)



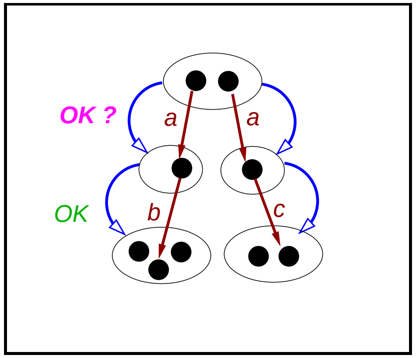


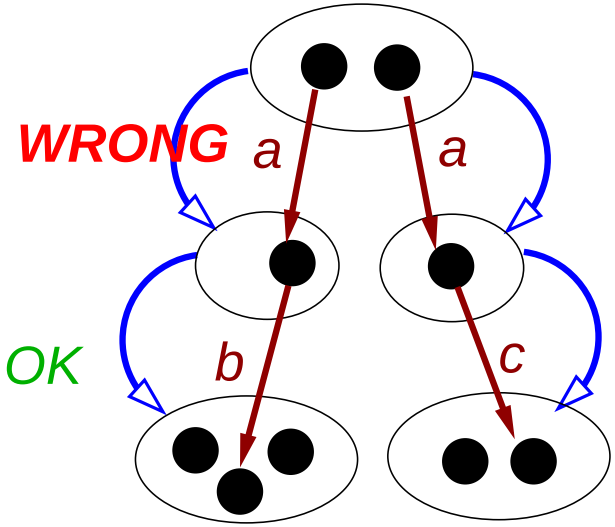
Partitioning algorithm: how it works (an example for b-models)



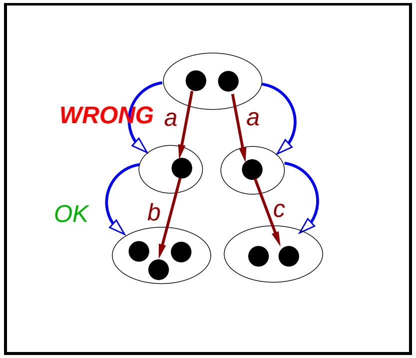


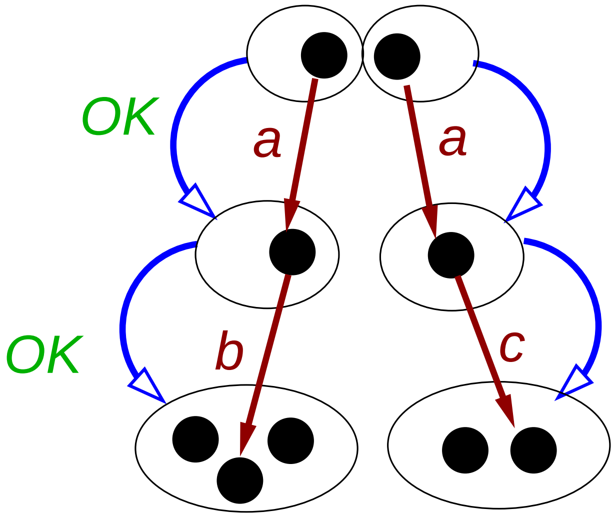
Partitioning algorithm: how it works
(an example for b-models)



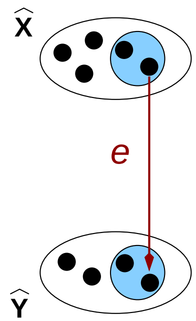


Partitioning algorithm: how it works (an example for b-models)

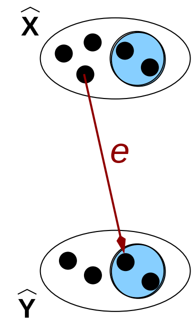




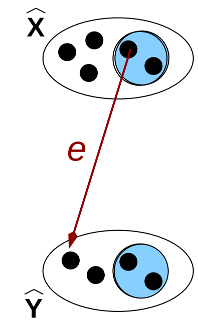
...and how partitioning works for *s-models*



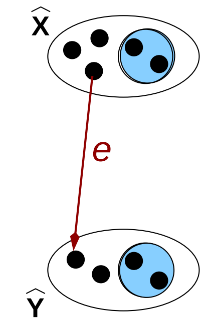
pseudo-e-stable



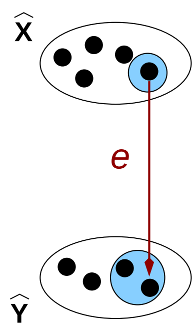
pseudo-e-unstable



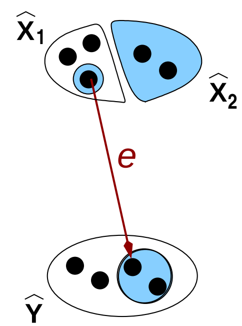
semi-e-unstable



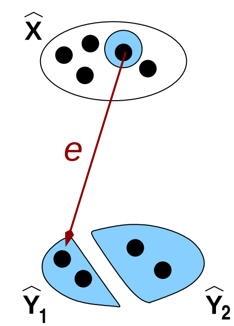
e-unstable



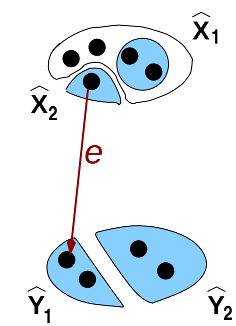
modify X^{cor}



split $\widehat{X}$



split $\widehat{Y}$ and modify X^{cor}



split $\widehat{X}$ and $\widehat{Y}$

Symbolic data structures

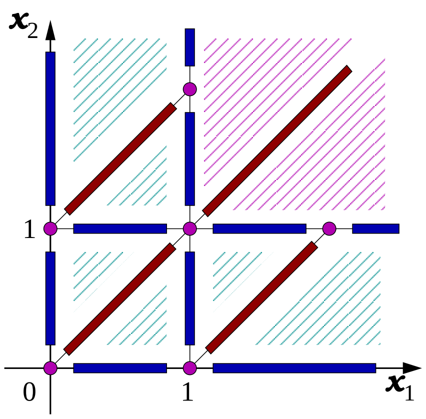
- Difference Bound Matrices (DBM) [Dill'89] - for representing state classes of TPNs or regions of TA.
- Clock Difference Diagrams (CDD) [Behrmann et al.'99] Clock Restriction Diagrams (CRD) [Wang'00], Difference Decision Diagrams (DDD) [Møller et al.'99] - for representing sets of regions.
- Propositional Logic (PL) - for representing sets of detailed regions.

Detailed zones and regions

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

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

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



$DZ(2)$

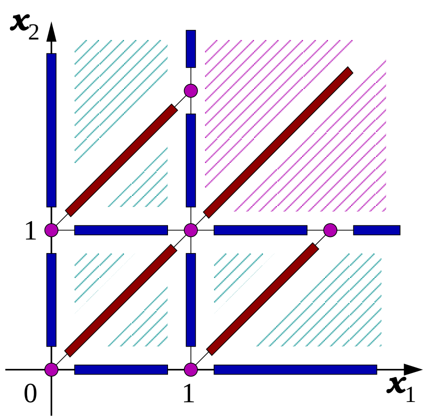
$c_{max} = 1$

Detailed zones and regions

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

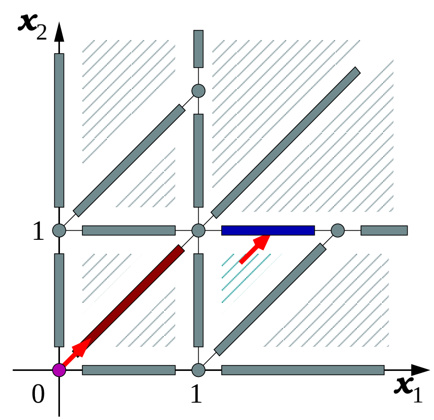
Detailed zones - the equivalence classes of the zone equivalence relation $\simeq$ in the set of the clock valuations.

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



$DZ(2)$

$c_{max} = 1$



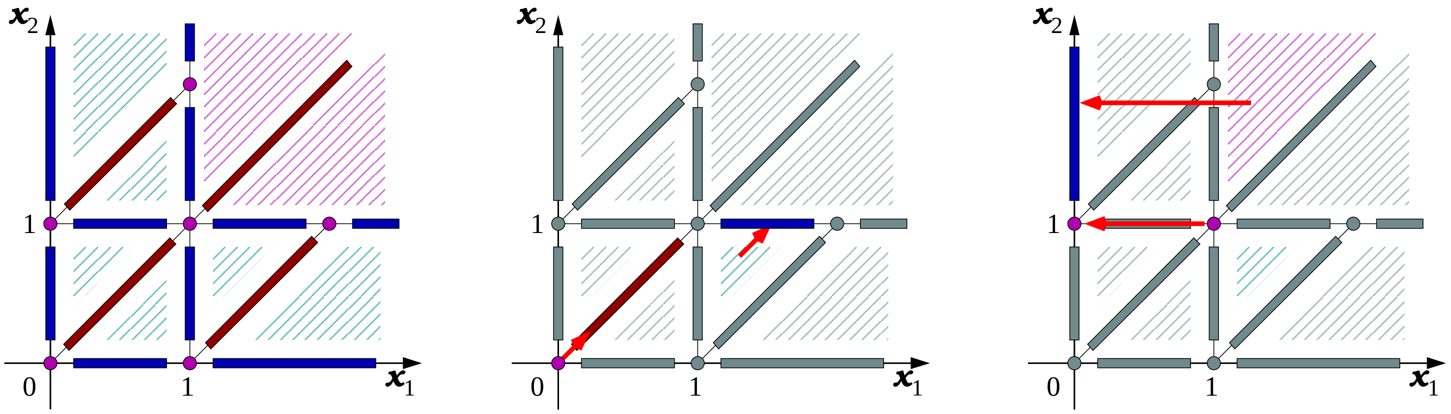
time steps

Detailed zones and regions

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

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

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



$DZ(2)$ — time steps — action steps

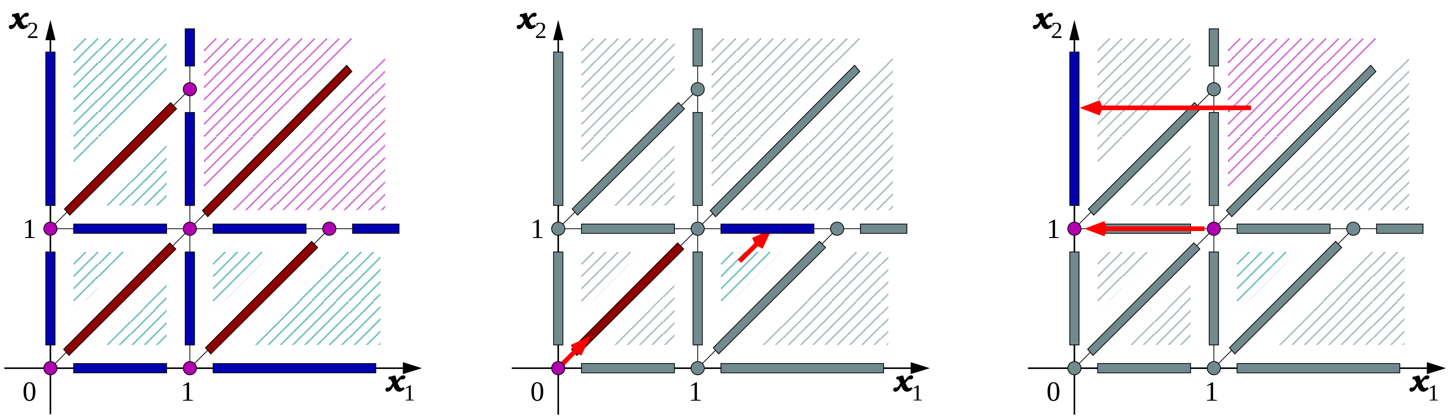
$c_{max} = 1$ — by transition $s \xrightarrow{x_1:=0} s'$

Detailed zones and regions

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

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

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

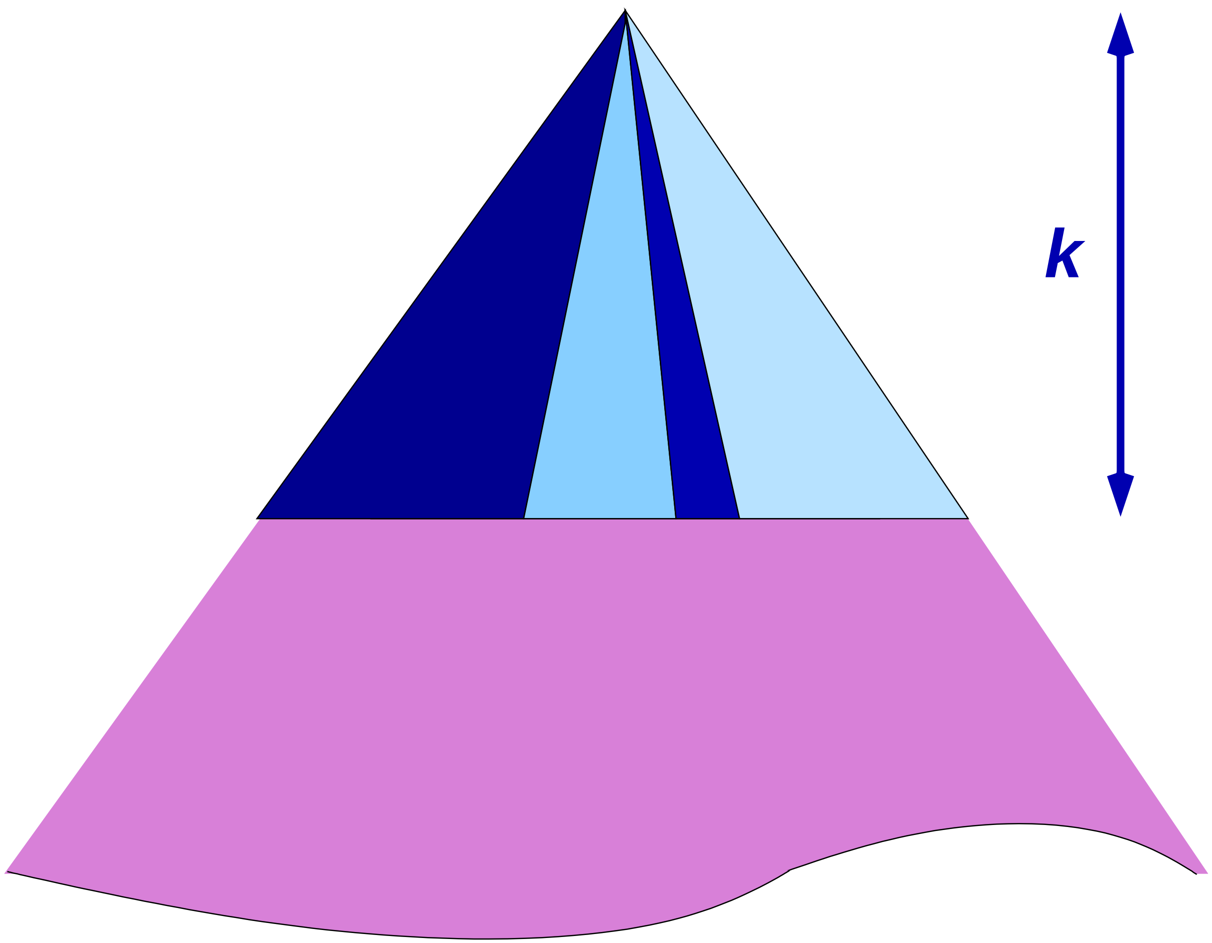


$DZ(2)$ time steps action steps

$c_{max} = 1$ by transition $s \xrightarrow{x_1 := 0} s'$

(l, Z) - a *(detailed) region*, where $l \in L$ and $Z \in DZ(n)$.

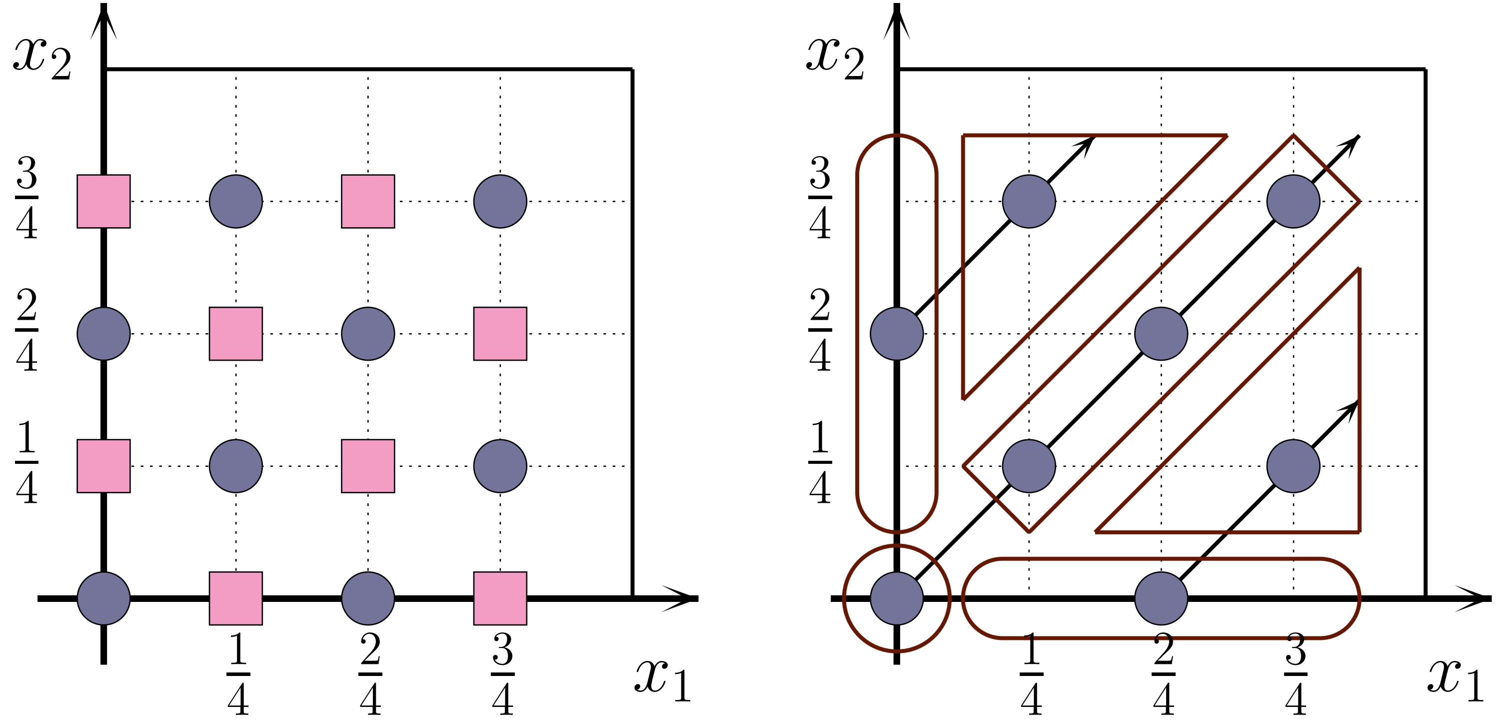
BMC: exploiting a part of the model



Selecting submodels of the k-model

Discretization scheme

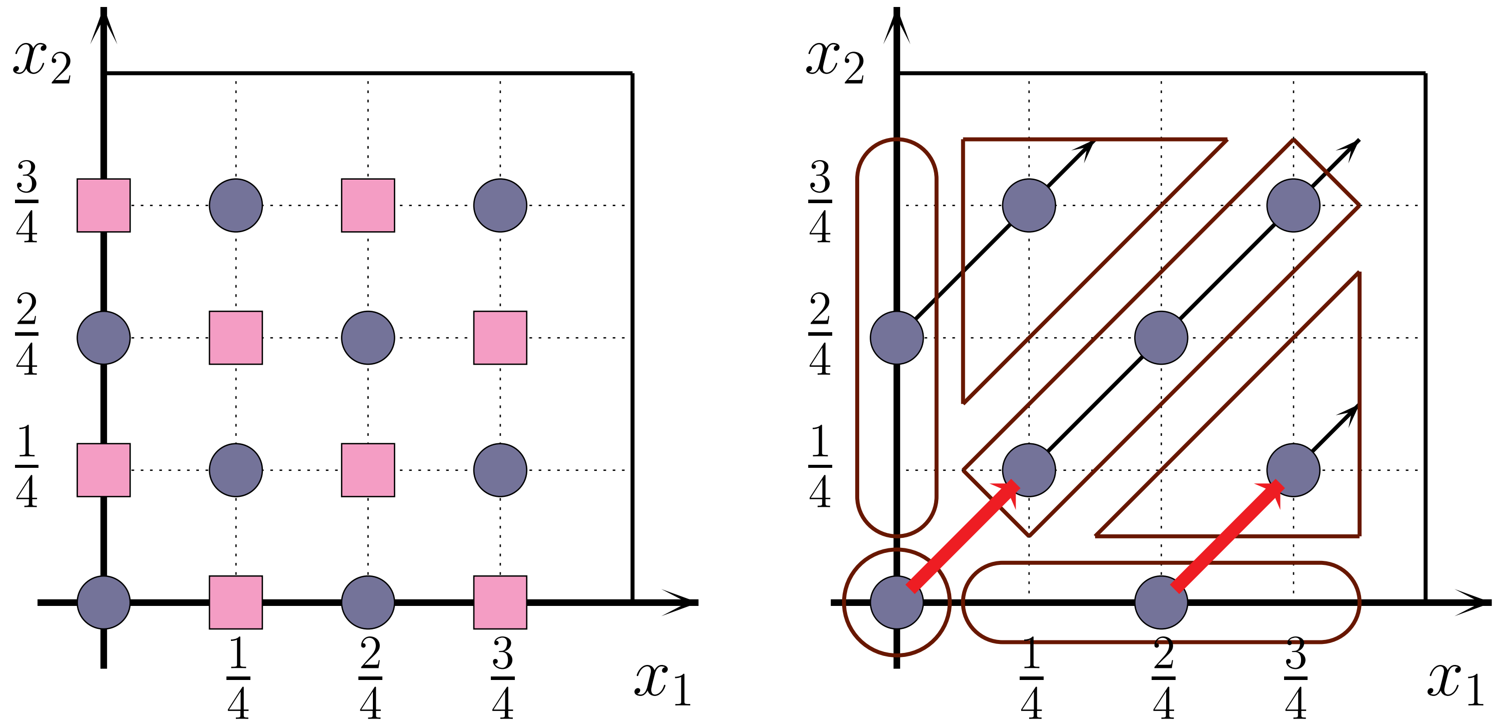
[Asarin, Bozga, Kerbrat, Maler, Pnueli, Rasse, *"Data-Structures for the Verification of Timed Automata"*]



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

Discretization scheme

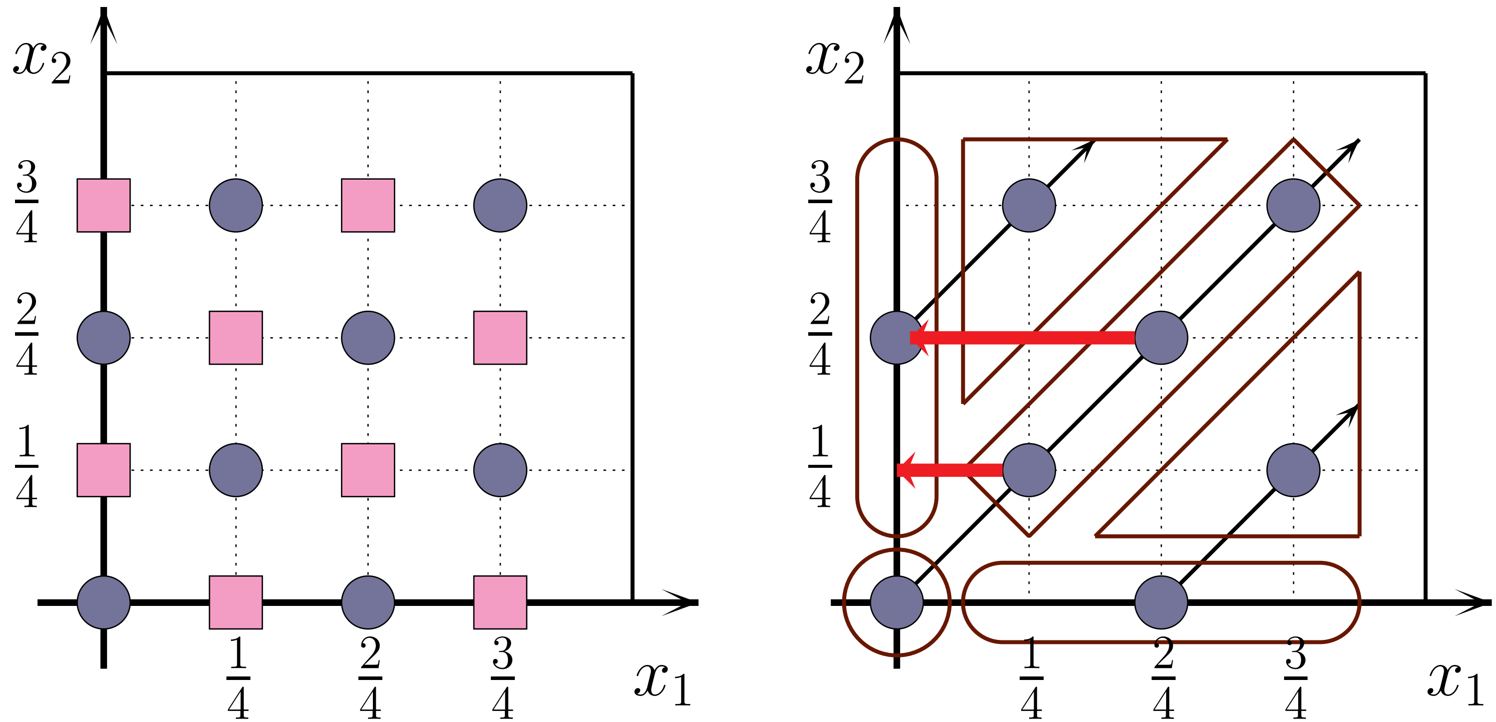
[Asarin, Bozga, Kerbrat, Maler, Pnueli, Rasse, *"Data-Structures for the Verification of Timed Automata"*]



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

Discretization scheme

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



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

Discretization

$\mathcal{A}$ - a timed automaton with n clocks

- $\Delta = \frac{1}{m}$ as a discretization step, where $m = 2^{\lceil \log_2 (2 \cdot n) \rceil}$
- $\mathbb{D} = \{l \cdot \Delta \mid 0 \leq l \cdot \Delta < 2 \cdot c_{max} + 2\}$ - the set of discretized values, and
- $E = \{l \cdot \Delta \mid 0 \leq l \cdot \Delta < c_{max} + 1\}$ - the set of labels.

The discrete representatives:

$$\mathbb{U} = \{u \in \mathbb{D}^n \mid (\forall x \in \mathcal{X})(\exists l \in \mathbb{N}) u(x) = 2l\Delta \ \vee$$
$$(\forall x \in \mathcal{X})(\exists l \in \mathbb{N}) u(x) = (2l+1)\Delta\}$$

"SMART" discretized region graph for TA

Discrete abstract model for a timed automaton:

$$DM(\mathcal{A}) = ((S,\ (l^0, (0, \ldots, 0)),\ \longrightarrow), V)$$

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

- ”SMART” time transitions
  (the transitive closure of timed steps)
- ”SMART” action transitions
  (action steps combined with ’adjust’ steps to remain within $\mathbb{U}$).

Special k-paths

A *special k-path* is a finite sequence $\pi = \langle u_0, \cdots, u_k \rangle$ of states of $DM(\mathcal{A})$ such that

- $u_0 = (l^0, (0, \ldots, 0))$ is the initial state of $DM(\mathcal{A})$,
- $u_i \longrightarrow u_{i+1}$ for each $0 \leq i < k$,
- the transition $u_0 \longrightarrow u_1$ is a time transition,
- each time transition is followed by an action transition,
- each action transition is followed by a time transition.

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

Checking reachability

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

$udp(\mathbf{u})$ - a propositional formula encoding an undesirable property.

$path_k(\mathbf{u}_0, \ldots, \mathbf{u}_k)$ - a propositional formula encoding all the special k-paths.

$$\alpha = path_k(\mathbf{u}_0, \ldots, \mathbf{u}_k) \wedge \bigvee_{i=0}^{k} udp(\mathbf{u}_i)$$

The property is reachable $\iff$ α is satisfiable.

Checking unreachability - an intuition

To prove unreachability of states satisfying $udp(\mathbf{u})$:

- Search for a longest path from an arbitrary state via states satisfying $\neg udp(\mathbf{u})$ to a state satisfying $udp(\mathbf{u})$
- if such a path π exists, then
  - a path from the initial state to a state satisfying $udp(\mathbf{u})$ cannot be longer than π,
  - therefore it is sufficient to test reachability only for $k = length(\pi)$

Free special k-paths

A *free special k-path* is a finite sequence $\pi = \langle u_0, \cdots, u_k \rangle$ of states of $DM(\mathcal{A})$ such that

- $u_i \longrightarrow u_{i+1}$ for each $0 \leq i < k$,
- the transition $u_0 \longrightarrow u_1$ is a time transition,
- each time transition is followed by an action transition,
- each action transition is followed by a time transition.

Searching for the longest witness

Find the length of a longest free special k-path π s.t.:

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

$freepath_k(\mathbf{u}_0, \ldots, \mathbf{u}_k)$ - a propositional formula encoding all the free special k-paths.

Check satisfiability of β:

$$\beta = freepath_k(\mathbf{u}_0, \ldots, \mathbf{u}_k) \wedge udp(\mathbf{u}_k) \wedge \bigwedge_{i=0}^{k-1} \neg udp(\mathbf{u}_i)$$

Checking unreachability

$$\beta = freepath_k(\mathbf{u}_0, \ldots, \mathbf{u}_k) \wedge udp(\mathbf{u}_k) \wedge \bigwedge_{i=0}^{k-1} \neg udp(\mathbf{u}_i)$$

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

$$\alpha = path_k(\mathbf{u}_0, \ldots, \mathbf{u}_k) \wedge \bigvee_{i=0}^{k} udp(\mathbf{u}_i)$$

only for $k = k_0 - 2$.

Selected tools for TPNs

- **Tina** - a toolbox for analysis of (time) Petri nets. It constructs (atomic) state class graphs and performs (CTL) LTL or reachability verification
- **Romeo** - provides several methods for translating TPNs to TA and computation of state class graphs.
- **Petri Net Toolbox** - a tool for simulation, analysis and synthesis of discrete event systems based on (Time) Petri net models.

Selected tools for TPNs - cont'd

- **PEP (Programming Environment based on Petri nets)** - various verification algorithms (e.g., reachability and deadlock-freeness checking, partial-order based model checking).
- **INA (Integrated Net Analyser)** - a Petri net analysis tool. INA provides verification by analysis of paths for TPNs.
- **CPN Tools** - a software package for modelling and analysis of both timed and untimed Coloured Petri Nets, enabling their simulation, generating occurrence (reachability) graph, and analysis by place invariants.

Selected tools for TA

- **Kronos** - uses DBM's to perform verification of TCTL using partitioning algorithms.
- **UppAal2k** uses CDD to represent unions of convex clock regions for modelling, simulation and verification of timed automata.
- **Red** is a model checker based on CRD. It supports TCTL model checking.

Selected tools for TA

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

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

Experimental results

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

Abstract models for nets of [YR98] by some different tools

Experimental results - cont'd

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

Abstract models for Fischer's protocol by some different tools

parameters: $\Delta = 1, \delta = 2$ or $\Delta = 2, \delta = 1$, max. 128 MB RAM and max. 1800 s

Experimental results - cont'd

	TPN to TA				TA			
NoP	vars	clauses	sec	MB	vars	clauses	sec	MB
8	61530	176319	10890.1	61.31	36461	103228	2326.3	34.5
8	22552	64442	8.0	21.9	13357	37666	0.7	20.5
10	29918	86002	14.5	23.5	17283	49034	1.1	20.2
50	378203	1118763	99.7	100.5	156941	459722	21.8	31.7
104	1411156	4200809	1397.6	577.9	528136	1562194	218.8	75.9
310	-	-	-	-	3873940	11557290	21723.6	648.3

BMC of VerICS for Fischer's protocol modelled by TPN and TA

parameters: $\Delta = 1$, $\delta = 2$ **(k=44) or** $\Delta = 2$, $\delta = 1$ **(k=17)**

References

[1] W. Penczek, A. Pólrola: Specification and Model Checking of Temporal Properties in Time Petri Nets and Timed Automata. ICATPN 2004: 37-76

[2] W. Penczek, A. Pólrola: Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach. Springer 2006.

[3] A Pólrola, W Penczek: Minimization Algorithms for Time Petri Nets. Fundam. Inform. 60(1-4): 307-331 (2004)