EuroMPI/ASIA 2014

September 12th, 2014
Kyoto, Japan

Distributed Behavioral Cartography of Timed Automata

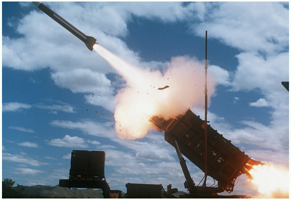
Étienne André, Camille Coti, Sami Evangelista

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

Context: Formal Verification of Timed Systems (1/2)

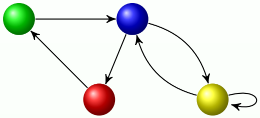
- Need for early bug detection
  - Bugs discovered when final testing: expensive

  ⇝ Need for a thorough modeling and verification phase



Context: Formal Verification of Timed Systems (2/2)

- Use formal methods



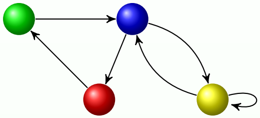
A model of the system

 is unreachable

A property to be satisfied

Context: Formal Verification of Timed Systems (2/2)

- Use formal methods



$$\overset{?}{\models}$$

 is unreachable

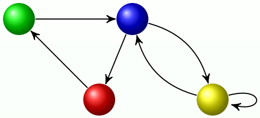
A model of the system

A property to be satisfied

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

Context: Formal Verification of Timed Systems (2/2)

- Use formal methods



$\overset{?}{\models}$

 is unreachable

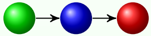
A model of the system

A property to be satisfied

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

Yes

No



Counterexample

Context: Parameter Synthesis

- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"

- Verification for one set of constants does not usually guarantee the correctness for other values

- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness: What happens if 50 is implemented with 49.99?

Context: Parameter Synthesis

- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"

- Verification for one set of constants does not usually guarantee the correctness for other values

- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness: What happens if 50 is implemented with 49.99?

- Parameter synthesis
  - Consider that timing constants are unknown constants (parameters)
  - Find good values for the parameters

Outline

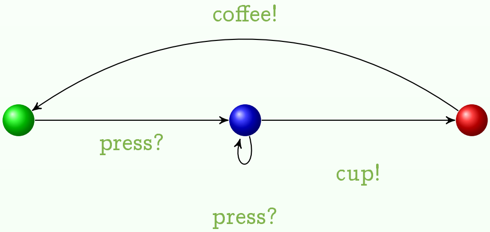
Outline

Timed Automaton (TA)

- Finite state automaton (sets of locations)

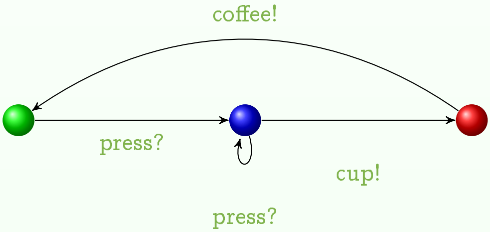
Timed Automaton (TA)

- Finite state automaton (sets of locations and actions)



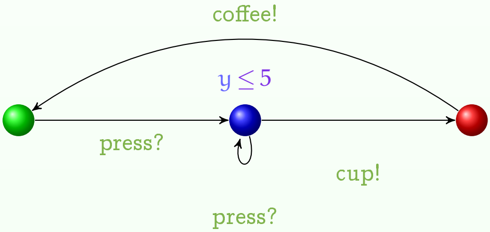
Timed Automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate



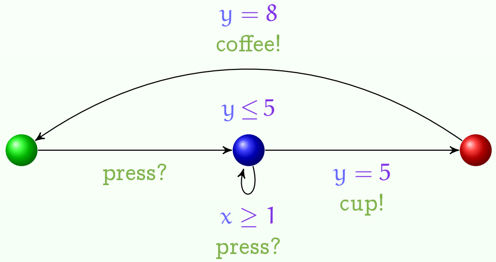
Timed Automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: property to be verified to stay at a location



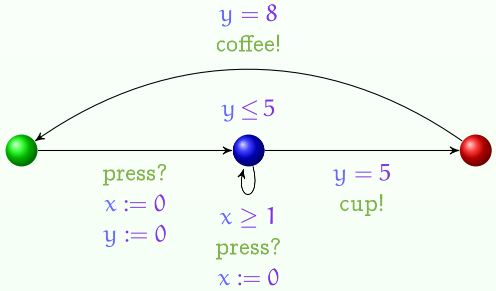
Timed Automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition

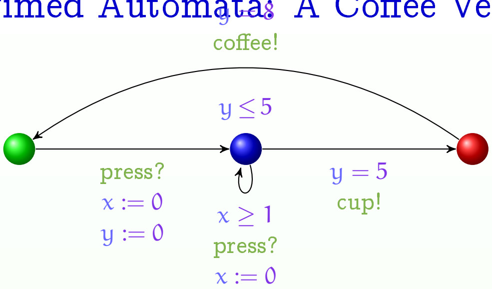


Timed Automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 at each transition

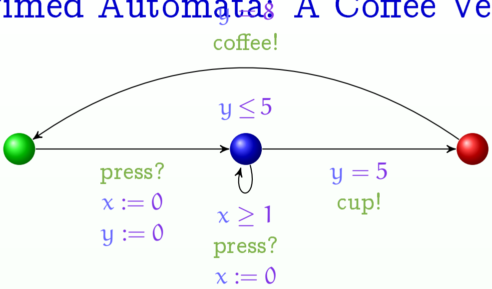


Timed Automata: A Coffee Vending Machine



- Examples of concrete runs

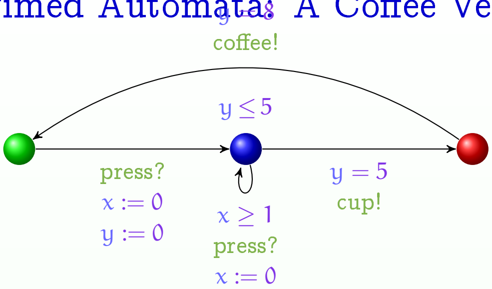
Timed Automata: A Coffee Vending Machine



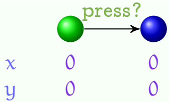
- Examples of concrete runs
  - Coffee with no sugar

x	0
y	0

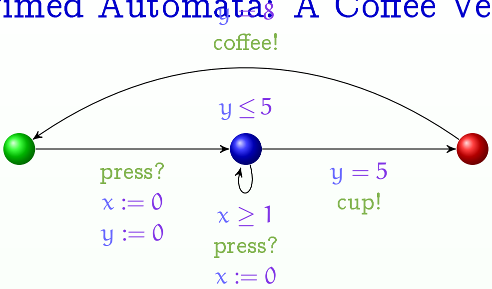
Timed Automata: A Coffee Vending Machine



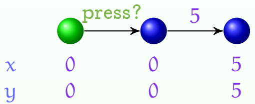
- Examples of concrete runs
  - Coffee with no sugar



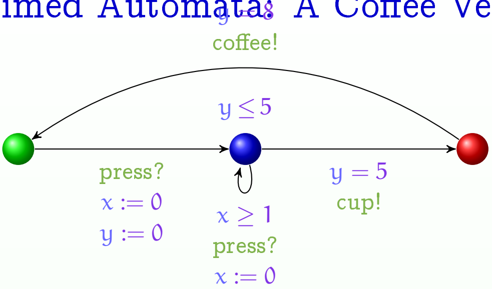
Timed Automata: A Coffee Vending Machine



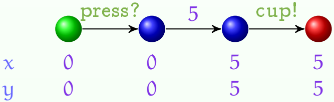
- Examples of concrete runs
  - Coffee with no sugar



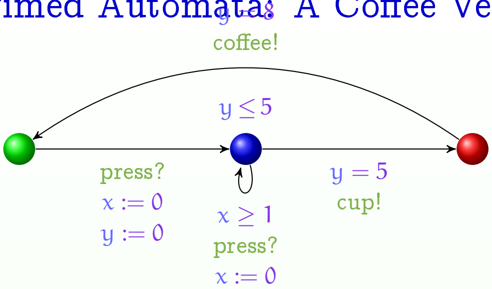
Timed Automata: A Coffee Vending Machine



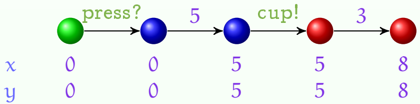
- Examples of concrete runs
  - Coffee with no sugar



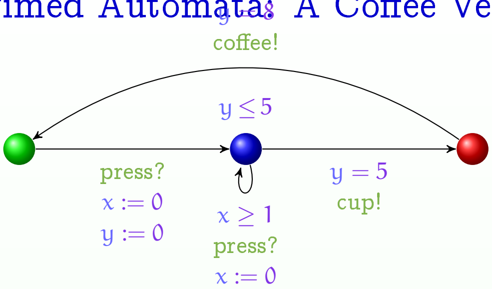
Timed Automata: A Coffee Vending Machine



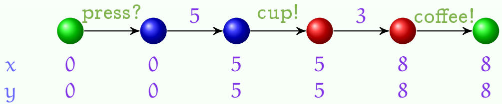
- Examples of concrete runs
  - Coffee with no sugar



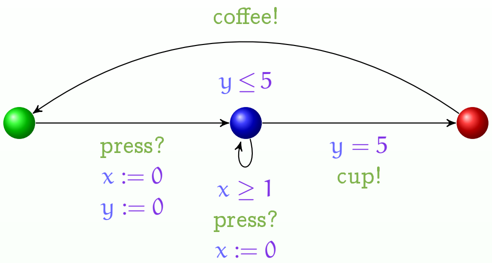
Timed Automata: A Coffee Vending Machine



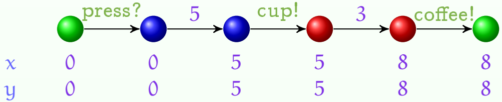
- Examples of concrete runs
  - Coffee with no sugar



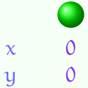
Timed Automata: A Coffee Vending Machine



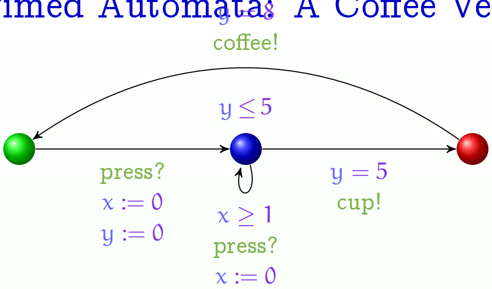
- Examples of concrete runs
  - Coffee with no sugar



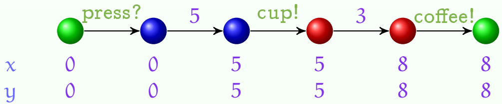
  - Coffee with 2 doses of sugar



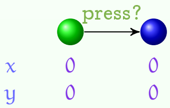
Timed Automata: A Coffee Vending Machine



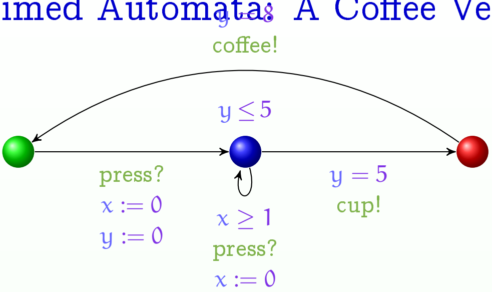
- Examples of concrete runs
  - Coffee with no sugar



  - Coffee with 2 doses of sugar



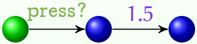
Timed Automata: A Coffee Vending Machine



- Examples of concrete runs
  - Coffee with no sugar

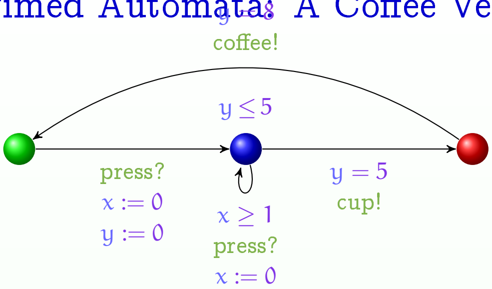
x	0	0	5	5	8	8
y	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



x	0	0	1.5
y	0	0	1.5

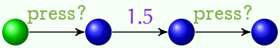
Timed Automata: A Coffee Vending Machine



- Examples of concrete runs
  - Coffee with no sugar

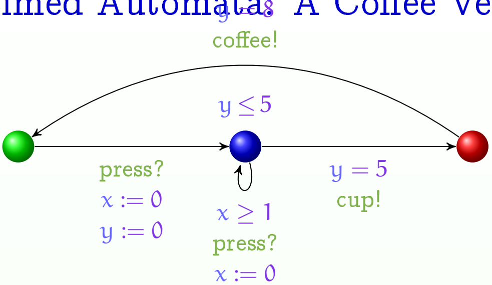
x	0	0	5	5	8	8
y	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



x	0	0	1.5	0
y	0	0	1.5	1.5

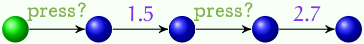
Timed Automata: A Coffee Vending Machine



- Examples of concrete runs
  - Coffee with no sugar

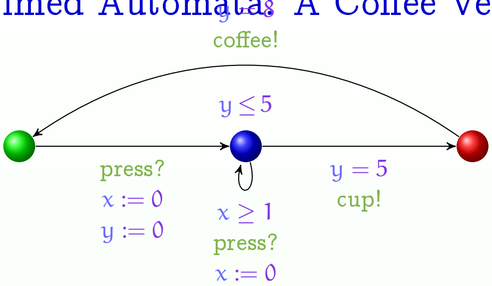
x	0	0	5	5	8	8
y	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

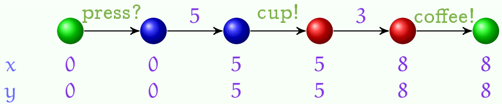


x	0	0	1.5	0	2.7
y	0	0	1.5	1.5	4.2

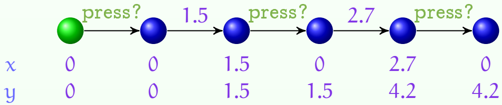
Timed Automata: A Coffee Vending Machine



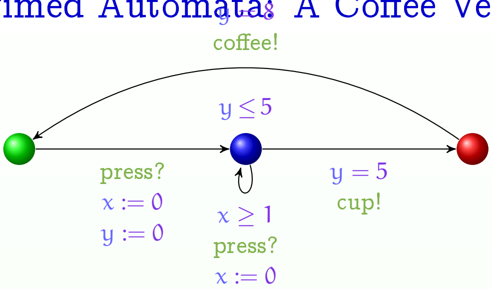
- Examples of concrete runs
  - Coffee with no sugar



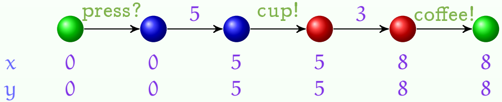
  - Coffee with 2 doses of sugar



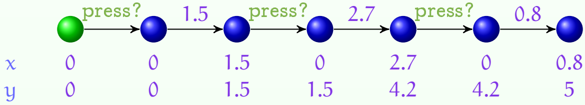
Timed Automata: A Coffee Vending Machine



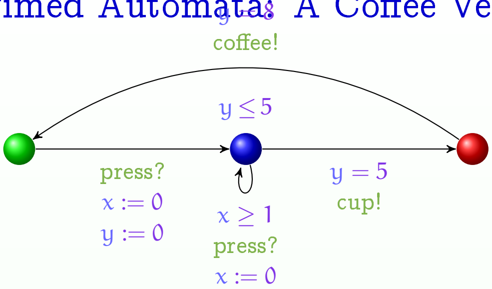
- Examples of concrete runs
  - Coffee with no sugar



  - Coffee with 2 doses of sugar



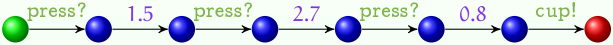
Timed Automata: A Coffee Vending Machine



- Examples of concrete runs
  - Coffee with no sugar

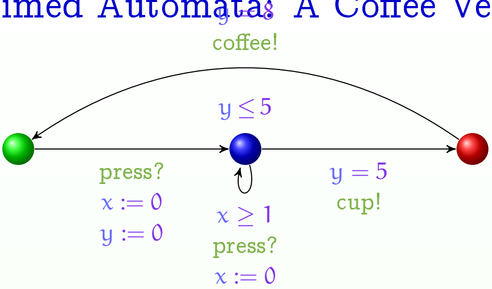
x	0	0	5	5	8	8
y	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



x	0	0	1.5	0	2.7	0	0.8	0.8
y	0	0	1.5	1.5	4.2	4.2	5	5

Timed Automata: A Coffee Vending Machine



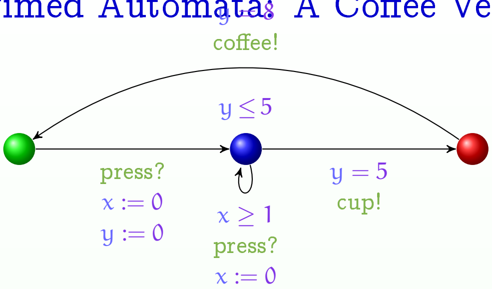
- Examples of concrete runs
  - Coffee with no sugar

		press?		5		cup!		3		coffee!	
x	0		0		5		5		8		8
y	0		0		5		5		8		8

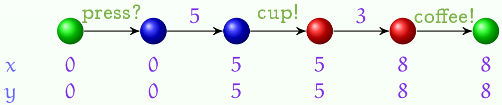
  - Coffee with 2 doses of sugar

		press?		1.5		press?		2.7		press?		0.8		cup!		3	
x	0		0		1.5		0		2.7		0		0.8		0.8		3.8
y	0		0		1.5		1.5		4.2		4.2		5		5		8

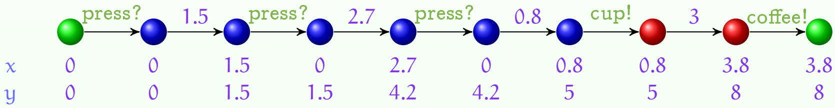
Timed Automata: A Coffee Vending Machine



- Examples of concrete runs
  - Coffee with no sugar

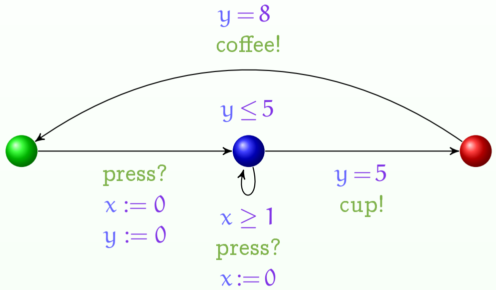


  - Coffee with 2 doses of sugar



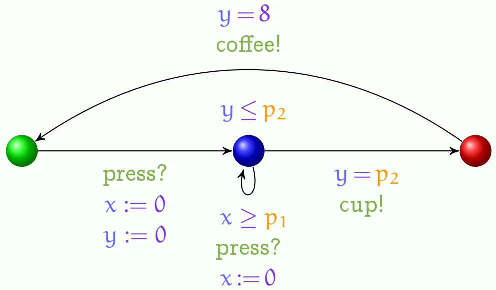
Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks)



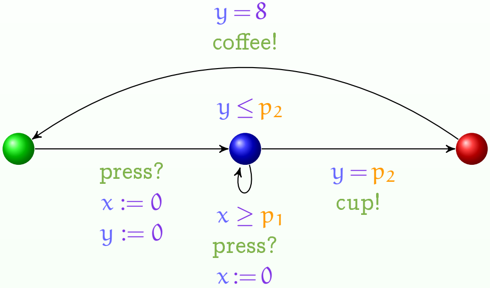
Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993]
  - Unknown constants used in guards and invariants



Parametric Timed Automaton (PTA)

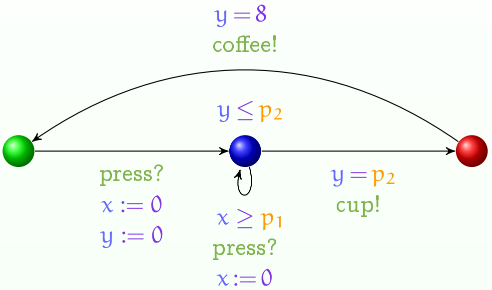
- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993]
  - Unknown constants used in guards and invariants



- Examples of problems
  - "Do there exist parameter valuations such that one can never get a coffee?"

Parametric Timed Automaton (PTA)

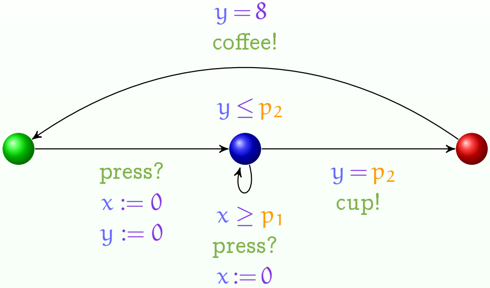
- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993]
  - Unknown constants used in guards and invariants



- Examples of problems
  - "Do there exist parameter valuations such that one can never get a coffee?" Yes! e.g.: $p_1 = 2, p_2 = 10$

Parametric Timed Automaton (PTA)

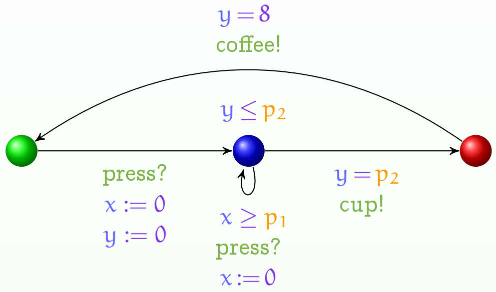
- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993]
  - Unknown constants used in guards and invariants



- Examples of problems
  - "Do there exist parameter valuations such that one can never get a coffee?" Yes! e.g.: $p_1 = 2, p_2 = 10$
  - "What are all possible parameter valuations such that one can get a coffee with 3 doses of sugar?"

Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993]
  - Unknown constants used in guards and invariants



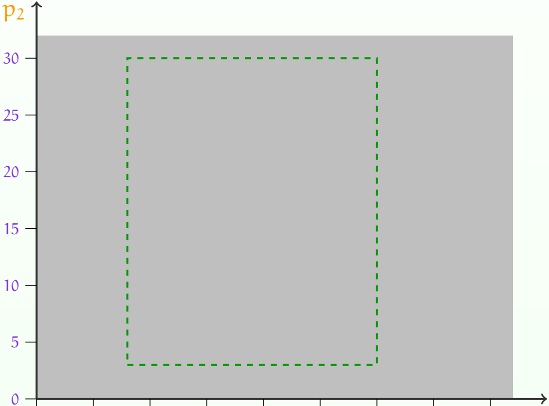
- Examples of problems
  - "Do there exist parameter valuations such that one can never get a coffee?" Yes! e.g.: $p_1 = 2, p_2 = 10$
  - "What are all possible parameter valuations such that one can get a coffee with 3 doses of sugar?" $p_2 \leq 8 \wedge p_2 \geq 3 \times p_1$

Behavioral Cartography

Partition the parameter state space into tiles

- Tile: constraint in which the discrete behavior ("same number of doses of sugar") is uniform

Method: done by calling the inverse method IM on integer points (parameter valuations) sequentially [André and Fribourg, 2010]

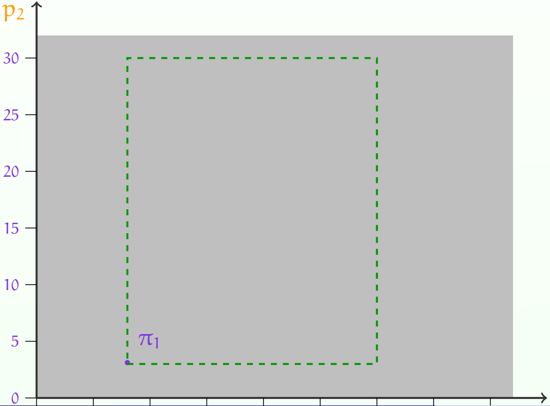


Behavioral Cartography

Partition the parameter state space into tiles

- Tile: constraint in which the discrete behavior ("same number of doses of sugar") is uniform

Method: done by calling the inverse method IM on integer points (parameter valuations) sequentially [André and Fribourg, 2010]

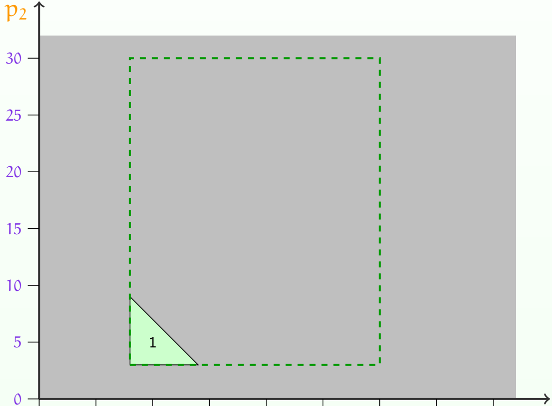


Behavioral Cartography

Partition the parameter state space into tiles

- Tile: constraint in which the discrete behavior ("same number of doses of sugar") is uniform

Method: done by calling the inverse method IM on integer points (parameter valuations) sequentially [André and Fribourg, 2010]

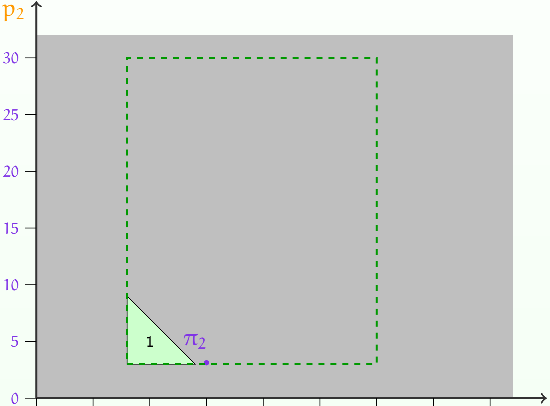


Behavioral Cartography

Partition the parameter state space into tiles

- Tile: constraint in which the discrete behavior ("same number of doses of sugar") is uniform

Method: done by calling the inverse method IM on integer points (parameter valuations) sequentially [André and Fribourg, 2010]

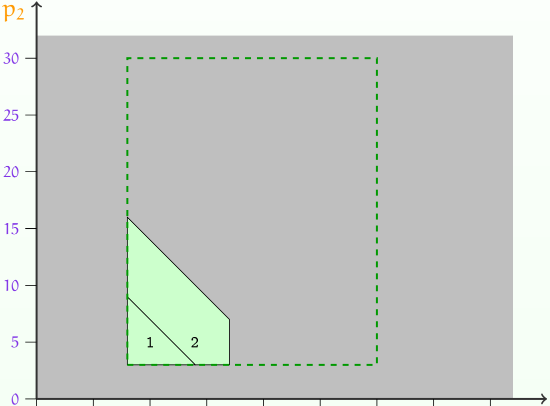


Behavioral Cartography

Partition the parameter state space into tiles

- Tile: constraint in which the discrete behavior ("same number of doses of sugar") is uniform

Method: done by calling the inverse method IM on integer points (parameter valuations) sequentially [André and Fribourg, 2010]

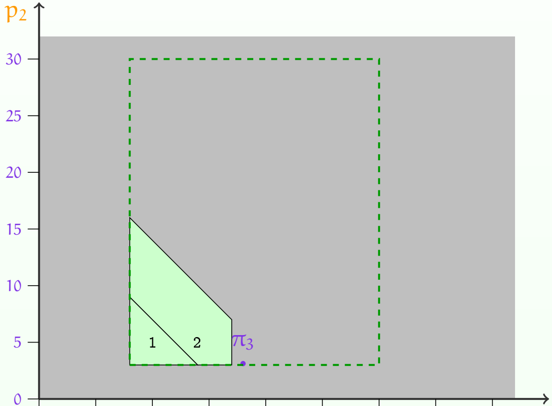


Behavioral Cartography

Partition the parameter state space into tiles

- Tile: constraint in which the discrete behavior ("same number of doses of sugar") is uniform

Method: done by calling the inverse method IM on integer points (parameter valuations) sequentially [André and Fribourg, 2010]

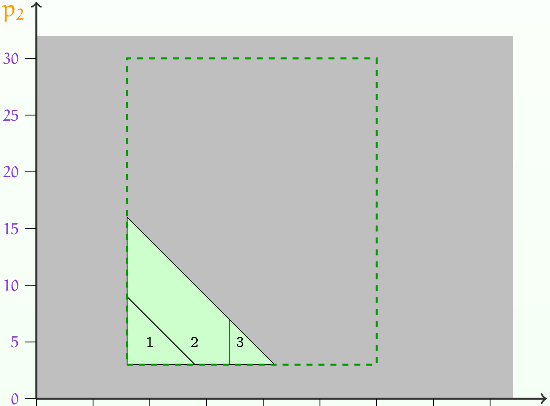


Behavioral Cartography

Partition the parameter state space into tiles

- Tile: constraint in which the discrete behavior ("same number of doses of sugar") is uniform

Method: done by calling the inverse method IM on integer points (parameter valuations) sequentially [André and Fribourg, 2010]

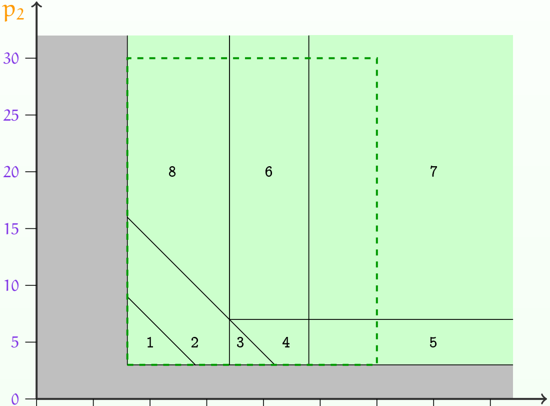


Behavioral Cartography

Partition the parameter state space into tiles

- Tile: constraint in which the discrete behavior ("same number of doses of sugar") is uniform

Method: done by calling the inverse method IM on integer points (parameter valuations) sequentially [André and Fribourg, 2010]

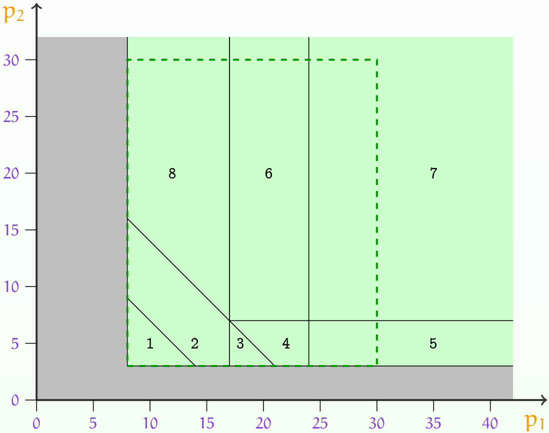


Behavioral Cartography: Partition

Application: given a linear-time property ("the coffee may have at least 3 doses of sugar"), one can partition the tiles into good and bad

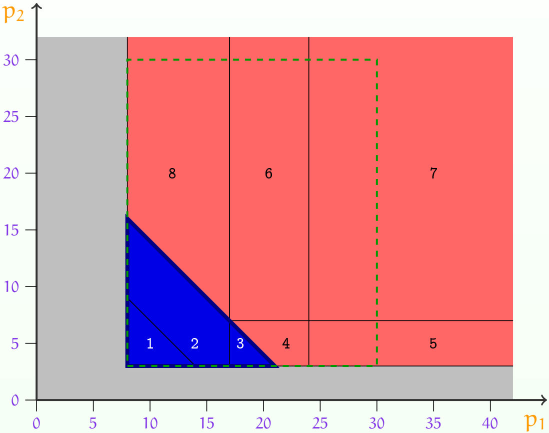
Behavioral Cartography: Partition

Application: given a linear-time property ("the coffee may have at least 3 doses of sugar"), one can partition the tiles into good and bad



Behavioral Cartography: Partition

Application: given a linear-time property (“the coffee may have at least 3 doses of sugar”), one can partition the tiles into good and bad



Outline

Distributing the cartography

Problem

Running the inverse method is long, and hence computing the cartography even more.

Distributing the cartography

Problem

Running the inverse method is long, and hence computing the cartography even more.

General goal

Distributing the cartography in order to take advantage of clusters.

Intrinsically easy since the cartography is built by calling the inverse method on a sequential set of points

Distributing the cartography

Problem

Running the inverse method is long, and hence computing the cartography even more.

General goal

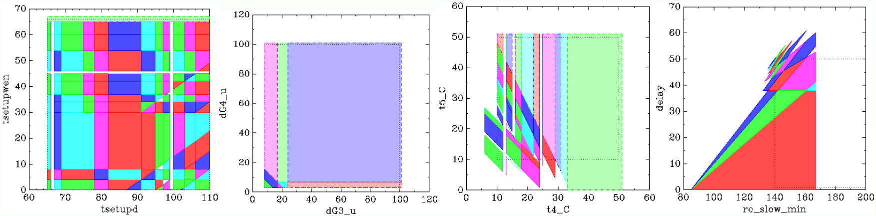
Distributing the cartography in order to take advantage of clusters.

Intrinsically easy since the cartography is built by calling the inverse method on a sequential set of points

. . . but doing it efficiently is far from trivial in practice!

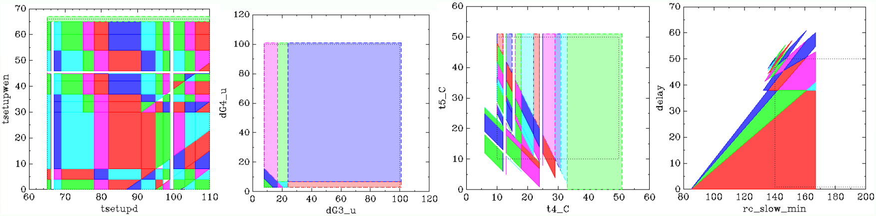
Problem 1

The general "shape" of the cartography is unknown in general



Problem 1

The general "shape" of the cartography is unknown in general

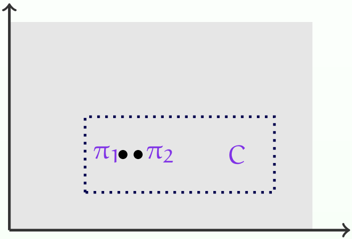


$\rightsquigarrow$ rules out the idea of partitioning the parameter space a priori

Problem 2

Calling the inverse method IM in parallel on two nodes starting from two close points will very probably yield the same tile

$\rightsquigarrow$ loss of efficiency

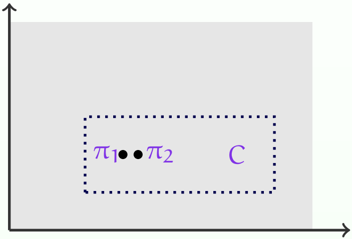


Idea: call the inverse method IM on points as far as possible

Problem 2

Calling the inverse method IM in parallel on two nodes starting from two close points will very probably yield the same tile

$\rightsquigarrow$ loss of efficiency



Idea: call the inverse method IM on points as far as possible

- But what does "as far as possible" mean for n nodes in m parameter dimensions?

Outline

A Master-Worker Scheme

Traditional Master-Worker communication scheme

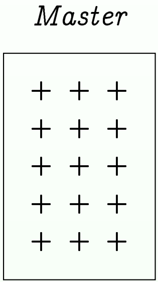
- Workers ask the master for a point, call IM on that point, and send the resulting tile to the master
- The master is responsible for the smart repartition of the data (*i.e.*, the points) between workers
  - In this work: 2 different algorithms for the master

Sequential Algorithm

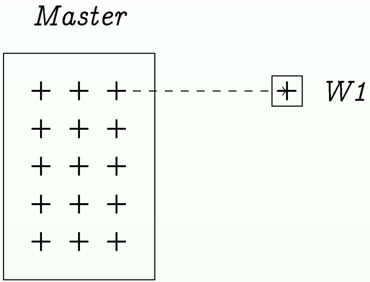
General idea

1. Enumerate all points starting from 0
2. When a point not yet covered by any tile is found, send it to the worker asking for work

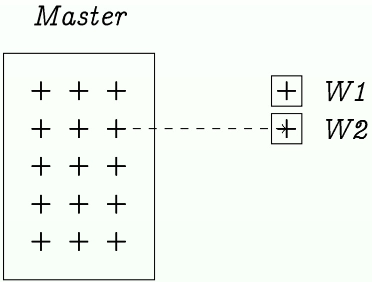
Sequential Algorithm: Graphical Explanation



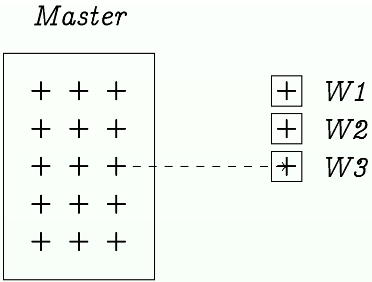
Sequential Algorithm: Graphical Explanation



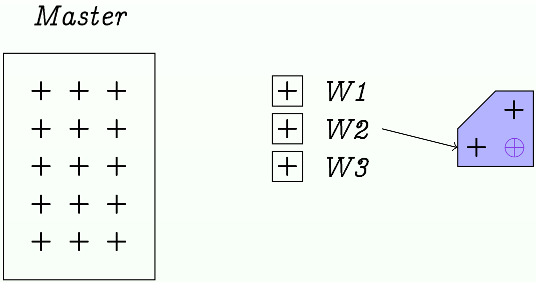
Sequential Algorithm: Graphical Explanation



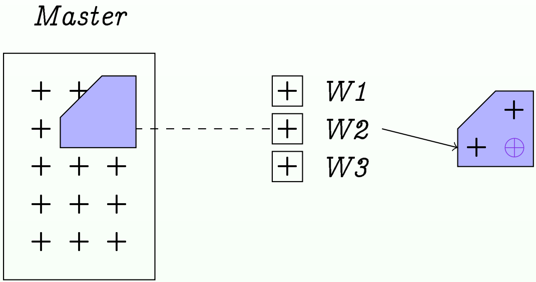
Sequential Algorithm: Graphical Explanation



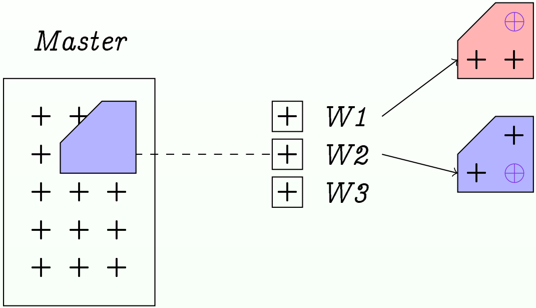
Sequential Algorithm: Graphical Explanation



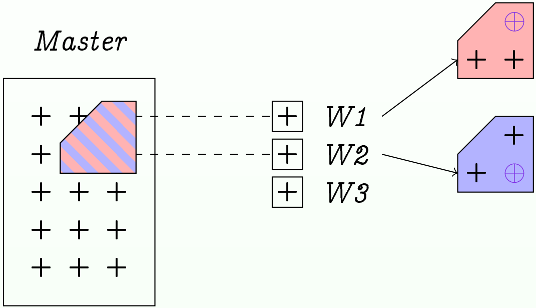
Sequential Algorithm: Graphical Explanation



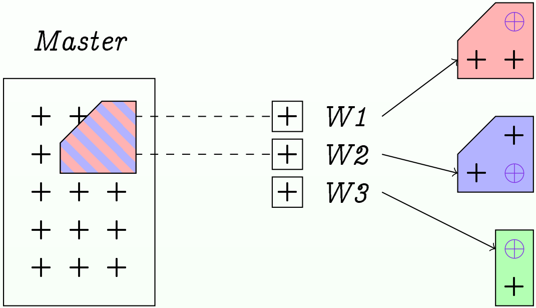
Sequential Algorithm: Graphical Explanation



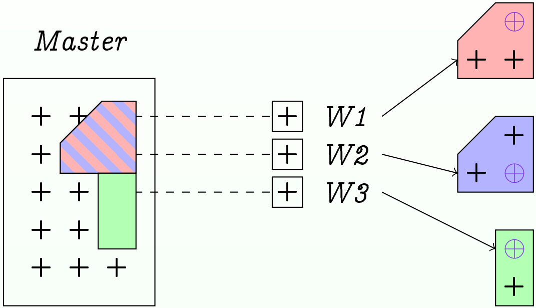
Sequential Algorithm: Graphical Explanation



Sequential Algorithm: Graphical Explanation



Sequential Algorithm: Graphical Explanation



Random+Sequential Algorithm

General idea

1. Try to find randomly a point not covered by any tile
2. After MAX consecutive failed attempts to find a point not covered by any tile, check sequentially all points starting from 0

Random+Sequential Algorithm

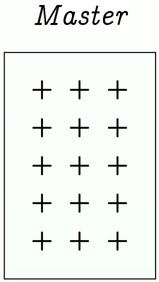
General idea

1. Try to find randomly a point not covered by any tile
2. After MAX consecutive failed attempts to find a point not covered by any tile, check sequentially all points starting from 0

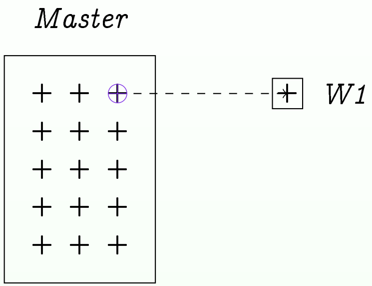
The second phase is costly, but necessary to ensure the full coverage of integer points

- Otherwise, would only guarantee a coverage of, e.g., 99 %

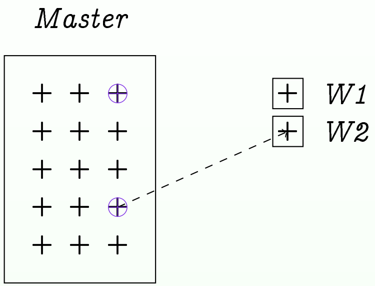
Random+Sequential: Graphical Explanation



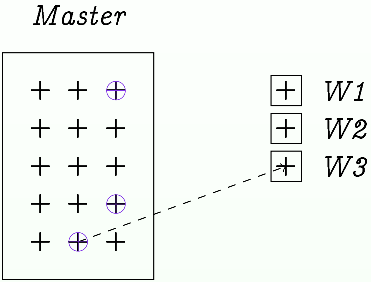
Random+Sequential: Graphical Explanation



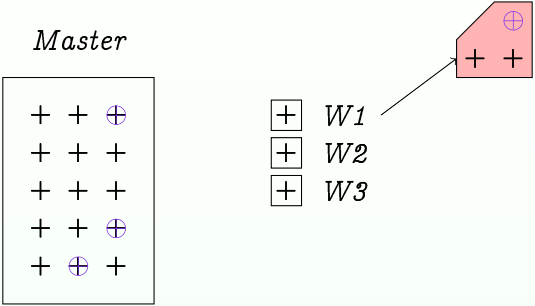
Random+Sequential: Graphical Explanation



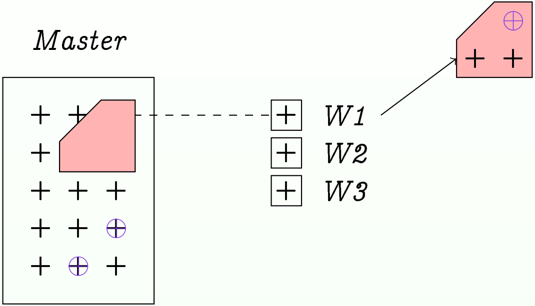
Random+Sequential: Graphical Explanation



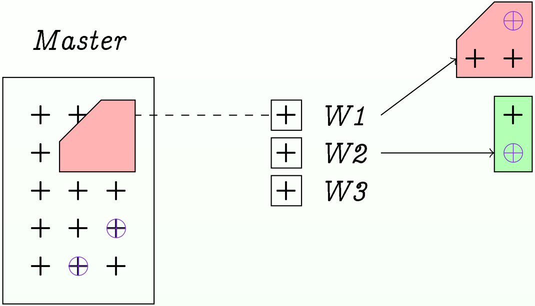
Random+Sequential: Graphical Explanation



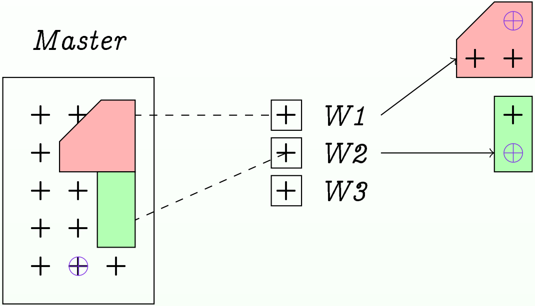
Random+Sequential: Graphical Explanation



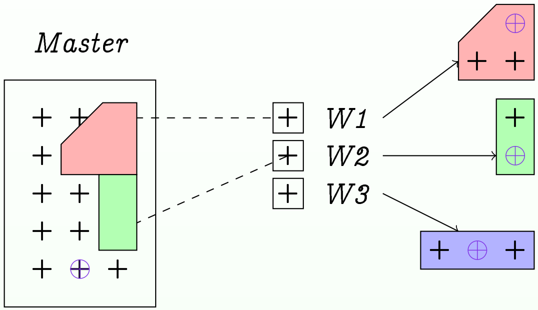
Random+Sequential: Graphical Explanation



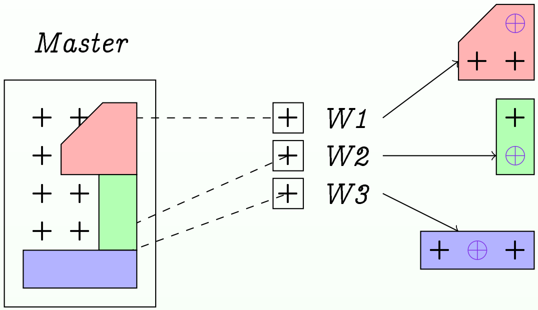
Random+Sequential: Graphical Explanation



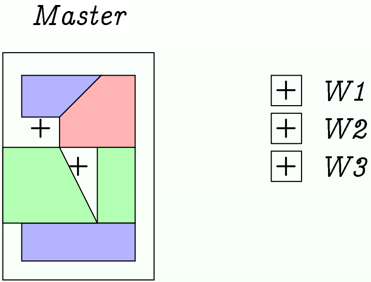
Random+Sequential: Graphical Explanation



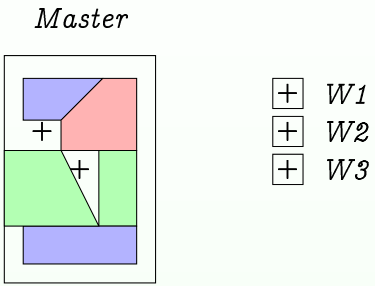
Random+Sequential: Graphical Explanation



Random+Sequential: Graphical Explanation



Random+Sequential: Graphical Explanation



...then switch to sequential enumeration to cover the remaining integer points

Outline

Implementation in a distributed version of IMITATOR

- IMITATOR [A., Fribourg, Kühne, Soulat, 2012]
  - "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 10,000 lines of OCaml code
  - Relies on the PPL library for operations on polyhedra [Bagnara et al., 2008]
  - Available under the GNU-GPL license

- Distributed extension of IMITATOR using MPI
  - Using the OcamlMPI library

Implementation in a distributed version of IMITATOR

- IMITATOR [A., Fribourg, Kühne, Soulat, 2012]
  - "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 10,000 lines of OCaml code
  - Relies on the PPL library for operations on polyhedra [Bagnara et al., 2008]
  - Available under the GNU-GPL license
- Distributed extension of IMITATOR using MPI
  - Using the OcamlMPI library
  - . . . in which we found a bug!

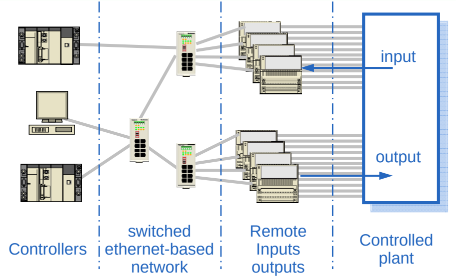
Description of the case studies

Sched3

- Parametric schedulability problem
- 2 parameters, 268 integer points

SIMOP

- Model of a networked automation system (NAS)
- 2 parameters, 10,201 integer points



Environment of the Experiments

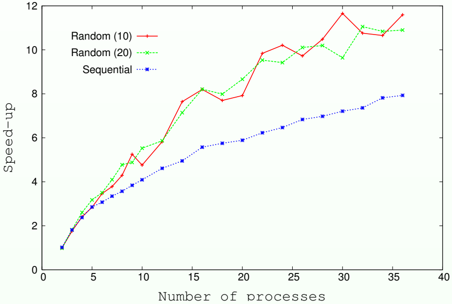
Magi cluster (Paris 13)

- Intel Xeon X5570, 2.93 GHz, 6 cores/CPU, 2 CPUs/node
- Memory: 24 GB/node (2 GB/core)
- 40 Gb InfiniBand network

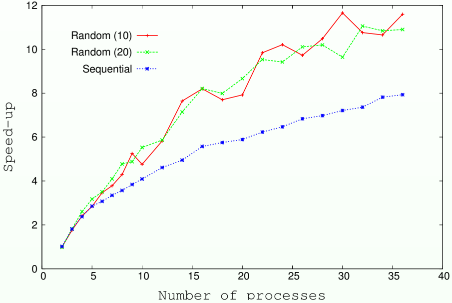
Software environment

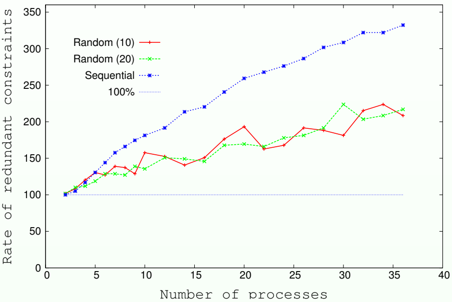
- Linux 3.2.0, 64 bits
- gcc 4.7.2, ocamlc 3.12.1
- Bullx OpenMPI 1.8.2, OCamlMPI 1.01

Graphical Comparison: Sched3

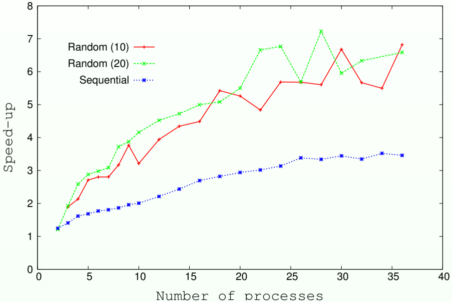


Graphical Comparison: Sched3

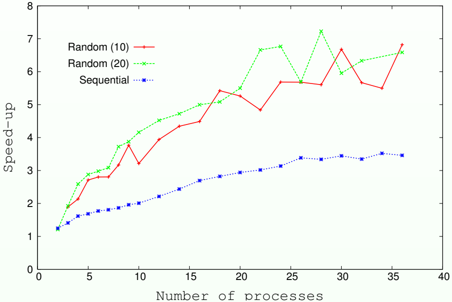


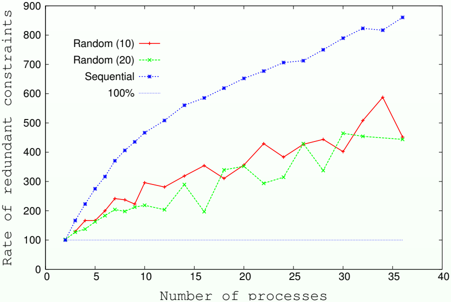


Graphical Comparison: Simop



Graphical Comparison: Simop





Analysis of the results: Sched3

Algorithm	**Sequential**	**Random10**	**Random20**
Time (seq)	40.29 s	N/A	N/A
# of cons. (seq)	59	N/A	N/A
Time (3 procs)	22.26 s	22.93 s	22.18 s
# of cons. (3 procs)	62	64	65
Time (36 procs)	5.08 s	3.48 s	3.70 s
# of cons. (36 procs)	196	123	128

Analysis of the results: Simop

Algorithm	**Sequential**	**Random10**	**Random20**
Time (seq)	121.91 s	N/A	N/A
# of cons. (seq)	48	N/A	N/A
Time (3 procs)	86.51 s	64.40 s	63.30 s
# of cons. (3 procs)	81	62	61
Time (36 procs)	35.23 s	17.87 s	18.51 s
# of cons. (36 procs)	413	217	213

Interpretation of the experiments

Summary of the experiments

- Adding more workers always decreases the computation time
  - Decrease by a factor of 8 (resp. 12) with 36 nodes
- Random+sequential much more efficient than sequential, despite the 2nd phase cost
- Random+sequential more or less linear for Sched3, less for SIMOP

Limitations of the use cases

- Few tiles (48 for SIMOP, 59 for Sched3): Intrinsically limits the efficiency for many workers

Outline

Conclusion

First attempt to distribute the behavioral cartography

- In fact first attempt for performing distributed parameter synthesis

Results quite promising

- . . . although there is still a lot of space for improvement!

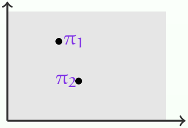
Perspectives

- Ongoing work: new algorithms
  - Master-worker with shuffle [completed]
  - Unsupervised workers with a common memory node [ongoing]
  - Totally decentralized [starting]

Perspectives

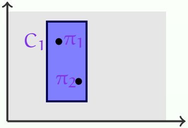
- Ongoing work: new algorithms
  - Master-worker with shuffle [completed]
  - Unsupervised workers with a common memory node [ongoing]
  - Totally decentralized [starting]

- Open questions and heuristics
  - Should we stop an ongoing execution of IM when its node was covered by another tile?



Perspectives

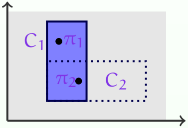
- Ongoing work: new algorithms
  - Master-worker with shuffle [completed]
  - Unsupervised workers with a common memory node [ongoing]
  - Totally decentralized [starting]
- Open questions and heuristics
  - Should we stop an ongoing execution of IM when its node was covered by another tile?



Perspectives

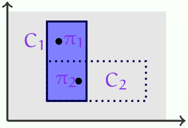
- Ongoing work: new algorithms
  - Master-worker with shuffle [completed]
  - Unsupervised workers with a common memory node [ongoing]
  - Totally decentralized [starting]

- Open questions and heuristics
  - Should we stop an ongoing execution of IM when its node was covered by another tile?



Perspectives

- Ongoing work: new algorithms
  - Master-worker with shuffle [completed]
  - Unsupervised workers with a common memory node [ongoing]
  - Totally decentralized [starting]
- Open questions and heuristics
  - Should we stop an ongoing execution of IM when its node was covered by another tile?



- Orthogonal problems
  - Coverage of almost all the state space (e.g., 99 %): towards a purely random algorithm?
  - Parallel parametric verification using multi-core (based on, e.g., [Evangelista et al., 2012])

Bibliography

References I

Alur, R. and Dill, D. L. (1994).
A theory of timed automata.
Theoretical Computer Science, 126(2):183–235.

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

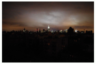
Bagnara, R., Hill, P. M., and Zaffanella, E. (2008).
The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems.
Science of Computer Programming, 72(1–2):3–21.

References II

Evangelista, S., Laarman, A., Petrucci, L., and Pol, J. V. D. (2012).
Improved multi-core nested depth-first search.
In *ATVA*, volume 7561 of *LNCS*, pages 269–283.

Additional explanation

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)

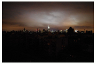
Allusion to any plane crash
(Picture actually from the happy-ending US Airways Flight 1549, 2009)

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 I



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

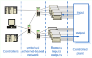
Title: Deepwater Horizon Offshore Drilling Platform on Fire
Author: ideum
Source: https://secure.flickr.com/photos/ideum/4711481781/
License: CC BY-SA 2.0

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

Source of the graphics used II

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: Example of a networked automation system
Author: unknown
Source: unknown
License: unknown (all rights reserved)

License of this document

Authors: Étienne André, Camille Coti and Sami Evangelista