Controlling Petri Net Behavior Using Time Constraints

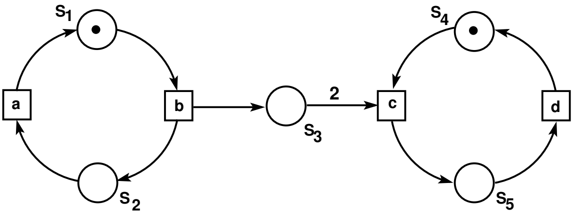
Irina Lomazova[1], Louchka Popova-Zeugmann[2]

[1]National Research University Higher School of Economics, Moscow, Russia
[2]Humboldt-Universität zu Berlin, Berlin, Germany

CS&P 2015

Outline

1. Preliminaries: Petri nets, boundedness, liveness.
2. Problem statement and motivating examples.
3. Time Petri nets.
4. Computing intervals for making a live net also bounded.
5. Validation.
6. Some comparisons.
7. Conclusions.



$(\mathcal{N}^*, m_0^*)$: a *live* Petri net with one *unbounded* place s_3.

A Petri net is *live* iff every transition is potentially enabled in any reachable marking.

A run is called *feasible* iff it starts from a reachable marking.

Problem statement

To transform a live and unbounded Petri nets into a live and bounded one by adding some control.

Problem statement

To transform a live and unbounded Petri nets into a live and bounded one by adding some control.

Important for applications in business process management (soundness = boundedness + liveness), biology, etc.

Problem statement

To transform a live and unbounded Petri nets into a live and bounded one by adding some control.

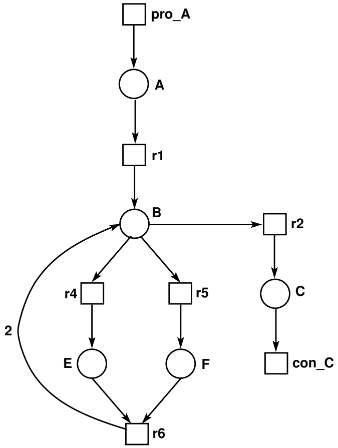
Important for applications in business process management (soundness = boundedness + liveness), biology, etc.

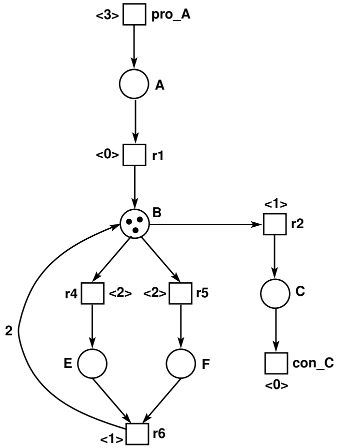
Adding time durations to transitions (M. Heiner, 2007)

When a Petri net is covered by transition invariants, these invariants can be used for computing time durations for transitions and thus transforming a live and unbounded Petri net into a live and bounded Timed Petri net with the same structure.

Some motivating examples: time durations

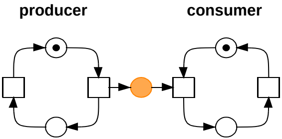
[Heiner, 2006]



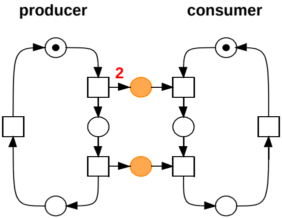


Live and unbounded Petri nets. The right one can be transformed into live and bounded by adding time durations.

Some motivating examples



weakly bounded

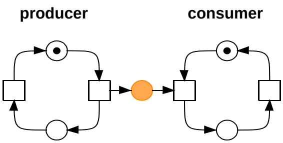


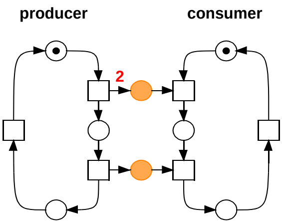
not weakly bounded

[DESEL 2006], WEAKLY BOUNDED PETRI NETS; AWPN 2006

A Petri net is *weakly bounded* iff it is unbounded, but for every reachable marking a bounded run is enabled.

Some motivating examples





weakly bounded

not weakly bounded

[DESEL 2006], WEAKLY BOUNDED PETRI NETS; AWPN 2006

A Petri net is *weakly bounded* iff it is unbounded, but for every reachable marking a bounded run is enabled.

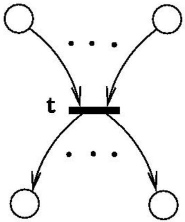
The distinction between bounded, weakly bounded and not weakly bounded Petri nets is very important for applications.

Time Petri nets

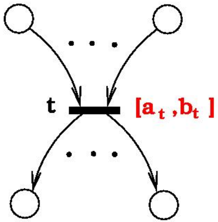
Let $(\mathcal{N}, m_0)$ be a live and unbounded Petri net,

We present an algorithm for finding time intervals (for transitions) that will make the net bounded, keeping its liveness.

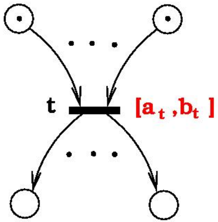
Time Petri nets



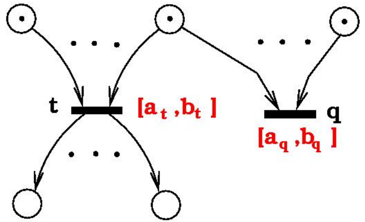
Time Petri nets



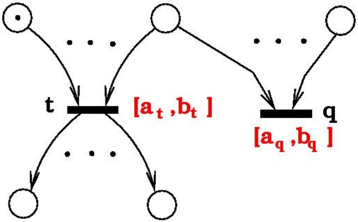
Time Petri nets



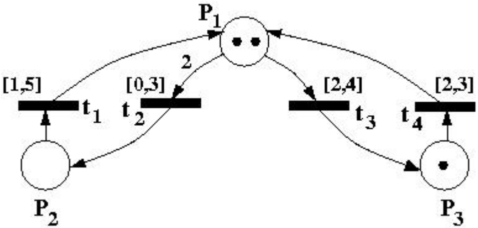
Time Petri nets



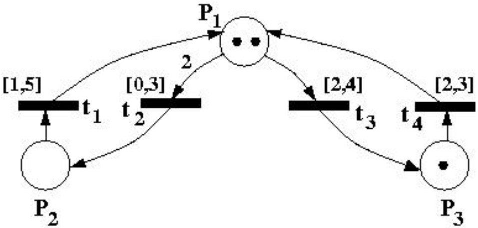
Time Petri nets



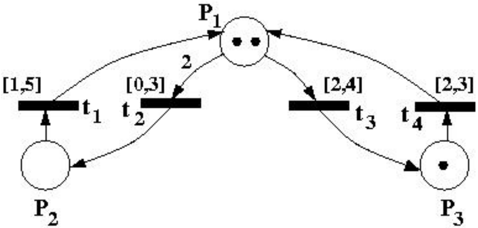
Time Petri net



Time Petri net

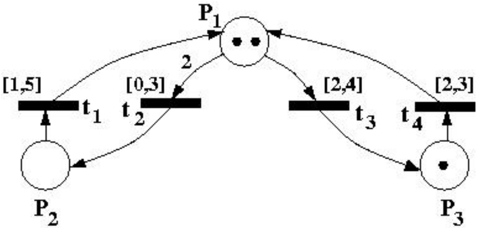


- $m_0 = (2, 0, 1)$ *p*-marking



- $m_0 = (2, 0, 1)$ p-marking
- $h_0 = (\sharp, 0, 0, 0)$ t-marking

Time Petri net



- $m_0 = (2, 0, 1)$ *p*-marking
- $h_0 = (\sharp, 0, 0, 0)$ *t*-marking
- $z := (m, h)$ state

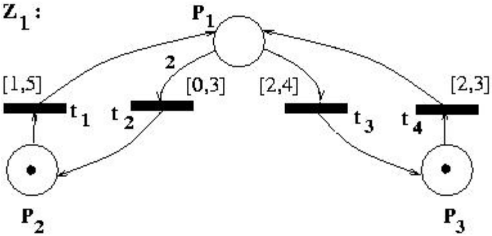
$h(t)$ is the time shown by the clock of t since the last enabling of t.

Let $\mathcal{Z}$ be a TPN and let $z = (m, h)$, $z' = (m', h')$ be two states.
$\mathcal{Z}$ changes from state $z = (m, h)$ into the state $z' = (m', h')$ by:

firing a transition / \ **time elapsing**

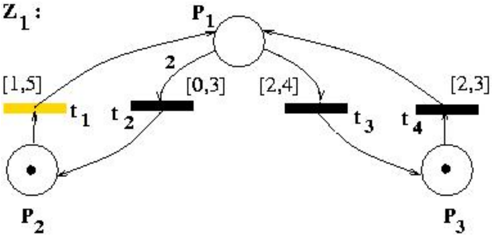
Notation: $z \xrightarrow{t} z'$ $z \xrightarrow{\tau} z'$

Time Petri net



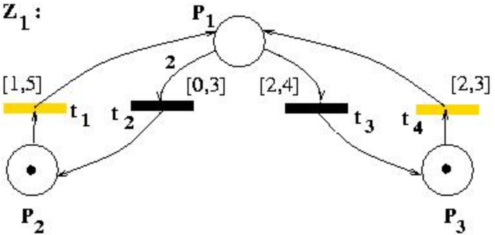
$$\left(m_0, \begin{pmatrix} 0 \\ \sharp \\ \sharp \\ 0 \end{pmatrix}\right)$$

Time Petri net



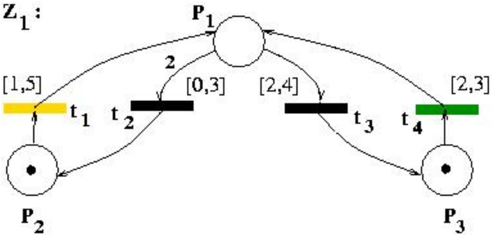
$$(m_0, \begin{pmatrix} 0 \\ \sharp \\ \sharp \\ 0 \end{pmatrix}) \xrightarrow{1.3} (m_1, \begin{pmatrix} 1.3 \\ \sharp \\ \sharp \\ 1.3 \end{pmatrix})$$

Time Petri net



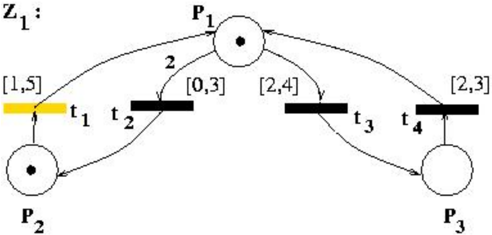
$$z_0 \xrightarrow{1.3} (m_1, \begin{pmatrix} 1.3 \\ \sharp \\ \sharp \\ 1.3 \end{pmatrix}) \xrightarrow{1.0} (m_2, \begin{pmatrix} 2.3 \\ \sharp \\ \sharp \\ 2.3 \end{pmatrix})$$

Time Petri net



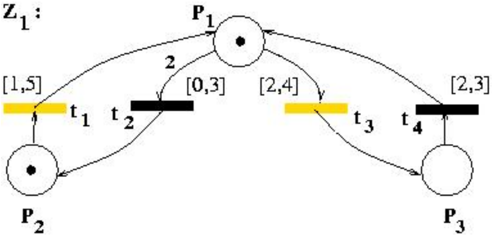
$$z_0 \xrightarrow{1.3} \xrightarrow{1.0} \left(m_2, \begin{pmatrix} 2.3 \\ \sharp \\ \sharp \\ 2.3 \end{pmatrix}\right) \xrightarrow{t_4}$$

Time Petri net



$$z_0 \xrightarrow{1.3} \xrightarrow{1.0} \left(m_2, \begin{pmatrix} 2.3 \\ \sharp \\ \sharp \\ 2.3 \end{pmatrix}\right) \xrightarrow{t_4} \left(m_3, \begin{pmatrix} 2.3 \\ \sharp \\ 0.0 \\ \sharp \end{pmatrix}\right)$$

Time Petri net



$$z_0 \xrightarrow{1.3} \xrightarrow{1.0} \xrightarrow{t_4} \left(m_3, \begin{pmatrix} 2.3 \\ \sharp \\ 0.0 \\ \sharp \end{pmatrix}\right) \xrightarrow{2.0} \left(m_4, \begin{pmatrix} 4.3 \\ \sharp \\ 2.0 \\ \sharp \end{pmatrix}\right)$$

Time Petri net

Notations:

- **transition sequence:** $\sigma = t_1 \cdots t_n$,
- **run:** $\sigma(\tau) = \tau_0\, t_1\, \tau_1\, \cdots\, \tau_{n-1}\, t_n\, \tau_n$ and $\tau_i \in \mathbb{R}_0^+$
- **parametric run:** $(\sigma(\tau) = \tau_0\, t_1\, \tau_1\, \cdots\, \tau_{n-1}\, t_n\, \tau_n, B_\sigma)$ and τ_i are variables which satisfied a set of conditions B_σ
- **parametric state:** $(z_\sigma, B\sigma)$
- **parametric state space:** $parStSp(\mathcal{Z}) = \bigcup\limits_{\sigma} (z_\sigma, B\sigma)$
- $StSp(\mathcal{Z}) = \bigcup\limits_{\sigma} K_\sigma$ where

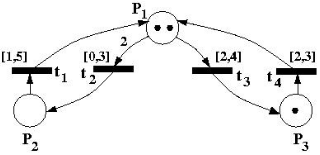
$$K_\sigma := \{z_{\sigma(\beta(x))} \mid \beta(x) \text{ is a solution of } B_\sigma\}.$$

Notations:

- **transition sequence:** $\sigma = t_1 \cdots t_n$,
- **run:** $\sigma(\tau) = \tau_0\, t_1\, \tau_1\, \cdots\, \tau_{n-1}\, t_n\, \tau_n$ and $\tau_i \in \mathbb{R}_0^+$
- **parametric run:** $(\sigma(x) = x_0\, t_1\, x_1\, \cdots\, x_{n-1}\, t_n\, x_n, B_\sigma)$ and x_i are variables which satisfied a set of conditions B_σ
- **parametric state:** $(z_\sigma, B\sigma)$
- **parametric state space:** $parStSp(\mathcal{Z}) = \bigcup_{\sigma}(z_\sigma, B\sigma)$
- $StSp(\mathcal{Z}) = \bigcup_{\sigma} K_\sigma$ where

$$K_\sigma := \{z_{\sigma(\beta(x))} \mid \beta(x) \text{ is a solution of } B_\sigma\}.$$

Time Petri net: Example



$K_\varepsilon = \{(\underbrace{(0,1,1)}_{m_\varepsilon}, \underbrace{(x_0, \sharp, \sharp, x_0)}_{h_\varepsilon}) \mid \underbrace{\{0 \le x_0 \le 3\}}_{B_\varepsilon}\}.$

$K_{t_4} = \{(\underbrace{(1,1,0)}_{m_{t_4}}, \underbrace{(x_0 + x_1, \sharp, x_1, \sharp)}_{h_{t_4}}) \mid \underbrace{\{2 \le x_0 \le 3, x_0 + x_1 \le 5, 0 \le x_1 \le 4\}}_{B_{t_4}}\}.$

The set of conditions B_{t_4} is the union of the three sets

B_ε, $\{eft(t_4) \le h_\varepsilon(t_4)\} = \{2 \le x_0\}$ and $\{0 \le h_\sigma(t) \le lft(t) \mid t^- \le m_\sigma\} = \left\{ \begin{array}{l} x_0 + x_1 \le 5, \\ 0 \le x_1 \le 4 \end{array} \right\}.$

Proposition

Let $(\mathcal{N}, m_0)$ be a live Petri net. Then there exists a feasible cyclic run, including all transitions in $\mathcal{N}$.

Liveness, bondedness, time

Proposition

Let $(\mathcal{N}, m_0)$ be a live Petri net. Then there exists a feasible cyclic run, including all transitions in $\mathcal{N}$.

Proposition

Let $(\mathcal{N}, m_0)$ be a Petri net, and let $\mathcal{Z} = (\mathcal{N}, m_0, I)$ be a Time Petri net, obtained from $\mathcal{N}$ by adding intervals to each transition. Then the reachability tree for $\mathcal{Z}$ is a subgraph of the reachability tree for $(\mathcal{N}, m_0)$.

Stage 1. Find all minimal feasible cycles, which include all transitions

Use the technique described in

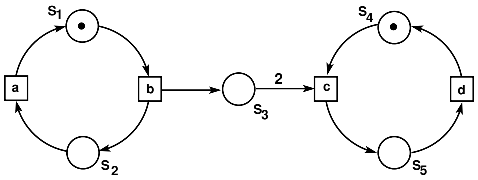
Jörg Desel

On Cyclic Behaviour of Unbounded Petri Nets, [ACSD-2013]

Following this technique for each minimal feasible cyclic run σ we also find a finite initial run τ, such that $\tau\sigma^{\star}$ is an initial run in $(\mathcal{N}, m_0)$.

If (N, m_0) does not have such cycles, then the problem does not have a solution.

Example



$(\mathcal{N}^*, m_0^*)$ has five minimal cyclic runs with all transitions. Three of them have empty prefixes, and two have prefixes $\tau_1 = b$ and $\tau_2 = ba$, respectively:

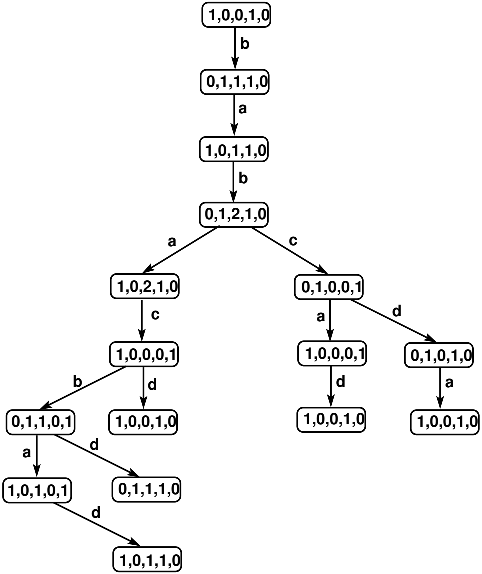
$$\begin{array}{l} babcda \\ babcad \\ babacd \end{array} \qquad \text{and} \qquad \begin{array}{ll} \overbrace{b}^{\tau_1 =} & abacbd \\ \underbrace{ba}_{\tau_2 =} & bacbad \end{array} .$$

Stage 2. Construct a spine tree

A *spine tree* is a subgraph of a reachability tree, containing exactly all minimal feasible cyclic runs together with their prefixes.

A spine tree contains the behavior that should be saved to keep a Petri net live.

The spine tree for the net $(\mathcal{N}^*, m_0^*)$:



Stage 3. Construct a spine-based coverability tree

A *spine-based coverability tree* is a special kind of a coverability tree that includes a spine tree as a backbone. Leafs in a spine-based coverability tree will be additionally colored with green or red. This coloring will be used then for computing time intervals.

Algorithm for computing a spine-based coverability tree:

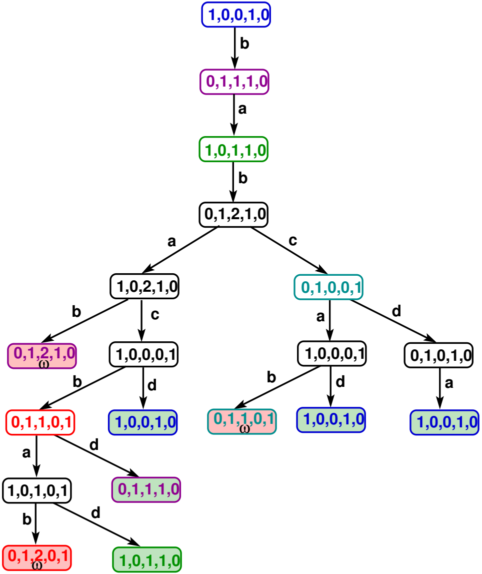
Step 1. Start with the given spine tree. Color all leafs in green.

Step 2. Repeat until all nodes are colored:

For each uncolored node v labeled with a marking m:

1. check whether there is a marking m', directly reachable from m and not included in the current tree. For each such marking m', where $m \xrightarrow{t} m'$:
    1. Add a node v' labeled with m' as well as the corresponding arc from v to v' labeled with t.
    2. If the marking m' strictly covers a marking in some node on the path from the root to v', then v' becomes a leaf and gets the red color.
    3. Otherwise, if the marking m' coincides with a marking labeling some node on the path from the root to v', then v' becomes a leaf and gets the green color.
    4. Otherwise, leave v' uncolored.
2. Color the node v in yellow.

The spine-based coverability tree for $(\mathcal{N}_1, m_0)$:



Stage 4. Compute parametric state space

- Let $\mathcal{T}$ be a spine-based coverability tree. By the construction of $\mathcal{T}$, all its leafs are colored either in green, or red.
- Add a time interval $[a_t, b_t]$ to each transition $t \in T$. All a_t, b_t are unknown and have to be calculated in stage 5. Thus, we are going to construct an interval function $I : T \to \mathbb{Q}_0^+ \times \mathbb{Q}_0^+$ and a TPN $(\mathcal{N}, m_0, I)$, respectively.
- For this we consider every path from the root to a green leaf as a parametric run. Additionally, we forbid a branching to a red leaf using strict inequality:

- (1) Let v_g be a green leaf and let σ be the path from the root to this leaf. Consider the parametric run $(\sigma(x), B_\sigma)$.

- (1) Let v_g be a green leaf and let σ be the path from the root to this leaf. Consider the parametric run $(\sigma(x), B_\sigma)$.
- (2) Let v_r be a red leaf. We consider σ, v_0, v^*, v_g, v_r (labeled with m_0, m^*, m_g, m_r, respectively) as follows:

$$m_0 \stackrel{\sigma^*}{\to} m^* \stackrel{\sigma_g}{\to} v_g, \quad m_0 \stackrel{\sigma^*}{\to} m^* \stackrel{t_r}{\to}\stackrel{\sigma_r}{\to} v_r, \quad \sigma = \sigma^* t_r \sigma_r,$$

where

  - σ is the path from the root to v_r,
  - v^* be the youngest ancestor of v_r such that at least one run goes from v^* to a green leaf v_g,
  - add to B_{σ^*} the constrain (strong inequality) $h_{\sigma^*}(t_r) < a_{t_r}$.

Solve the system of linear inequalities B,

$$B := \bigcup\{B_\sigma \mid \sigma \text{ is an initial run to a green node}\} \cup$$
$$\{0 \leq a_t \leq b_t \mid t \in T\} \cup$$
$$\bigcup\{h_{\sigma^*}(t_r) < a_{t_r} \mid w.r.t.\ \ Stage\ \ 4\}.$$

in $\mathbb{Q}_0^+$.

Example

$$B := \left\{ \begin{array}{lllll}
a_b \leq x_0 \leq b_b & a_d \leq x_4 \leq b_d & 0 \leq x_8 & 0 \leq x_{13} < a_b & a_a \leq x_3 + x_4 + x_5 \leq b_a \\
a_a \leq x_1 \leq b_a & 0 \leq x_5 & x_9 < a_b & 0 \leq x_{14} & a_c \leq x_3 + x_9 \leq b_c \\
a_b \leq x_2 \leq b_b & 0 \leq x_6 \leq b_b & a_d \leq x_{10} & 0 \leq x_{15} & x_9 + x_{10} + x_{11} \leq b_b \\
a_a \leq x_3 \leq b_a & 0 \leq x_7 \leq b_d & 0 \leq x_{11} & a_b \leq x_9 + x_{10} & a_d \leq x_{10} + x_{12} + x_{13} \leq b_d \\
a_c \leq x_3 & x_7 < a_b & a_a \leq x_{12} \leq b_a & x_{13} + x_{14} \leq b_b & x7 + x_8 \leq b_b \\
0 \leq a_a & 0 \leq a_b & 0 \leq a_c & a_d \leq x_{10} + x_{12} & x_{12} + x_{15} \leq b_a \\
0 \leq a_d & & & &
\end{array} \right\}$$

Subsequently, with respect to the properties of a interval function, it has to be true:

$$a_b \geq 1, \;\; b_b \geq 1.$$

A solution (with minimal values) for the interval function I^* is, e.g., $a_a = b_a = 0, a_b = b_b = 1, a_c = b_c = 0, a_d = 0$ and $b_d = 1$.

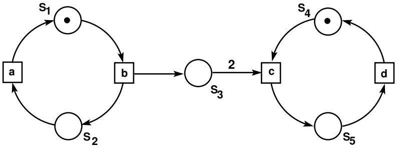
Validation

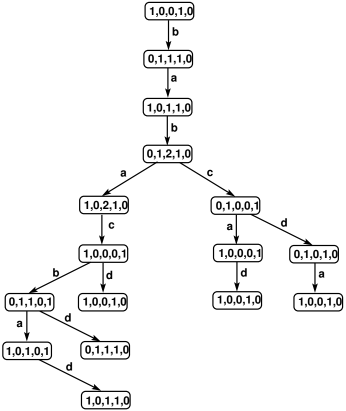
Theorem

Let $(\mathcal{N}, m_0)$ be a live and unbounded Petri net, for which there exists a feasible cyclic run, which includes all transitions in $\mathcal{N}$. Let then B be the set of inequalities constructed for $(\mathcal{N}, m_0)$ according to the algorithm described above. If B has a solution in $\mathbb{Q}_0^+$, then this defined an interval function I such that the Time Petri net $\mathcal{Z} = (\mathcal{N}, m_0, I)$ is live and bounded.

Some Comparisons

Let consider the PN $(\mathcal{N}^*, m_0^*)$ again:

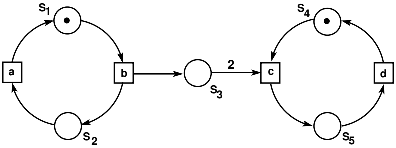


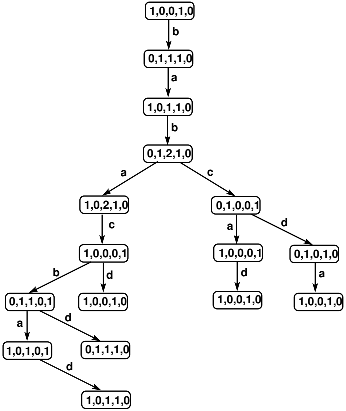


1. The PN $(\mathcal{N}^*, m_0^*)$ has 5 minimal cyclic runs:
   babcda *babcad* *babacd* *babacbd* *babacbad*

Some Comparisons

Let consider the PN $(\mathcal{N}^*, m_0^*)$ again:

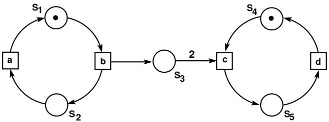


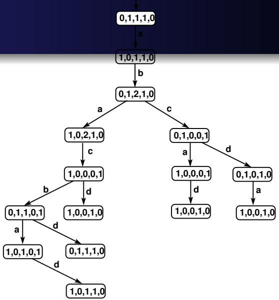


1. The PN $(\mathcal{N}^*, m_0^*)$ has 5 minimal cyclic runs:
   babcda *babcad* *babacd* *babacbd* *babacbad*

Some Comparisons

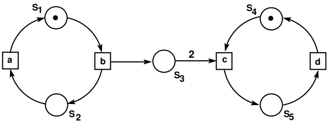
Let consider the PN $(\mathcal{N}^*, m_0^*)$ again:

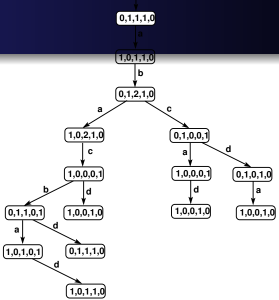




1. The PN $(\mathcal{N}^*, m_0^*)$ has 5 minimal cyclic runs:
   babcda *babcad* *babacd* *babacbd* *babacbad*

Let consider the PN $(\mathcal{N}^*, m_0^*)$ again:

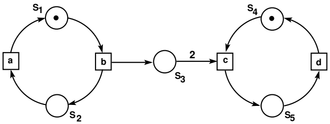


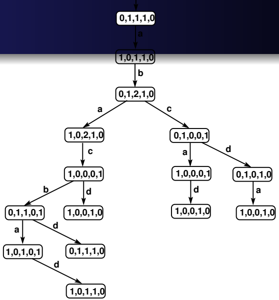


1. The PN $(\mathcal{N}^*, m_0^*)$ has 5 minimal cyclic runs:
   babcda *babcad* *babacd* *babacbd* *babacbad*
2. The Timed PN $\mathcal{D}^* = (\mathcal{N}^*, m_0^*, D^*)$ calculated using T-invariants has 1 minimal cycle:

   babacbad (or more exactly: $bab \begin{matrix} a \\ c \end{matrix} b \begin{matrix} a \\ d \end{matrix}$)

Let consider the PN $(\mathcal{N}^*, m_0^*)$ again:

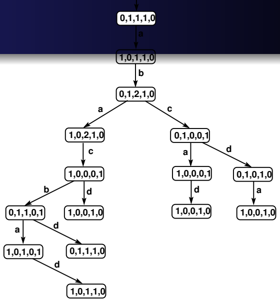




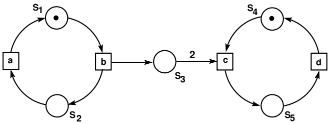
1. The PN $(\mathcal{N}^*, m_0^*)$ has 5 minimal cyclic runs: *babcda* *babcad* *babacd* *babacbd* *babacbad*
2. The Timed PN $\mathcal{D}^* = (\mathcal{N}^*, m_0^*, D^*)$ calculated using T-invariants has 1 minimal cycle:

   babacbad (or more exactly: $bab \begin{matrix} a \\ c \end{matrix} b \begin{matrix} a \\ d \end{matrix}$)
3. The PN with priorities $\mathcal{P}_1 = (\mathcal{N}_1, \ll, m_0)$ has 3 minimal cyclic runs: *babcda* *babcad* *babacd*

Some Comparisons



Let consider the PN $(\mathcal{N}^*, m_0^*)$ again:



1. The PN $(\mathcal{N}^*, m_0^*)$ has 5 minimal cyclic runs:
   babcda *babcad* *babacd* *babacbd* *babacbad*
2. The Timed PN $\mathcal{D}^* = (\mathcal{N}^*, m_0^*, D^*)$ calculated using T-invariants has 1 minimal cycle:

   babacbad (or more exactly: $bab \begin{matrix} a \\ c \end{matrix}\ b \begin{matrix} a \\ d \end{matrix}$)
3. The PN with priorities $\mathcal{P}_1 = (\mathcal{N}_1, \ll, m_0)$ has 3 minimal cyclic runs: *babcda* *babcad* *babacd*
4. The Time PN $\mathcal{Z}^* = (\mathcal{N}^*, m_0^*, I^*)$ has all 5 minimal cyclic runs.

Conclusions

- The possibility for obtaining a live and bounded Petri net from a live and unbounded one by adding time intervals was investigated.

Conclusions

- The possibility for obtaining a live and bounded Petri net from a live and unbounded one by adding time intervals was investigated.
- Necessary conditions for existence of such intervals were obtained. This conditions are not sufficient. But ...

Conclusions

- The possibility for obtaining a live and bounded Petri net from a live and unbounded one by adding time intervals was investigated.
- Necessary conditions for existence of such intervals were obtained. This conditions are not sufficient. But ...
- An algorithm for computing time intervals for transforming a live and unbounded Petri net into live and bounded net by ascribing these intervals to transitions was developed.
  This algorithm guarantees that the computed interval function solve the problem, i.e. the Time Petri net is live and bounded.

Thank you!