TiAMo: Timed Automata Model-checker

P. Bouyer, M. Colange, N. Markey, O. Sankur

April 8, 2016

Motivation (1)

Timed systems

- Models for real-time systems (transportation systems, production lines, networks . . .)
- Need for verification
  - is a task realizable under given real-time constraints?
  - how fast can a task be?
  - . . .

Motivation (2)

We need dedicated algorithms for timed systems

Motivation (2)

We need dedicated algorithms for timed systems

Discrete time is not enough:
(even for reachability) computing the appropriate discrete granularity is undecidable [CHR02]

Motivation (2)

We need dedicated algorithms for timed systems

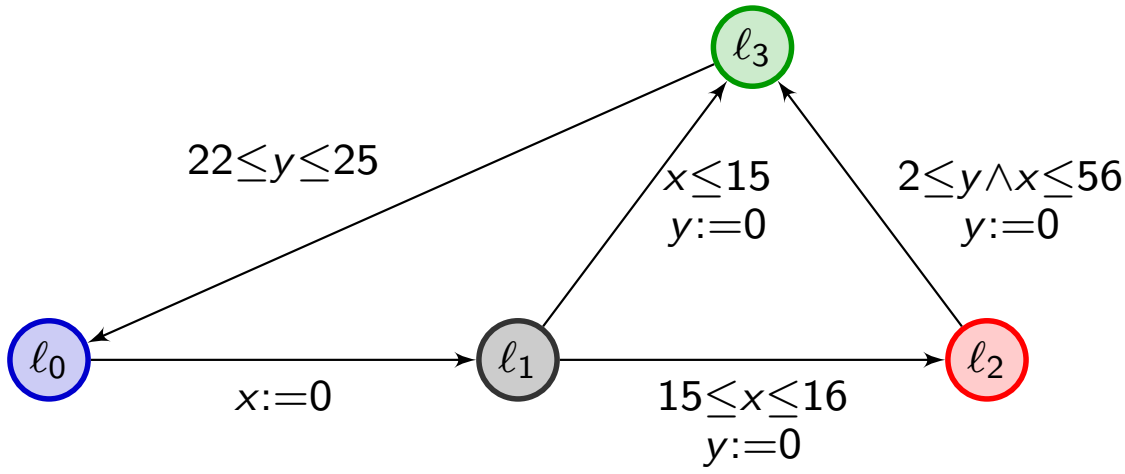
Discrete time is not enough:
(even for reachability) computing the appropriate discrete granularity is undecidable [CHR02]

TiAMo

- a platform for experiments
- assert new algorithms
- compare algorithms

Currently focused on (weighted) reachability

Timed Automata [AD94]



$$
\begin{aligned}
& (\ell_0, x = 0, y = 0) && \Rightarrow_{\text{delay } 0.32} && (\ell_0, x = 0.32, y = 0.32) \\
\Rightarrow & (\ell_1, x = 0, y = 0.32) && \Rightarrow_{\text{delay } 2.64} && (\ell_1, x = 2.64, y = 2.96) \\
\Rightarrow & (\ell_3, x = 2.64, y = 0) && \Rightarrow_{\text{delay } 23.13} && (\ell_1, x = 25.77, y = 23.13) \\
\Rightarrow & (\ell_0, x = 25.77, y = 23.13) && \Rightarrow \ \ldots &&
\end{aligned}
$$

Timed Automata formally

Definition

- Q finite set of locations (or discrete states)
- X finite set of clocks
- $T \subseteq Q \times \mathcal{G} \times 2^X \times Q$ where
  - $\mathcal{G}$ set of guards, i.e. conjunctions of comparisons $(<, \leq, =, \geq, >)$ of a clock with a constant integer

$v \in \mathbb{R}^X_{\geq 0}$: clock valuation
$(\ell, v) \in Q \times \mathbb{R}^X_{\geq 0}$: (extended) state

2 flavors of transitions

- time elapse: delay all clocks by $\delta > 0$
- discrete transition

Reachability Analysis

Question

Is a given location reachable?

Reachability Analysis

Question

Is a given location reachable?

Problems: State space $\subseteq Q \times \mathbb{R}_{\geq 0}^{X}$

Reachability Analysis

Question

Is a given location reachable?

Problems: State space $\subseteq Q \times \mathbb{R}_{\geq 0}^{X}$

- uncountably infinite

Reachability Analysis

Question

Is a given location reachable?

Problems: State space $\subseteq Q \times \mathbb{R}_{\geq 0}^{X}$

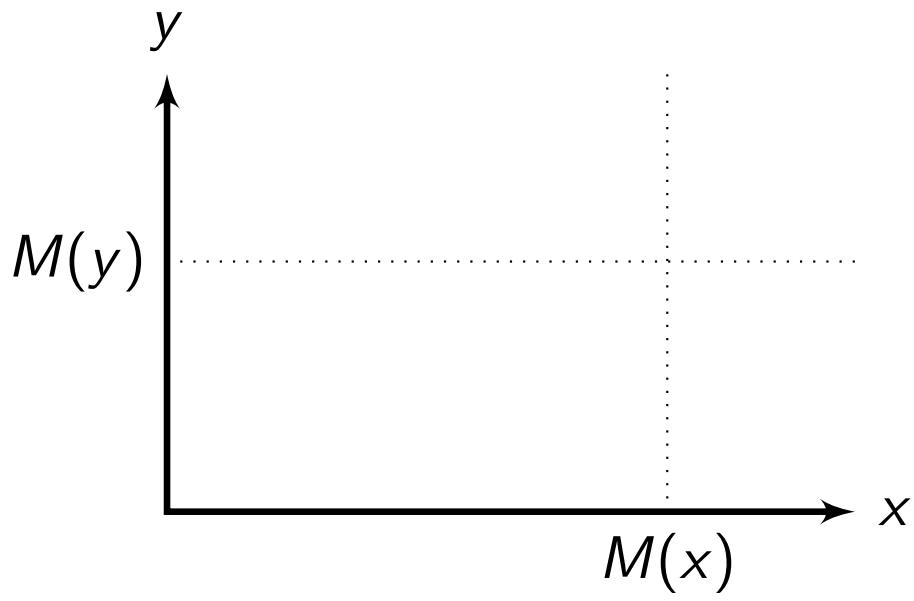
- uncountably infinite
- uncountably infinitely branching

Regions (1)

For a clock x, $M(x) = \max\{k \mid x \bowtie k \text{ in the automaton}\}$

Inactive clock

If $x > M(x)$, its exact value no longer matters: *inactive clock*.



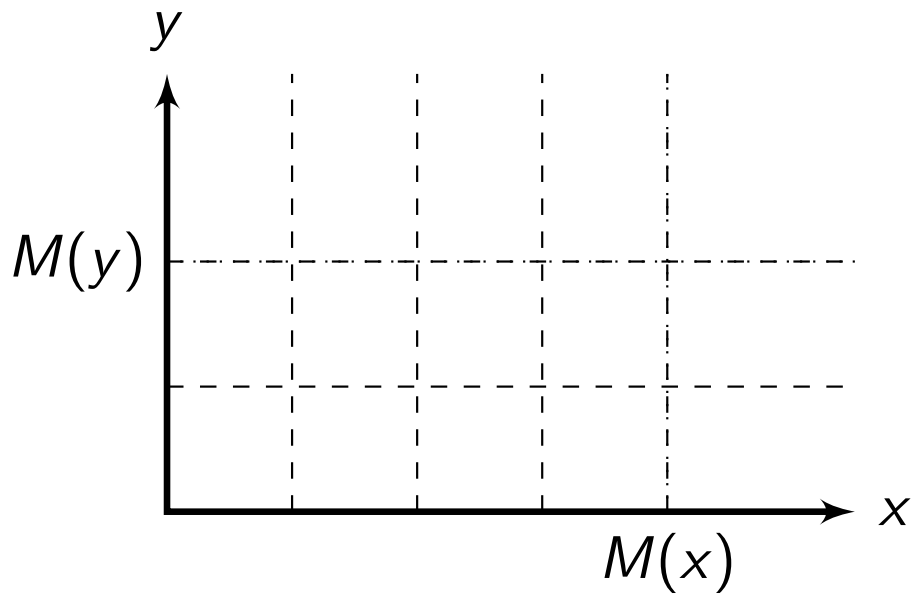
Regions (1)

For a clock x, $M(x) = \max\{k \mid x \bowtie k \text{ in the automaton}\}$

Inactive clock

If $x > M(x)$, its exact value no longer matters: *inactive clock*.

- comparison with integers



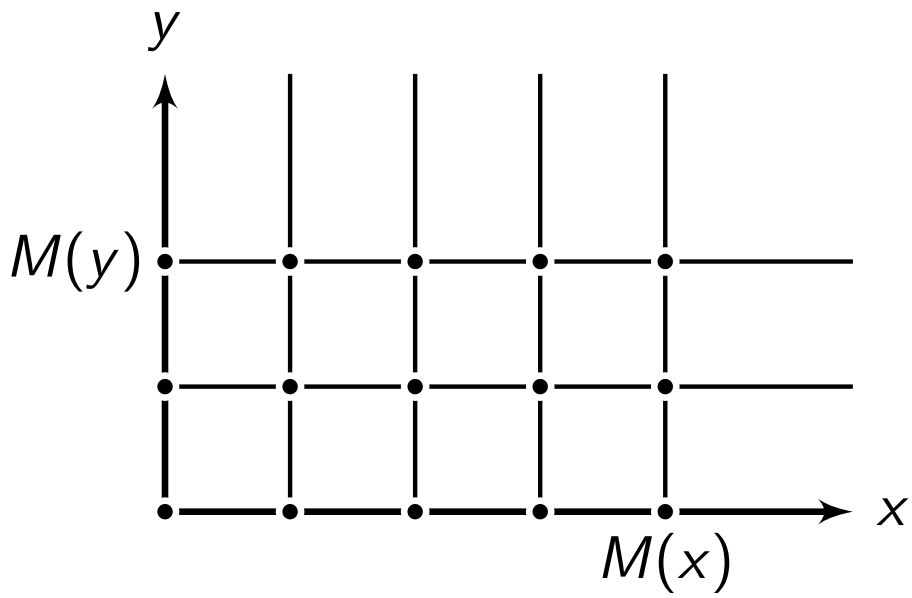
Regions (1)

For a clock x, $M(x) = \max\{k \mid x \bowtie k \text{ in the automaton}\}$

Inactive clock

If $x > M(x)$, its exact value no longer matters: *inactive clock*.

- comparison with integers
- strict/large inequalities



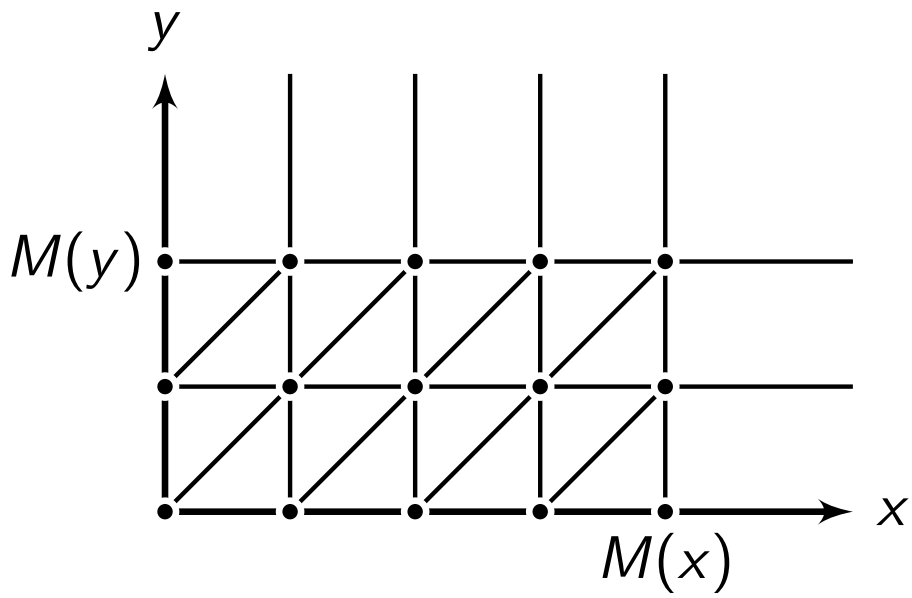
Regions (1)

For a clock x, $M(x) = \max\{k \mid x \bowtie k \text{ in the automaton}\}$

Inactive clock

If $x > M(x)$, its exact value no longer matters: *inactive clock*.

- comparison with integers
- strict/large inequalities
- time elapse



Regions (2)

Time-abstract bisimulation

If v and v' are in the same region, then
same locations reachable from (ℓ, v) and (ℓ, v').

Reachability is decidable [AD94]

- ℓ reachable iff ℓ reachable in the region graph
- the region graph is *finite*

At most $(2M + 2)^{|X|} 2^{|X|} |X|!$ regions
$\Rightarrow$ combinatorial explosion

The general algorithm

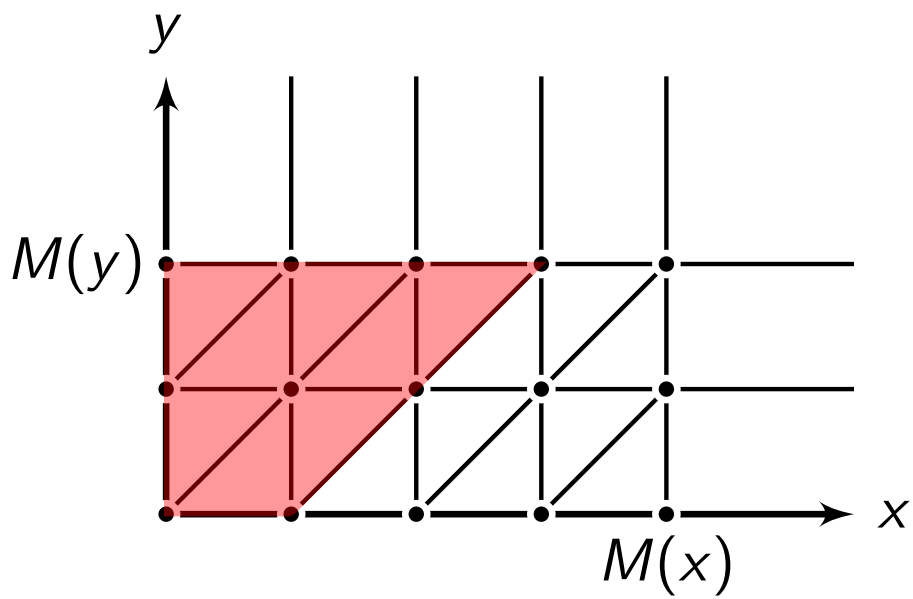
Algorithm 1: Symbolic algorithm for reachability

```
1  PASSED ← ∅
2  WAITING ← {(ℓ0, Z0)}
3  while WAITING ≠ ∅ do
4      select (ℓ, Z) from WAITING
5      if ℓ ∈ Goal then
6          return Reachable
7      if (ℓ, Z) has not been explored yet then
8          add (ℓ, Z) to PASSED
9          add Post(ℓ, Z) to WAITING
10 return Not reachable
```

Zones (1)

The regions are too small.

$$(0 \leq x) \wedge (0 \leq y) \wedge (x \leq y - 1) \wedge (y \leq 2)$$



Zones (2)

Zone [Dil89]

Convex set of valuations, intended to capture many regions at once

- regions are zones
- nice algorithmic representation and manipulation (DBM)
  - the successor of a zone is a zone
  - polynomial-time algorithm for resets, time elapse, guard evaluation, inclusion

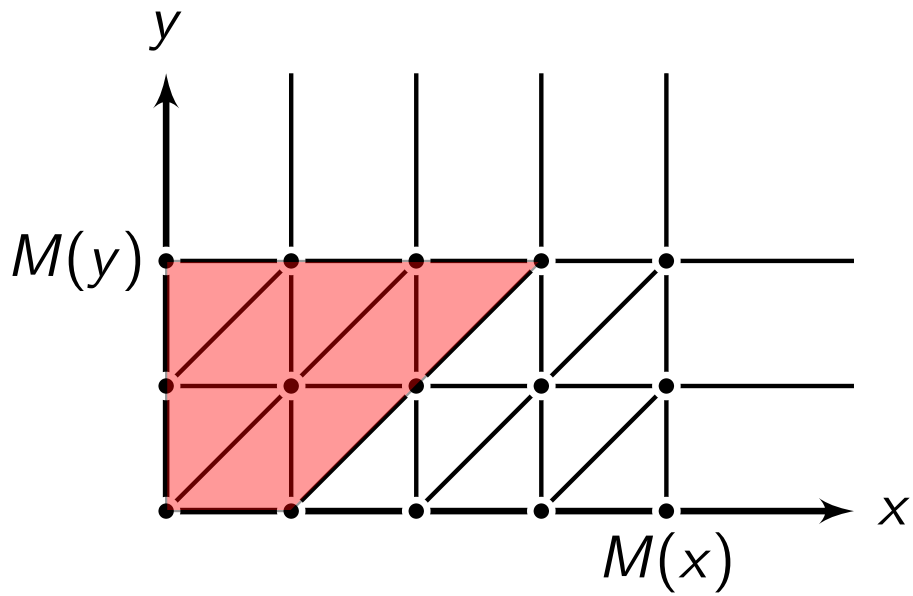
The zone graph is still correct and sound for reachability properties. Implementations use zones (as DBM) instead of regions.

The problem with zones

Drawback of zones

convexity hinders proper representation of inactive clocks

Let the time elapse in the red zone.



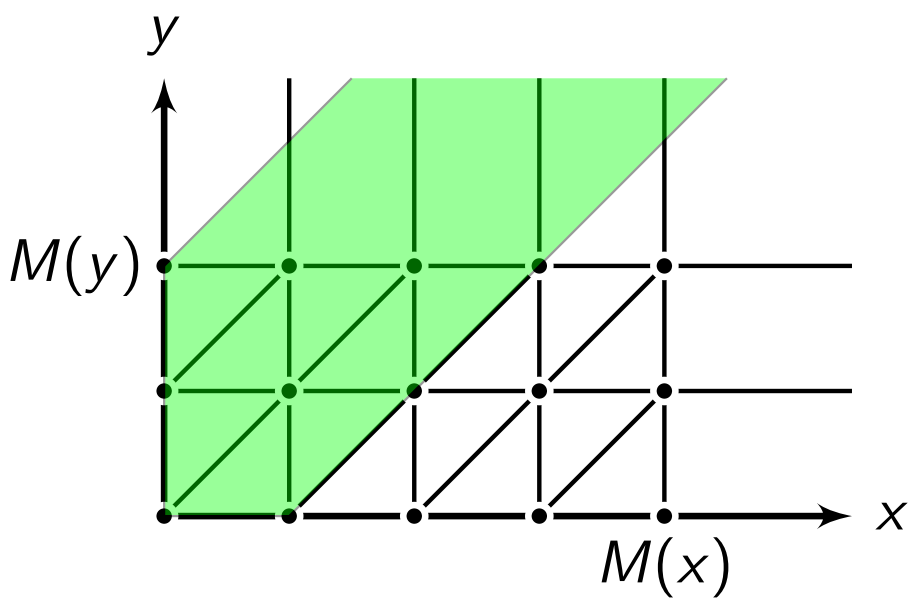
The problem with zones

Drawback of zones

convexity hinders proper representation of inactive clocks

Let the time elapse in the red zone.

- the green zone is not a union of regions



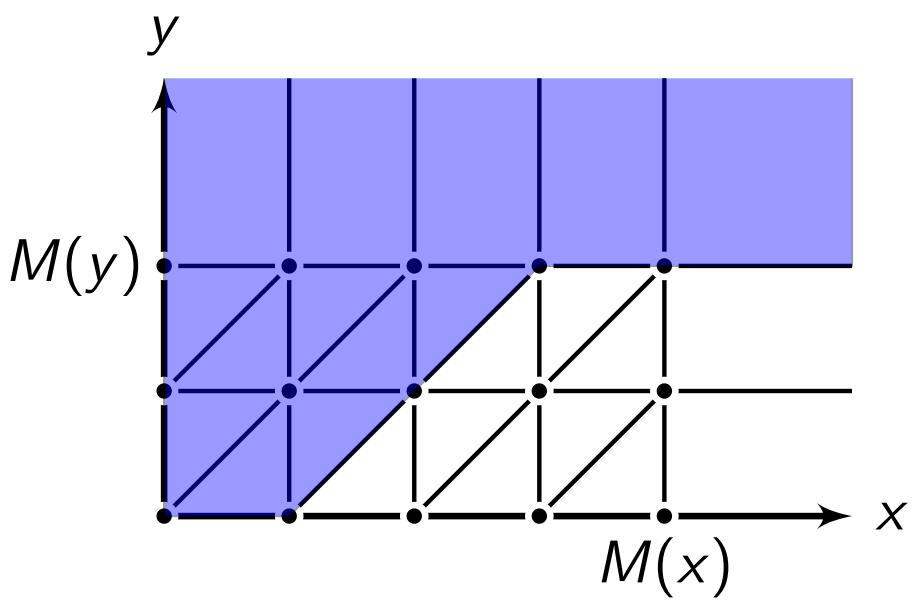
The problem with zones

Drawback of zones

convexity hinders proper representation of inactive clocks

Let the time elapse in the red zone.

- the green zone is not a union of regions
- the blue "zone" is not convex



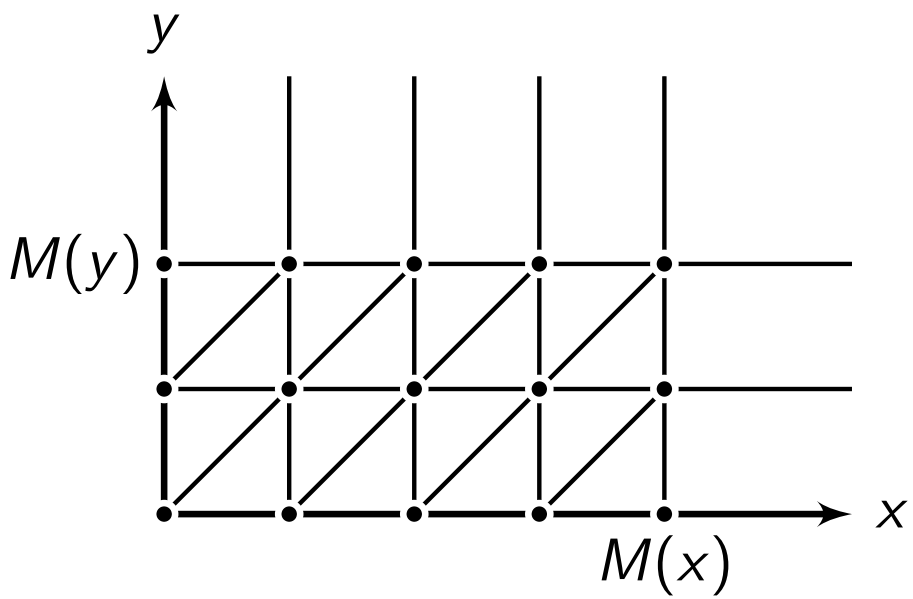
The problem with zones

Drawback of zones

convexity hinders proper representation of inactive clocks

Let the time elapse in the red zone.

- the green zone is not a union of regions
- the blue "zone" is not convex



Problem: the zone graph is no longer finite (!)

Termination: abstractions and comparisons

zone abstraction [DT98]

Enlarge zones, in an attempt to guarantee termination

- try to capture entire regions

Termination: abstractions and comparisons

zone abstraction [DT98]

Enlarge zones, in an attempt to guarantee termination

- try to capture entire regions

Abstractions do not always preserve zones (cf. previous slide)

Termination: abstractions and comparisons

zone abstraction [DT98]

Enlarge zones, in an attempt to guarantee termination

- try to capture entire regions

Abstractions do not always preserve zones (cf. previous slide)

- use weaker, zone-preserving abstractions [BBLP04, BDL$^+$06]

Termination: abstractions and comparisons

zone abstraction [DT98]

Enlarge zones, in an attempt to guarantee termination

- try to capture entire regions

Abstractions do not always preserve zones (cf. previous slide)

- use weaker, zone-preserving abstractions [BBLP04, BDL$^+$06]
- push abstractions in the "already explored?" test

Termination: abstractions and comparisons

zone abstraction [DT98]

Enlarge zones, in an attempt to guarantee termination

- try to capture entire regions

Abstractions do not always preserve zones (cf. previous slide)

- use weaker, zone-preserving abstractions [BBLP04, BDL$^+$06]
- push abstractions in the "already explored?" test

zone comparison

given a newly discovered zone, has it already been explored?

Termination: abstractions and comparisons

zone abstraction [DT98]

Enlarge zones, in an attempt to guarantee termination

- try to capture entire regions

Abstractions do not always preserve zones (cf. previous slide)

- use weaker, zone-preserving abstractions [BBLP04, BDL$^+$06]
- push abstractions in the "already explored?" test

zone comparison

given a newly discovered zone, has it already been explored?

- plain set inclusion

Termination: abstractions and comparisons

zone abstraction [DT98]

Enlarge zones, in an attempt to guarantee termination

- try to capture entire regions

Abstractions do not always preserve zones (cf. previous slide)

- use weaker, zone-preserving abstractions [BBLP04, BDL$^+$06]
- push abstractions in the "already explored?" test

zone comparison

given a newly discovered zone, has it already been explored?

- plain set inclusion
- inclusion of abstractions [HKSW11, HSW12]: $Z \subseteq \mathit{abstr}(Z')$

Termination: abstractions and comparisons

zone abstraction [DT98]

Enlarge zones, in an attempt to guarantee termination

- try to capture entire regions

Abstractions do not always preserve zones (cf. previous slide)

- use weaker, zone-preserving abstractions [BBLP04, BDL$^+$06]
- push abstractions in the "already explored?" test

zone comparison

given a newly discovered zone, has it already been explored?

- plain set inclusion
- inclusion of abstractions [HKSW11, HSW12]: $Z \subseteq \mathit{abstr}(Z')$
  - abstraction is not explicitly build
  - same asymptotic complexity as set inclusion (on DBM)

The parameterized algorithm

Algorithm 2: Symbolic zone algorithm for reachability

Parameter: α `// Abstraction operator`
Parameter: $\preceq$ `// Zone comparison`

```
1  PASSED ← ∅
2  WAITING ← {(ℓ0, Z0)}
3  while WAITING ≠ ∅ do
4      select (ℓ, Z) from WAITING
5      if ℓ ∈ Goal then
6          return Reachable
7      if for all (ℓ, Z') ∈ PASSED, Z ⋠ Z' then
8          add (ℓ, α(Z)) to PASSED
9          add Post(ℓ, Z) to WAITING
10 return Not reachable
```

Termination

Termination ensured by an appropriate combination of

- abstraction
- comparison
- (sometimes) restriction on the input models, e.g. all clocks are upper-bounded

(usually) termination $\Leftarrow$ comparison is well-founded over the set of abstracted zones

Termination

Termination ensured by an appropriate combination of

- abstraction
- comparison
- (sometimes) restriction on the input models, e.g. all clocks are upper-bounded

(usually) termination $\Leftarrow$ comparison is well-founded over the set of abstracted zones

Usually, trade-off between

- computational complexity of abstraction/comparison
- number of zones explored (hence number of abstractions/comparisons performed)

Algorithmic bottleneck

Main bottleneck (from experimental observations)

Number of zone comparisons

Algorithmic bottleneck

Main bottleneck (from experimental observations)

Number of zone comparisons

- merge PASSED and WAITING [BBD+02]
  already explored states are NOT added to WAITING

Algorithmic bottleneck

Main bottleneck (from experimental observations)

Number of zone comparisons

- merge PASSED and WAITING [BBD+02]
  already explored states are NOT added to WAITING
- order of exploration matters:
  BFS (larger zones) *often* better than DFS (smaller zones)

Algorithmic bottleneck

Main bottleneck (from experimental observations)

Number of zone comparisons

- merge PASSED and WAITING [BBD+02]
  already explored states are NOT added to WAITING
- order of exploration matters:
  BFS (larger zones) *often* better than DFS (smaller zones)
- smart exploration [HT15]:
  if s subsumes s', the successors of s subsume those of s'
  if s subsumes s', remove s' and its successors from WAITING

Reachability with weights

Add non-negative weights to the timed automata model

- spends t time in location ℓ: weight $t \times \mathit{weight}(\ell)$
- take edge e: weight $\mathit{weight}(e)$

Optimal reachability

Given a location, what is the smallest weight to reach it?

Similar algorithms and problems:

Reachability with weights

Add non-negative weights to the timed automata model

- spends t time in location ℓ: weight $t \times \mathit{weight}(\ell)$
- take edge e: weight $\mathit{weight}(e)$

Optimal reachability

Given a location, what is the smallest weight to reach it?

Similar algorithms and problems:

- priced regions to show decidability [BFH$^+$01]

Reachability with weights

Add non-negative weights to the timed automata model

- spends t time in location ℓ: weight $t \times \mathit{weight}(\ell)$
- take edge e: weight $\mathit{weight}(e)$

Optimal reachability

Given a location, what is the smallest weight to reach it?

Similar algorithms and problems:

- priced regions to show decidability [BFH$^+$01]
- priced zones with nice algorithmic properties [LBB$^+$01]

Reachability with weights

Add non-negative weights to the timed automata model

- spends t time in location ℓ: weight $t \times \mathit{weight}(\ell)$
- take edge e: weight $\mathit{weight}(e)$

Optimal reachability

Given a location, what is the smallest weight to reach it?

Similar algorithms and problems:

- priced regions to show decidability [BFH$^+$01]
- priced zones with nice algorithmic properties [LBB$^+$01]
- abstraction/comparisons [BCM16]

To sum up

A symbolic algorithm for reachability analysis, parameterized by:

To sum up

A symbolic algorithm for reachability analysis, parameterized by:

- zone abstraction

To sum up

A symbolic algorithm for reachability analysis, parameterized by:

- zone abstraction
- zone comparison

To sum up

A symbolic algorithm for reachability analysis, parameterized by:

- zone abstraction
- zone comparison
- order of exploration

To sum up

A symbolic algorithm for reachability analysis, parameterized by:

- zone abstraction
- zone comparison
- order of exploration
- weighted vs. unweighted

To sum up

A symbolic algorithm for reachability analysis, parameterized by:

- zone abstraction
- zone comparison
- order of exploration
- weighted vs. unweighted

Non-trivial interaction between parameters
Efficiency results from a trade-off between parameters

A new tool, why?

Many tools out there

A new tool, why?

Many tools out there

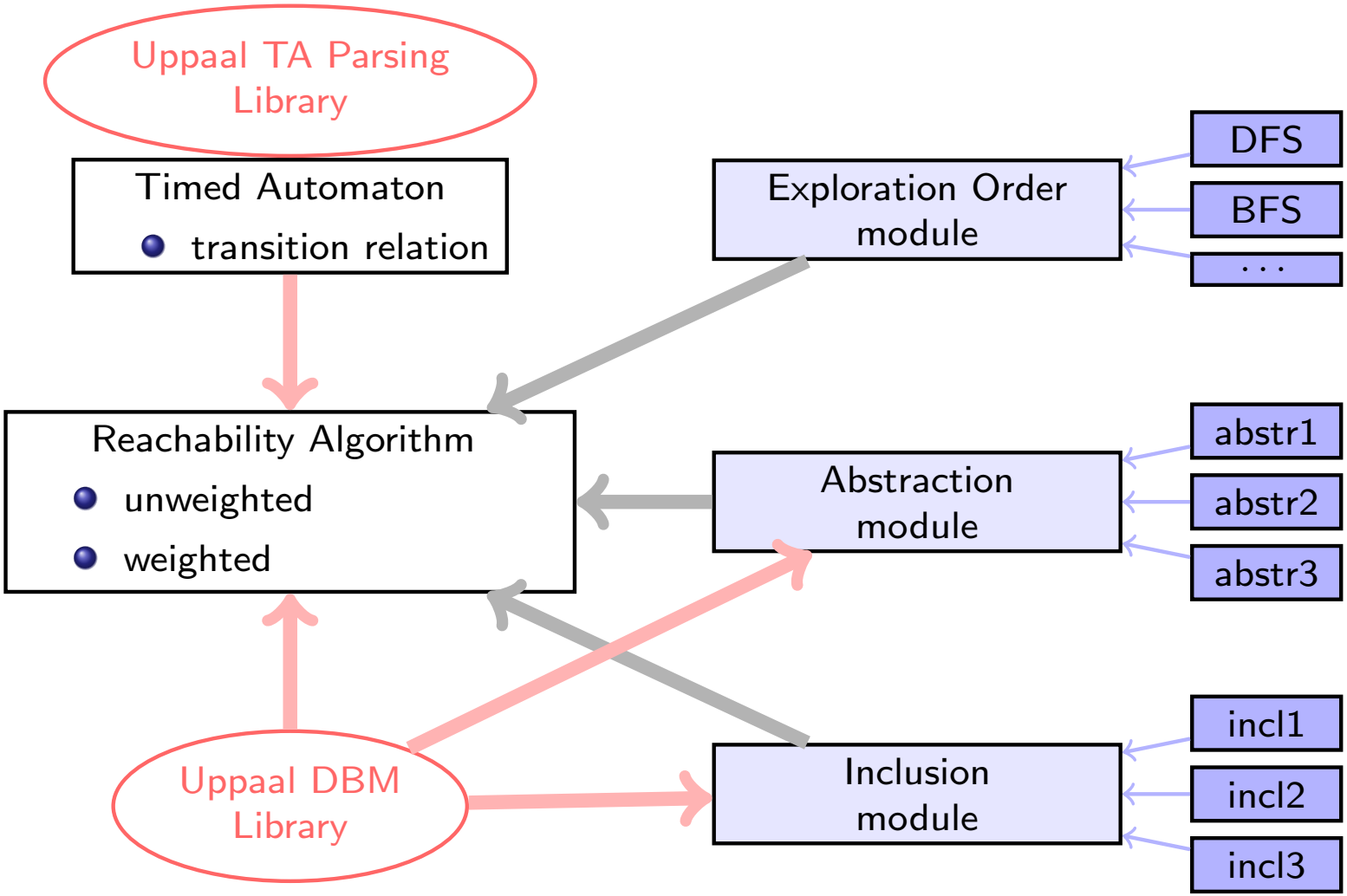
- most of them are no longer active

A new tool, why?

Many tools out there

- most of them are no longer active
- source code access problems
  - it is not always clear what is exactly implemented
  - not suitable for rigorous comparisons (change one parameter at a time)

TiAMo architecture



What is implemented

Exploration strategies

- BFS, DFS
- best cost first (for weighted models)
- preference-based (use a special "preference" variable in the model)
- "smart": approximate implementation of [HT15]

What is implemented

Exploration strategies

- BFS, DFS
- best cost first (for weighted models)
- preference-based (use a special "preference" variable in the model)
- "smart": approximate implementation of [HT15]

Abstractions

- identity (i.e. no abstraction), to be used with abstract inclusion tests
- LU abstraction [BBLP04]

What is implemented

Exploration strategies

- BFS, DFS
- best cost first (for weighted models)
- preference-based (use a special "preference" variable in the model)
- "smart": approximate implementation of [HT15]

Abstractions

- identity (i.e. no abstraction), to be used with abstract inclusion tests
- LU abstraction [BBLP04]

Inclusions

- set inclusion
- abstract inclusion [HKSW11, HSW12]
- weighted set inclusion [RLS06]
- abstract weighted inclusion [BCM16]

A concrete application

We designed a new abstract inclusion for weighted zones [BCM16]

Experimental comparison with state-of-the-art algorithm with TiAMo

- various metrics
  - wall-clock time
  - number of symbolic states explored
  - number of comparisons done
- in various contexts
  - with different exploration strategies

What's next?

Towards Games

Reachability games (both weighted and unweighted)

- controller synthesis

Models Database

- collect models
- format issues

Thank you!

https://git.lsv.fr/colange/tiamo

Licensed under GPL

Any questions?

References I

Rajeev Alur and David L. Dill.

A theory of timed automata.

Theoretical Computer Science, 126(2):183–235, 1994.

Gerd Behrmann, Johan Bengtsson, Alexandre David, Kim G. Larsen, Paul Pettersson, and Wang Yi.

UPPAAL implementation secrets.

In *Proc. of 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems*, 2002.

Gerd Behrmann, Patricia Bouyer, Kim G. Larsen, and Radek Pelánek.

Lower and upper bounds in zone based abstractions of timed automata.

In *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, volume 2988 of *Lecture Notes in Computer Science*, pages 312–326. Springer, 2004.

Patricia Bouyer, Maximilien Colange, and Nicolas Markey.

Symbolic optimal reachability in weighted timed automata.

CoRR, abs/1602.00481, 2016.

Gerd Behrmann, Alexandre David, Kim G. Larsen, John Hakansson, Paul Petterson, Wang Yi, and Martijn Hendriks.

Uppaal 4.0.

In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems*, QEST '06, pages 125–126, Washington, DC, USA, 2006. IEEE Computer Society.

Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager.

Minimum-cost reachability for priced timed automata.

In *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

References II

Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin.

A comparison of control problems for timed and hybrid systems.
In Claire Tomlin and Mark R. Greenstreet, editors, *Hybrid Systems: Computation and Control, 5th International Workshop, HSCC 2002, Stanford, CA, USA, March 25-27, 2002, Proceedings*, volume 2289 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2002.

David L. Dill.

Timing assumptions and verification of finite-state concurrent systems.
In *Proc. of the Workshop on Automatic Verification Methods for Finite State Systems (1989)*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.

Conrado Daws and Stavros Tripakis.

Model-checking of real-time reachability properties using abstractions.
In *Proc. 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98)*, volume 1384 of *Lecture Notes in Computer Science*, pages 313–329. Springer, 1998.

Frédéric Herbreteau, Dileep Kini, B Srivathsan, and Igor Walukiewicz.

Using non-convex approximations for efficient analysis of timed automata.
arXiv preprint arXiv:1110.3704, 2011.

Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz.

Better abstractions for timed automata.
In *Proc. 27th Annual Symposium on Logic in Computer Science (LICS'12)*, pages 375–384. IEEE Computer Society Press, 2012.

Frédéric Herbreteau and Thanh-Tung Tran.

Improving search order for reachability testing in timed automata.
In *Formal Modeling and Analysis of Timed Systems - 13th International Conference, FORMATS 2015*, pages 124–139, 2015.

References III

Kim Larsen, Gerd Behrmann, Ed Brinksma, Ansgar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn.
As cheap as possible: Effcient cost-optimal reachability for priced timed automata.
In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 493–505. Springer Berlin Heidelberg, 2001.

Jacob I. Rasmussen, Kim G. Larsen, and K. Subramani.
On using priced timed automata to achieve optimal scheduling.
Formal Methods in System Design, 29(1):97–114, 2006.