A Concurrency-Preserving Translation from Time Petri Nets to Networks of Timed Automata

Sandie Balaguer, Thomas Chatain, Stefan Haar

LSV – ENS Cachan, INRIA, CNRS – France

ACTS – January 28, 2011

Motivation

Concurrency

- Two actions that might be performed in any order leading to the same state are concurrent. Concurrency can be used to improve the analysis of distributed systems.
- The definition of concurrency in timed systems is not clear since events are ordered both by their occurrence dates and by causality.

2 formalisms

- Networks of timed automata (NTA)
- Time Petri nets (TPN)

Translation between formalisms

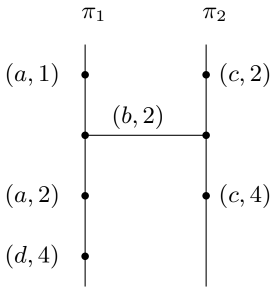
- Theoretical reasons (comparison)
- Practical reasons (verification tools)

Motivation

- Translations from TPN to NTA with preservation of timed words but loss of concurrency

Concurrency-preserving translation

- Runs are represented as timed traces $\neq$ timed words.
- The translation preserves timed traces.
- Some hidden dependencies caused by time are made explicit.

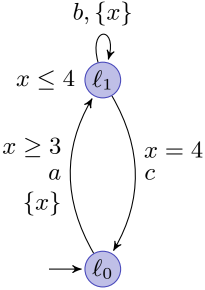


Timed Automata [Alur, Dill, 94]

Definition (Timed Automaton)

A timed automaton is a tuple $\mathcal{A} = (L, \ell_0, C, \Sigma, E, Inv)$ where:

- L is a set of locations,
- $\ell_0 \in L$ is the initial location,
- C is a finite set of clocks,
- Σ is a finite set of actions,
- $E \subseteq L \times \mathcal{B}(C) \times \Sigma \times 2^C \times L$ is a set of edges,
- $Inv : L \to \mathcal{B}(C)$ assigns invariants to locations.



- A location must be left when its invariant reaches its limit.
- An edge cannot be taken if its guard is not satisfied.

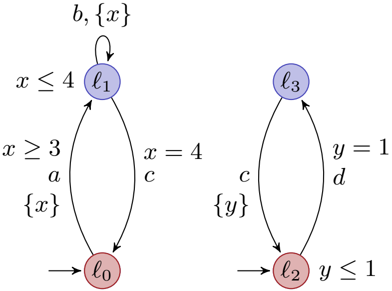
Networks of Timed Automata: $\mathcal{A}_1 \| \ldots \| \mathcal{A}_n$

Action step: $(\vec{\ell}, v) \xrightarrow{a} (\vec{\ell'}, v')$

- If all the automata that share a are ready to perform it.
- Edges labeled by a are taken simultaneously in these automata.

Delay step: $\forall d \in \mathbb{R}_{\geq 0}$, $(\vec{\ell}, v) \xrightarrow{d} (\vec{\ell}, v + d)$

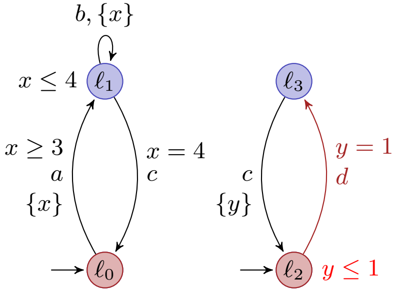
- $v + d$ respects the invariants of the current locations.



$\begin{matrix}(\ell_0, \ell_2) \\ (0, 0)\end{matrix} \xrightarrow{1}$

Networks of Timed Automata: $\mathcal{A}_1 \| \ldots \| \mathcal{A}_n$

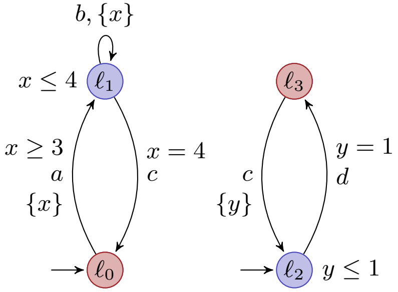
Example run



$$\begin{matrix}(\ell_0, \ell_2) \\ (0, 0)\end{matrix} \xrightarrow{1} \begin{matrix}(\ell_0, \ell_2) \\ (1, 1)\end{matrix} \xrightarrow{d}$$

Networks of Timed Automata: $\mathcal{A}_1 \| \ldots \| \mathcal{A}_n$

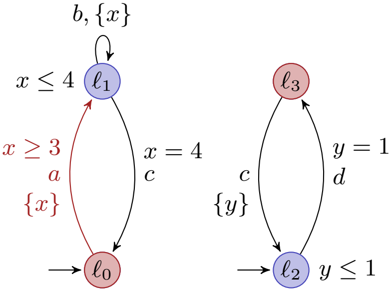
Example run



$$\begin{matrix}(\ell_0,\ell_2)\\(0,0)\end{matrix} \xrightarrow{1} \begin{matrix}(\ell_0,\ell_2)\\(1,1)\end{matrix} \xrightarrow{d} \begin{matrix}(\ell_0,\ell_3)\\(1,1)\end{matrix} \xrightarrow{2.5}$$

Networks of Timed Automata: $\mathcal{A}_1 \| \ldots \| \mathcal{A}_n$

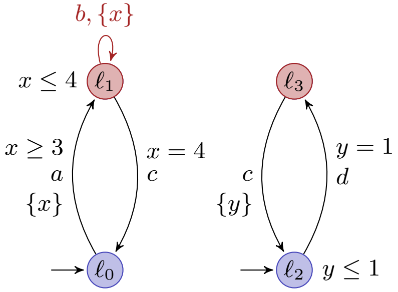
Example run



$$\begin{matrix}(\ell_0, \ell_2)\\(0,0)\end{matrix} \xrightarrow{1} \begin{matrix}(\ell_0, \ell_2)\\(1,1)\end{matrix} \xrightarrow{d} \begin{matrix}(\ell_0, \ell_3)\\(1,1)\end{matrix} \xrightarrow{2.5} \begin{matrix}(\ell_0, \ell_3)\\(3.5,3.5)\end{matrix} \xrightarrow{a}$$

Networks of Timed Automata: $\mathcal{A}_1 \| \ldots \| \mathcal{A}_n$

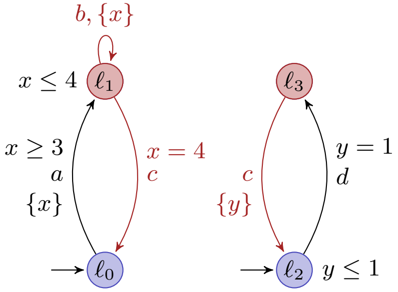
Example run



$$\begin{pmatrix}\ell_0,\ell_2\\0,0\end{pmatrix} \xrightarrow{1} \begin{pmatrix}\ell_0,\ell_2\\1,1\end{pmatrix} \xrightarrow{d} \begin{pmatrix}\ell_0,\ell_3\\1,1\end{pmatrix} \xrightarrow{2.5} \begin{pmatrix}\ell_0,\ell_3\\3.5,3.5\end{pmatrix} \xrightarrow{a} \begin{pmatrix}\ell_1,\ell_3\\0,3.5\end{pmatrix} \xrightarrow{4}$$

Networks of Timed Automata: $\mathcal{A}_1 \| \ldots \| \mathcal{A}_n$

Example run

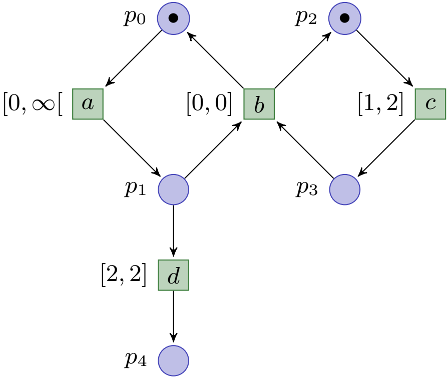


$$\begin{matrix}(\ell_0,\ell_2)\\(0,0)\end{matrix} \xrightarrow{1} \begin{matrix}(\ell_0,\ell_2)\\(1,1)\end{matrix} \xrightarrow{d} \begin{matrix}(\ell_0,\ell_3)\\(1,1)\end{matrix} \xrightarrow{2.5} \begin{matrix}(\ell_0,\ell_3)\\(3.5,3.5)\end{matrix} \xrightarrow{a} \begin{matrix}(\ell_1,\ell_3)\\(0,3.5)\end{matrix} \xrightarrow{4} \begin{matrix}(\ell_1,\ell_3)\\(4,7.5)\end{matrix} \xrightarrow{c} \ldots$$

Time Petri Nets [Merlin, 74]

(P, T, F, M_0, efd, lfd)

- $efd : T \to \mathbb{R}$ earliest firing delay
- $lfd : T \to \mathbb{R} \cup \{\infty\}$ latest firing delay



TPN Semantics

- t is enabled in M: $t \in enabled(M) \Leftrightarrow {}^\bullet t \subseteq M$
- firing t from M: $M \xrightarrow{t} (M' = M - {}^\bullet t + t^\bullet)$
- t' is newly enabled by the firing of t from M: intermediate semantics
  $\uparrow enabled(t', M, t) = \big(t' \in enabled(M')\big) \wedge \big(t' \notin enabled(M - {}^\bullet t)\big)$

Discrete transition: $\forall t \in enabled(M), (M, \nu) \xrightarrow{t} (M', \nu')$ iff

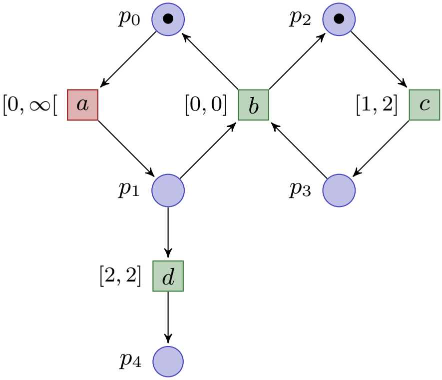
- $efd(t) \leq \nu(t)$,
- $\forall t' \in T, \nu'(t') = \begin{cases} 0 & \text{if } \uparrow enabled(t', M, t) \\ \nu(t') & \text{otherwise.} \end{cases}$

Continuous transition: $\forall d \in \mathbb{R}_{\geq 0}, (M, \nu) \xrightarrow{d} (M, \nu + d)$ iff

- $\forall t \in enabled(M), \nu(t) + d \leq lfd(t)$ urgency

TPN Semantics

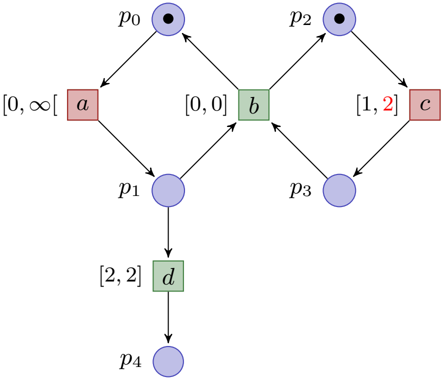
Example run



$$\begin{array}{c}\{p_0, p_2\} \\ (0, \_, 0, \_)\end{array} \xrightarrow{2}$$

TPN Semantics

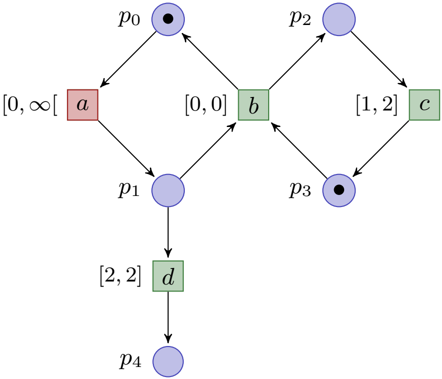
Example run



$$\begin{matrix}\{p_0, p_2\} \\ (0, \_, 0, \_)\end{matrix} \xrightarrow{2} \begin{matrix}\{p_0, p_2\} \\ (2, \_, 2, \_)\end{matrix} \xrightarrow{c}$$

TPN Semantics

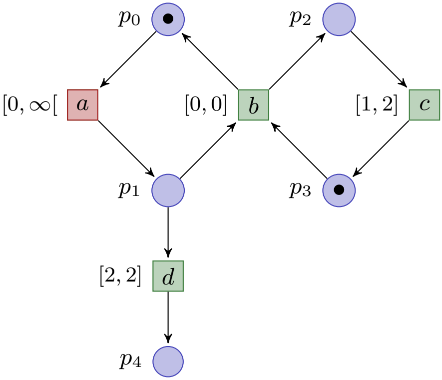
Example run



$$\begin{matrix}\{p_0, p_2\} \\ (0, \_, 0, \_)\end{matrix} \xrightarrow{2} \begin{matrix}\{p_0, p_2\} \\ (2, \_, 2, \_)\end{matrix} \xrightarrow{c} \begin{matrix}\{p_0, p_3\} \\ (2, \_, \_, \_)\end{matrix} \xrightarrow{10}$$

TPN Semantics

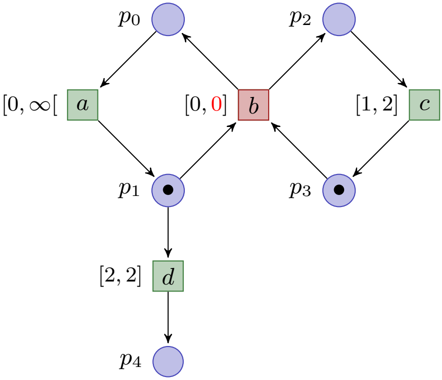
Example run



$$\begin{matrix}\{p_0, p_2\} \\ (0, \_, 0, \_)\end{matrix} \xrightarrow{2} \begin{matrix}\{p_0, p_2\} \\ (2, \_, 2, \_)\end{matrix} \xrightarrow{c} \begin{matrix}\{p_0, p_3\} \\ (2, \_, \_, \_)\end{matrix} \xrightarrow{10} \begin{matrix}\{p_0, p_3\} \\ (12, \_, \_, \_)\end{matrix} \xrightarrow{a}$$

TPN Semantics

Example run

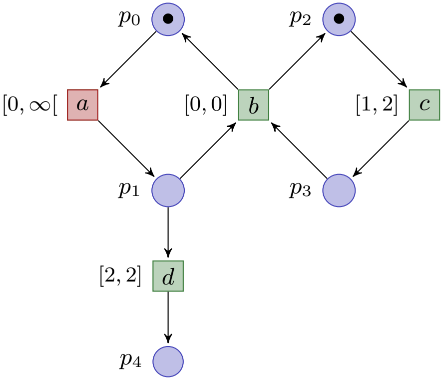


$$\begin{matrix}\{p_0, p_2\} \\ (0, \_, 0, \_)\end{matrix} \xrightarrow{2} \begin{matrix}\{p_0, p_2\} \\ (2, \_, 2, \_)\end{matrix} \xrightarrow{c} \begin{matrix}\{p_0, p_3\} \\ (2, \_, \_, \_)\end{matrix} \xrightarrow{10} \begin{matrix}\{p_0, p_3\} \\ (12, \_, \_, \_)\end{matrix} \xrightarrow{a} \begin{matrix}\{p_1, p_3\} \\ (\_, 0, \_, 0)\end{matrix} \xrightarrow{b}$$

b and d are newly enabled.

TPN Semantics

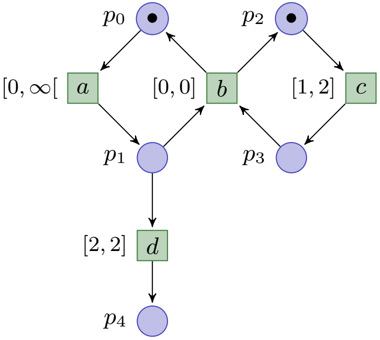
Example run

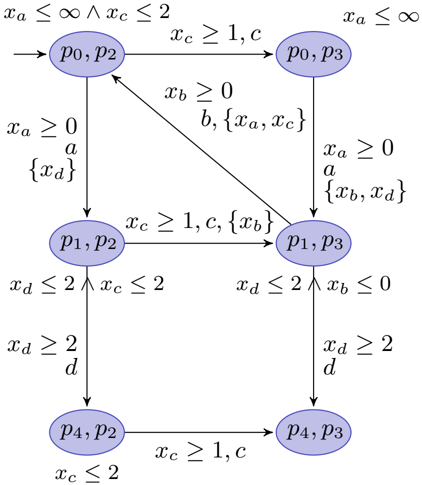


$$\begin{matrix}\{p_0, p_2\} \\ (0, \_, 0, \_)\end{matrix} \xrightarrow{2} \begin{matrix}\{p_0, p_2\} \\ (2, \_, 2, \_)\end{matrix} \xrightarrow{c} \begin{matrix}\{p_0, p_3\} \\ (2, \_, \_, \_)\end{matrix} \xrightarrow{10} \begin{matrix}\{p_0, p_3\} \\ (12, \_, \_, \_)\end{matrix} \xrightarrow{a} \begin{matrix}\{p_1, p_3\} \\ (\_, 0, \_, 0)\end{matrix} \xrightarrow{b} \begin{matrix}\{p_0, p_2\} \\ (0, \_, 0, \_)\end{matrix}$$

TPN Semantics

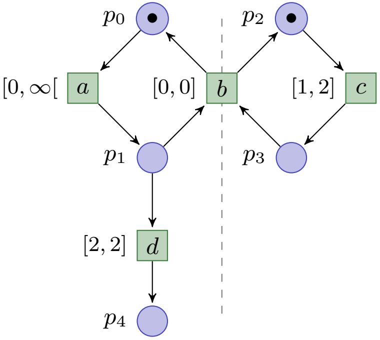
Can be seen as a TA

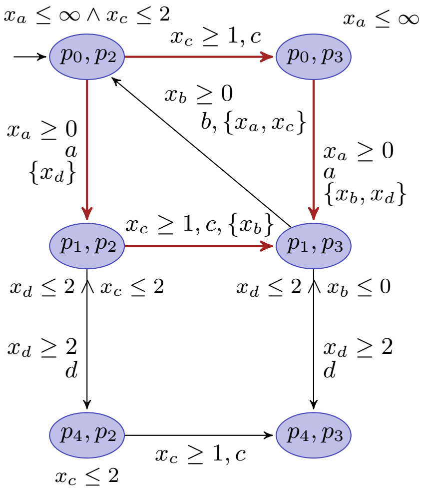




TPN Semantics

Can be seen as a TA





Partial order semantics for distributed systems

NTA and TPN represent distributed systems

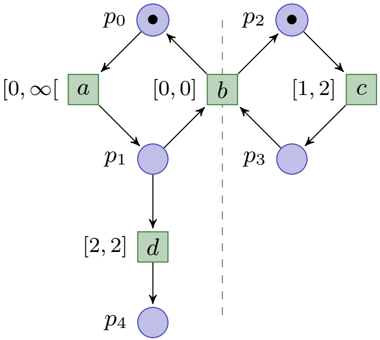
- Composition of several (physical) components
- Notion of process
  - In a NTA, each automaton is a process.
  - PNs usually built as products of transition systems

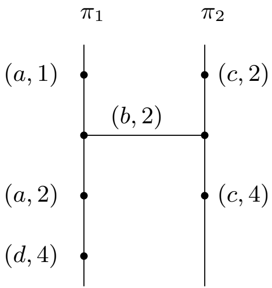
Usual semantics as timed words does not reflect the distribution of actions.
Partial order semantics reflects the distribution of actions.

Timed traces

A timed trace over the alphabet Σ, and the set of processes $\Pi = (\pi_1, \ldots, \pi_n)$ is a tuple $\mathcal{W} = (E, \preccurlyeq, \lambda, t, proc)$ where:

- E is a set of events,
- $\preccurlyeq \subseteq (E \times E)$ is a partial order on E ($\preccurlyeq_{|\pi_i}$ is a total order),
- $\lambda : E \to \Sigma$ is a labeling function,
- $t : E \to \mathbb{R}_{\geq 0}$ maps each event to a date,
- $proc : \Sigma \to 2^{\Pi}$ is a distribution of actions.





Distributed timed language

Definition (Distributed timed language)

A distributed timed language is a set of timed traces.

- A timed trace is defined by a timed word and a distribution of actions ($proc : \Sigma \rightarrow 2^{\Pi}$).
- A distributed timed language is defined by a timed language and a distribution of actions.

S-invariants [Lautenbach, 75], [Reisig, 85], [Desel, Esparza, 95]...

$X : P \to \mathbb{N}$, solution of the equation $X \cdot \mathbf{N} = \mathbf{0}$, where $\mathbf{N}$ is the incidence matrix.

We consider S-invariants X s.t. $X : P \to \{0, 1\}$ (subsets of places).

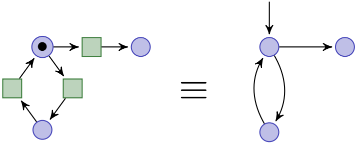
Definition, properties

- X is an S-invariant of $N \Leftrightarrow \forall t \in T, \sum_{p \in {}^\bullet t} X(p) = \sum_{p \in t^\bullet} X(p)$
- X is an S-invariant of $N \Rightarrow \forall M, \sum_{p \in X} M(p) = \sum_{p \in X} M_0(p)$

S-invariants as processes

- A net (P, T, F) is an S-net if $\forall t \in T$, $|{}^\bullet t| = |t^\bullet| = 1$.

An S-net with one token can be seen as an automaton.



- The subnet (P', T', F') of N is a P-closed subnet of N if $T' = {}^\bullet P' \cup P'^\bullet$.

Definition

The net $N = (P, T, F)$ is decomposable iff there exists a set of P-closed S-nets $N_i = (P_i, T_i, F_i)$ that covers N.

[Desel, Esparza, 95] Well-formed free-choice nets are covered by strongly connected P-closed S-nets (S-components).

Decomposition

Proposition

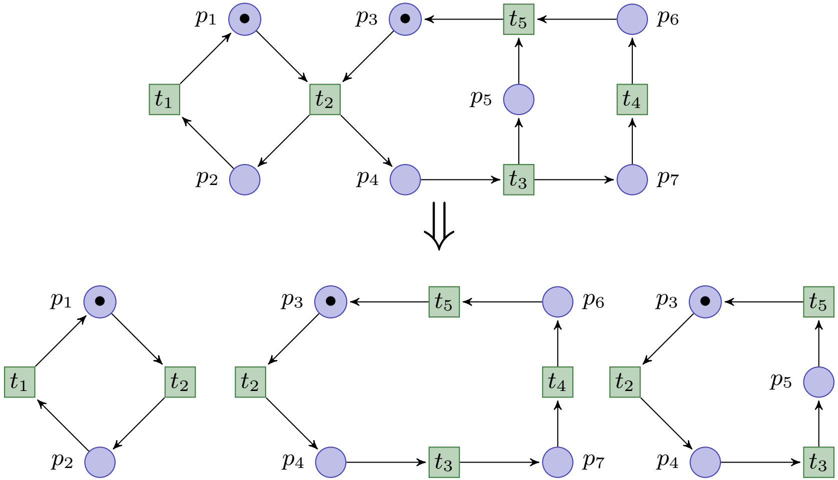
A Petri net (P, T, F) is decomposable in the subnets $N_1, \ldots, N_n$ iff there exists a set of S-invariants $\{X_1, \ldots X_n\}$ such that,

- $\forall i \in [1..n], X_i : P \rightarrow \{0, 1\}$,
  X_i is the characteristic function of P_i over P.
- $\forall i \in [1..n], \forall t \in T, \sum_{p \in {}^\bullet t} X_i(p) = 1 \left(= \sum_{p \in t^\bullet} X_i(p)\right)$,
  N_i is an S-net.
- $\forall p \in P, \sum_i X_i(p) \geq 1$
  The set covers the net.

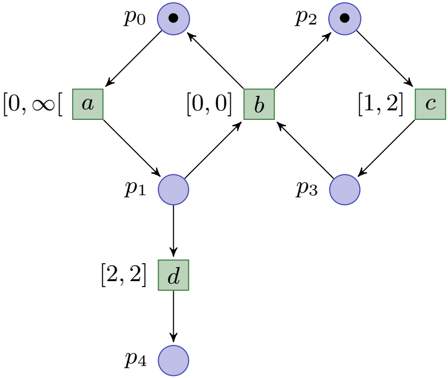
The processes are the subnets spanned by the supports of these S-invariants.

Decomposition

An example

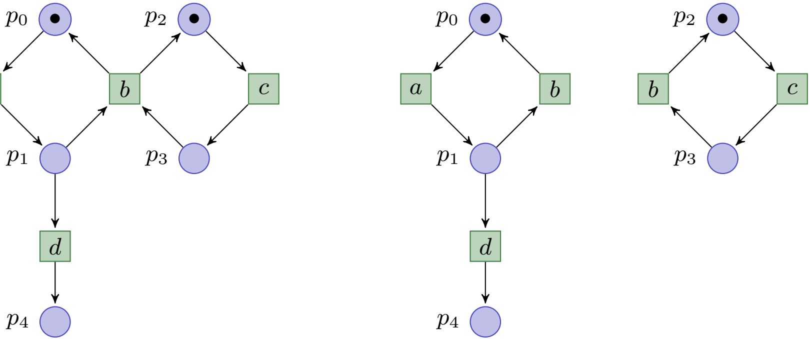


Translation



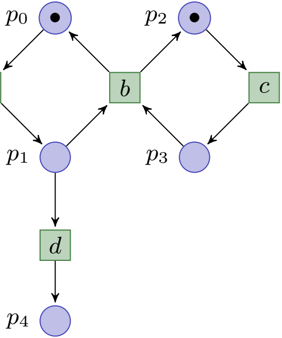
Translation

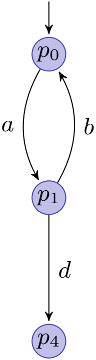
Decomposing the untimed PN.

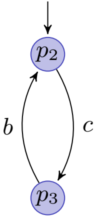


Translation

Translating each subnet into an automaton.

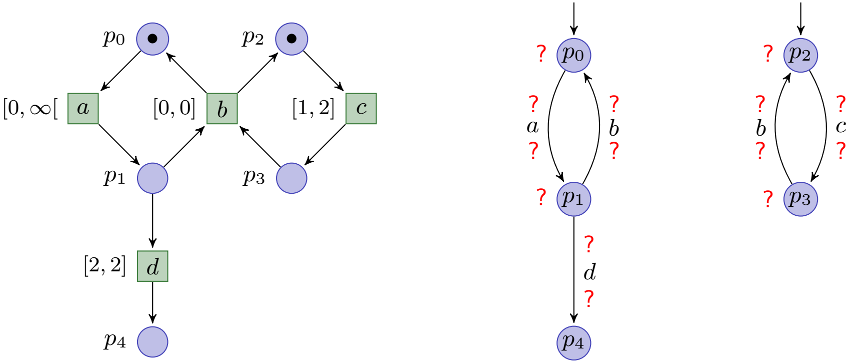






Translation

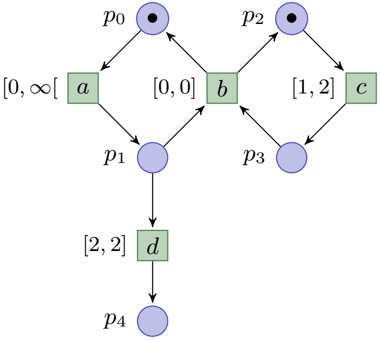
Adding timing constraints (resets, guards and invariants).

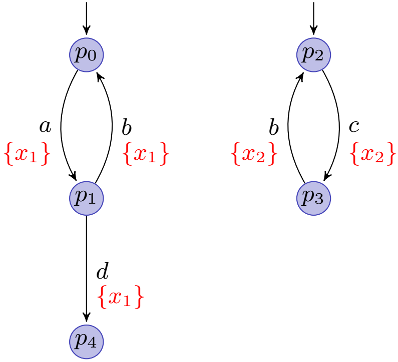


Translation

t enabled $\implies \nu(t) = \min_{\{i|t\in\Sigma_i\}} \big(v(x_i)\big)$

We add one clock to each automaton. The clock is reset on each edge.

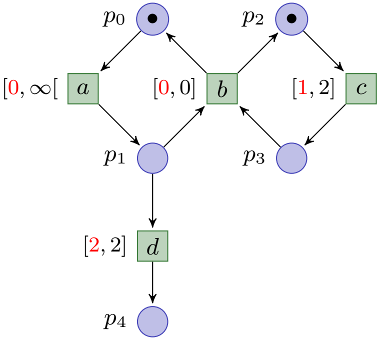


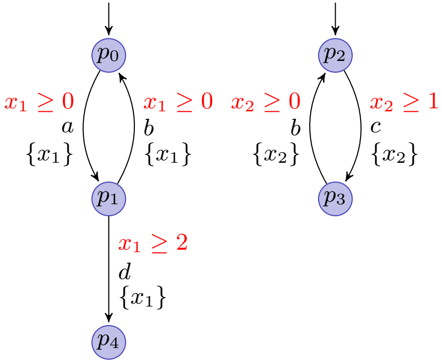


Translation

t enabled $\Longrightarrow \nu(t) = \min_{\{i|t\in\Sigma_i\}} \big(v(x_i)\big)$

We add guards. $\min_{\{i|t\in\Sigma_i\}} \big(v(x_i)\big) \geq efd(t) \Leftrightarrow \forall i$ s.t. $t \in \Sigma_i, v(x_i) \geq efd(t)$

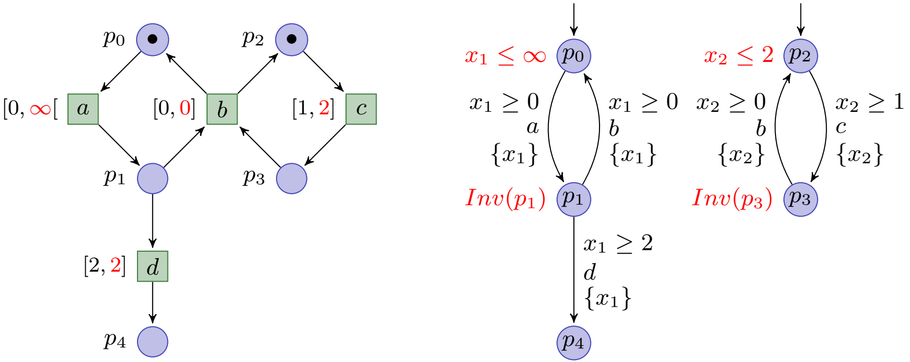




Translation

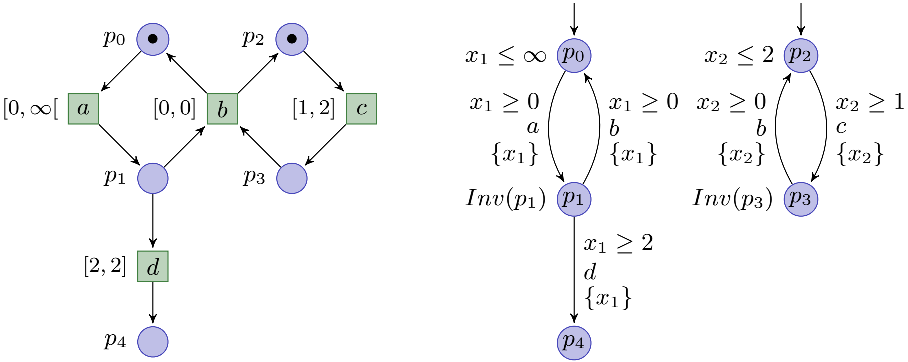
$$t \text{ enabled} \implies \nu(t) = \min_{\{i | t \in \Sigma_i\}} \left(v(x_i)\right)$$

We add invariants. $Inv_i(p) \equiv \bigwedge_{t \in p^\bullet} \left(t \text{ enabled } \Rightarrow \nu(t) \leq lfd(t)\right)$



$$
\begin{aligned}
Inv(p_1) &\equiv \overbrace{(p_1 \Rightarrow x_1 \leq 2)}^{Inv(d)} \wedge \overbrace{((p_1 \wedge p_3) \Rightarrow (\min(x_1, x_2) \leq 0)))}^{Inv(b)} \\
&\equiv (x_1 \leq 2) \wedge (\neg p_3 \vee (x_1 \leq 0 \vee x_2 \leq 0)) \\
Inv(p_3) &\equiv (p_1 \wedge p_3) \Rightarrow (\min(x_1, x_2) \leq 0)) \\
&\equiv (\neg p_1 \vee (x_1 \leq 0 \vee x_2 \leq 0))
\end{aligned}
$$

Translation



$$Inv(p_1) \equiv (x_1 \leq 2) \wedge (\neg p_3 \vee (x_1 \leq 0 \vee x_2 \leq 0))$$
$$Inv(p_3) \equiv (\neg p_1 \vee (x_1 \leq 0 \vee x_2 \leq 0))$$

It is unavoidable to share clocks and states.

Properties of the translation

1. Timed bisimulation: (M, v) denotes a state of the NTA $\mathcal{S}$ and (M, ν) a state of the TPN $\mathcal{N}$.

$$(M, v)\mathcal{R}(M, \nu) \Leftrightarrow \forall t \in enabled(M), \nu(t) = \min_{\{i | t \in \Sigma_i\}} \big(v(x_i)\big)$$

   We show that $\mathcal{R}$ is a timed bisimulation.

2. Distributed timed language equivalence:
   - Timed bisimulation between the TTS of $\mathcal{S}$ and $\mathcal{N}$.
   - Bijection between the processes of $\mathcal{S}$ and those of $\mathcal{N}$ (same distribution of actions up to a renaming of processes).

Size of the resulting NTA

Decomposition: at most $|P|$ processes

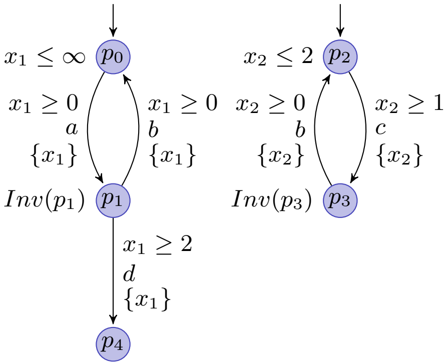
- at most $|P|^2$ locations,
- at most $|T| \times |P|$ edges (exactly $\sum_{t \in T} |\{i \mid t \in \Sigma_i\}|$ edges).

Timing information:

- at most $|P|$ clocks,
- $\sum_{t \in T} |\{i \mid t \in \Sigma_i\}|$ guards,
- $\sum_{t \in T} |\{i \mid t \in \Sigma_i\}|$ clock comparisons in the invariants ($Inv(t)$ can be attached to one place).

Know thy neighbour!

Given a TPN $\mathcal{N}$, in general, there does not exist any NTA $\mathcal{S}$ using the local syntax (clocks and current locations are not shared) such that $\mathcal{N}$ and $\mathcal{S}$ have the same distributed timed language.



$Inv(p_1) \equiv (x_1 \leq 2) \wedge (\neg p_3 \vee (x_1 \leq 0 \vee x_2 \leq 0))$
$Inv(p_3) \equiv (\neg p_1 \vee (x_1 \leq 0 \vee x_2 \leq 0))$

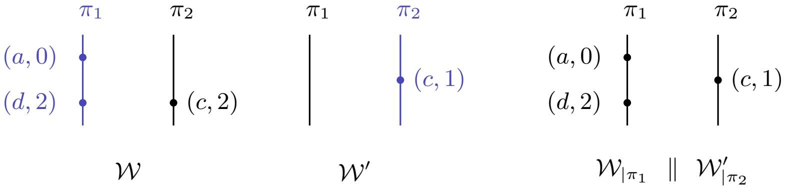
Know thy neighbour!

Given a TPN $\mathcal{N}$, in general, there does not exist any NTA $\mathcal{S}$ using the local syntax (clocks and current locations are not shared) such that $\mathcal{N}$ and $\mathcal{S}$ have the same distributed timed language.

Lemma

Let $\mathcal{S}$ be a network of n timed automata that do not read the state of the other automata, then for any $\mathcal{W}_1, \ldots, \mathcal{W}_n$ admissible timed traces without synchronization and stopping at a same date θ, $\mathcal{W}_{1|\pi_1} \parallel \cdots \parallel \mathcal{W}_{n|\pi_n}$ is also an admissible timed trace stopping at θ.

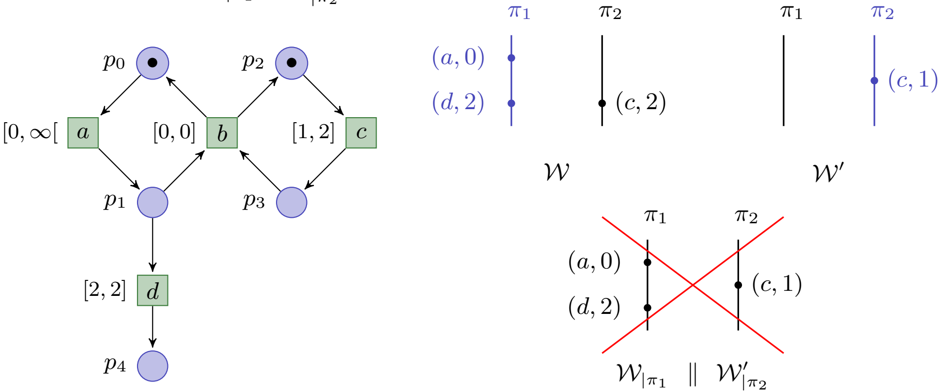
Proof



Know thy neighbour!

Given a TPN $\mathcal{N}$, in general, there does not exist any NTA $\mathcal{S}$ using the local syntax (clocks and current locations are not shared) such that $\mathcal{N}$ and $\mathcal{S}$ have the same distributed timed language.

Counterexample: $\mathcal{W}_{|\pi_1} \parallel \mathcal{W}'_{|\pi_2}$ should be admissible.

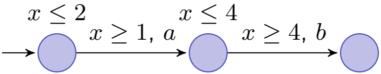


Reverse translation: from NTA to TPN

Sequential semantics: [Bérard, Cassez, Haddad, Lime, Roux, 06] When are Timed Automata weakly timed bisimilar to Time Petri Nets?
But we want to preserve the distributed semantics.

1. Translation of each TA in a finite "time S-net" with one token

   But finite time S-nets with 1 token are strictly less expressive than TA with 1 clock



$\neq$ time S-net with one token

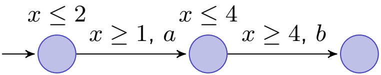
Reverse translation: from NTA to TPN

Sequential semantics: [Bérard, Cassez, Haddad, Lime, Roux, 06] When are Timed Automata weakly timed bisimilar to Time Petri Nets?
But we want to preserve the distributed semantics.

1. Translation of each TA in a finite "time S-net" with one token

   But finite time S-nets with 1 token are strictly less expressive than TA with 1 clock

2. Considering the translation into more general nets,

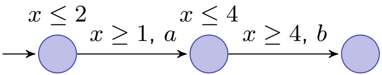


$\neq$ time S-net with one token

Reverse translation: from NTA to TPN

Sequential semantics: [Bérard, Cassez, Haddad, Lime, Roux, 06] When are Timed Automata weakly timed bisimilar to Time Petri Nets?
But we want to preserve the distributed semantics.

1. Translation of each TA in a finite "time S-net" with one token
   But finite time S-nets with 1 token are strictly less expressive than TA with 1 clock
2. Considering the translation into more general nets,
3. Composing the nets.



$\neq$ time S-net with one token

Conclusion

Summary

- Timed trace and distributed timed language: description of a distributed semantics where concurrency is not erased
- Translation from a TPN to a NTA based on the decomposition in processes
  - Correctness w.r.t. the distributed timed language
  - Usable in practice (small tests with Uppaal)
  - Readable and close to the modeled system: processes are preserved

Future work

- Identification of TPN with good decompositional properties (no need to share clocks).
- Explore timed concurrency
  - Definition and properties
  - Use in verification tools

    [Lugiez, Niebert, Zennou, 05] A partial order semantics approach to the clock explosion problem of timed automata

    [Niebert, Qu, 06] invariants