ICFEM 2015

November 5th, 2015
Paris, France

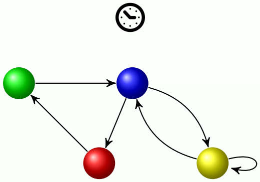
Enhanced Distributed Behavioral Cartography of Parametric Timed Automata

Étienne André, Camille Coti, Hoang Gia Nguyen

LIPN, UMR 7030, Université Paris 13, Sorbonne Paris Cité, CNRS, France

Context: Formal Verification of Timed Systems

- Use formal methods



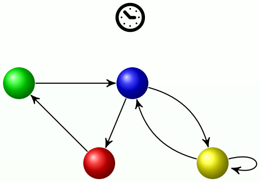
A model of the system

$AG\neg$ 

A property to be satisfied

Context: Formal Verification of Timed Systems

- Use formal methods



$\overset{?}{\models}$

$AG\neg$ 

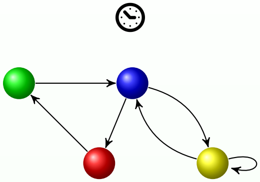
A model of the system

A property to be satisfied

- Question: does the model of the system satisfy the property?

Context: Formal Verification of Timed Systems

- Use formal methods



$\overset{?}{\models}$

AG¬ 

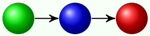
A model of the system

A property to be satisfied

- Question: does the model of the system satisfy the property?

Yes

No



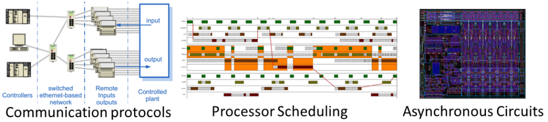
Counterexample

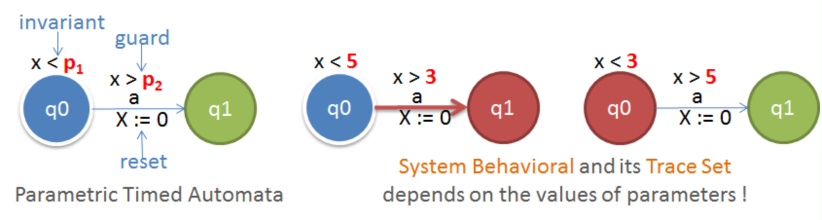
Outline

Outline

Parametric Timed Automata (PTA)

A formalism to model and verify concurrent real-time systems [Alur et al., 1993]



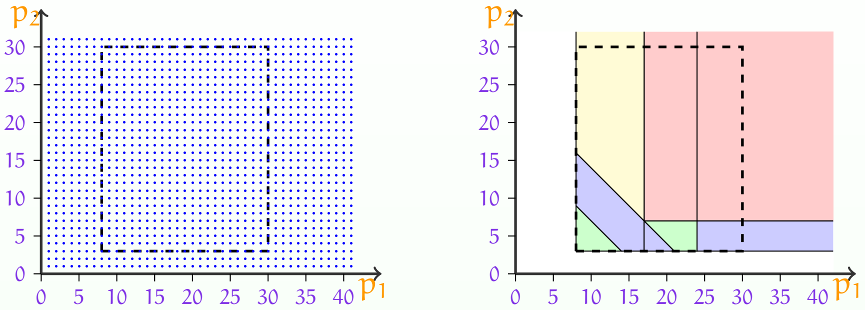


x: Clock

p: Parameters allow to represent unknown values (e.g., a transmission delay or a timeout)

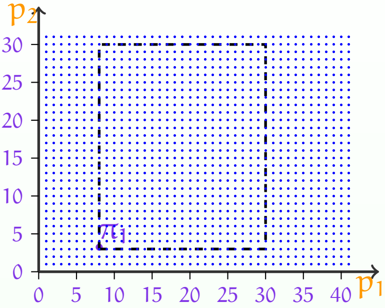
Trace set = set of all sequences of (untimed) actions

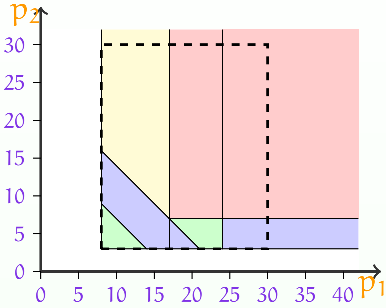
Behavioral Cartography (BC)



- BC: Partitions a parameter domain (bounded rectangle) into tiles, i.e., parametric zones of uniform behavior [André and Fribourg, 2010]
- Method: Enumerate integer points and generate a tile using an existing algorithm (the inverse method IM)
- All parameter valuations in a tile have the same possible behaviors (same "trace set"), and verify the same linear-time properties

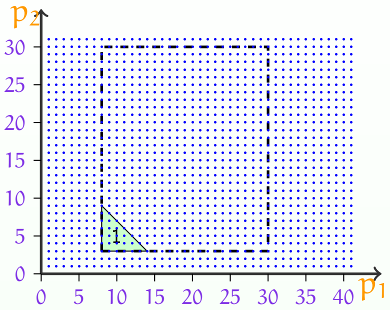
Behavioral Cartography (BC)

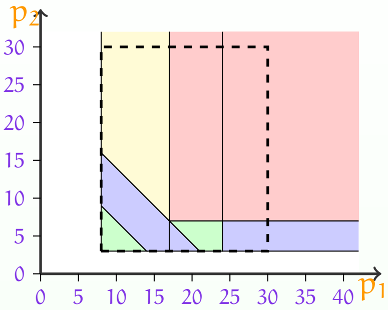




- BC: Partitions a parameter domain (bounded rectangle) into tiles, i.e., parametric zones of uniform behavior [André and Fribourg, 2010]
- Method: Enumerate integer points and generate a tile using an existing algorithm (the inverse method IM)
- All parameter valuations in a tile have the same possible behaviors (same "trace set"), and verify the same linear-time properties

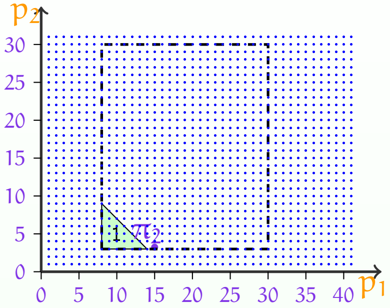
Behavioral Cartography (BC)

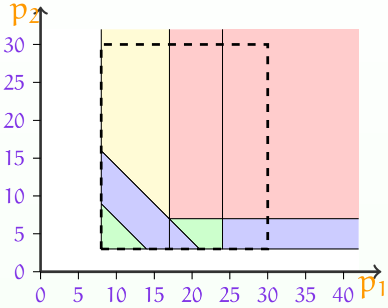




- BC: Partitions a parameter domain (bounded rectangle) into tiles, i.e., parametric zones of uniform behavior [André and Fribourg, 2010]
- Method: Enumerate integer points and generate a tile using an existing algorithm (the inverse method IM)
- All parameter valuations in a tile have the same possible behaviors (same "trace set"), and verify the same linear-time properties

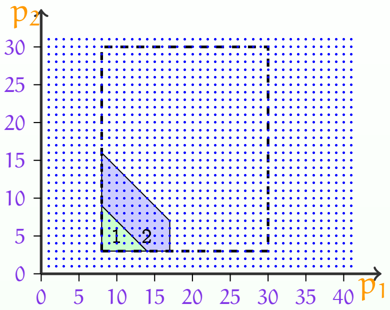
Behavioral Cartography (BC)

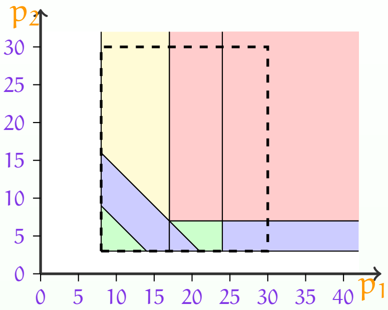




- BC: Partitions a parameter domain (bounded rectangle) into tiles, i.e., parametric zones of uniform behavior [André and Fribourg, 2010]
- Method: Enumerate integer points and generate a tile using an existing algorithm (the inverse method IM)
- All parameter valuations in a tile have the same possible behaviors (same "trace set"), and verify the same linear-time properties

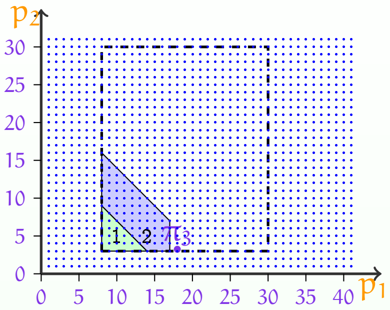
Behavioral Cartography (BC)

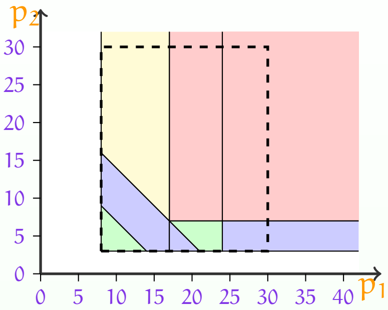




- BC: Partitions a parameter domain (bounded rectangle) into tiles, i.e., parametric zones of uniform behavior [André and Fribourg, 2010]
- Method: Enumerate integer points and generate a tile using an existing algorithm (the inverse method IM)
- All parameter valuations in a tile have the same possible behaviors (same "trace set"), and verify the same linear-time properties

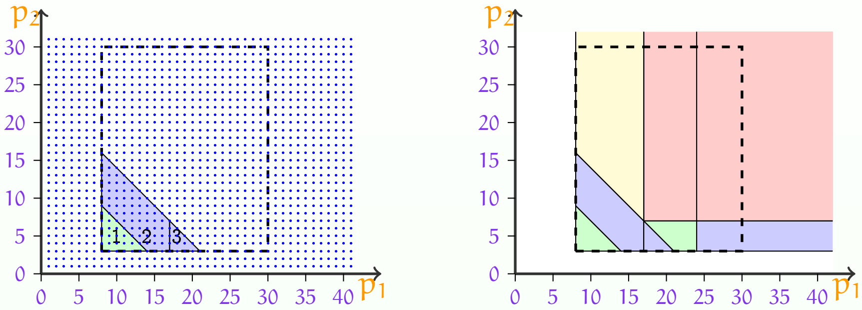
Behavioral Cartography (BC)





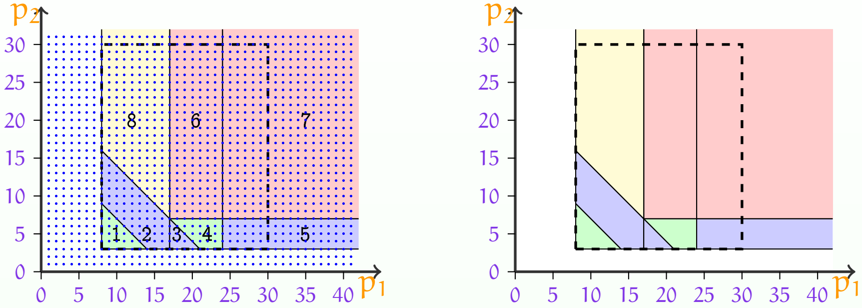
- BC: Partitions a parameter domain (bounded rectangle) into tiles, i.e., parametric zones of uniform behavior [André and Fribourg, 2010]
- Method: Enumerate integer points and generate a tile using an existing algorithm (the inverse method IM)
- All parameter valuations in a tile have the same possible behaviors (same "trace set"), and verify the same linear-time properties

Behavioral Cartography (BC)



- BC: Partitions a parameter domain (bounded rectangle) into tiles, i.e., parametric zones of uniform behavior [André and Fribourg, 2010]
- Method: Enumerate integer points and generate a tile using an existing algorithm (the inverse method IM)
- All parameter valuations in a tile have the same possible behaviors (same "trace set"), and verify the same linear-time properties

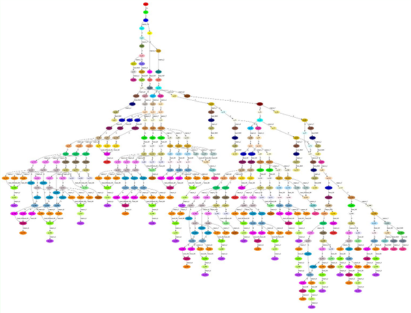
Behavioral Cartography (BC)



- BC: Partitions a parameter domain (bounded rectangle) into tiles, i.e., parametric zones of uniform behavior [André and Fribourg, 2010]
- Method: Enumerate integer points and generate a tile using an existing algorithm (the inverse method IM)
- All parameter valuations in a tile have the same possible behaviors (same "trace set"), and verify the same linear-time properties

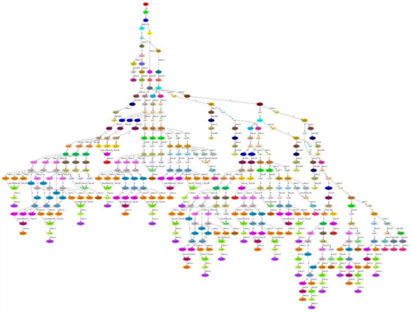
Behavioral Cartography (BC) Problem

- But state space explosion is always painful! Especially for real-time systems.



Behavioral Cartography (BC) Problem

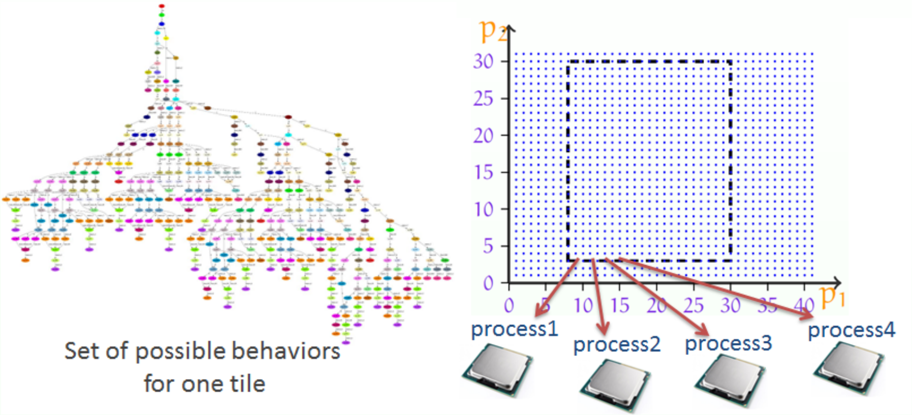
- But state space explosion is always painful! Especially for real-time systems.



- One solution:
  - Extend to distributed fashion

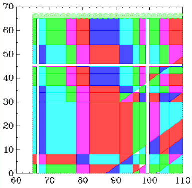
Outline

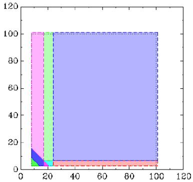
Distributing BC

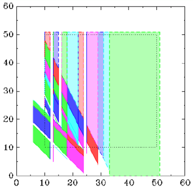


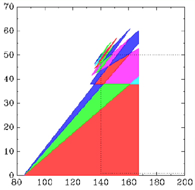
- Problem: BC is very slow! (up to several hours)
- Goal: distribute BC on a cluster to increase the computation speed

Distributing BC: Problems

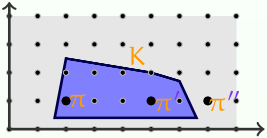








The general shape of the Cartography is unknown and the time to compute each tile varies a lot (more or less complex trace sets)
→ Load balancing problem

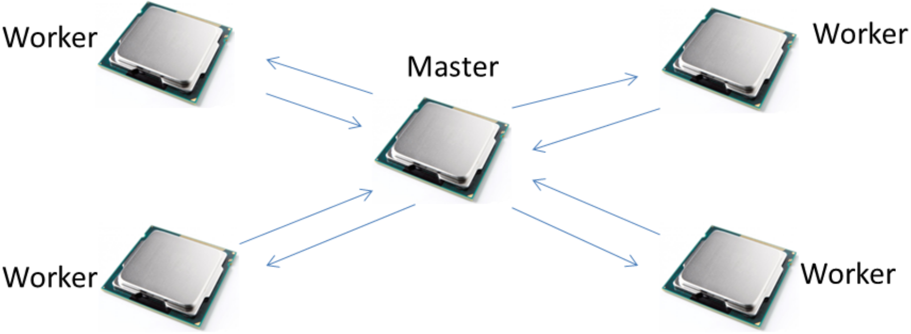


Since tiles are unpredictable, two close points will very probably yield the same tile

→ Choosing point problem

Outline

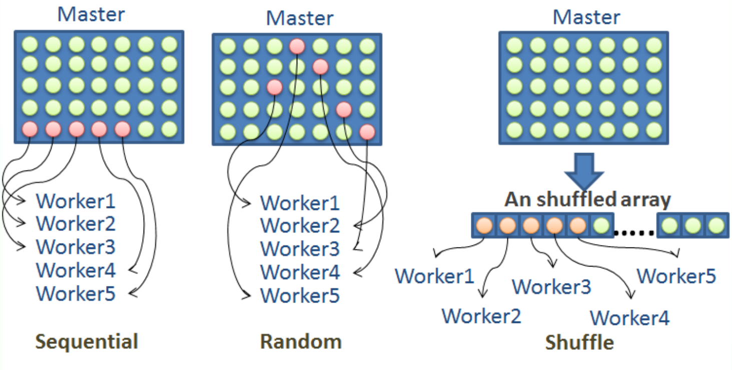
Master Workers Scheme



Traditional Master-Worker communication scheme: [André, Coti, Evangelista, 2014]

- Workers: ask the master for a point, and send the result ("tiles") to the master
- Master: is responsible for smart repartition of data between the workers

Point-based BC Algorithms



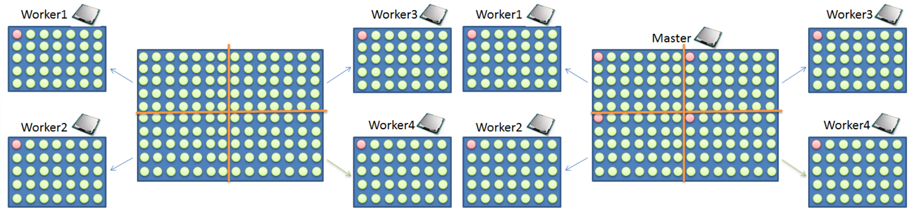
1. Sequential: Each point is sent to a worker sequentially
2. Random: Points selected randomly, then switches to Sequential
3. Shuffle: Similar to the Sequential, but the master must statically compute the array of all points, then shuffle all points, then store them back in array (new)

Outline

Domain-based BC Algorithm Scheme

- Master
  1. initially splits the parameter domain into sub-domains and send them to the workers
  2. when a worker has completed its sub-domain, the master splits another sub-domain, and sends it to the idle worker
- Workers
  1. receives the sub-domain from the master
  2. calls IM on the points of this sub-domain
  3. sends the results (tiles) back to the master
  4. asks for more work

Domain-based BC Algorithms



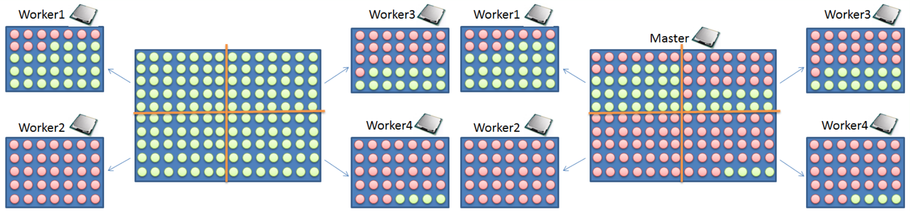
Static:

- One of Workers splits the domain then sends to other workers and gathers result of all workers
- Drawback: computing time depends on the slowest process

Dynamic:

- Master is only responsible for gathering tiles and splitting domain/sub-domains
- Master monitors all Workers then it can balance workload (by splitting) between workers

Domain-based BC Algorithms



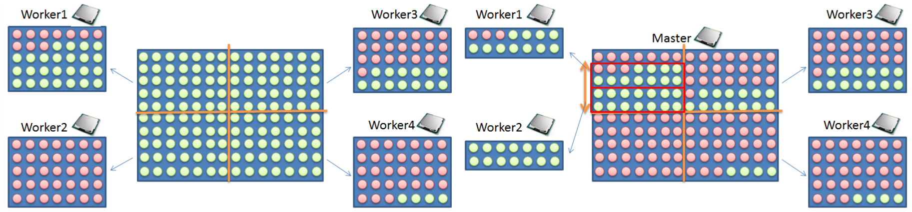
Static:

- One of Workers splits the domain then sends to other workers and gathers result of all workers
- Drawback: computing time depends on the slowest process

Dynamic:

- Master is only responsible for gathering tiles and splitting domain/sub-domains
- Master monitors all Workers then it can balance workload (by splitting) between workers

Domain-based BC Algorithms



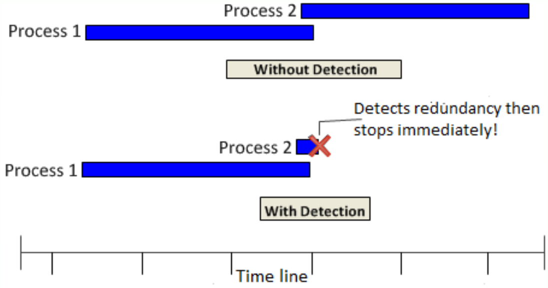
Static:

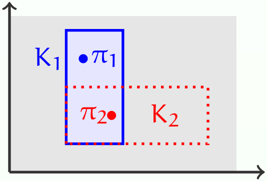
- One of Workers splits the domain then sends to other workers and gathers result of all workers
- Drawback: computing time depends on the slowest process

Dynamic:

- Master is only responsible for gathering tiles and splitting domain/sub-domains
- Master monitors all Workers then it can balance workload (by splitting) between workers

Redundancy Detection – Heuristic (For Dynamic Only)





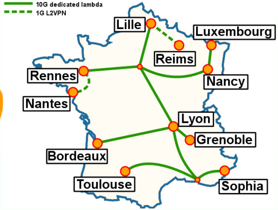
- Redundancy detection: a mechanism to detect and stop process which calling point is in a covered tile
- Example: better to stop immediately when the reference point ("π_2") is covered by another tile ("K_1"). Note that: There is non-determinism, so a same point can yield different tiles

Outline

Implementation in IMITATOR

- IMITATOR [André, Fribourg, Kühne, Soulat, 2012]
  - $26,000$ lines of OCaml code
    - Including $> 3,000$ lines for the distribution algorithms
  - Relies on the PPL library for operations on polyhedra [Bagnara et al., 2008]
  - Available under the GNU-GPL license at `www.imitator.fr`
- Distributed version of IMITATOR relying on MPI
  - Using the OcamlMPI library for passing messages between Master and Workers
- Implementation of the new algorithms: not trivial at all in practice

Experimental Validation



Experimental conducted on a 2 clusters of Grid'5000:

1. **Pastel(Located in Toulouse, FR): 140 nodes**
2. **Griffon(Located in Nancy, FR): 92 nodes**

Each node's	Pastel	Griffon
Processors	2x Dual-core AMD Opteron 2218@2.6GHz	2x Quad-core Intel Xeon L5420@2.5GHz
Memory (GB)	8	16
Interconnection network	GigaEthernet	GigaEthernet + 20G InfiniBand

Experiment Summary

Case study	Flip-flop4	RCP	Sched3-2	Sched3B-2	Sched3B-3	Sched5	SiMoP
Model							
Clocks	5	6	13	13	13	21	8
Parameters	4	2	2	2	3	2	2
$\|D\|$	386400	3050	286	14746	530856	1681	10201
Cartography							
# Tiles	190	19	59	71	378	177	48
N_{max}	128	32	64	128	128	128	64
Execution time at N_{max}							
Static	33.0	2108.0	4.0	26.6	181.0	213.0	21.4
Seq	2059.0	653.0	4.6	11.0	810.0	219.0	36.1
Random	652.0	635.0	3.6	8.4	524.0	148.0	23.6
Shuffle	670.0	624.0	3.1	7.6	243.0	140.0	18.7
Subdomain	48.0	1286.0	7.2	15.8	217.0	273.0	32.4
Subdomain + H	24.0	622.0	4.0	11.0	81.0	199.0	23.2
Hybrid	24.0	624.0	3.1	7.6	81.0	140.0	18.7

- Shuffle and Subdomain + H(Heuristic) are the fastest algorithms
- Hybrid:
  - $<$100k points: Uses Shuffle
  - $>$100k points: Uses Subdomain + H

Outline

Conclusion and Perspectives

- Conclusion:
  - Proposed a new efficient distributed algorithm for Behavioral Cartography
  - Proposed a new heuristic approach improving all BC distribution algorithms
  - Implemented the new algorithms in IMITATOR
- Future works:
  - Design an autonomous distribution scheme for BC
    - No master!
  - Try BC in GPU's or CPU+GPU's environment
  - ... and formally prove the deadlock-freeness of our master-worker communication scheme!

Bibliography

References I

Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993).
Parametric real-time reasoning.
In *STOC*, pages 592–601. ACM.

André, É., Coti, C., and Evangelista, S. (2014).
Distributed behavioral cartography of timed automata.
In Dongarra, J., Ishikawa, Y., and Atsushi, H., editors, *21st European MPI Users' Group Meeting (EuroMPI/ASIA'14)*, pages 109–114. ACM.

André, É. and Fribourg, L. (2010).
Behavioral cartography of timed automata.
In *RP*, volume 6227 of *Lecture Notes in Computer Science*, pages 76–90. Springer.

André, É., Fribourg, L., Kühne, U., and Soulat, R. (2012).
IMITATOR 2.5: A tool for analyzing robustness in scheduling problems.
In *FM*, volume 7436 of *Lecture Notes in Computer Science*, pages 33–36. Springer.

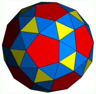
Bagnara, R., Hill, P. M., and Zaffanella, E. (2008).
The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems.
Science of Computer Programming, 72(1–2):3–21.

Licensing

Source of the graphics used I

Title: Smiley green alien big eyes (aaah)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

Title: Smiley green alien big eyes (cry)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain



Title: Polyhedron
Author: Robert Webb
Source: http://commons.wikimedia.org/wiki/File:Uniform_polyhedron-53-s012.png
License: public domain

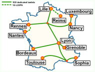
Title: MPI logo
Author: Unknown
Source: http://www.open-mpi.org
License: Unknown

Source of the graphics used II

Title: Ocaml logo
Author: Amir Chaudhry
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: CC BY-SA 4.0

Title: IMITATOR logo (Typing Monkey)
Author: Kater Begemot
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: CC BY-SA 3.0

Title: Logo Grid'5000
Author: Unknown
Source: https://www.lri.fr/~fci/Grid5000/images/logo.jpg
License: Unknown



Title: Grid'5000
Author: Unknown
Source: https://www.grid5000.fr/mediawiki/index.php/File:Renater5-g5k.jpg
License: Unknown

License of this document