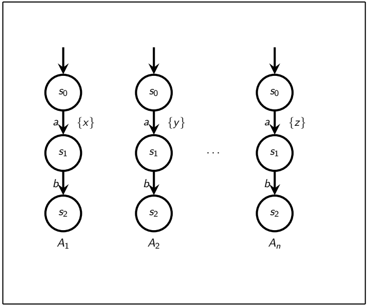
Partial Order Reduction for Timed Systems

Frédéric Herbreteau[⋆] B Srivathsan[⋆] Igor Walukiewicz[⋆]
Govind R[⋆⋆]

LaBRI, Université de Bordeaux
Chennai Mathematical Institute

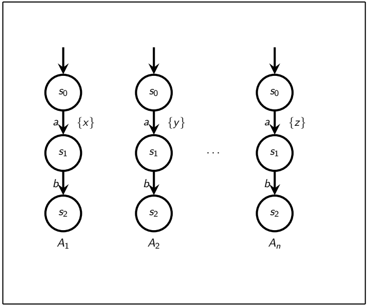
10 July 2021
Formal Methods Update Meeting 2021

Networks of Timed automata: used to model several safety-critical systems.



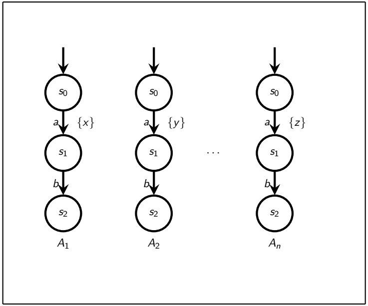
Networks of Timed automata: used to model several safety-critical systems.

State-space explosion is a major challenge in their verification!



Networks of Timed automata: used to model several safety-critical systems.

State-space explosion is a major challenge in their verification!



Our work

Algorithms for networks of timed automata motivated by **partial order reduction**.

Overview of the talk

Partial order reduction

Timed automata and our problem

Local time semantics

Local zone graph

Solution 1

Local time semantics and aggregated zones

Sync-subsumption

Interleavings in local sync graphs

Solution 2

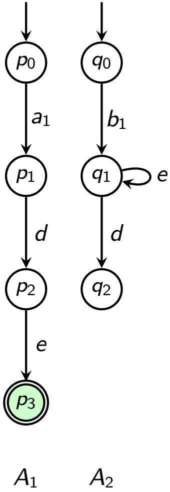
Local time semantics and POR

Local sync graphs + POR ?

No finite subsumption for LZG that allows POR

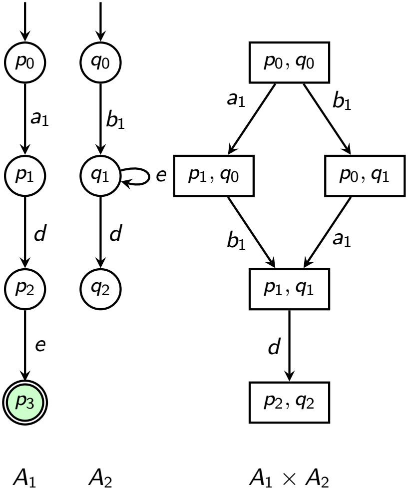
POR for spread-bounded systems

Reachability in networks of automata



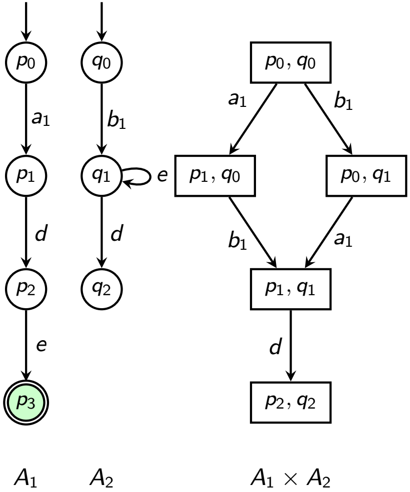
Synchronises on joint actions

Reachability in networks of automata



Synchronises on joint actions

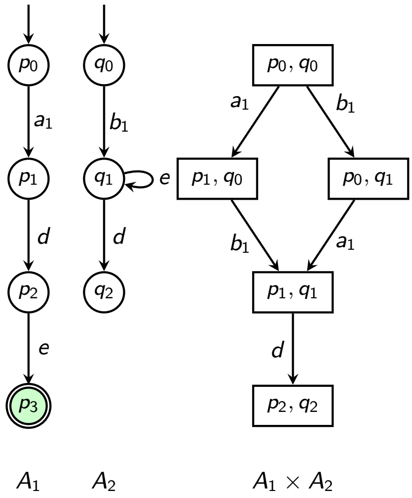
Reachability in networks of automata



State space explosion
due to multiple interleavings
of same actions

Synchronises on joint actions

Reachability in networks of automata

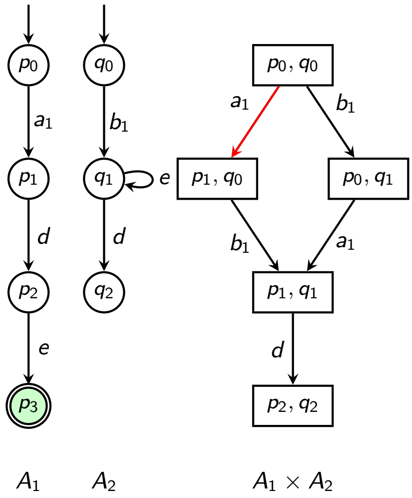


State space explosion
due to multiple interleavings
of same actions

Can we do better than
exhaustively explore $A_1 \times A_2$?

Synchronises on joint actions

Reachability in networks of automata

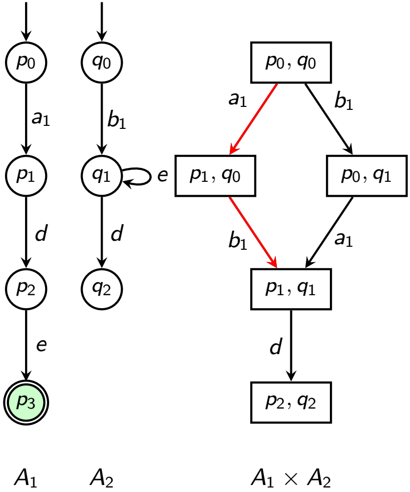


Synchronises on joint actions

State space explosion
due to multiple interleavings
of same actions

Can we do better than
exhaustively explore $A_1 \times A_2$?

Reachability in networks of automata

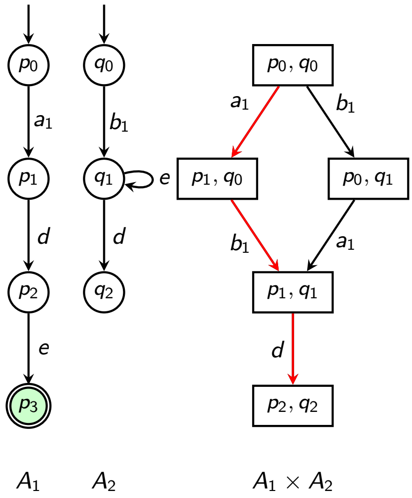


State space explosion
due to multiple interleavings
of same actions

Can we do better than
exhaustively explore $A_1 \times A_2$?

Synchronises on joint actions

Reachability in networks of automata

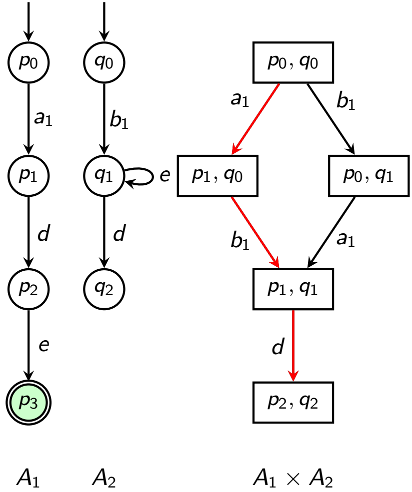


State space explosion
due to multiple interleavings
of same actions

Can we do better than
exhaustively explore $A_1 \times A_2$?

Synchronises on joint actions

Reachability in networks of automata



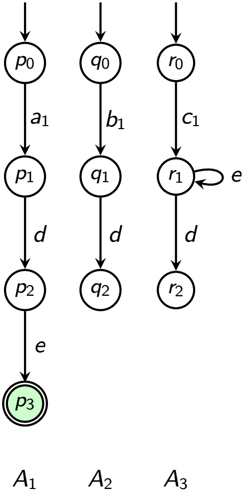
State space explosion
due to multiple interleavings
of same actions

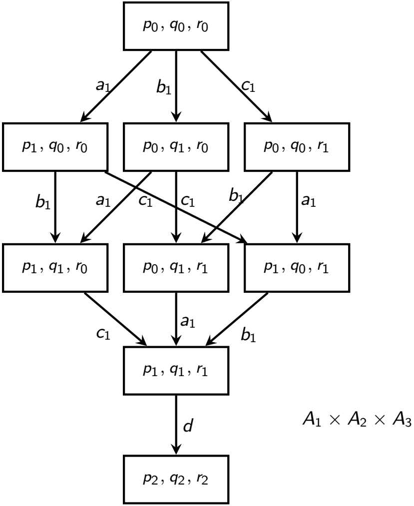
Can we do better than
exhaustively explore $A_1 \times A_2$?

Sufficient to explore a
part of $A_1 \times A_2$

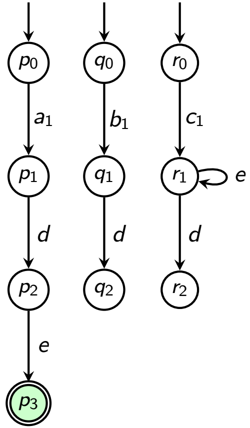
Synchronises on joint actions

Reachability in networks of automata

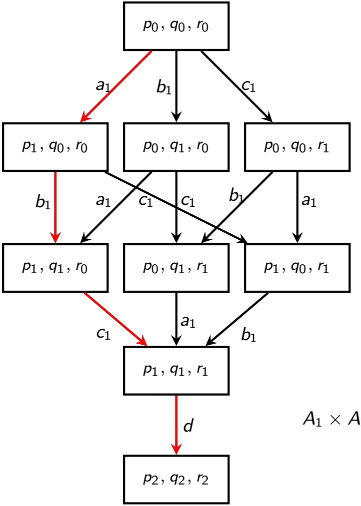




Reachability in networks of automata



A_1 A_2 A_3



$A_1 \times A_2 \times A_3$

Reachability in networks of automata

The graphs occuring in practice are very large, in general!

Reachability in networks of automata

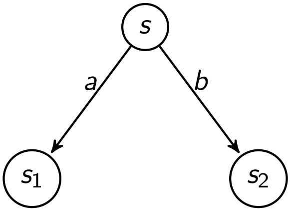
The graphs occuring in practice are very large, in general!

A network with 10 processes
5 states per process $\quad \rightarrow \quad 5^{10} \approx 10$ million states.

Partial Order Reduction (POR)

Partial Order Reduction (POR)

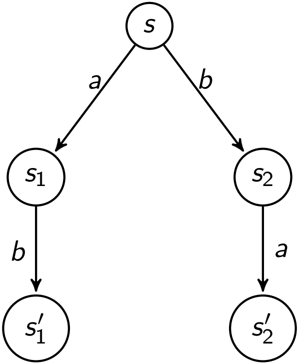
Actions a and b are *independent*, if for every state s from which they are enabled:



Partial Order Reduction (POR)

Actions a and b are *independent*, if for every state s from which they are enabled:

Forward Diamond : If $s \xrightarrow{a} s_1$ and $s \xrightarrow{b} s_2$, then b is enabled from s_1 and a is enabled from s_2.

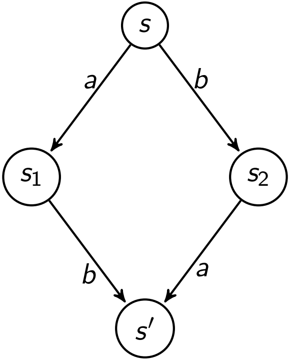


Partial Order Reduction (POR)

Actions a and b are *independent*, if for every state s from which they are enabled:

Forward Diamond : If $s \xrightarrow{a} s_1$ and $s \xrightarrow{b} s_2$, then b is enabled from s_1 and a is enabled from s_2.

Diamond : If $s \xrightarrow{ab} s'$, then $s \xrightarrow{ba} s'$.

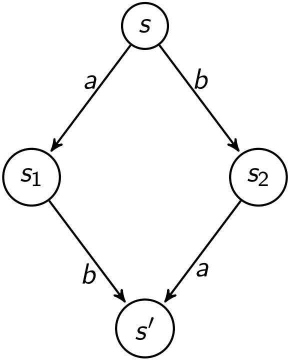


Partial Order Reduction (POR)

Actions a and b are *independent*, if for every state s from which they are enabled:

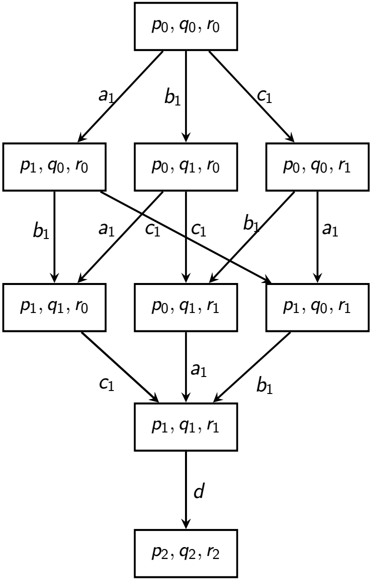
Forward Diamond : If $s \xrightarrow{a} s_1$ and $s \xrightarrow{b} s_2$, then b is enabled from s_1 and a is enabled from s_2.

Diamond : If $s \xrightarrow{ab} s'$, then $s \xrightarrow{ba} s'$.



Actions a, b, are independent if they belong to different processes.

Partial Order Reduction (POR)



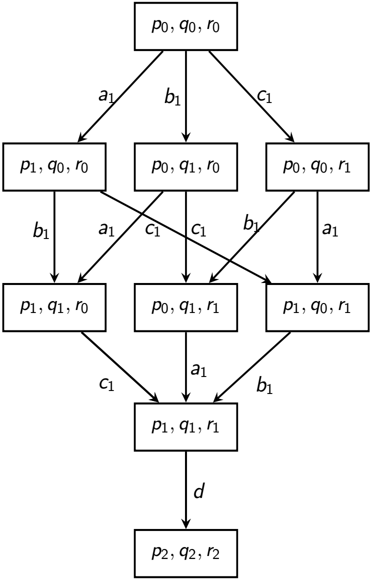
Partial Order Reduction (POR)

Equivalence relation between paths

$u \sim w$ - when w can be obtained from u by permuting adjacent independent actions.

$a_1 b_1 c_1 d$

$b_1 a_1 c_1 d$



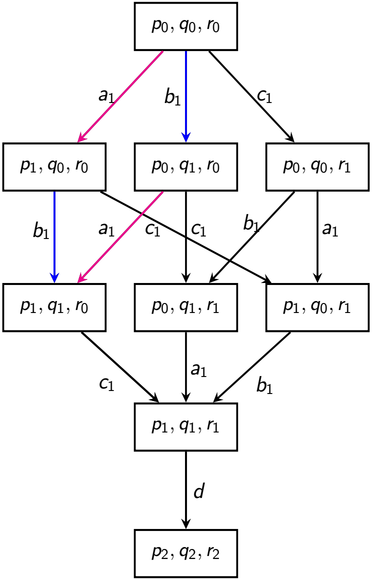
Partial Order Reduction (POR)

Equivalence relation between paths

$u \sim w$ - when w can be obtained from u by permuting adjacent independent actions.

$a_1 b_1 c_1 d$

$b_1 a_1 c_1 d$



Partial Order Reduction (POR)

Equivalence relation between paths

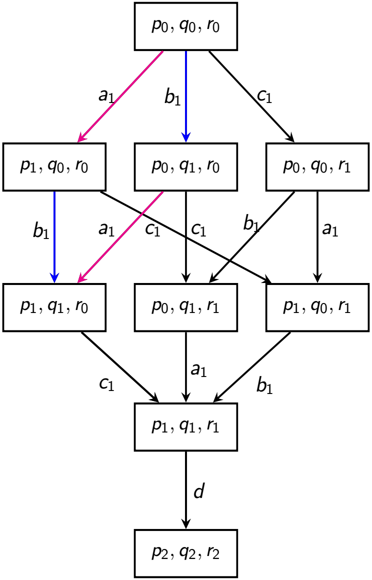
$u \sim w$ - when w can be obtained from u by permuting adjacent independent actions.

$$a_1 b_1 c_1 d$$

$$b_1 a_1 c_1 d$$

Strategy

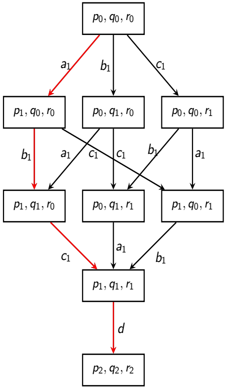
- Classify paths into equivalence classes and explore only one path from each equivalence class.
- Avoid exploring multiple interleavings of independent actions.



Partial Order Reduction (POR)

Goal

To compute a subset of successors from each state seen during exploration.



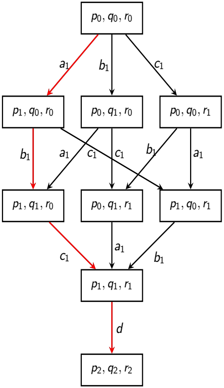
Partial Order Reduction (POR)

Goal

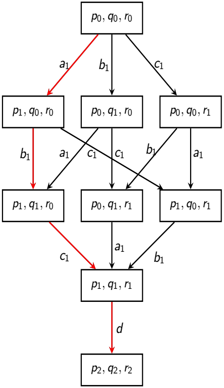
To compute a subset of successors from each state seen during exploration.

Challenges

- Completeness.
- Computing subsets should be "much more efficient" than testing reachability.
- Subset should be as small as possible.



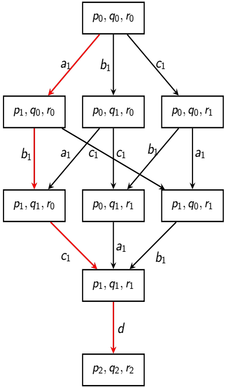
Partial Order Reduction (POR)



Partial order reduction methods

- Extensively studied.
- Still an active research field.
- Several known variants.

Partial Order Reduction (POR)



Partial order reduction methods

- Extensively studied.
- Still an active research field.
- Several known variants.

Variants

- Stubborn sets [Valmari '89]
- Ample sets [Godefroid '90]
- Persistent sets [Peled '93]
- Source sets [Abdulla et al. '16]

Similar ideas, but differ in the choice of the subset of actions to be played.

Overview of the talk

✓ Partial order reduction

Timed automata and our problem

Local time semantics

Local zone graph

Solution 1

Local time semantics and aggregated zones

Sync-subsumption

Interleavings in local sync graphs

Solution 2

Local time semantics and POR

Local sync graphs + POR ?

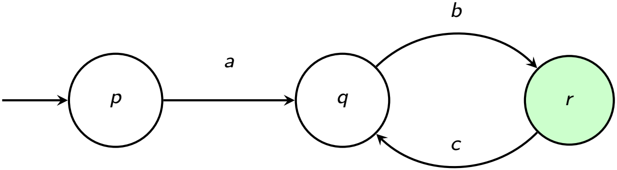
No finite subsumption for LZG that allows POR

POR for spread-bounded systems

Our focus

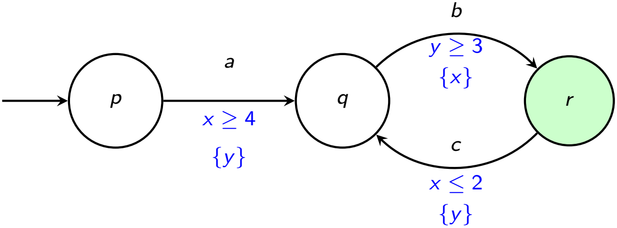
Reachability in networks of timed automata

Timed automaton [Alur Dill '94]



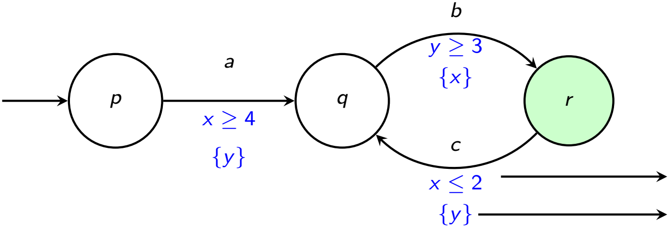
Timed automaton [Alur Dill '94]

Clocks: $X = \{x, y\}$



Timed automaton [Alur Dill '94]

Clocks: $X = \{x, y\}$

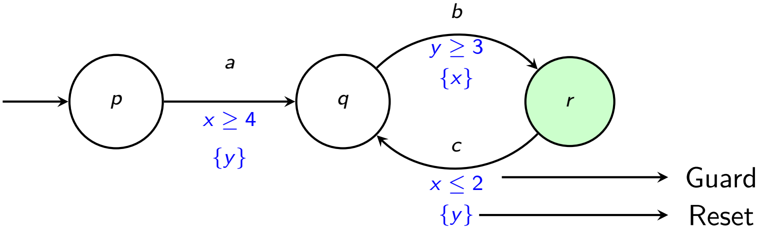


Guard

Reset

Timed automaton [Alur Dill '94]

Clocks: $X = \{x, y\}$



Configuration: (q, v)

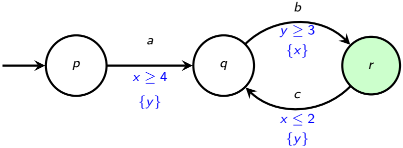
State

Valuation

$v : X \mapsto \mathbb{R}_{\geq 0}$

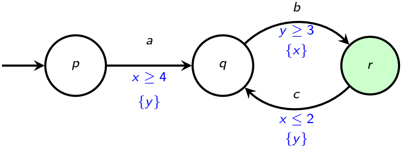
Timed automaton [Alur Dill '94]

Timed automaton



Timed automaton [Alur Dill '94]

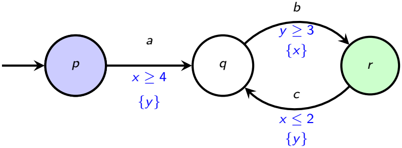
Timed automaton



Run of the timed automaton

Timed automaton [Alur Dill '94]

Timed automaton

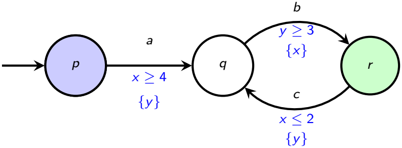


Run of the timed automaton

$(p, x = 0, y = 0)$

Timed automaton [Alur Dill '94]

Timed automaton



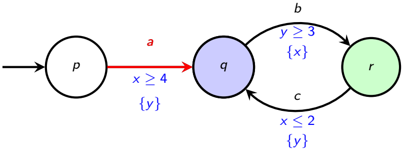
Run of the timed automaton

Delay

$$(p, x = 0, y = 0) \xrightarrow{5} (p, x = 5, y = 5)$$

Timed automaton [Alur Dill '94]

Timed automaton



Run of the timed automaton

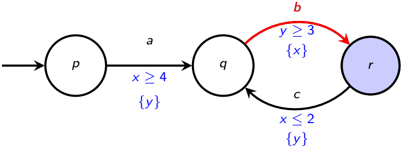
Delay

Action

$$(p, x = 0, y = 0) \xrightarrow{5} (p, x = 5, y = 5) \xrightarrow{a} (q, x = 5, y = 0)$$

Timed automaton [Alur Dill '94]

Timed automaton



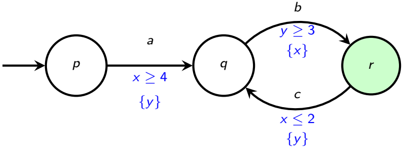
Run of the timed automaton

Delay ↑ Action ↑

$$(p, x=0, y=0) \xrightarrow{5} (p, x=5, y=5) \xrightarrow{a} (q, x=5, y=0) \xrightarrow{3} \xrightarrow{b} (r, x=0, y=3)$$

Timed automaton [Alur Dill '94]

Timed automaton



Run of the timed automaton

Delay

Action

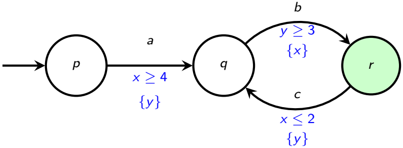
$$(p, x = 0, y = 0) \xrightarrow{5} (p, x = 5, y = 5) \xrightarrow{a} (q, x = 5, y = 0) \xrightarrow{3} \xrightarrow{b} (r, x = 0, y = 3)$$

Reachability Problem

Decide if a given timed automaton has a run reaching a green state.

Timed automaton [Alur Dill '94]

Timed automaton



Run of the timed automaton

$$(p, x = 0, y = 0) \xrightarrow{\substack{\text{Delay} \\ \uparrow \\ 5}} (p, x = 5, y = 5) \xrightarrow{\substack{\text{Action} \\ \uparrow \\ a}} (q, x = 5, y = 0) \xrightarrow{3} \xrightarrow{b} (r, x = 0, y = 3)$$

Reachability Problem

Decide if a given timed automaton has a run reaching a green state.

This problem is PSPACE-complete
[Alur Dill '94]

Standard Reachability Algorithm [Daws Tripakis '98]

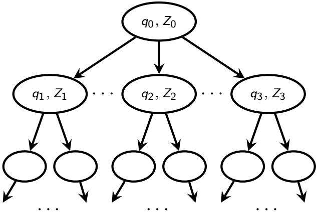
Based on explicit enumeration of
sets of valuations called **zones**

Standard Reachability Algorithm [Daws Tripakis '98]

Based on explicit enumeration of
sets of valuations called **zones**

Zone graph

Directed graph with nodes of the form (state, zone).

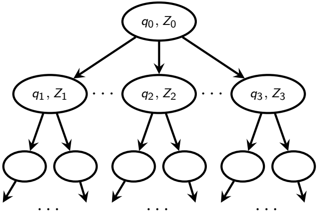


Standard Reachability Algorithm [Daws Tripakis '98]

Based on explicit enumeration of
sets of valuations called **zones**

Zone graph

Directed graph with nodes of the form (state, zone).



Used in tools like
UPPAAL, TChecker, PAT, KRONOS

Standard Reachability Algorithm

Zone graphs are infinite in general!

Standard Reachability Algorithm

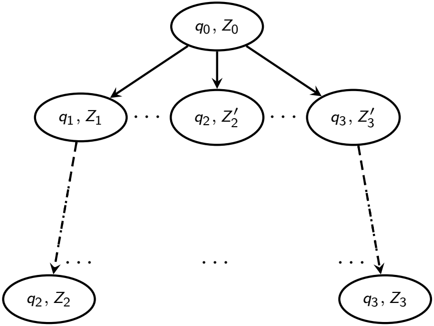
Zone graphs are infinite in general!

How do we ensure termination?

Standard Reachability Algorithm

Idea

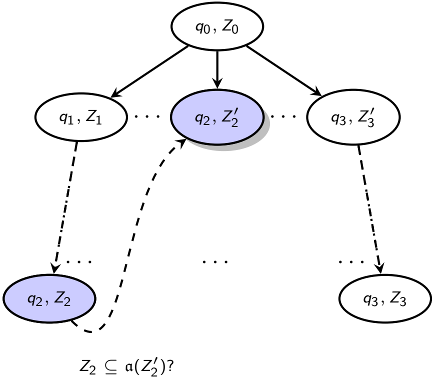
For each newly seen node (q, Z), check if there exists an already visited node (q, Z') such that Z is subsumed by Z'.



Standard Reachability Algorithm

Idea

For each newly seen node (q, Z), check if there exists an already visited node (q, Z') such that Z is subsumed by Z'.

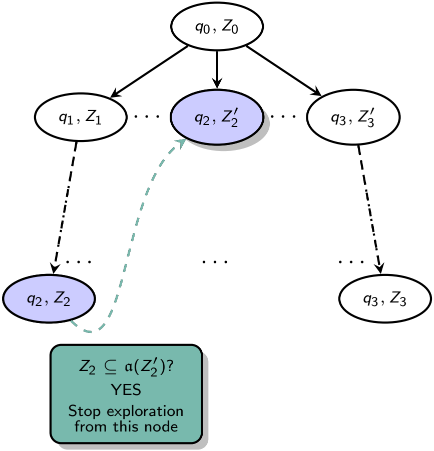


Standard Reachability Algorithm

Idea

For each newly seen node (q, Z), check if there exists an already visited node (q, Z') such that Z is subsumed by Z'.

- If yes, no need to explore further from that node.



$Z_2 \subseteq \mathfrak{a}(Z'_2)$?

YES

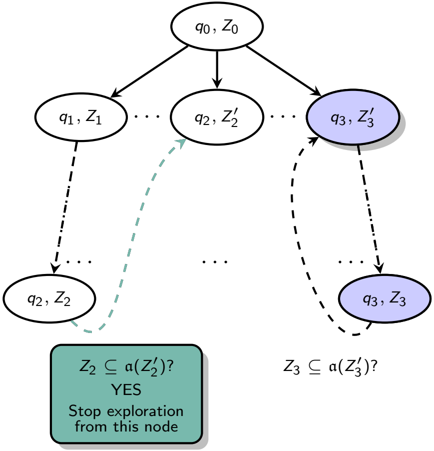
Stop exploration from this node

Standard Reachability Algorithm

Idea

For each newly seen node (q, Z), check if there exists an already visited node (q, Z') such that Z is subsumed by Z'.

- If yes, no need to explore further from that node.
- If not, add (q, Z) to the set of visited nodes and continue exploration.



$Z_2 \subseteq \mathfrak{a}(Z'_2)$?
YES
Stop exploration from this node

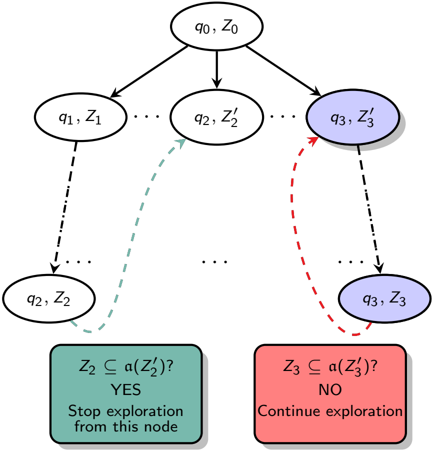
$Z_3 \subseteq \mathfrak{a}(Z'_3)$?

Standard Reachability Algorithm

Idea

For each newly seen node (q, Z), check if there exists an already visited node (q, Z') such that Z is subsumed by Z'.

- If yes, no need to explore further from that node.
- If not, add (q, Z) to the set of visited nodes and continue exploration.



$Z_2 \subseteq \mathfrak{a}(Z'_2)$?
YES
Stop exploration from this node

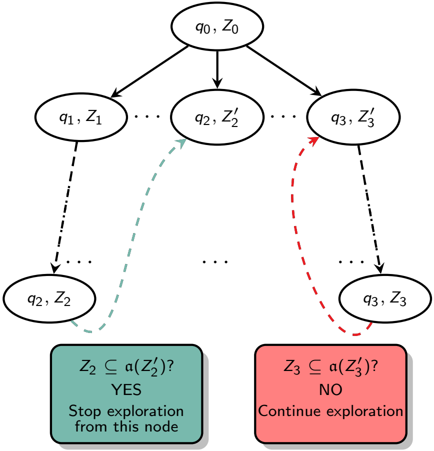
$Z_3 \subseteq \mathfrak{a}(Z'_3)$?
NO
Continue exploration

Standard Reachability Algorithm

Idea

For each newly seen node (q, Z), check if there exists an already visited node (q, Z') such that Z is subsumed by Z'.

- If yes, no need to explore further from that node.
- If not, add (q, Z) to the set of visited nodes and continue exploration.



$Z_2 \subseteq \mathfrak{a}(Z'_2)$?
YES
Stop exploration from this node

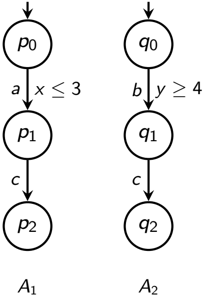
$Z_3 \subseteq \mathfrak{a}(Z'_3)$?
NO
Continue exploration

Subsumptions used to ensure finiteness for zone graphs

Well-studied subsumptions: $\mathfrak{a}_{LU}$, $\mathfrak{a}_M$ [BBLP 06, HSW12]

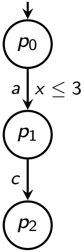
Network of timed automata

Network of timed automata

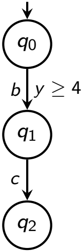


No shared clocks

Network of timed automata

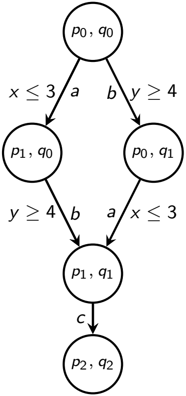


A_1



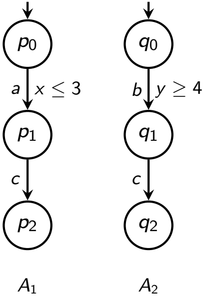
A_2

No shared clocks



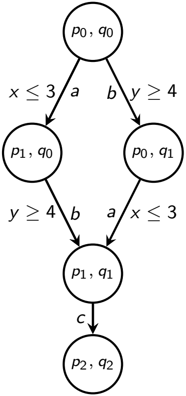
Synchronized
product

Network of timed automata

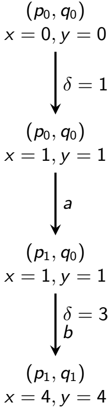


A_1 A_2

No shared clocks



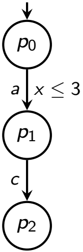
Synchronized
product



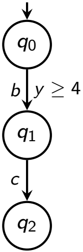
(p_0, q_0)
$x = 0, y = 0$
$\delta = 1$
(p_0, q_0)
$x = 1, y = 1$
a
(p_1, q_0)
$x = 1, y = 1$
$\delta = 3$
b
(p_1, q_1)
$x = 4, y = 4$

Run of the network

Network of timed automata

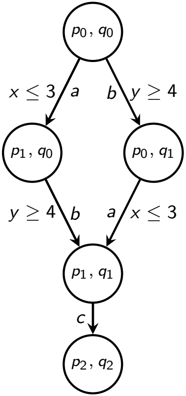


A_1

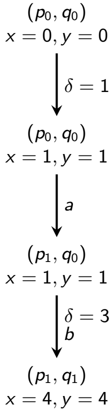


A_2

No shared clocks



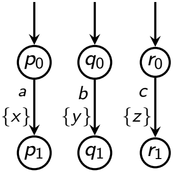
Synchronized product



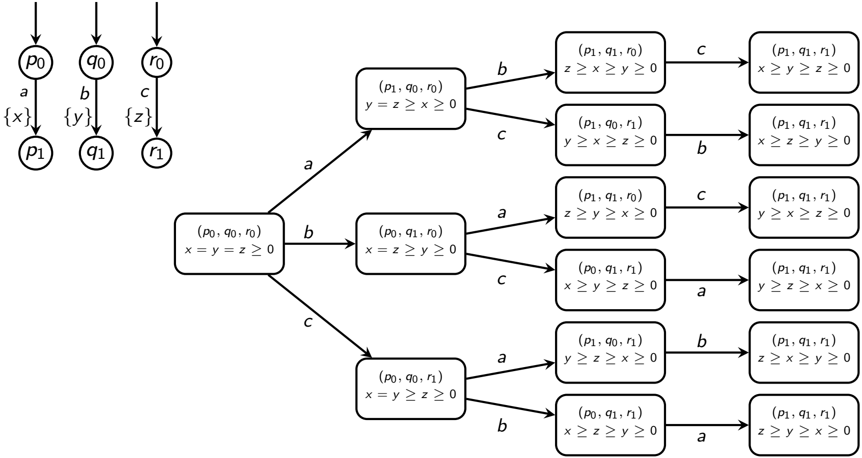
Run of the network

Time elapses synchronously for all processes

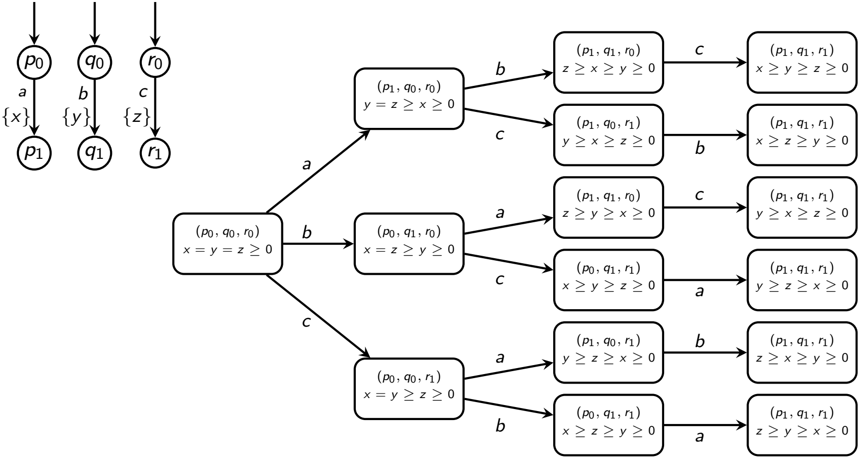
Zone Graph of a network of timed automata



Zone Graph of a network of timed automata



Zone Graph of a network of timed automata



Different interleavings of independent actions lead to different nodes!

Our problem: State-space explosion due to interleavings for networks of timed automata

Our problem: State-space explosion due to interleavings for networks of timed automata

Explosion is even worse than for untimed networks as
different interleavings lead to different nodes!

Our problem: State-space explosion due to interleavings for networks of timed automata

Explosion is even worse than for untimed networks as different interleavings lead to different nodes!

Our contribution:

Our problem: State-space explosion due to interleavings for networks of timed automata

Explosion is even worse than for untimed networks as different interleavings lead to different nodes!

Our contribution:

Two algorithms for alleviating effects of this explosion

Our problem: State-space explosion due to interleavings for networks of timed automata

Explosion is even worse than for untimed networks as
different interleavings lead to different nodes!

Our contribution:

Two algorithms for alleviating effects of this explosion

⋆ Modified zone graph that merges different interleavings into one node

Our problem: State-space explosion due to interleavings for networks of timed automata

Explosion is even worse than for untimed networks as
different interleavings lead to different nodes!

Our contribution:

Two algorithms for alleviating effects of this explosion

- ⋆ Modified zone graph that merges different interleavings into one node
- ⋆ Applying POR on this modified zone graph, technical challenges, solutions

Our problem: State-space explosion due to interleavings for networks of timed automata

Explosion is even worse than for untimed networks as different interleavings lead to different nodes!

Our contribution:

Two algorithms for alleviating effects of this explosion

⋆ Modified zone graph that merges different interleavings into one node

⋆ Applying POR on this modified zone graph, technical challenges, solutions

Main idea: **local time semantics**

Overview of the talk

✓ Partial order reduction

✓ Timed automata and our problem

Local time semantics

Local zone graph

Solution 1
Local time semantics and aggregated zones

Sync-subsumption

Interleavings in local sync graphs

Solution 2
Local time semantics and POR

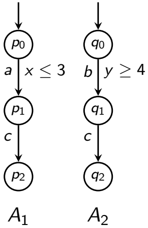
Local sync graphs + POR ?

No finite subsumption for LZG
that allows POR

POR for spread-bounded systems

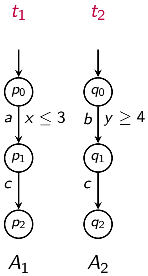
Local time semantics [Bengtsson et al. '98]

Idea: Allow each process to elapse time independently



Local time semantics [Bengtsson et al. '98]

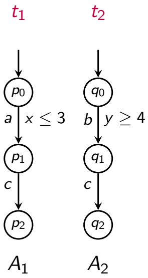
Idea: Allow each process to elapse time independently



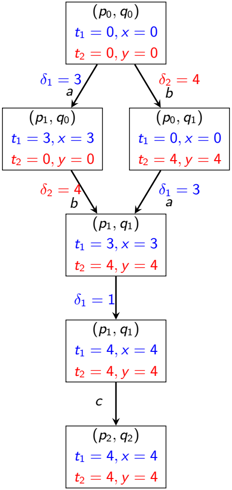
Reference clocks to measure time elapsed in each process

Local time semantics [Bengtsson et al. '98]

Idea: Allow each process to elapse time independently



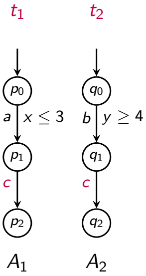
Reference clocks to measure time elapsed in each process



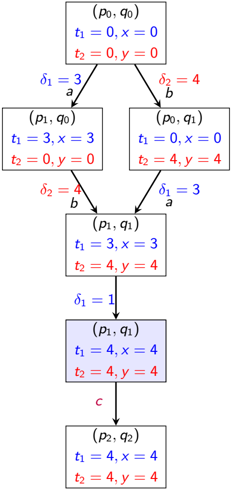
Local time semantics [Bengtsson et al. '98]

Idea: Allow each process to elapse time independently

Reference clocks to measure time elapsed in each process

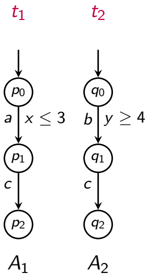


Shared actions only executed from valuations in which processes agree on the value of reference clocks



Local time semantics [Bengtsson et al. '98]

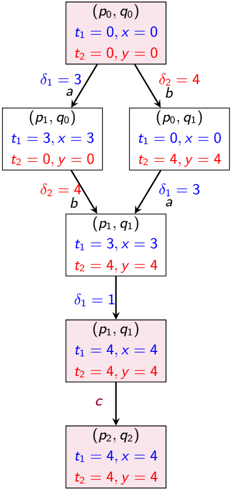
Idea: Allow each process to elapse time independently



Reference clocks to measure time elapsed in each process

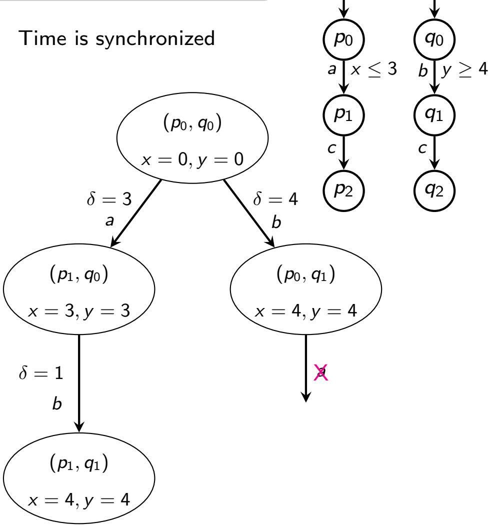
Shared actions only executed from valuations in which processes agree on the value of reference clocks

Synchronized valuations: reference clocks of all the processes have the same value



Global time semantics (Standard)

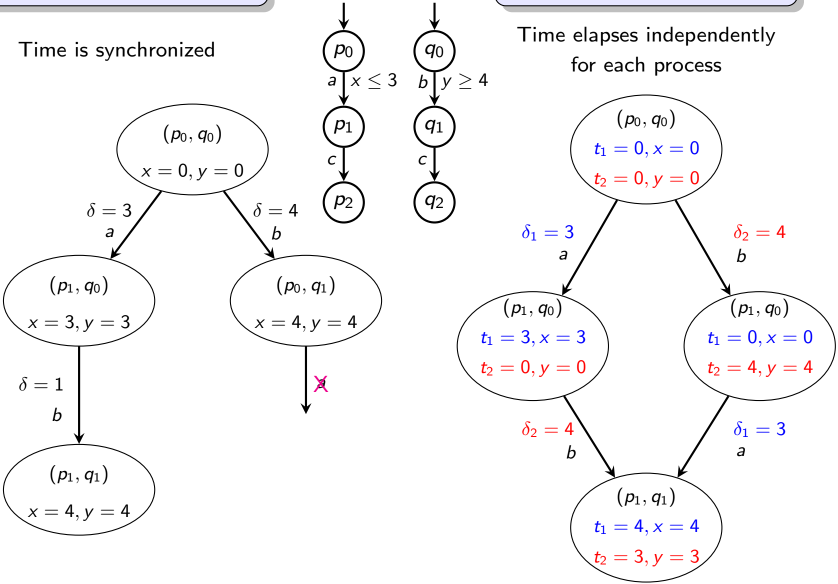
Time is synchronized



Global time semantics (Standard)

Local time semantics

[Bengtsson et al. '98]



Time is synchronized

Time elapses independently for each process

Local time semantics

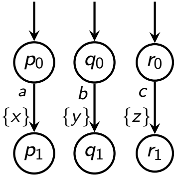
Independence [Bengtsson et al. '98]

If $(q_0, v_0) \xrightarrow{\sigma_1}_{lt} (q, v)$ and $\sigma_1 \sim \sigma_2$, then $(q_0, v_0) \xrightarrow{\sigma_2}_{lt} (q, v)$.

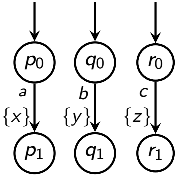
Correctness [Bengtsson et al. '98]

- Every global run is a local time run.
- If there is a local time run to a (q, v), where v is a synchronized valuation, then there is a global run to (q, v).

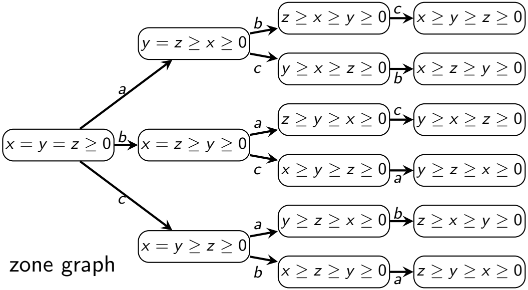
Local Zone Graph



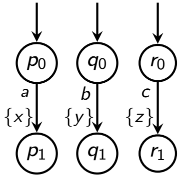
Local Zone Graph



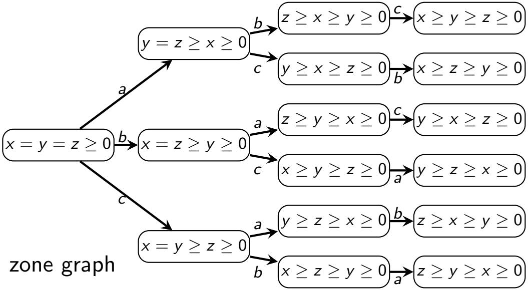
Global zone graph



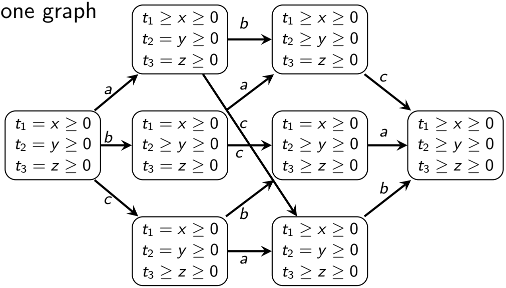
Local Zone Graph



Global zone graph



Local zone graph



Local zone graphs

Challenge: Local zone graphs are not finite in general!

Overview of the talk

✓ Partial order reduction

✓ Timed automata and our problem

✓ Local time semantics	✓ Local zone graph
Solution 1 **Local time semantics and aggregated zones**	Sync-subsumption Interleavings in local sync graphs
Solution 2 **Local time semantics and POR**	Local sync graphs + POR ? No finite subsumption for LZG that allows POR POR for spread-bounded systems

Local zone graphs

Challenge: Local zone graphs are not finite in general!

Local zone graphs

Challenge: Local zone graphs are not finite in general!

Current attempts at finiteness for local zone graphs

- Catchup equivalence
  [Bengtsson et al. '98]
- Extrapolation operator
  [Minea '99]

Local zone graphs

Challenge: Local zone graphs are not finite in general!

Current attempts at finiteness for local zone graphs

- Catchup equivalence
  [Bengtsson et al. '98]

Not efficiently computable.

- Extrapolation operator
  [Minea '99]

Our solution: Sync-subsumption

Our solution: Sync-subsumption

Idea: Focus on **synchronized valuations**

Our solution: Sync-subsumption

Idea: Focus on **synchronized valuations**

Synchronized valuations
all reference clocks are equal

Our solution: Sync-subsumption

Idea: Focus on **synchronized valuations**

Synchronized valuations
all reference clocks are equal

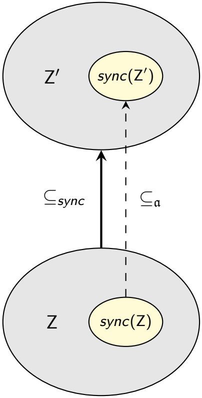
sync operator when applied to a local zone gives a standard zone.

Our solution: Sync-subsumption

Idea: Focus on **synchronized valuations**

Synchronized valuations
all reference clocks are equal

sync operator when applied to a local zone gives a standard zone.



Sync-subsumption

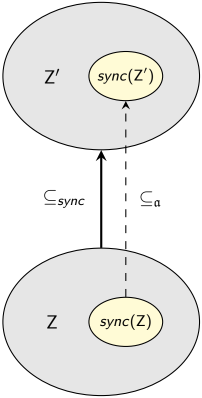
$Z \subseteq_{sync} Z'$ if
$sync(Z) \subseteq \mathfrak{a}_{LU}(sync(Z'))$

Our solution: Sync-subsumption

Idea: Focus on **synchronized valuations**

Synchronized valuations
all reference clocks are equal

sync operator when applied to a local zone gives a standard zone.



Sync-subsumption

$$Z \subseteq_{sync} Z' \text{ if}$$
$$sync(Z) \subseteq \mathfrak{a}_{LU}(sync(Z'))$$

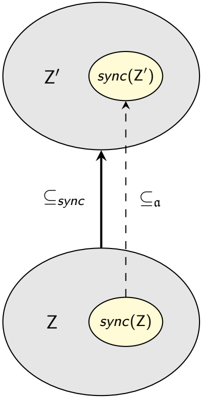
Allows use of subsumption techniques for standard zones.

Our solution: Sync-subsumption

Idea: Focus on **synchronized valuations**

Synchronized valuations
all reference clocks are equal

sync operator when applied to a local zone gives a standard zone.



Sync-subsumption

$$Z \subseteq_{sync} Z' \text{ if}$$
$$sync(Z) \subseteq \mathfrak{a}_{LU}(sync(Z'))$$

Allows use of subsumption techniques for standard zones.

Local sync graph - local zone graph with sync-subsumption

Overview of the talk

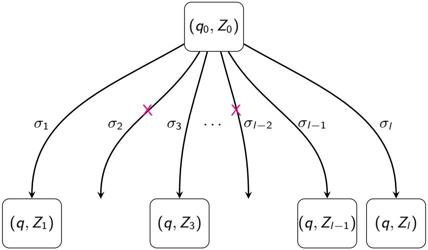
✓ Partial order reduction

✓ Timed automata and our problem

✓ Local time semantics	Local zone graph
Solution 1 **Local time semantics and aggregated zones**	✓ Sync-subsumption Interleavings in local sync graphs
Solution 2 **Local time semantics and POR**	Local sync graphs + POR ? No finite subsumption for LZG that allows POR POR for spread-bounded systems

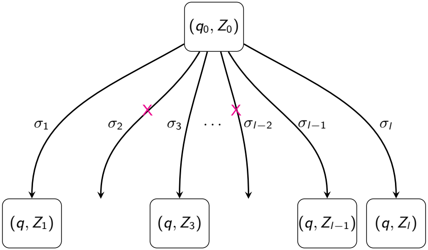
Aggregated zones in local zone graph

Aggregated zones in local zone graph

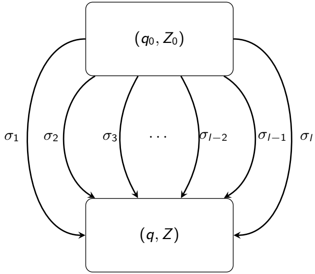


Global zone graph

Aggregated zones in local zone graph

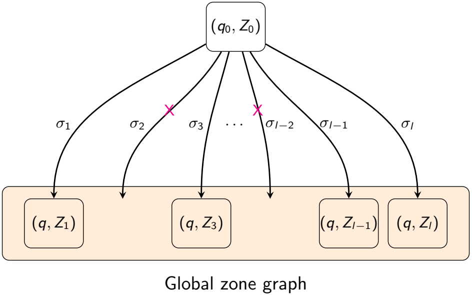


Global zone graph

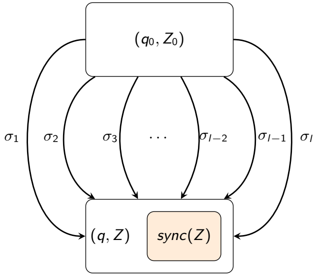


Local zone graph

Aggregated zones in local zone graph



Global zone graph



Local zone graph

$$Z_1 \cup Z_2 \cup \cdots \cup Z_l = sync(Z)$$

Overview of the talk

✓ Partial order reduction

✓ Timed automata and our problem

✓ Local time semantics

Local zone graph

Solution 1

Local time semantics and aggregated zones

✓ Sync-subsumption

✓ Interleavings in local sync graphs

Solution 2

Local time semantics and POR

Local sync graphs + POR ?

No finite subsumption for LZG
that allows POR

POR for spread-bounded systems

Experimental Results

Models	UPPAAL			Global ZG			Local ZG		
(# processes)	visited	stored	sec.	visited	stored	sec.	visited	stored	sec.
CorSSO 3	64378	61948	1.48	64378	61948	1.41	1962	1962	0.05
CorSSO 4	timeout			timeout			23784	23784	0.69
CorSSO 5	timeout			timeout			281982	281982	16.71
Critical reg. 4	78049	53697	1.45	75804	53697	2.27	44490	28400	2.40
Critical reg. 5	timeout			timeout			709908	389614	75.55
Dining Phi. 7	38179	38179	34.61	38179	38179	7.28	2627	2627	0.32
Dining Phi. 8	timeout			timeout			8090	8090	1.65
Dining Phi. 9	timeout			timeout			24914	24914	7.10
Dining Phi. 10	timeout			timeout			76725	76725	30.20
Parallel 6	11743	11743	4.82	11743	11743	1.09	256	256	0.02
Parallel 7	timeout			timeout			576	576	0.04
Parallel 8	timeout			timeout			1280	1280	0.11
CSMACD 4	258	258	0.03	258	258	0.04	258	258	0.04
CSMACD 5	850	850	0.04s	850	850	0.07	850	850	0.11

Timeout is set to 90 seconds.

TChecker: an open-source model-checker for timed systems.
Available at `https://github.com/ticktac-project/tchecker`

Experimental Results

Models	UPPAAL			Global ZG			Local ZG		
(# processes)	visited	stored	sec.	visited	stored	sec.	visited	stored	sec.
CorSSO 3	**64378**	**61948**	**1.48**	**64378**	**61948**	**1.41**	**1962**	**1962**	**0.05**
CorSSO 4	timeout			timeout			23784	23784	0.69
CorSSO 5	timeout			timeout			281982	281982	16.71
Critical reg. 4	78049	53697	1.45	75804	53697	2.27	44490	28400	2.40
Critical reg. 5	timeout			timeout			709908	389614	75.55
Dining Phi. 7	38179	38179	34.61	38179	38179	7.28	2627	2627	0.32
Dining Phi. 8	timeout			timeout			8090	8090	1.65
Dining Phi. 9	timeout			timeout			24914	24914	7.10
Dining Phi. 10	timeout			timeout			76725	76725	30.20
Parallel 6	11743	11743	4.82	11743	11743	1.09	256	256	0.02
Parallel 7	timeout			timeout			576	576	0.04
Parallel 8	timeout			timeout			1280	1280	0.11
CSMACD 4	258	258	0.03	258	258	0.04	258	258	0.04
CSMACD 5	850	850	0.04s	850	850	0.07	850	850	0.11

Timeout is set to 90 seconds.

TChecker: an open-source model-checker for timed systems.
Available at `https://github.com/ticktac-project/tchecker`

Experimental Results

Models	UPPAAL			Global ZG			Local ZG		
(# processes)	visited	stored	sec.	visited	stored	sec.	visited	stored	sec.
CorSSO 3	64378	61948	1.48	64378	61948	1.41	1962	1962	0.05
CorSSO 4	timeout			timeout			23784	23784	0.69
CorSSO 5	timeout			timeout			281982	281982	16.71
Critical reg. 4	78049	53697	1.45	75804	53697	2.27	44490	28400	2.40
Critical reg. 5	timeout			timeout			709908	389614	75.55
Dining Phi. 7	38179	38179	34.61	38179	38179	7.28	2627	2627	0.32
Dining Phi. 8	timeout			timeout			8090	8090	1.65
Dining Phi. 9	timeout			timeout			24914	24914	7.10
Dining Phi. 10	timeout			timeout			76725	76725	30.20
Parallel 6	11743	11743	4.82	11743	11743	1.09	256	256	0.02
Parallel 7	timeout			timeout			576	576	0.04
Parallel 8	timeout			timeout			1280	1280	0.11
CSMACD 4	**258**	**258**	**0.03**	**258**	**258**	**0.04**	**258**	**258**	**0.04**
CSMACD 5	850	850	0.04s	850	850	0.07	850	850	0.11

Timeout is set to 90 seconds.

TChecker: an open-source model-checker for timed systems.
Available at `https://github.com/ticktac-project/tchecker`

Results

Local-sync graph implementation

Results

Local-sync graph implementation

Reachability algorithm based on local time semantics.

Results

Local-sync graph implementation

Reachability algorithm based on local time semantics.

Merges interleavings into a single node.

Results

Local-sync graph implementation

Reachability algorithm based on local time semantics.

Merges interleavings into a single node.

Good experimental results

- Very good performance on some examples.
- At least as good as the standard algorithm on every example.

Overview of the talk

✓ Partial order reduction

✓ Timed automata and our problem

✓ Local time semantics

Local zone graph

✓ Solution 1

Local time semantics and aggregated zones

✓ Sync-subsumption

✓ Interleavings in local sync graphs

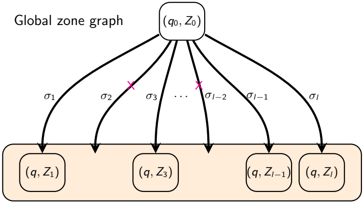
Solution 2

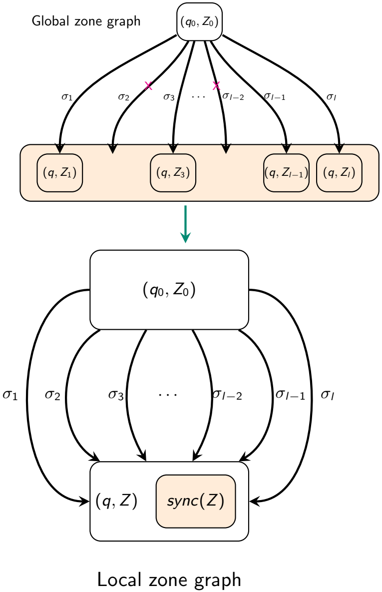
Local time semantics and POR

Local sync graphs + POR ?

No finite subsumption for LZG that allows POR

POR for spread-bounded systems

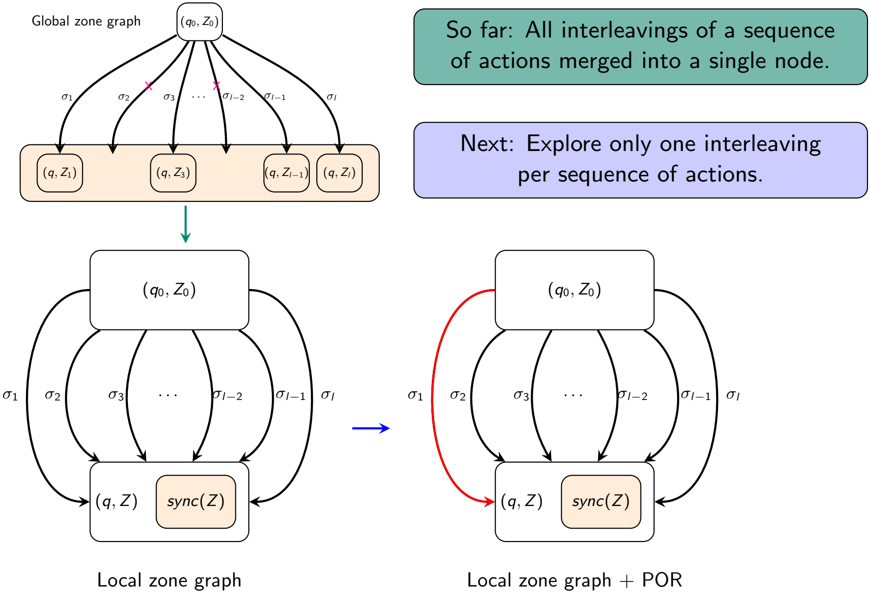

Global zone graph
(q_0, Z_0)
σ_1
σ_2
σ_3
$\cdots$
σ_{l-2}
σ_{l-1}
σ_l
(q, Z_1)
(q, Z_3)
(q, Z_{l-1})
(q, Z_l)



So far: All interleavings of a sequence of actions merged into a single node.

So far: All interleavings of a sequence of actions merged into a single node.

Next: Explore only one interleaving per sequence of actions.



Current object: Local sync graph

All interleavings of a sequence of actions
are merged into a single node.

Current object: Local sync graph

All interleavings of a sequence of actions
are merged into a single node.

Can we directly apply an existing POR technique on local sync graphs?

Current object: Local sync graph

All interleavings of a sequence of actions
are merged into a single node.

Can we directly apply an existing POR technique on local sync graphs?

Sync-subsumption does not preserve enabled actions

Current object: Local sync graph

All interleavings of a sequence of actions
are merged into a single node.

Can we directly apply an existing POR technique on local sync graphs?

Sync-subsumption does not preserve enabled actions
Local sync graph does not have diamonds!

Current object: Local sync graph

All interleavings of a sequence of actions
are merged into a single node.

Can we directly apply an existing POR
technique on local sync graphs?

NO!

Sync-subsumption does not preserve enabled actions
Local sync graph does not have diamonds!

Overview of the talk

✓ Partial order reduction

✓ Timed automata and our problem

✓ Local time semantics

Local zone graph

✓ Solution 1

Local time semantics and aggregated zones

✓ Sync-subsumption

✓ Interleavings in local sync graphs

Solution 2

Local time semantics and POR

✓ Local sync graphs + POR ?

No finite subsumption for LZG that allows POR

POR for spread-bounded systems

Challenge: To obtain a finite version of local zone graph that allows application of POR.

Challenge: To obtain a finite version of local zone graph that allows application of POR.

What do we need?

Challenge: To obtain a finite version of local zone graph that allows application of POR.

What do we need?

- Finiteness

Challenge: To obtain a finite version of local zone graph that allows application of POR.

What do we need?

- Finiteness
- Soundness

Challenge: To obtain a finite version of local zone graph that allows application of POR.

What do we need?

- Finiteness
- Soundness
- Preservation of enabled actions

Our Results

Our Results

No subsumption that is finite, sound and preserves enabled actions.

Our Results

No subsumption that is finite, sound and preserves enabled actions.

A new subsumption $\mathfrak{a}^*_M$ that guarantees soundness and preservation of enabled actions, but not finiteness.

Our Results

No subsumption that is finite, sound and preserves enabled actions.

A new subsumption $\mathfrak{a}^*_M$ that guarantees soundness and preservation of enabled actions, but not finiteness.

Subclass of networks where a modification of $\mathfrak{a}^*_M$ also guarantees finiteness.

Our Results

No subsumption that is finite, sound and preserves enabled actions.

A new subsumption $\mathfrak{a}_M^*$ that guarantees soundness and preservation of enabled actions, but not finiteness.

Subclass of networks where a modification of $\mathfrak{a}_M^*$ also guarantees finiteness.

Spread-bounded systems:

Our Results

No subsumption that is finite, sound and preserves enabled actions.

A new subsumption $\mathfrak{a}^*_M$ that guarantees soundness and preservation of enabled actions, but not finiteness.

Subclass of networks where a modification of $\mathfrak{a}^*_M$ also guarantees finiteness.

Spread-bounded systems: Networks for which every run can be converted to a run with only a bounded divergence between reference clocks.

Finite subsumption for spread-bounded systems

Finite subsumption for spread-bounded systems

$\mathfrak{a}_D^M$ **subsumption**

Finite subsumption for spread-bounded systems

$\mathfrak{a}_D^M$ subsumption

Theorem

Local zone graph + $\mathfrak{a}_D^M$ subsumption is finite, sound and preserves enabled actions for a D-spread-bounded system.

Overview of the talk

✓ Partial order reduction

✓ Timed automata and our problem

✓ Local time semantics

Local zone graph

✓ Solution 1

Local time semantics and aggregated zones

✓ Sync-subsumption

✓ Interleavings in local sync graphs

Solution 2

Local time semantics and POR

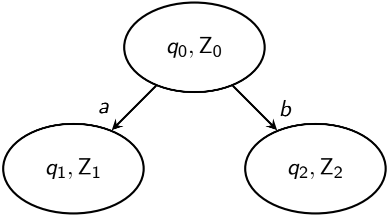
✓ Local sync graphs + POR ?

✓ No finite subsumption for LZG that allows POR

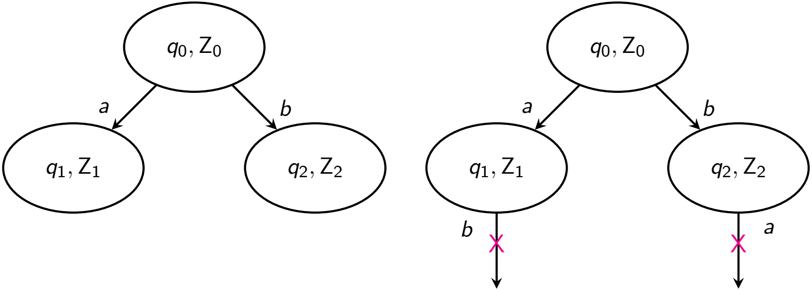
POR for spread-bounded systems

Can we directly apply an off-the-shelf POR technique on local zone graphs with $\mathfrak{a}_D^M$ subsumption?

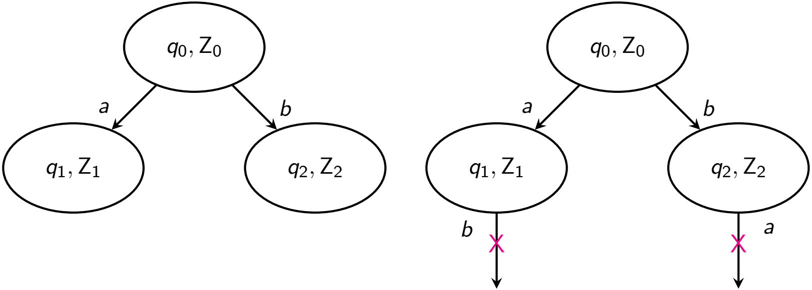
Can we directly apply an off-the-shelf POR technique on local zone graphs with $\mathfrak{a}_D^M$ subsumption?



Can we directly apply an off-the-shelf POR technique on local zone graphs with $\mathfrak{a}_D^M$ subsumption?

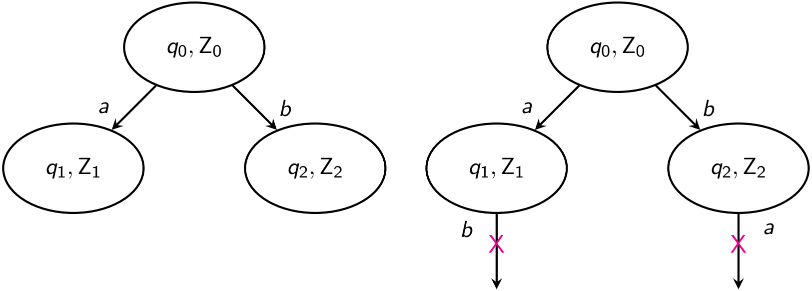


Can we directly apply an off-the-shelf POR technique on local zone graphs with $\mathfrak{a}_D^M$ subsumption?



Problem: a and b are enabled, but neither ab nor ba are feasible

Can we directly apply an off-the-shelf POR technique on local zone graphs with $\mathfrak{a}_D^M$ subsumption?



Problem: a and b are enabled, but neither ab nor ba are feasible

No forward diamonds!

We cannot directly apply an off-the-shelf POR technique on finite zone graph.

We cannot directly apply an off-the-shelf POR technique on finite zone graph.

We give specialized partial order reduction algorithms for spread-bounded systems.

We cannot directly apply an off-the-shelf POR technique on finite zone graph.

We give specialized partial order reduction algorithms for spread-bounded systems.

- Global-local POR
- Client-server POR

Experiments for POR implementation

Global-local POR

Models	Global ZG			Local Sync Graph			POR-LZG		
(# processes)	visited	stored	time	visited	stored	time	visited	stored	time
GL Toy 4	257	257	1.745s	257	257	2.295s	14	14	0.062s
GL Toy 5	1,025	1,025	10.264s	1,025	1,025	10.159s	17	17	0.091s
GL Toy 8	65,537	65,537	14m51s	65,537	65,537	14m52s	26	26	0.113s
Fire-alarm 4	271	271	2.269s	271	271	0.114s	17	17	0.299s
Fire-alarm 5	1,055	1,055	0.102s	1,055	1,055	0.123s	21	21	0.085s
Fire-alarm 8	**65,791**	**65,791**	**7.752s**	**65,791**	**65,791**	**29.622s**	**33**	**33**	**1.740s**
Fire-alarm 10	1mil	1mil	> 1hr	1mil	1mil	> 1hr	41	41	1.8s
CSMA CD 3	70	70	0.03s	70	70	0.03s	76	76	0.04s
CSMA CD 4	258	258	0.07s	258	258	0.12s	306	306	0.14s
CSMA CD 5	850	850	0.24s	850	850	0.46s	1100	1100	0.6s

Good results for **Fire-alarm**

Implemented in **TChecker**.

Available at https://github.com/ticktac-project/tchecker

Experiments for POR implementation

Client-server POR

Models (# processes)	Global ZG			Local Sync Graph			POR-LZG		
	visited	stored	sec.	visited	stored	sec.	visited	stored	sec.
CS Toy 3	216	216	1.3s	216	216	1.3s	56	56	0.2s
CS Toy 4	1296	1296	9.8s	1296	1296	9.6s	144	144	0.4s
CS Toy 5	7776	7776	1m27s	7776	7776	1m15s	352	352	4.2s
CS Toy 8	1.6M	1,6M	98m	1,6M	1,6M	95m	4352	4352	25.2s
WCET 1	138	138	2.2s	138	138	2.3s	113	101	2.3s
WCET 2	9,379	9,379	1m5s	8,803	8,803	53s	6260	4520	1m5s
WCET 3	**647,338**	**647,338**	$>$ **1hr**	**524,650**	**524,650**	$>$ **1hr**	**168,463**	**168,463**	**11m**
WCET 4	47mil	47mil	$>$ 1,027m	29mil	29mil	$>$ 472m	3mil	2.2mil	200m

Good results for **WCET**

Implemented in **TChecker**.
Available at `https://github.com/ticktac-project/tchecker`

Overview of the talk

✓ Partial order reduction

✓ Timed automata and our problem

✓ Local time semantics

Local zone graph

✓ Solution 1

Local time semantics and aggregated zones

✓ Sync-subsumption

✓ Interleavings in local sync graphs

✓ Solution 2

Local time semantics and POR

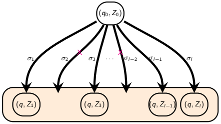
✓ Local sync graphs + POR ?

✓ No finite subsumption for LZG that allows POR

✓ POR for spread-bounded systems

Conclusion

Challenge: State-space explosion in networks of timed automata.

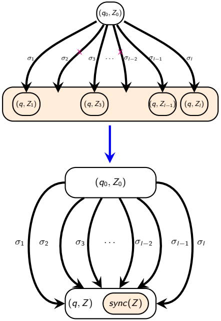


Conclusion

Challenge: State-space explosion in networks of timed automata.

Reachability algorithm based on exploration of local zone graph.

Already better than the standard reachability algorithm.



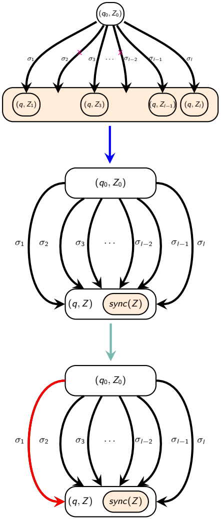
Conclusion

Challenge: State-space explosion in networks of timed automata.

Reachability algorithm based on exploration of local zone graph.

Already better than the standard reachability algorithm.

Partial order reduction algorithm for some classes of networks of timed automata.



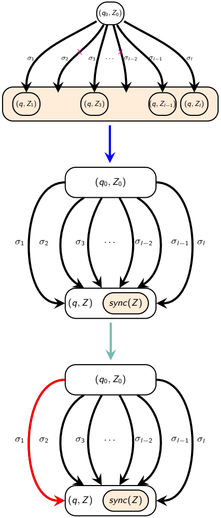
Conclusion

Challenge: State-space explosion in networks of timed automata.

Reachability algorithm based on exploration of local zone graph.

Already better than the standard reachability algorithm.

Partial order reduction algorithm for some classes of networks of timed automata.



Evaluation of a prototype of the implementation of these methods using the tool TChecker.

Open problems and future directions

Open problems and future directions

Local sync graphs

Open problems and future directions

Local sync graphs

Extending our algorithm to other models

Open problems and future directions

Local sync graphs

Extending our algorithm to other models

- Timed automata with diagonal constraints.

Open problems and future directions

Local sync graphs

Extending our algorithm to other models

- Timed automata with diagonal constraints.
- Updatable timed automata.

Open problems and future directions

Local sync graphs

Extending our algorithm to other models

- Timed automata with diagonal constraints.
- Updatable timed automata.

POR-reachability algorithm

Open problems and future directions

Local sync graphs

Extending our algorithm to other models

- Timed automata with diagonal constraints.
- Updatable timed automata.

POR-reachability algorithm

- Improving our POR-technique.

Open problems and future directions

Local sync graphs

Extending our algorithm to other models

- Timed automata with diagonal constraints.
- Updatable timed automata.

POR-reachability algorithm

- Improving our POR-technique.
- Combining our techniques with other solutions for state-space explosion such as symmetry reduction.

Open problems and future directions

Local sync graphs

Extending our algorithm to other models

- Timed automata with diagonal constraints.
- Updatable timed automata.

POR-reachability algorithm

- Improving our POR-technique.
- Combining our techniques with other solutions for state-space explosion such as symmetry reduction.

Thank you!