Computing Students Talks

26th January 2011

The Good, the Bad and the Unknown

Synthesis of Timing Parameters in Concurrent Systems

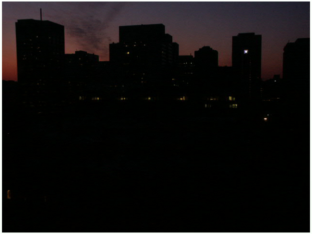
Étienne ANDRÉ

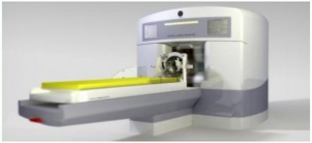
PAT Team

School of Computing, National University of Singapore

Verification of Real Time Systems

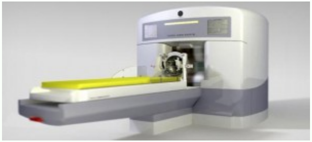
- Motivation





Context: Model Checking Timed Systems (1/2)

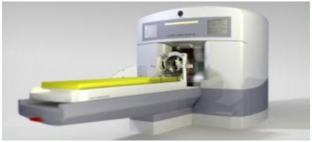
- Input



A timed concurrent system

Context: Model Checking Timed Systems (1/2)

- Input

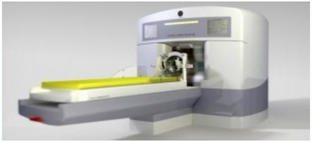


A timed concurrent system

A good behavior expected for the system

Context: Model Checking Timed Systems (1/2)

- Input



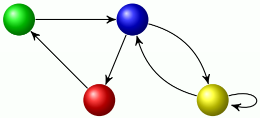
A timed concurrent system

A good behavior expected for the system

- Question: does the system always behave well?

Context: Model Checking Timed Systems (2/2)

- Use of formal methods



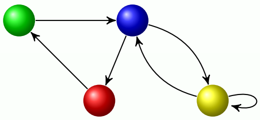
$\mathsf{AG}\neg \mathit{bad}$

A finite model of the system

A formula to be satisfied

Context: Model Checking Timed Systems (2/2)

- Use of formal methods



$\overset{?}{\models}$

$AG\neg bad$

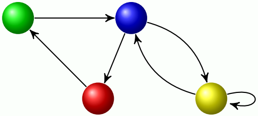
A finite model of the system

A formula to be satisfied

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

Context: Model Checking Timed Systems (2/2)

- Use of formal methods



$\models^{?}$ $\quad$ $AG\neg bad$

A finite model of the system $\qquad$ A formula to be satisfied

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

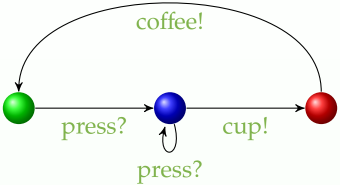
Yes $\qquad$ **No**

Counterexample

Outline

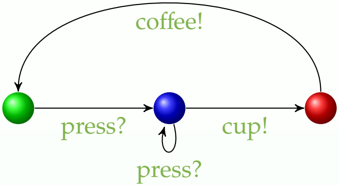
Outline

Model



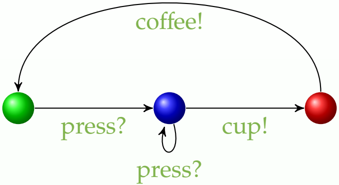
- Waiting
- Adding sugar
- Delivering coffee

Model



- Example of runs
  - Coffee with no sugar

Model

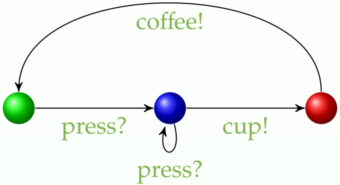


- Waiting
- Adding sugar
- Delivering coffee

- Example of runs
  - Coffee with no sugar

  - Coffee with 2 doses of sugar

Model



- Waiting
- Adding sugar
- Delivering coffee

- Example of runs
  - Coffee with no sugar

  - Coffee with 2 doses of sugar

  - And so on

Temporal Logics

- Specify properties on the order between events
- Example: CTL (Computation Tree Logic) [Clarke and Emerson, 1982]
    - "After the button is pressed, a coffee is always eventually delivered."

Temporal Logics

- Specify properties on the order between events
- Example: CTL (Computation Tree Logic) [Clarke and Emerson, 1982]
  - "After the button is pressed, a coffee is always eventually delivered." ($\times$)

Temporal Logics

- Specify properties on the order between events
- Example: CTL (Computation Tree Logic) [Clarke and Emerson, 1982]
  - "After the button is pressed, a coffee is always eventually delivered." ($\times$)
  - "After the button is pressed, there exists an execution such that a coffee is eventually delivered."

Temporal Logics

- Specify properties on the order between events
- Example: CTL (Computation Tree Logic) [Clarke and Emerson, 1982]
  - "After the button is pressed, a coffee is always eventually delivered." ($\times$)
  - "After the button is pressed, there exists an execution such that a coffee is eventually delivered." ($\surd$)

Temporal Logics

- Specify properties on the order between events
- Example: CTL (Computation Tree Logic) [Clarke and Emerson, 1982]
  - "After the button is pressed, a coffee is always eventually delivered." ($\times$)
  - "After the button is pressed, there exists an execution such that a coffee is eventually delivered." ($\surd$)
  - "It is possible to get a coffee with 2 doses of sugar."

Temporal Logics

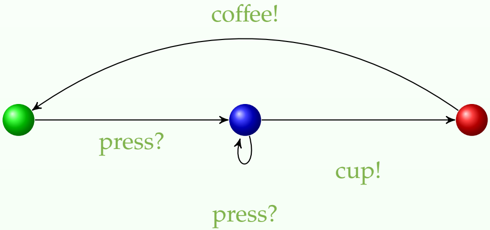
- Specify properties on the order between events
- Example: CTL (Computation Tree Logic) [Clarke and Emerson, 1982]
    - “After the button is pressed, a coffee is always eventually delivered.” ($\times$)
    - “After the button is pressed, there exists an execution such that a coffee is eventually delivered.” ($\surd$)
    - “It is possible to get a coffee with 2 doses of sugar.” ($\surd$)

Timed Automaton

- Finite state automaton (sets of locations)

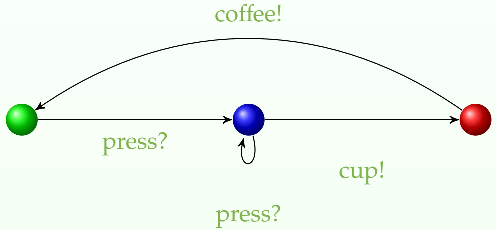
Timed Automaton

- Finite state automaton (sets of locations and actions)



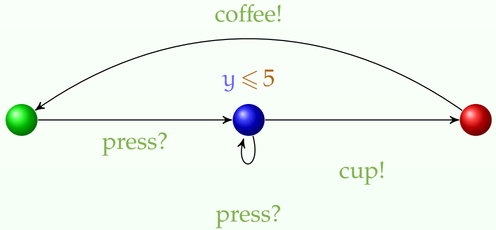
Timed Automaton

- 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



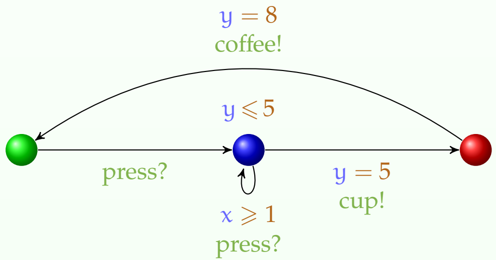
Timed Automaton

- 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



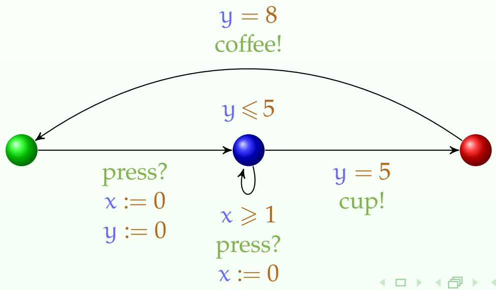
Timed Automaton

- 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

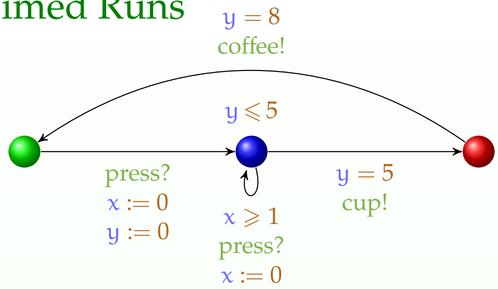


Timed Automaton

- 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

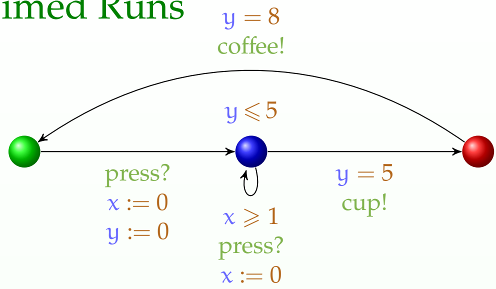


Timed Runs

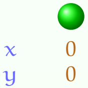


- Examples of timed runs

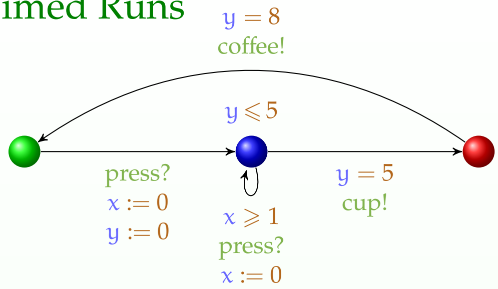
Timed Runs



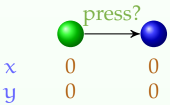
- Examples of timed runs
  - Coffee with no sugar



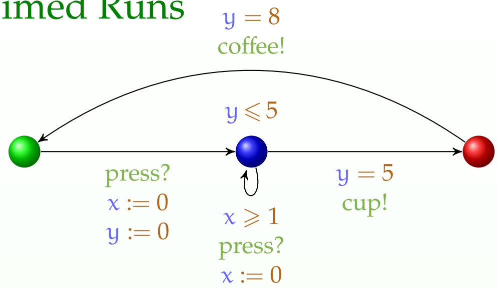
Timed Runs



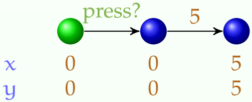
- Examples of timed runs
  - Coffee with no sugar



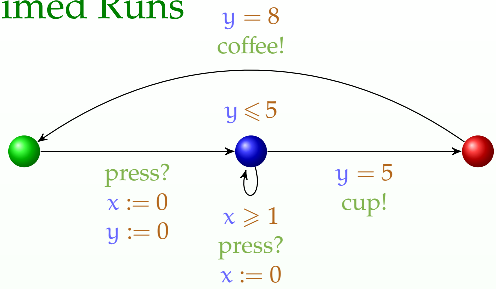
Timed Runs



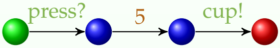
- Examples of timed runs
  - Coffee with no sugar



Timed Runs

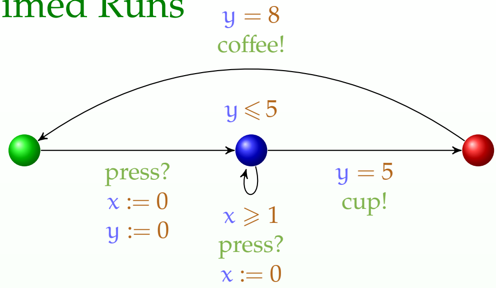


- Examples of timed runs
  - Coffee with no sugar

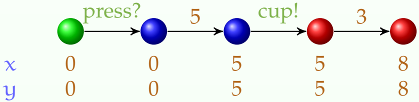


x	0	0	5	5
y	0	0	5	5

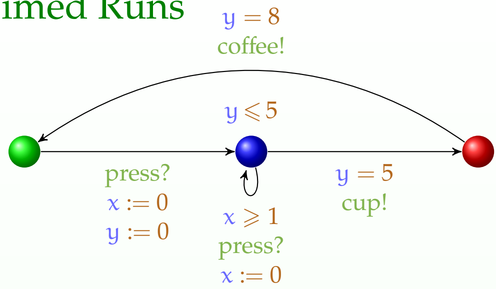
Timed Runs



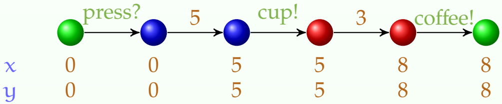
- Examples of timed runs
  - Coffee with no sugar



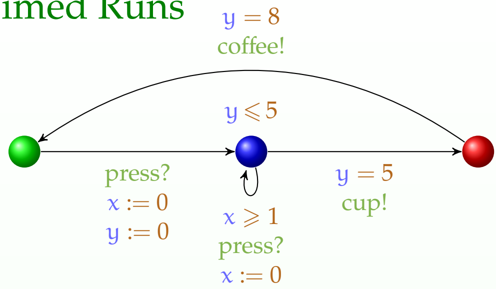
Timed Runs



- Examples of timed runs
  - Coffee with no sugar



Timed Runs



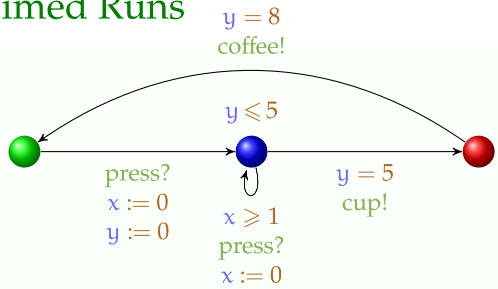
- Examples of timed runs
  - Coffee with no sugar

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

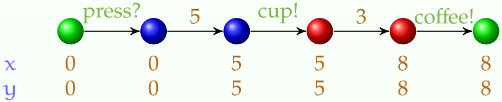
  - Coffee with 2 doses of sugar

x	0
y	0

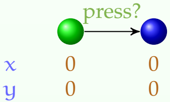
Timed Runs



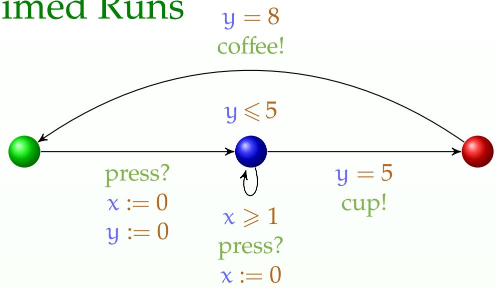
- Examples of timed runs
  - Coffee with no sugar



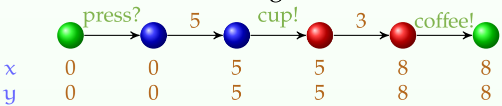
  - Coffee with 2 doses of sugar



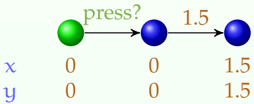
Timed Runs



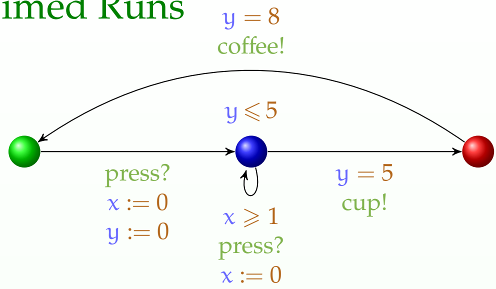
- Examples of timed runs
  - Coffee with no sugar



  - Coffee with 2 doses of sugar



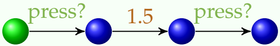
Timed Runs



- Examples of timed runs
  - Coffee with no sugar

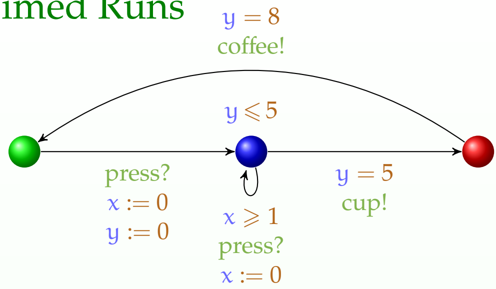
x	0	0	5	5	8	8
y	0	0	5	5	8	8

  - Coffee with 2 doses of sugar

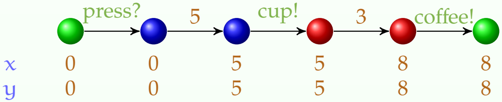


x	0	0	1.5	0
y	0	0	1.5	1.5

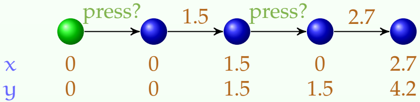
Timed Runs



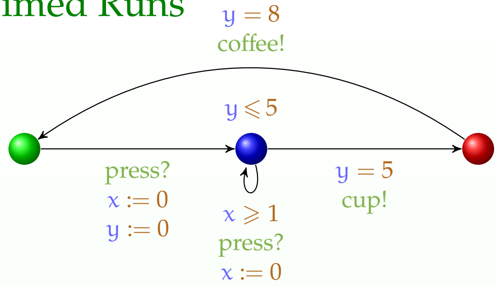
- Examples of timed runs
  - Coffee with no sugar



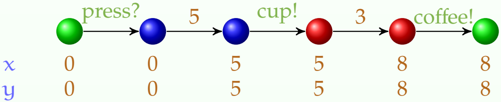
  - Coffee with 2 doses of sugar



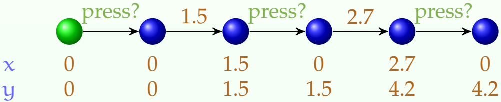
Timed Runs



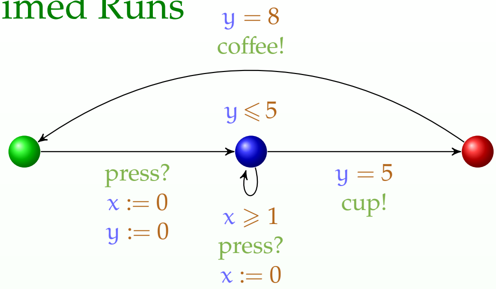
- Examples of timed runs
  - Coffee with no sugar



  - Coffee with 2 doses of sugar



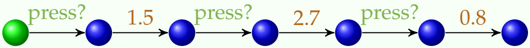
Timed Runs



- Examples of timed runs
  - Coffee with no sugar

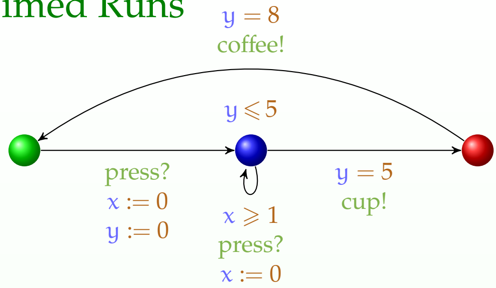
x	0	0	5	5	8	8
y	0	0	5	5	8	8

  - Coffee with 2 doses of sugar



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

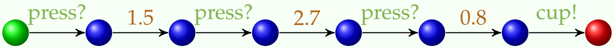
Timed Runs



- Examples of timed runs
  - Coffee with no sugar

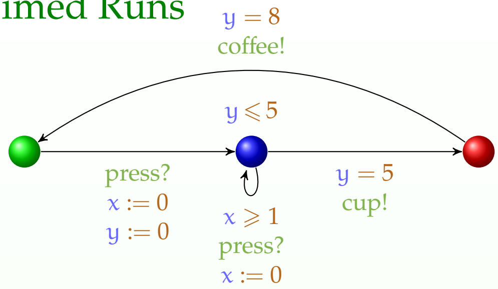
x	0	0	5	5	8	8
y	0	0	5	5	8	8

  - 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 Runs



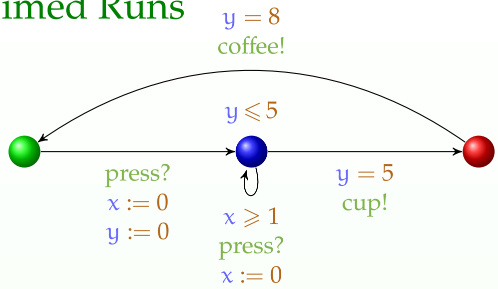
- Examples of timed runs
  - Coffee with no sugar

		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 Runs



- Examples of timed 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	3.8	3.8
y	0	0	1.5	1.5	4.2	4.2	5	5	8	8

Dense Time

- Time is dense: transitions can be taken anytime
  - Infinite number of timed runs
  - Model checking needs a finite structure!
- Some runs are equivalent
  - Taking the press? action at $t = 1.5$ or $t = 1.57$ is equivalent w.r.t. the possible actions
- Idea: reason with abstractions
  - Region automaton [Alur and Dill, 1994]
  - Example: in location , all clock values in the following region are equivalent

$$x \geqslant 1 \land y \leqslant 5 \land x = y$$

  - This abstraction is finite

Timed Temporal Logics

- Specify properties on the order and the delay between events
- Example: TCTL (Timed CTL) [Alur et al., 1993a]
  - "After the first time the button is pressed, a coffee is always eventually delivered within 10 units of time."

Timed Temporal Logics

- Specify properties on the order and the delay between events
- Example: TCTL (Timed CTL) [Alur et al., 1993a]
  - "After the first time the button is pressed, a coffee is always eventually delivered within 10 units of time." ($\surd$)

Timed Temporal Logics

- Specify properties on the order and the delay between events
- Example: TCTL (Timed CTL) [Alur et al., 1993a]
  - "After the first time the button is pressed, a coffee is always eventually delivered within 10 units of time." ($\surd$)
  - "It must never happen that the button can be pressed twice within 1 unit of time."

Timed Temporal Logics

- Specify properties on the order and the delay between events
- Example: TCTL (Timed CTL) [Alur et al., 1993a]
  - “After the first time the button is pressed, a coffee is always eventually delivered within 10 units of time.” ($\surd$)
  - “It must never happen that the button can be pressed twice within 1 unit of time.” ($\times$)

Timed Temporal Logics

- Specify properties on the order and the delay between events
- Example: TCTL (Timed CTL) [Alur et al., 1993a]
  - "After the first time the button is pressed, a coffee is always eventually delivered within 10 units of time." ($\surd$)
  - "It must never happen that the button can be pressed twice within 1 unit of time." ($\times$)
  - "It must never happen that the button can be pressed twice within a time strictly less than 1 unit of time."

Timed Temporal Logics

- Specify properties on the order and the delay between events
- Example: TCTL (Timed CTL) [Alur et al., 1993a]
  - "After the first time the button is pressed, a coffee is always eventually delivered within 10 units of time." ($\surd$)
  - "It must never happen that the button can be pressed twice within 1 unit of time." ($\times$)
  - "It must never happen that the button can be pressed twice within a time strictly less than 1 unit of time." ($\surd$)

Towards a Parametrization...

- Interesting problems
  - Robustness
    - Does the system still behave the same if one of the delays (slightly) changes?
  - Optimization of timing constants
    - Up to which value of the delay between two actions press? can I still order a coffee with 3 doses of sugar?
  - Avoidance of numerous verifications
    - If one of the timing delays of the model changes, should I model check again the whole system?

Towards a Parametrization...

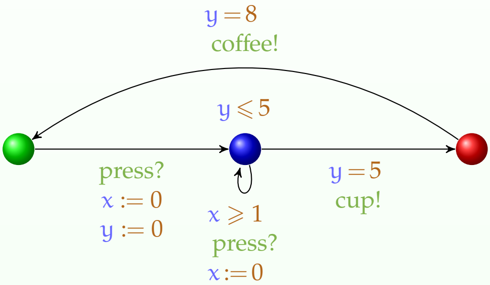
- Interesting problems
  - Robustness
    - Does the system still behave the same if one of the delays (slightly) changes?
  - Optimization of timing constants
    - Up to which value of the delay between two actions press? can I still order a coffee with 3 doses of sugar?
  - Avoidance of numerous verifications
    - If one of the timing delays of the model changes, should I model check again the whole system?

- Idea: reason with parameters (unknown constants)

Outline

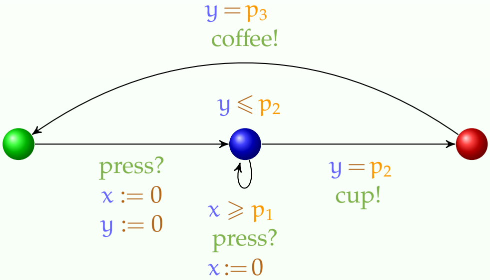
Parametric Timed Automaton

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



Parametric Timed Automaton

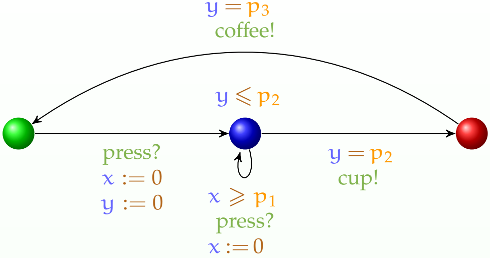
- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993b]
  - Unknown constants used in guards and invariants

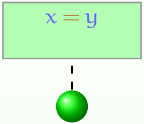


Symbolic Exploration

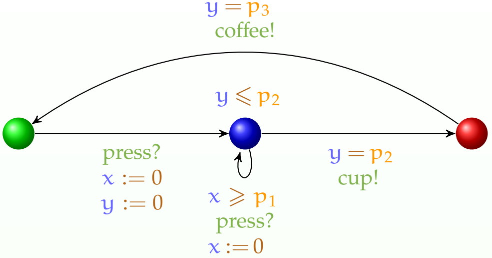
- Iterative exploration of symbolic states
  - Symbolic state: location and constraint on the clocks and parameters

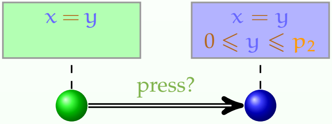
Symbolic Exploration: Coffee Machine



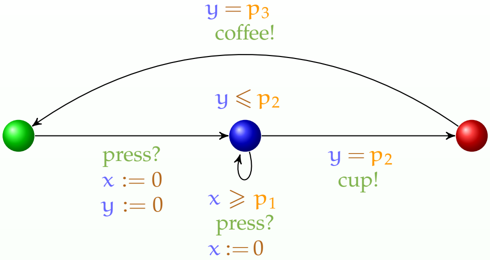


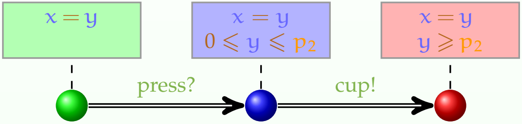
Symbolic Exploration: Coffee Machine



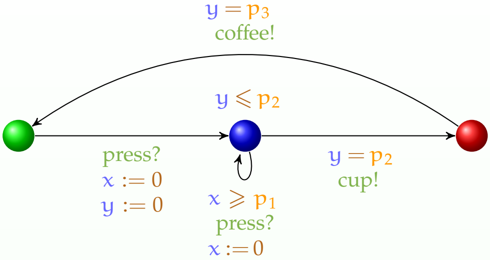


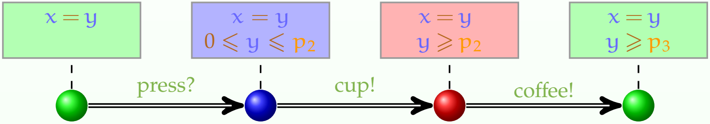
Symbolic Exploration: Coffee Machine



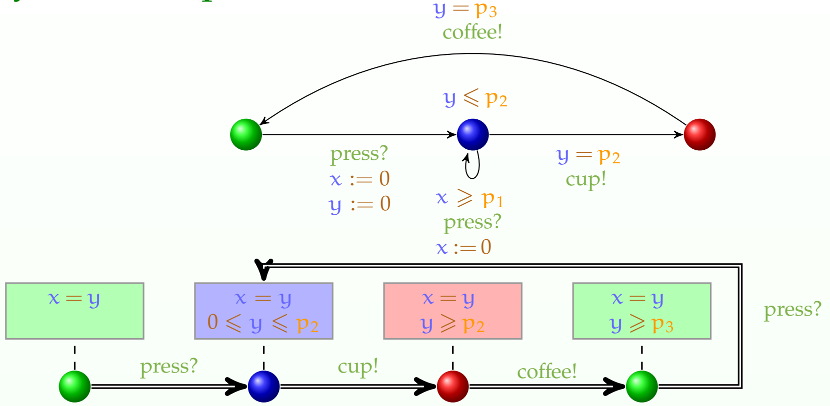


Symbolic Exploration: Coffee Machine

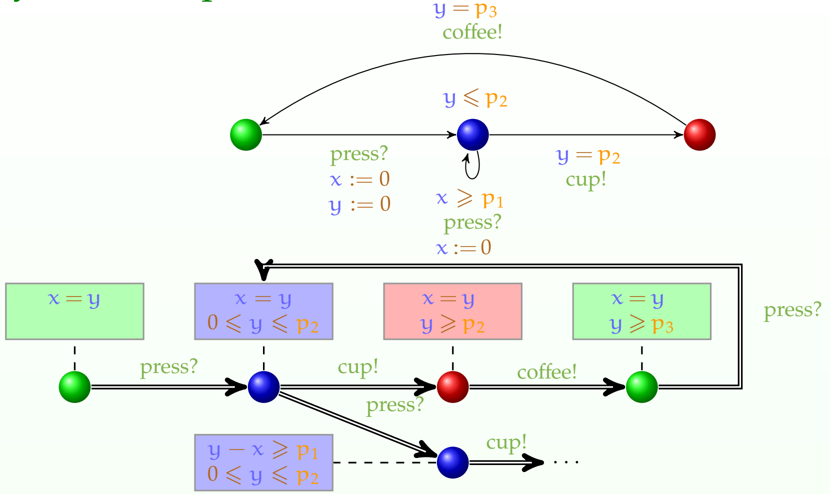




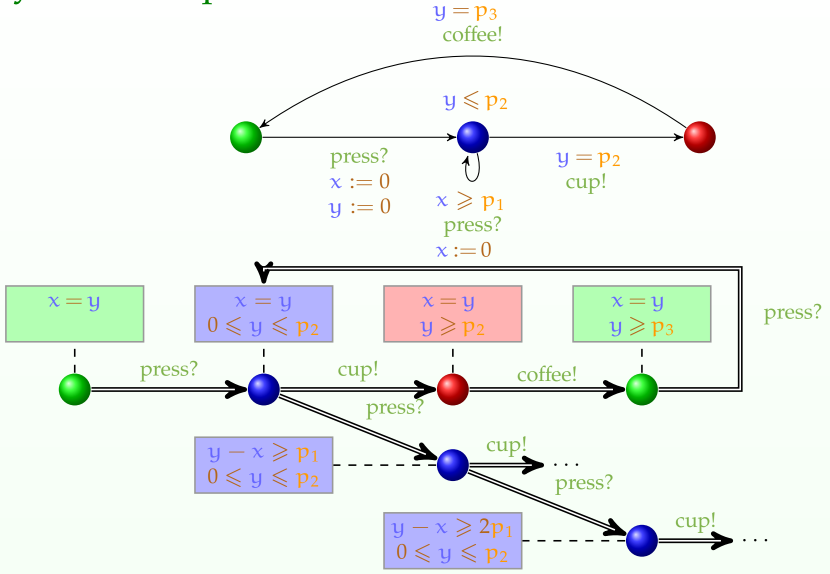
Symbolic Exploration: Coffee Machine



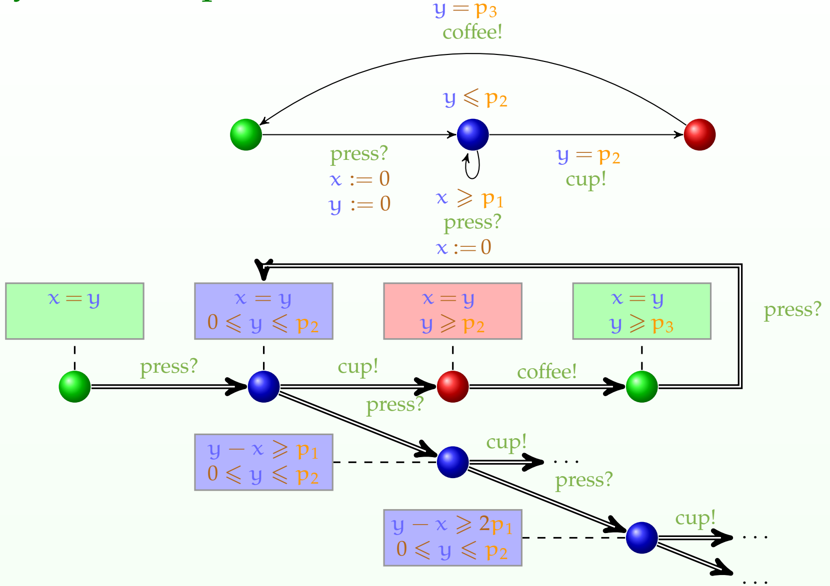
Symbolic Exploration: Coffee Machine



Symbolic Exploration: Coffee Machine



Symbolic Exploration: Coffee Machine



Undecidability

- The symbolic exploration is infinite in general
- No possible abstraction like for Timed Automata

Undecidability

- The symbolic exploration is infinite in general
- No possible abstraction like for Timed Automata

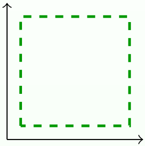
Bad News

(Almost) all interesting problems are undecidable for Parametric Timed Automata.

Outline

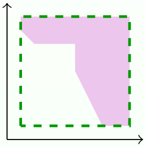
Synthesis of Parameters (1/2)

- The good parameters problem
  - "Given a bounded parameter domain, find a set of parameter valuations of good behavior"



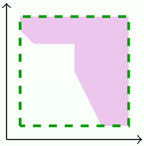
Synthesis of Parameters (1/2)

- The good parameters problem
  - "Given a bounded parameter domain, find a set of parameter valuations of good behavior"



Synthesis of Parameters (1/2)

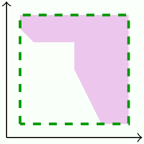
- The good parameters problem
  - "Given a bounded parameter domain, find a set of parameter valuations of good behavior"



- Interesting problem

Synthesis of Parameters (1/2)

- The good parameters problem
  - “Given a bounded parameter domain, find a set of parameter valuations of good behavior”



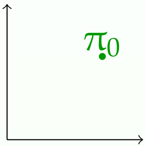
- Interesting problem
  - But undecidable!

Synthesis of Parameters (2/2)

- Possible options to deal with undecidability
  - Semi algorithms
    - Example: Computation of all the reachable states, and intersection with the bad states [Henzinger and Wong-Toi, 1996]
  - Approximations
    - Example: Use of octahedra [Clarisó and Cortadella, 2007]
  - Restrictions to decidable subclasses
    - Example: L/U automata [Hune et al., 2002]

An Inverse Method for PTAs

- Original method for synthesis of parameters [André et al., 2009]
  - "Given a reference parameter valuation π_0, find other valuations around π_0 of same (linear-time) behavior"



Input

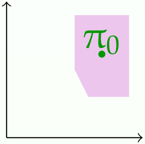
$p_1 = 1$

$p_2 = 5$

$p_3 = 8$

An Inverse Method for PTAs

- Original method for synthesis of parameters [André et al., 2009]
  - "Given a reference parameter valuation π_0, find other valuations around π_0 of same (linear-time) behavior"



Input

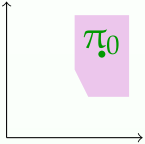
$p_1 = 1$
$p_2 = 5$
$p_3 = 8$

Output

$p_3 \geqslant p_2$
$\wedge\ 6p_1 > p_2$
$\wedge\ p_2 \geqslant 5p_1$

An Inverse Method for PTAs

- Original method for synthesis of parameters [André et al., 2009]
  - "Given a reference parameter valuation π_0, find other valuations around π_0 of same (linear-time) behavior"



Input

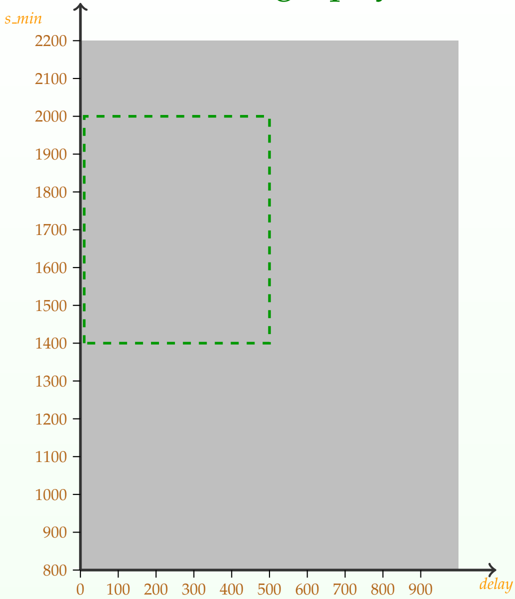
$p_1 = 1$
$p_2 = 5$
$p_3 = 8$

Output

$p_3 \geqslant p_2$
$\wedge\ 6p_1 > p_2$
$\wedge\ p_2 \geqslant 5p_1$

- Properties
  - Semi-algorithm
  - Not complete
- Advantages
  - Exact method
  - Behaves well in practice!

Behavioral Cartography of Timed Automata (1/2)



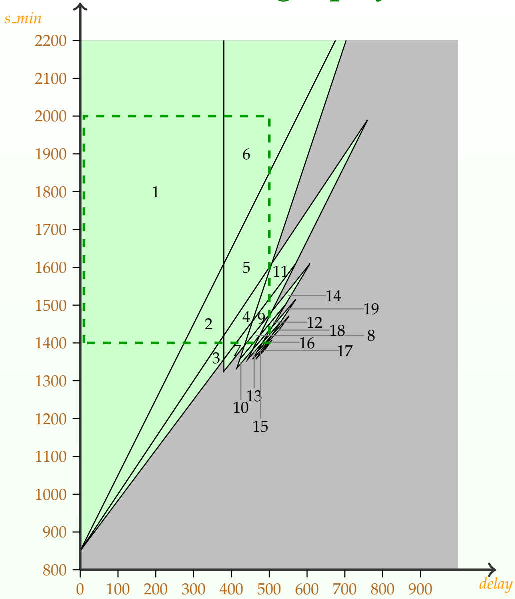
- Idea: repeatedly call the inverse method [André and Fribourg, 2010]

  ⇝ Cover the parametric space with tiles

  - Parametric zones with uniform behavior

Example: Root Contention Protocol

Behavioral Cartography of Timed Automata (1/2)



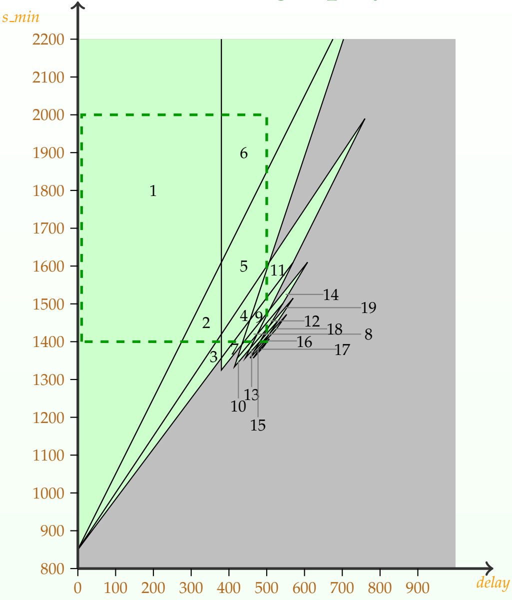
- Idea: repeatedly call the inverse method [André and Fribourg, 2010]

  ⇝ Cover the parametric space with tiles

  - Parametric zones with uniform behavior

Example: Root Contention Protocol

Behavioral Cartography of Timed Automata (1/2)



- Idea: repeatedly call the inverse method [André and Fribourg, 2010]

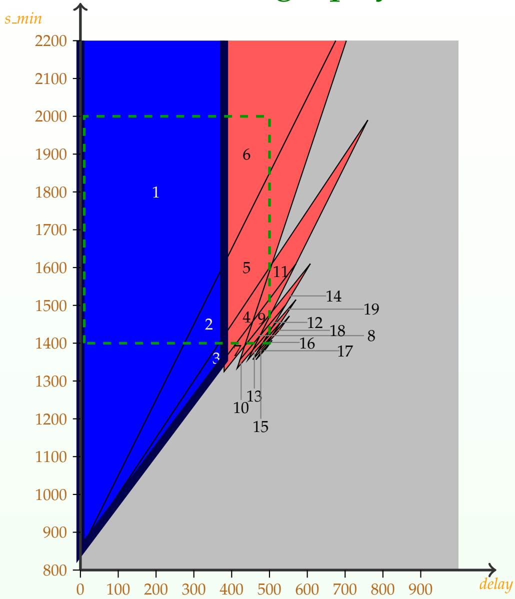
  ⇝ Cover the parametric space with tiles

  - Parametric zones with uniform behavior

- Remarks
  - Tiles may be infinite
  - The cartography may not cover the whole real-valued space

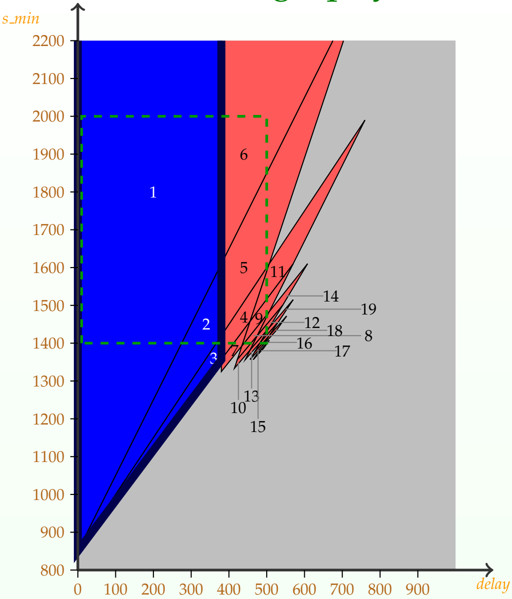
Example: Root Contention Protocol

Behavioral Cartography of Timed Automata (2/2)



- Partition into good and bad subspaces
  - Good if the property is true
  - Bad if the property is false
  - Unknown if not covered by any tile

Behavioral Cartography of Timed Automata (2/2)



- Partition into good and bad subspaces
  - Good if the property is true
  - Bad if the property is false
  - Unknown if not covered by any tile
- Advantages
  - Independent of the property (Only the partition depends on the property)
  - Covers much of the parametric space in practice

Summary

- Synthesis of parameters for real-time concurrent systems important
  - Robustness
  - Optimization of timing constants
  - Avoidance of numerous verifications
- Undecidable in the general case
  - Possible solutions
    - Semi algorithms
    - Approximations
    - Restrictions to decidable subclasses
- Still ongoing work!

References I

Alur, R., Courcoubetis, C., and Dill, D. L. (1993a).
Model-checking in dense real-time.
Information and Computation, 104(1):2–34.

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

Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993b).
Parametric real-time reasoning.
In *STOC '93*, pages 592–601. ACM.

André, É., Chatain, T., Encrenaz, E., and Fribourg, L. (2009).
An inverse method for parametric timed automata.
International Journal of Foundations of Computer Science, 20(5):819–836.

André, É. and Fribourg, L. (2010).
Behavioral cartography of timed automata.
In *RP'10*, volume 6227 of *LNCS*, pages 76–90. Springer.

Clarisó, R. and Cortadella, J. (2007).
The octahedron abstract domain.
Sci. Comput. Program., 64(1):115–139.

References II

Clarke, E. M. and Emerson, E. A. (1982).
Design and synthesis of synchronization skeletons using branching-time temporal logic.
In *Logic of Programs, Workshop*, pages 52–71, London, UK. Springer-Verlag.

Henzinger, T. A. and Wong-Toi, H. (1996).
Using HyTech to synthesize control parameters for a steam boiler.
In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control, LNCS 1165*. Springer-Verlag.

Hune, T., Romijn, J., Stoelinga, M., and Vaandrager, F. (2002).
Linear parametric model checking of timed automata.
Journal of Logic and Algebraic Programming.

Summary of Experiments: *IM*

- Computation times of various case studies
  - Experiments conducted on an Intel Core2 Duo 2.4 GHz with 2 Gb

Example	PTAs	loc./PTA	$\lvert X \rvert$	$\lvert P \rvert$	iter.	$\lvert K_0 \rvert$	states	trans.	Time
SR-latch	3	[3, 8]	3	3	5	2	4	3	0.007
Flip-flop	5	[4, 16]	5	12	9	6	11	10	0.122
And–Or	3	[4, 8]	4	12	14	4	13	13	0.15
Latch circuit	7	[2, 5]	8	13	12	6	18	17	0.345
CSMA/CD	3	[3, 8]	3	3	19	2	219	342	1.01
RCP	5	[6, 11]	6	5	20	2	327	518	2.3
BRP	6	[2, 6]	7	6	30	7	429	474	34
SIMOP	5	[5, 16]	8	7	53	9	1108	1404	67
SPSMALL	28	[2, 11]	28	62	94	45	129	173	461

Summary of Experiments: *BC*

- Computation time for the cartography algorithm
  - Experiments conducted on an Intel Core2 Duo 2.4 GHz with 2 Gb

Example	PTAs	loc./PTA	$\|X\|$	$\|P\|$	$\|V_0\|$	tiles	states	trans.	Time (s)
SR-latch	3	$[3, 8]$	3	3	1331	6	5	4	0.3
Flip-flop	5	$[4, 16]$	5	2	644	8	15	14	3
Latch circuit	7	$[2, 5]$	8	4	73062	5	21	20	96.3
And–Or	3	$[4, 8]$	4	6	75600	4	64	72	118
CSMA/CD	3	$[3, 8]$	3	3	2000	140	349	545	269
RCP	5	$[6, 11]$	6	3	186050	19	5688	9312	7018
SPSMALL	28	$[2, 11]$	28	3	784	213	145	196	31641