FORMATS 2013

31st August 2013
Buenos Aires, Argentina

Precise Robustness Analysis of Time Petri Nets with Inhibitor Arcs

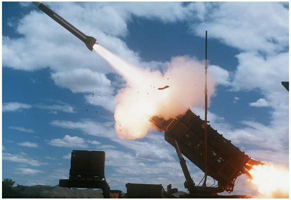
Étienne André, Giuseppe Pellegrino, Laure Petrucci

Laboratoire d'Informatique de Paris Nord
Université Paris 13, Sorbonne Paris Cité, France

Context: Verifying Complex Timed Systems

- Need for early bug detection
  - Bugs discovered when final testing: expensive

⇝ Need for a thorough modeling and verification phase



Motivation: Robustness Analysis

- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"
- Challenge: Robustness [Markey, 2011]
  - What happens if 50 is implemented with 49.99?
  - In which neighbourhood of 50 does the system behave well?

Motivation: Robustness Analysis

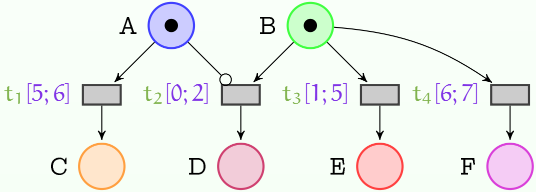
- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"
- Challenge: Robustness [Markey, 2011]
  - What happens if 50 is implemented with 49.99?
  - In which neighbourhood of 50 does the system behave well?
- Parametric analysis
  - Consider that timing constants are parameters
  - Find good values for the parameters, such that the system still behaves well

Outline

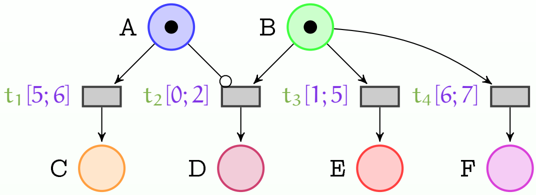
Outline

Time Petri Nets With Inhibitor Arcs (ITPNs)

- Powerful formalism for verifying real-time systems [Merlin, 1974]
- Transition t_1 can fire from 5 to 6 units of time after being enabled
- An enabled transition must fire before (or at) its upper bound
- An inhibitor arc enables its transition (t_2) when its source place (A) is empty



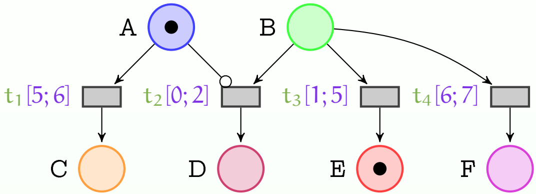
Time Petri Nets With Inhibitor Arcs: Example



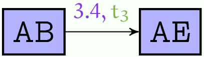
Some possible runs

AB

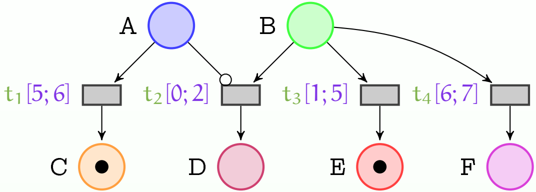
Time Petri Nets With Inhibitor Arcs: Example



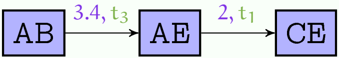
Some possible runs



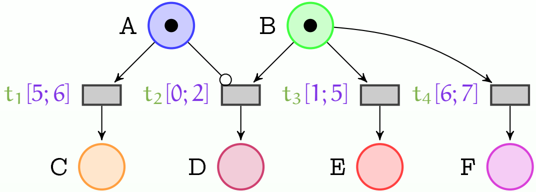
Time Petri Nets With Inhibitor Arcs: Example



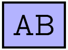
Some possible runs



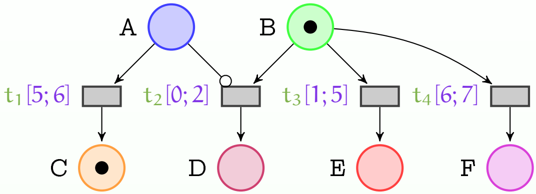
Time Petri Nets With Inhibitor Arcs: Example



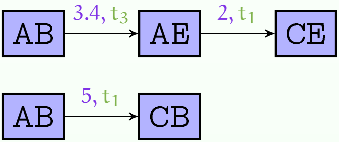
Some possible runs



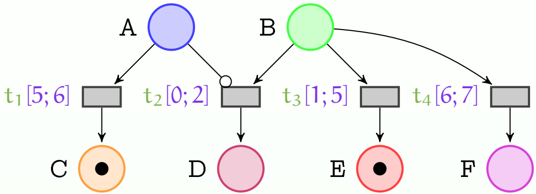
Time Petri Nets With Inhibitor Arcs: Example



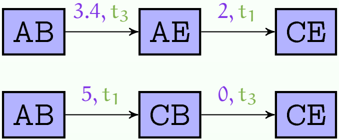
Some possible runs



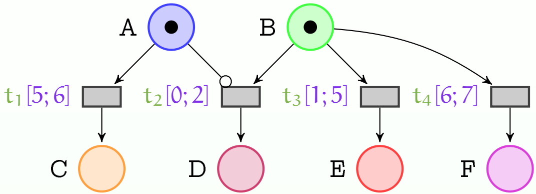
Time Petri Nets With Inhibitor Arcs: Example



Some possible runs



Time Petri Nets With Inhibitor Arcs: Example



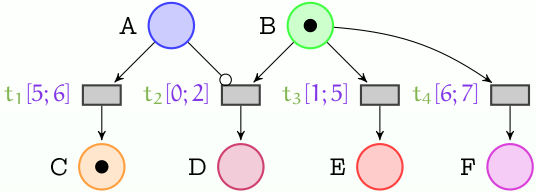
Some possible runs

AB $\xrightarrow{3.4,\ t_3}$ AE $\xrightarrow{2,\ t_1}$ CE

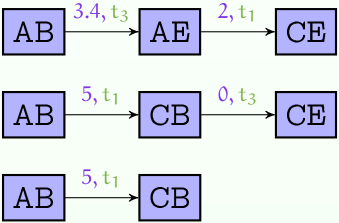
AB $\xrightarrow{5,\ t_1}$ CB $\xrightarrow{0,\ t_3}$ CE

AB

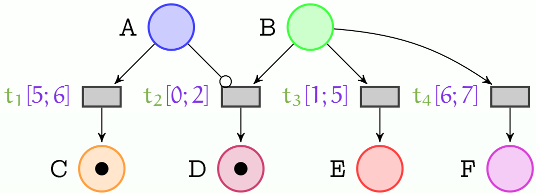
Time Petri Nets With Inhibitor Arcs: Example



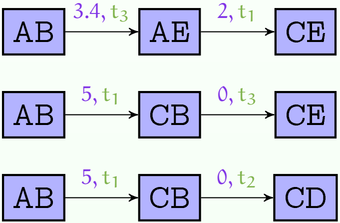
Some possible runs



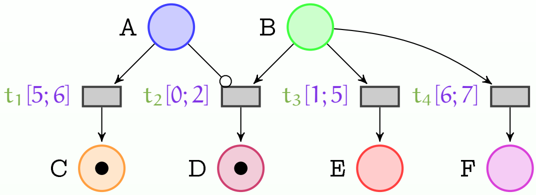
Time Petri Nets With Inhibitor Arcs: Example



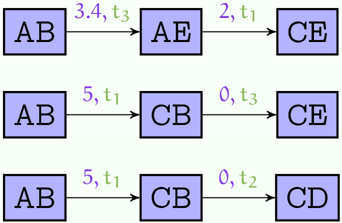
Some possible runs



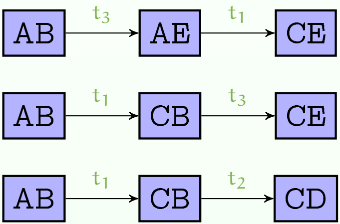
Time Petri Nets With Inhibitor Arcs: Example



Some possible runs

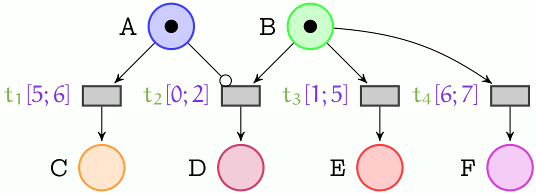


Trace set



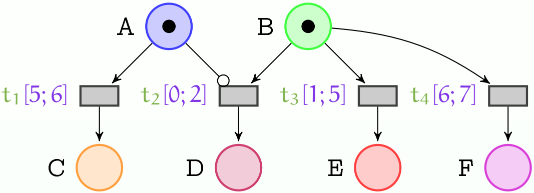
Trace: time-abstract behaviour

Robustness (1/2)



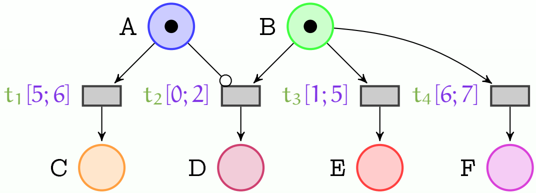
- What happens if $t_2[0;2]$ is implemented with $t_2[0.01;2]$?
  - Trace AB $\xrightarrow{t_1}$ CB $\xrightarrow{t_2}$ CD cannot happen anymore

Robustness (1/2)



- What happens if $t_2[0;2]$ is implemented with $t_2[0.01;2]$?
  - Trace $AB \xrightarrow{t_1} CB \xrightarrow{t_2} CD$ cannot happen anymore
- What happens if $t_3[1;5]$ is implemented with $t_3[1;4.99]$?
  - Trace $AB \xrightarrow{t_1} CB \xrightarrow{t_3} CE$ cannot happen anymore

Robustness (1/2)



- What happens if $t_2[0;2]$ is implemented with $t_2[0.01;2]$?
  - Trace AB $\xrightarrow{t_1}$ CB $\xrightarrow{t_2}$ CD cannot happen anymore
- What happens if $t_3[1;5]$ is implemented with $t_3[1;4.99]$?
  - Trace AB $\xrightarrow{t_1}$ CB $\xrightarrow{t_3}$ CE cannot happen anymore

$\rightsquigarrow$ This system is not robust, in the sense that infinitesimal variations of the bounds lead to a different discrete behaviour (trace set).

Robustness (2/2)

Definition (LT-robustness)

An ITPN N is LT-robust if there exists $\gamma > 0$ such that N_γ and N have the same trace sets.
(where N_γ is any ITPN similar to N where each timing bound c can be replaced with any value within $[c - \gamma, c + \gamma]$)

Challenges:

- Is an ITPN robust?
- If not, why is it non-robust?
- Is it possible to render robust a non-robust ITPN? If so, how?

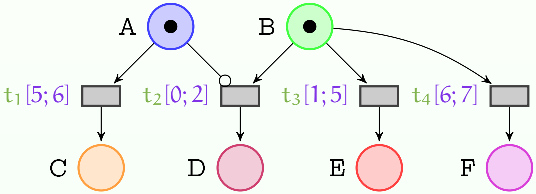
Outline

Parametric Time Petri Nets

Idea: Reason parametrically (using unknown constants)

Parametric Time Petri Nets with Inhibitor Arcs (PITPNs)

- Constants in firing intervals replaced with parameters [Traonouez et al., 2009]

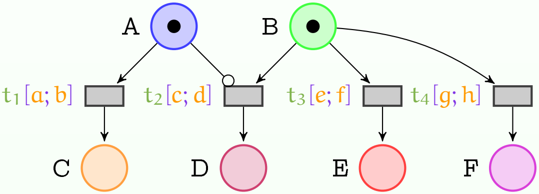


Parametric Time Petri Nets

Idea: Reason parametrically (using unknown constants)

Parametric Time Petri Nets with Inhibitor Arcs (PITPNs)

- Constants in firing intervals replaced with parameters [Traonouez et al., 2009]

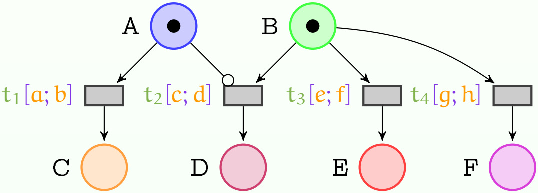


Parametric Time Petri Nets

Idea: Reason parametrically (using unknown constants)

Parametric Time Petri Nets with Inhibitor Arcs (PITPNs)

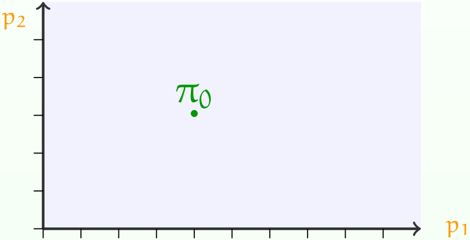
- Constants in firing intervals replaced with parameters [Traonouez et al., 2009]



- Notation: given a PITPN $\mathcal{N}$ and a valuation π of the parameters, we denote by $[\![\mathcal{N}]\!]_\pi$ the ITPN obtained from $\mathcal{N}$ by replacing all parameters with their valuation in π

The Inverse Method (IM)

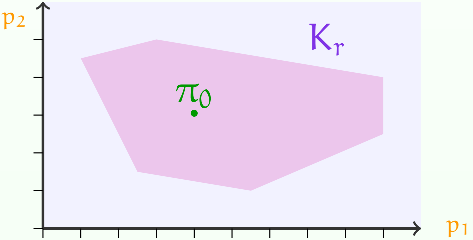
- Input
  - A PITPN $\mathcal{N}$
  - A reference valuation π_0 of all the parameters of $\mathcal{N}$



The Inverse Method (*IM*)

- Input
  - A PITPN $\mathcal{N}$
  - A reference valuation π_0 of all the parameters of $\mathcal{N}$

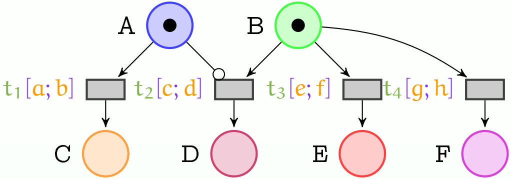
- Output: K_r
  - Convex constraint on the parameters such that
    - $\pi_0 \models K_r$
    - For all points $\pi \models K_r$, $[\![\mathcal{N}]\!]_\pi$ and $[\![\mathcal{N}]\!]_{\pi_0}$ have the same trace sets



The Inverse Method: General Idea

- Initially defined for timed automata [A., Chatain, Encrenaz, Fribourg, 2009]
- Extended to PITPNs [A., Pellegrino, Petrucci, 2013]
- The idea
  - Exploration of the parametric state space
  - Instead of negating bad states (as in "CEGAR" approaches), remove π_0-incompatible states
  - Return the intersection of all constraints on the parameters

Application to an Example

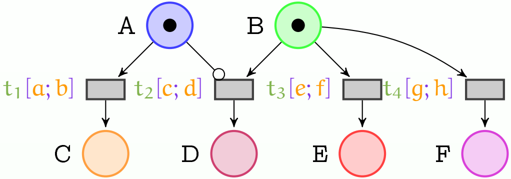


π_0

$a = 5$	$b = 6$
$c = 0$	$d = 2$
$e = 1$	$f = 5$
$g = 6$	$h = 7$

Forward analysis

Application to an Example



π_0

$a = 5 \quad b = 6$

$c = 0 \quad d = 2$

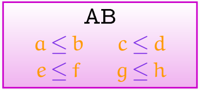
$e = 1 \quad f = 5$

$g = 6 \quad h = 7$

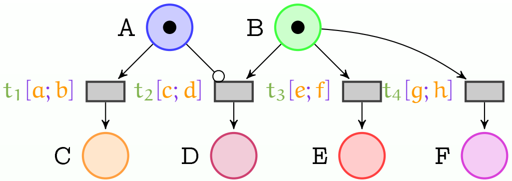
Forward analysis

K :

true



Application to an Example



π_0

$a = 5 \quad b = 6$

$c = 0 \quad d = 2$

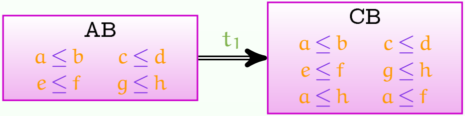
$e = 1 \quad f = 5$

$g = 6 \quad h = 7$

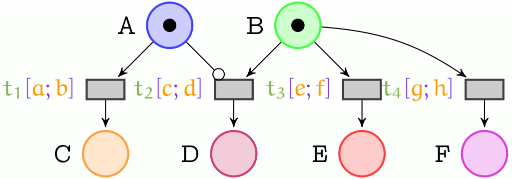
Forward analysis

K :

`true`



Application to an Example



π_0

$a = 5 \quad b = 6$

$c = 0 \quad d = 2$

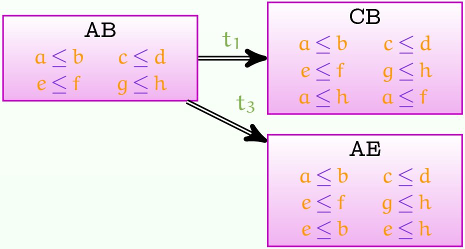
$e = 1 \quad f = 5$

$g = 6 \quad h = 7$

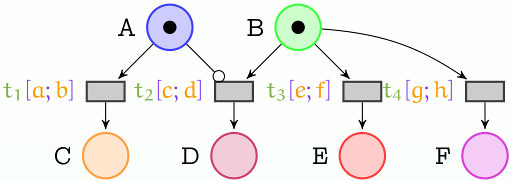
Forward analysis

K :

`true`



Application to an Example



π_0

$a = 5 \quad b = 6$

$c = 0 \quad d = 2$

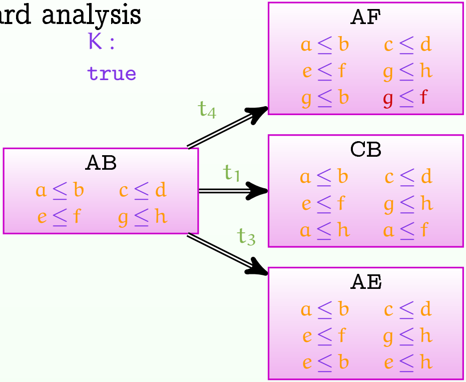
$e = 1 \quad f = 5$

$g = 6 \quad h = 7$

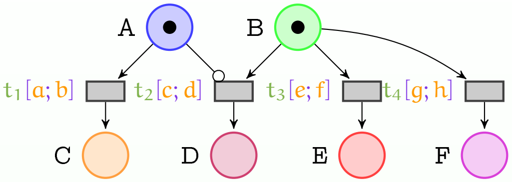
Forward analysis

K :

true



Application to an Example



π_0

$a = 5 \quad b = 6$

$c = 0 \quad d = 2$

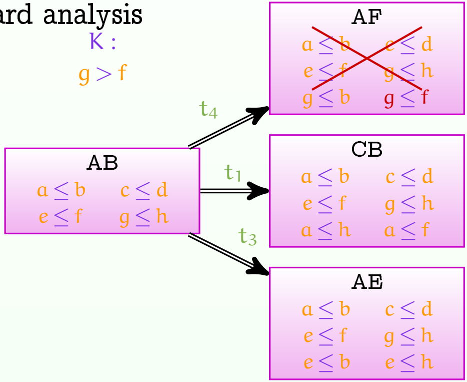
$e = 1 \quad f = 5$

$g = 6 \quad h = 7$

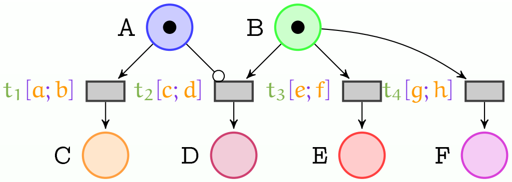
Forward analysis

K :

$g > f$



Application to an Example



π_0

$a = 5$ $b = 6$

$c = 0$ $d = 2$

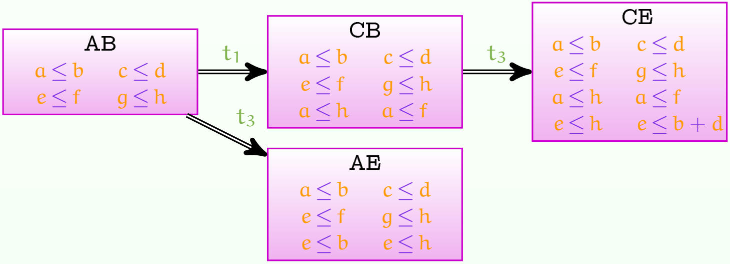
$e = 1$ $f = 5$

$g = 6$ $h = 7$

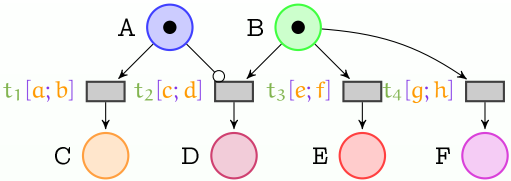
Forward analysis

K :

$g > f$



Application to an Example



π_0

$a = 5 \quad b = 6$

$c = 0 \quad d = 2$

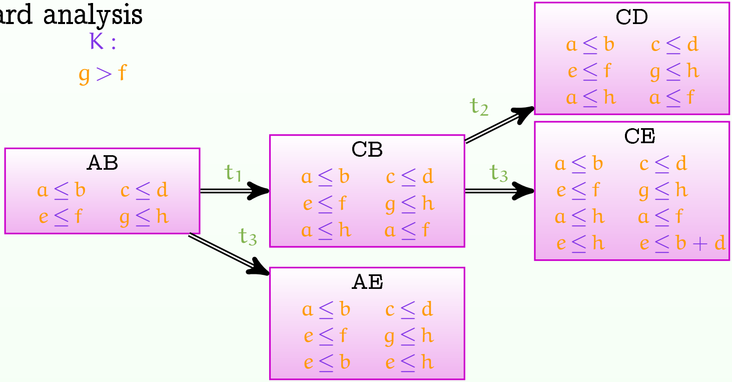
$e = 1 \quad f = 5$

$g = 6 \quad h = 7$

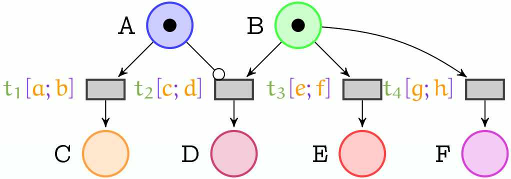
Forward analysis

K :

$g > f$



Application to an Example



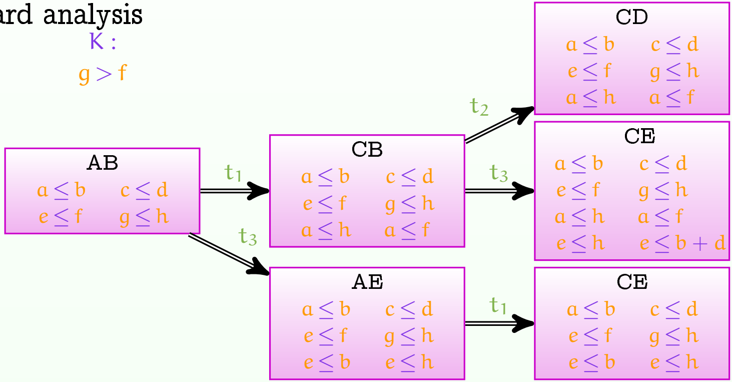
π_0

$a = 5$	$b = 6$
$c = 0$	$d = 2$
$e = 1$	$f = 5$
$g = 6$	$h = 7$

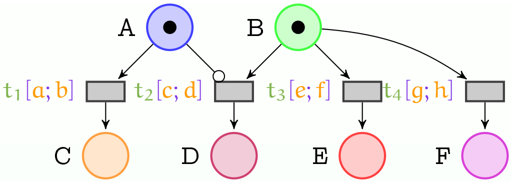
Forward analysis

K :

$g > f$



Application to an Example



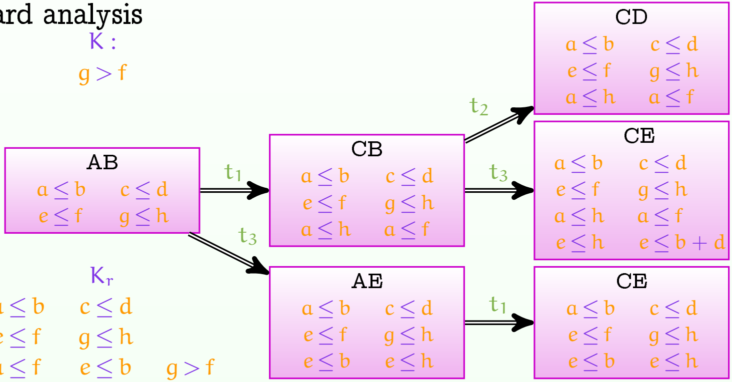
π_0

$a = 5$	$b = 6$
$c = 0$	$d = 2$
$e = 1$	$f = 5$
$g = 6$	$h = 7$

Forward analysis

K :

$g > f$



K_r

$a \leq b$ $c \leq d$
$e \leq f$ $g \leq h$
$a \leq f$ $e \leq b$ $g > f$

Properties

- Correctness
  - $\pi_0 \models K_r$ and
  - $\forall \pi \models K_r$, $[\![\mathcal{N}]\!]_\pi$ and $[\![\mathcal{N}]\!]_{\pi_0}$ have the same trace set.
- *IM* is non-confluent
  - Several executions with the same input may lead to different outputs
- *IM* is non-complete
  - K_r may not be the maximum set of parameter valuations with the same trace set as $[\![\mathcal{N}]\!]_{\pi_0}$
- Termination of *IM* is not guaranteed in general
  - Parameter synthesis for PITPNs undecidable [Traonouez et al., 2009]

Outline

Robustness Using the Inverse Method

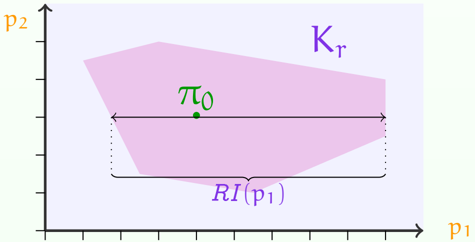
Let N be an ITPN.

General idea

1. Construct the parametric version $\mathcal{N}$ of N, and π_0 the reference valuation such that $[\![\mathcal{N}]\!]_{\pi_0} = \mathsf{N}$
2. Call $IM(\mathcal{N}, \pi_0)$ and assume K_r is the resulting constraint
3. Measure the system robustness
4. If the system is non-robust, render it robust (if possible)

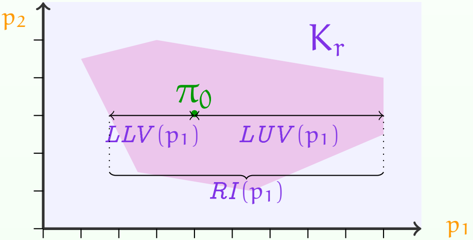
Metrics for Measuring Local Robustness

- Ranging interval of a parameter $RI(p)$
  - Minimum and maximum admissible values within K_r



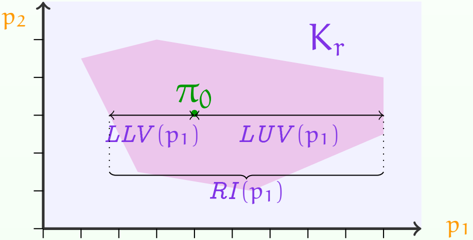
Metrics for Measuring Local Robustness

- Ranging interval of a parameter $RI(p)$
  - Minimum and maximum admissible values within K_r
- Local lower/upper variability of a parameter
  - Distance between $\pi_0(p)$ and and the lower/upper bound of $RI(p)$
  - Given $RI(p) = (a, b)$, then $LLV(p) = \pi_0(p) - a$ and $LUV(p) = b - \pi_0(p)$



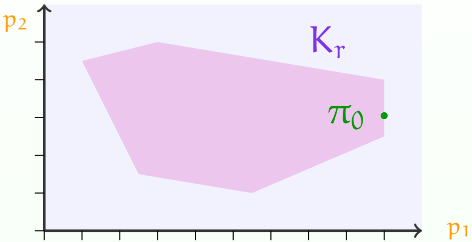
Metrics for Measuring Local Robustness

- Ranging interval of a parameter $RI(p)$
  - Minimum and maximum admissible values within K_r
- Local lower/upper variability of a parameter
  - Distance between $\pi_0(p)$ and and the lower/upper bound of $RI(p)$
  - Given $RI(p) = (a, b)$, then $LLV(p) = \pi_0(p) - a$ and $LUV(p) = b - \pi_0(p)$
- Local robustness: distance between $\pi_0(p)$ and the closest border within K_r
  - $LR(p) = \min(LLV(p), LUV(p))$



Critical Timing Bounds

- Critical timing bounds are those with a null local robustness



Remark

If any of the timing bounds is critical, classical ("Δ-based") approaches will just classify the system as non-robust.

Relaxing Timing Bounds

Definition (Potential robustness)

An ITPN N is potentially robust if, for all timing bounds p_i, $LLV(p_i) > 0$ or $LUV(p_i) > 0$.

Intuitively: A system is potentially robust if each parameter can vary within K_r.

Relaxing Timing Bounds

Definition (Potential robustness)

An ITPN N is potentially robust if, for all timing bounds p_i, $LLV(p_i) > 0$ or $LUV(p_i) > 0$.

Intuitively: A system is potentially robust if each parameter can vary within K_r.

Theorem (Rendering a system robust)

If N is potentially robust, then there exists π_R such that $[\![\mathcal{N}]\!]_{\pi_R}$ is LT-robust, and has the same trace set as N.

Construction: choose $\frac{LLV(p)+LUV(p)}{2}$ for each parameter p.

Relaxing Timing Bounds: Remarks

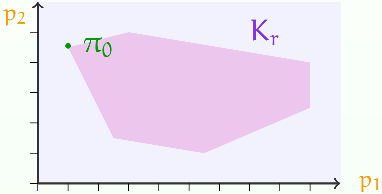
The potential robustness is a non-necessary condition to render a system robust

1. The potential robustness is based on the local robustness, that comes from K_r, that may be non-complete

Relaxing Timing Bounds: Remarks

The potential robustness is a non-necessary condition to render a system robust

1. The potential robustness is based on the local robustness, that comes from K_r, that may be non-complete
2. The potential robustness considers the variability of each timing bound in an independent manner



- In that case, the system is not potentially robust (since $LLV(p_2) = LUV(p_2) = 0$), but could still be made robust (by choosing a point in the middle of K_r)

Comparison with Related Work (1/2)

- Robustness studied for timed automata and time Petri nets (see [Markey, 2011] for a survey)
- "Δ-based" approaches
  - Robustness studied with respect to a single enlargement Δ for all bounds
  - or to a single shrinking Δ for all bounds
  - Extension to a (constant) vector
  - Extension to independent variations Δ for each parameter, but for shrinking only [Sankur, 2013]

Comparison with Related Work (2/2)

- Recent approaches
  - Parameterized robust reachability in timed automata is decidable [Bouyer et al., 2012]
  - Computing the greatest acceptable variation Δ is decidable for flat timed automata with progressive clocks [Jaubert and Reynier, 2011]
  - CEGAR-based approach using parametric techniques to evaluate the greatest acceptable variation Δ for parametric timed automata (not decidable in general) [Traonouez, 2012]

Comparison with Related Work (2/2)

- Recent approaches
  - Parameterized robust reachability in timed automata is decidable [Bouyer et al., 2012]
  - Computing the greatest acceptable variation Δ is decidable for flat timed automata with progressive clocks [Jaubert and Reynier, 2011]
  - CEGAR-based approach using parametric techniques to evaluate the greatest acceptable variation Δ for parametric timed automata (not decidable in general) [Traonouez, 2012]

- In contrast to most approaches, we consider a local robustness measure for each delay
  - ☺ For linear-time properties
  - ☺ More flexible: Bounds can be both enlarged and shrinked
  - ☺ More precise: Exhibits the critical timing bounds
  - ☹ May not terminate

Outline

Conclusion

- Local robustness analysis of timed systems
  - For linear-time properties
  - Using the inverse method
  - Quantifies the robustness of each timing bound

    $\rightsquigarrow$ Identifies critical bounds
- Sufficient condition for rendering a non-robust system robust
- Comparison with related approaches
  - ☺ More precise than most existing approaches
  - ☹ May not terminate

Conclusion

- Local robustness analysis of timed systems
  - For linear-time properties
  - Using the inverse method
  - Quantifies the robustness of each timing bound

  $\rightsquigarrow$ Identifies critical bounds

- Sufficient condition for rendering a non-robust system robust

- Comparison with related approaches
  - ☺ More precise than most existing approaches
  - ☹ May not terminate

- Linear-time properties, hence untimed
  - But timed properties can be considered using observers

Perspectives

- Implementation
  - Work in progress
  - Comparison with other tools such as Shrinktech [Sankur, 2013]
- Improve conditions for rendering non-robust systems robust
- Variation of the clocks speed ("ϵ")
  - Addition of two parameters for the admissible decrease and increase of the clock rate
  - Extension of the inverse method to non-linear (hybrid) systems

Bibliography

References I

André, É., Chatain, Th., Encrenaz, E., and Fribourg, L. (2009).
An inverse method for parametric timed automata.
IJFCS, 20(5):819–836.

André, É., Petrucci, L., and Pellegrino, G. (2013).
Precise robustness analysis of time Petri nets with inhibitor arcs.
In *FORMATS*, volume 8053 of *Lecture Notes in Computer Science*, pages 1–15. Springer.

Bouyer, P., Markey, N., and Sankur, O. (2012).
Robust reachability in timed automata: A game-based approach.
In *ICALP*, volume 7392 of *Lecture Notes in Computer Science*, pages 128–140. Springer.

Jaubert, R. and Reynier, P.-A. (2011).
Quantitative robustness analysis of flat timed automata.
In *FoSSaCS*, volume 6604 of *Lecture Notes in Computer Science*, pages 229–244. Springer-Verlag.

Markey, N. (2011).
Robustness in real-time systems.
In *SIES*, pages 28–34. IEEE Computer Society Press.

References II

Merlin, P. M. (1974).
A study of the recoverability of computing systems.
PhD thesis, University of California, Irvine, CA, USA.

Sankur, O. (2013).
Shrinktech: A tool for the robustness analysis of timed automata.
In *CAV'13*, volume 8044 of *Lecture Notes in Computer Science*, pages 1006–1012. Springer.

Traonouez, L.-M. (2012).
A parametric counterexample refinement approach for robust timed specifications.
In *FIT*, volume 87 of *EPTCS*, pages 17–33.

Traonouez, L.-M., Lime, D., and Roux, O. H. (2009).
Parametric model-checking of stopwatch Petri nets.
Journal of Universal Computer Science, 15(17):3273–3304.

Additional explanation

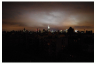
The Algorithm

Algorithm 1: *IM*$(\mathcal{N}, \pi)$

input : PITPN $\mathcal{N}$ of initial class c_0 and initial constraint K_0, valuation π
output: Constraint K_r

$i \leftarrow 0$; $K_c \leftarrow K_0$; $C \leftarrow \{c_0\}$
while true **do**
  while $\exists$ π-*incompatible classes in* C **do**
    Select a π-incompatible class (M, D) of C
    Select a π-incompatible J in $D{\downarrow_P}$
    $K_c \leftarrow K_c \wedge \neg J$; $C \leftarrow \bigcup_{j=0}^{i} Post^{j}_{\mathcal{N}(K_c)}(\{c_0\})$
  if $Post_{\mathcal{N}(K_c)}(C) \subseteq C$ **then**
    return $K_r \leftarrow \bigcap_{(M,D) \in C} D{\downarrow_P}$
  $i \leftarrow i + 1$; $C \leftarrow C \cup Post_{\mathcal{N}(K_c)}(C)$

Explanation for the 4 pictures in the beginning



Allusion to the Northeast blackout (USA, 2003)
Computer bug
Consequences: 11 fatalities, huge cost
(Picture actually from the Sandy Hurricane, 2012)

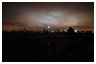
Allusion to any plane crash
(Picture actually from the happy-ending US Airways Flight 1549, 2009)

Allusion to the sinking of the Sleipner A offshore platform (Norway, 1991)
No fatalities
Computer bug: inaccurate finite element analysis modeling
(Picture actually from the Deepwater Horizon Offshore Drilling Platform)

Allusion to the MIM-104 Patriot Missile Failure (Iraq, 1991)
28 fatalities, hundreds of injured
Computer bug: software error (clock drift)
(Picture of an actual MIM-104 Patriot Missile, though not the one of 1991)

Licensing

Source of the graphics used



Title: Hurricane Sandy Blackout New York Skyline
Author: David Shankbone
Source: https://commons.wikimedia.org/wiki/File:Hurricane_Sandy_Blackout_New_York_Skyline.JPG
License: CC BY 3.0

Title: Miracle on the Hudson
Author: Janis Krums (cropped by Étienne André)
Source: https://secure.flickr.com/photos/davidwatts1978/3199405401/
License: CC BY 2.0

Title: Deepwater Horizon Offshore Drilling Platform on Fire
Author: ideum
Source: https://secure.flickr.com/photos/ideum/4711481781/
License: CC BY-SA 2.0

Title: DA-SC-88-01663
Author: imcomkorea
Source: https://secure.flickr.com/photos/imcomkorea/3017886760/
License: CC BY-NC-ND 2.0

License of this document

Author: Étienne André