Laboratoire Méthodes Formelles

Zone-based verification of timed automata: Extrapolations, simulations and what next?

Patricia Bouyer

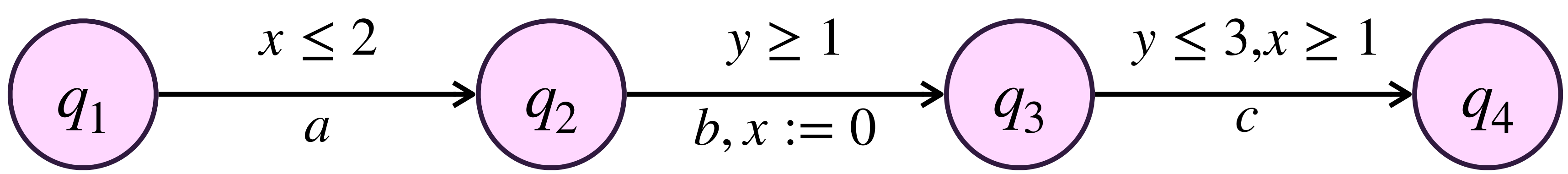
Laboratoire Méthodes Formelles
Université Paris-Saclay, CNRS, ENS Paris-Saclay
France

Based on a survey paper written with Paul Gastin, Frédéric Herbreteau, Ocan Sankur and B. Srivathsan

Partly supported by ANR projet Ticktac

Timed Automata

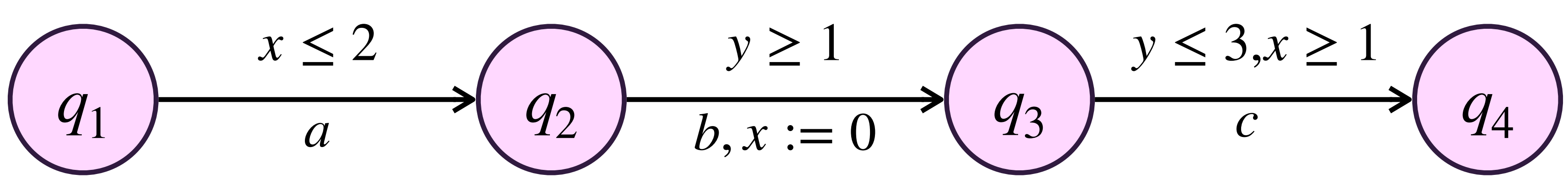
[AD90,AD94]



[AD90] Alur, Dill: Automata For Modeling Real-Time Systems (ICALP'90)
[AD94] Alur, Dill: A Theory of Timed Automata (TCS)

Timed Automata

[AD90,AD94]



- Infinitely many configurations!
- Decidability proven using regions
- Reachability is PSPACE-complete

[AD90] Alur, Dill: Automata For Modeling Real-Time Systems (ICALP'90)
[AD94] Alur, Dill: A Theory of Timed Automata (TCS)

Zones and DBMs

Enumerative approach: not possible
Region construction: not feasible in general
Alternative: **zone-based symbolic computation**

Zones and DBMs

Enumerative approach: not possible
Region construction: not feasible in general
Alternative: **zone-based symbolic computation**

- Zone = symbolic representation

$$Z = (x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$

Zones and DBMs

Enumerative approach: not possible
Region construction: not feasible in general
Alternative: **zone-based symbolic computation**

- Zone = symbolic representation

$$Z = (x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$

- DBM = data structure

$$\begin{array}{cc} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} +\infty & -3 & 0 \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{pmatrix} \end{array}$$

Zones and DBMs

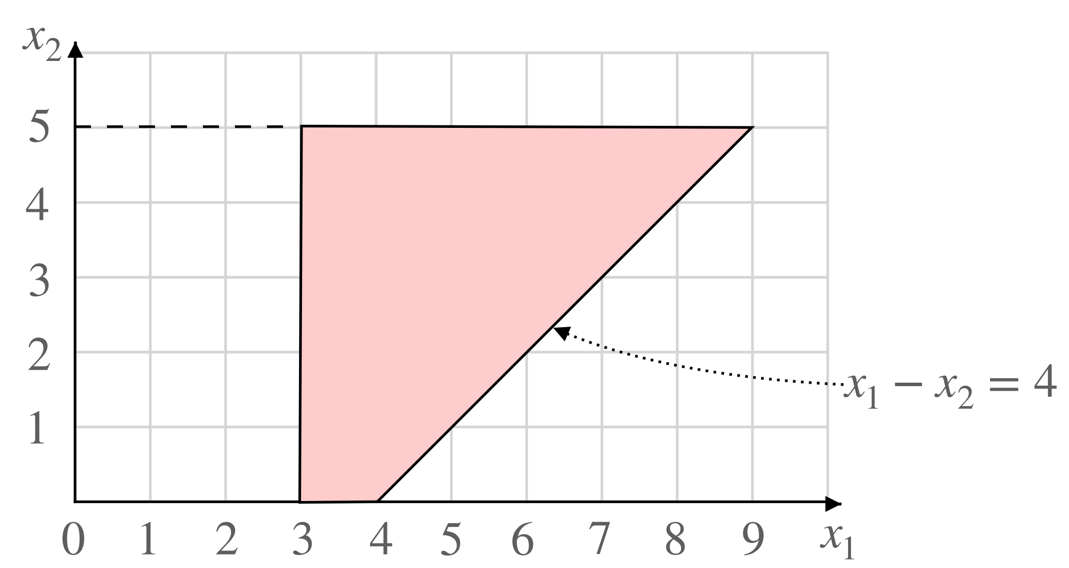
Enumerative approach: not possible
Region construction: not feasible in general
Alternative: **zone-based symbolic computation**

- Zone = symbolic representation

$$Z = (x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$

- DBM = data structure

$$\begin{array}{cc} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} +\infty & -3 & 0 \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{pmatrix} \end{array}$$



Zones and DBMs

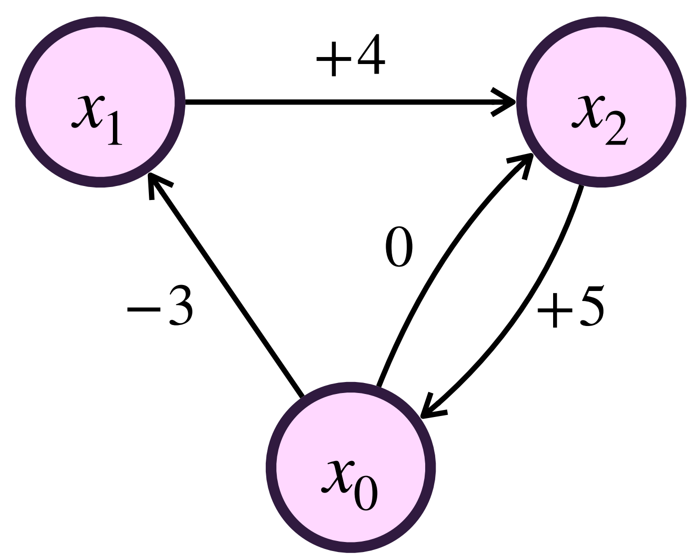
Enumerative approach: not possible
Region construction: not feasible in general
Alternative: **zone-based symbolic computation**

- Zone = symbolic representation

$$Z = (x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$

- DBM = data structure

$$\begin{array}{cc} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} +\infty & -3 & 0 \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{pmatrix} \end{array}$$



Zones and DBMs

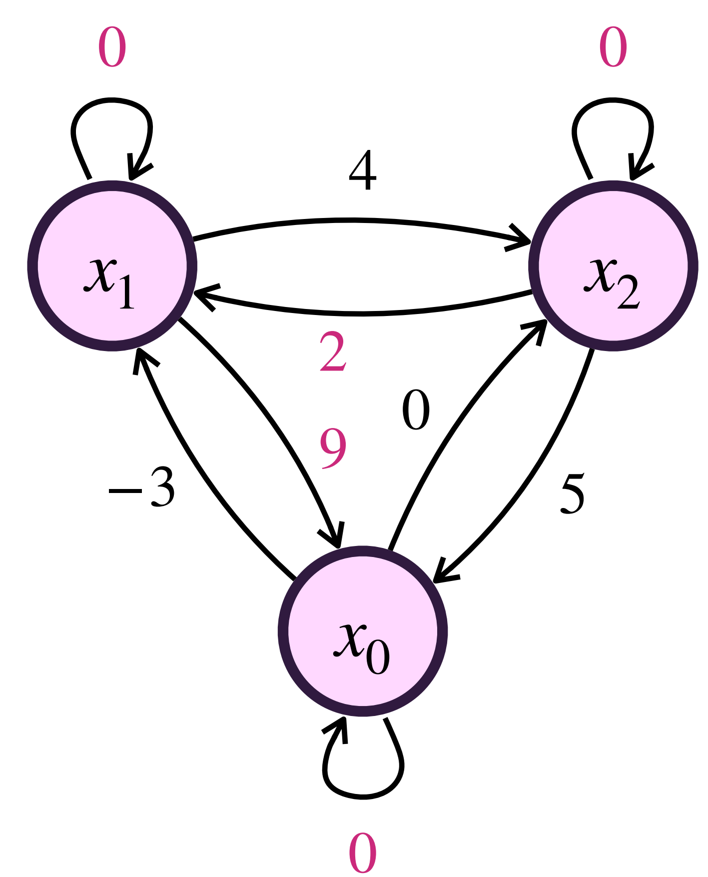
Enumerative approach: not possible
Region construction: not feasible in general
Alternative: **zone-based symbolic computation**

- Zone = symbolic representation

$$Z = (x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$

- DBM = data structure

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} +\infty & -3 & 0 \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{pmatrix} \end{array}$$



Zones and DBMs

Enumerative approach: not possible
Region construction: not feasible in general
Alternative: **zone-based symbolic computation**

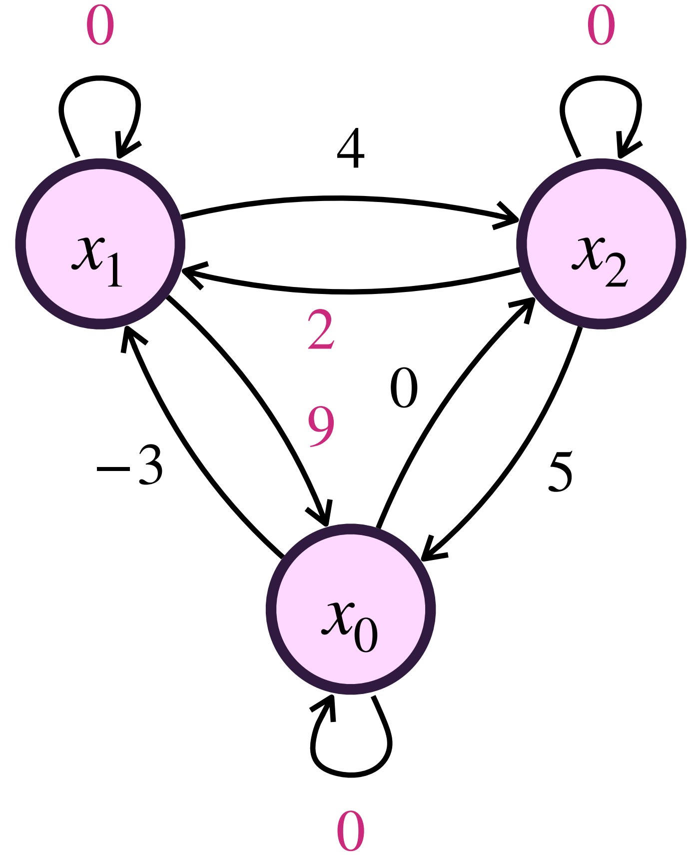
- Zone = symbolic representation

$$Z = (x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$

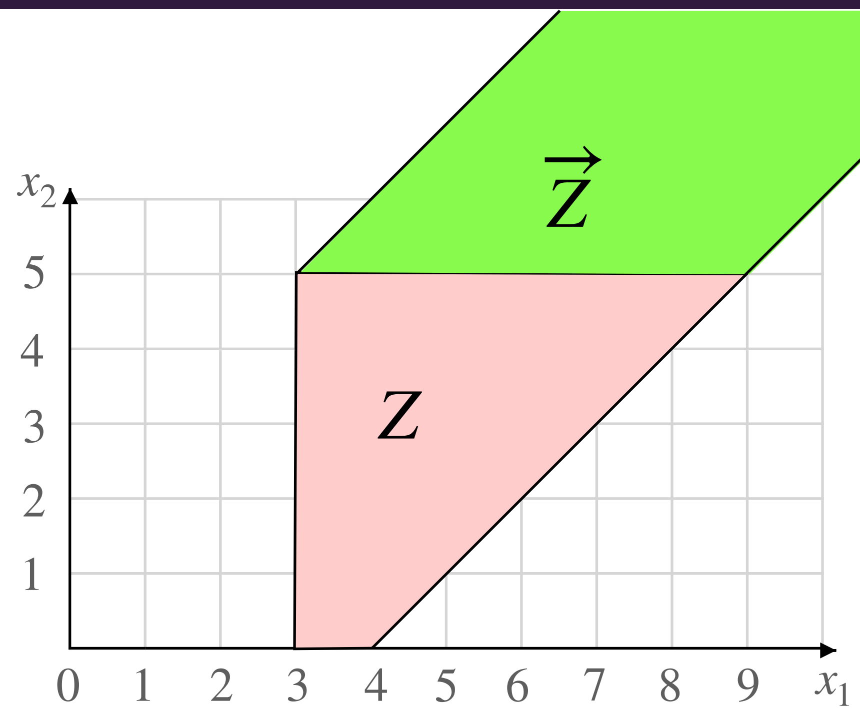
- DBM = data structure

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix} \end{array}$$

Normal form

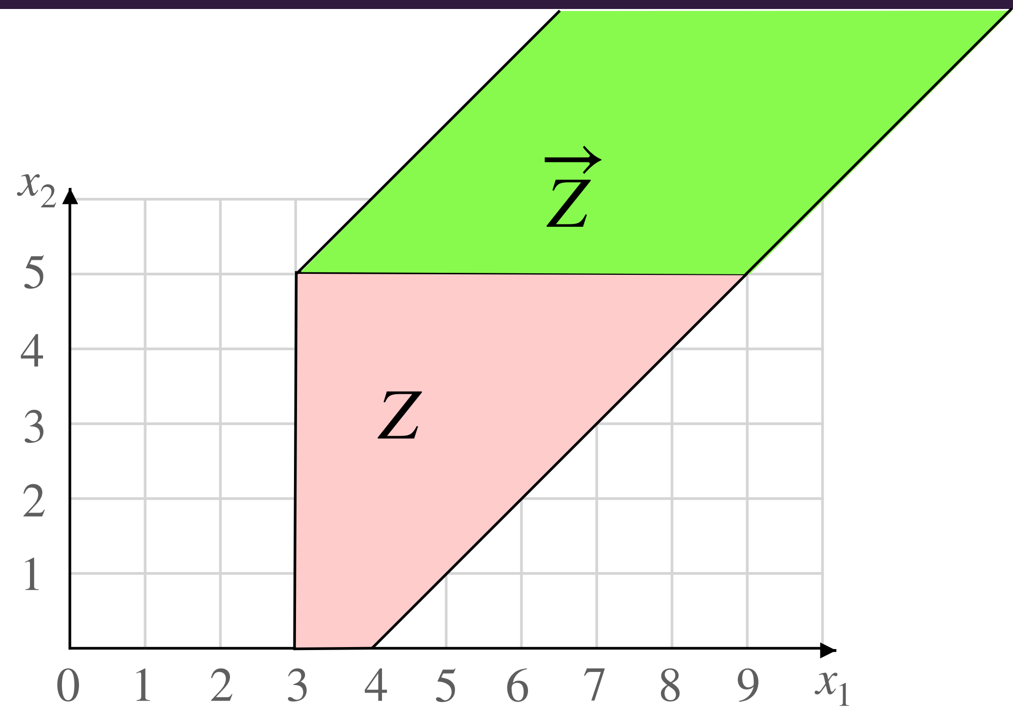


Operations on zones

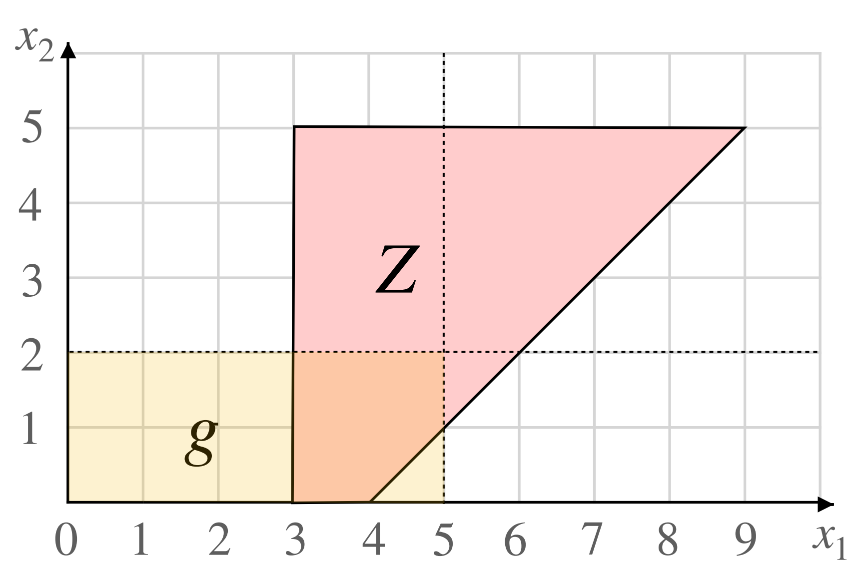


$$\begin{array}{cc} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix} \end{array} \rightsquigarrow \begin{array}{cc} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} 0 & -3 & 0 \\ +\infty & 0 & 4 \\ +\infty & 2 & 0 \end{pmatrix} \end{array}$$

Operations on zones

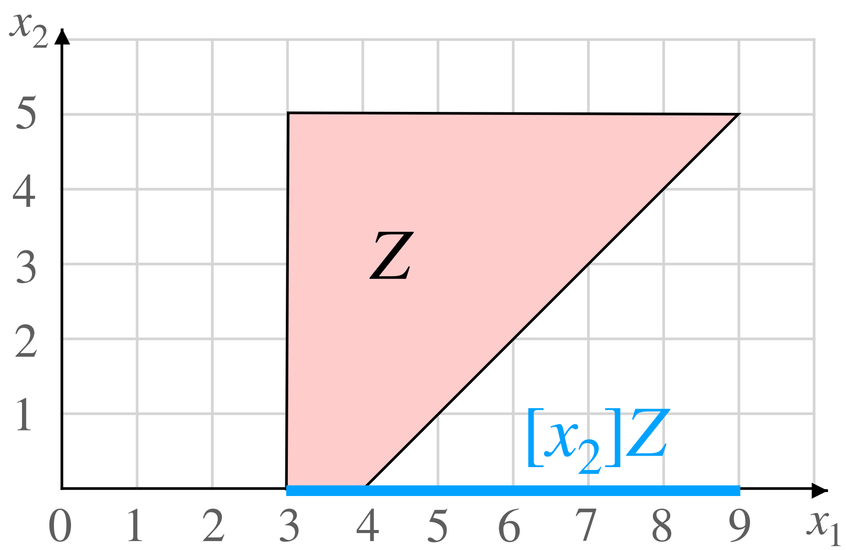


$$\begin{array}{c|ccc} & x_0 & x_1 & x_2 \\ x_0 & 0 & -3 & 0 \\ x_1 & 9 & 0 & 4 \\ x_2 & 5 & 2 & 0 \end{array} \rightsquigarrow \begin{array}{c|ccc} & x_0 & x_1 & x_2 \\ x_0 & 0 & -3 & 0 \\ x_1 & +\infty & 0 & 4 \\ x_2 & +\infty & 2 & 0 \end{array}$$



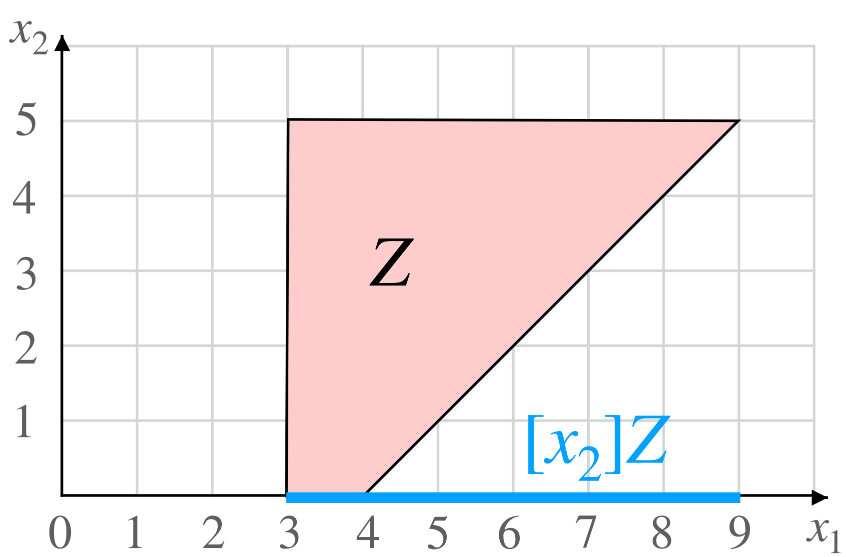
$$\begin{array}{c|ccc} & x_0 & x_1 & x_2 \\ x_0 & 0 & -3 & 0 \\ x_1 & 9 & 0 & 4 \\ x_2 & 5 & 2 & 0 \end{array} \rightsquigarrow \begin{array}{c|ccc} & x_0 & x_1 & x_2 \\ x_0 & 0 & -3 & 0 \\ x_1 & 5 & 0 & 4 \\ x_2 & 2 & -1 & 0 \end{array}$$

Operations on zones



$$
\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array}
\begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix} \end{array}
\quad \rightsquigarrow \quad
\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array}
\begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 9 \\ 0 & -3 & 0 \end{pmatrix} \end{array}
$$

Operations on zones



$$\begin{array}{c|ccc} & x_0 & x_1 & x_2 \\ \hline x_0 & & & \\ x_1 & & & \\ x_2 & & & \end{array}$$

$$\begin{array}{cc} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix} \end{array} \rightsquigarrow \begin{array}{cc} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 9 \\ 0 & -3 & 0 \end{pmatrix} \end{array}$$

If Z is a zone, then $Z' = \overrightarrow{[Y](Z \cap g)}$ is a zone

The computation can be made in $\mathcal{O}(|X|^2 \cdot |g|)$

Standard forward computation

- Initialize $\mathcal{S}$ with $(q_0, \vec{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add (q', Z') to $\mathcal{S}$, where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$ unless there is $(q', Z'') \in \mathcal{S}$ s.t. $Z' \subseteq Z''$

Standard forward computation

- Initialize $\mathcal{S}$ with $(q_0, \vec{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add (q', Z') to $\mathcal{S}$,
    where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$
    unless there is $(q', Z'') \in \mathcal{S}$ s.t. $Z' \subseteq Z''$

Inclusion test
Can be made in $\mathcal{O}(|X|^2)$

Standard forward computation

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add (q', Z') to $\mathcal{S}$,
    where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$
    unless there is $(q', Z'') \in \mathcal{S}$ s.t. $Z' \subseteq Z''$

Inclusion test
Can be made in $\mathcal{O}(|X|^2)$

Three properties

- Soundness: for every $(q, Z) \in \mathcal{S}$, there is $v \in Z$ s.t. (q, v) reachable

Standard forward computation

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add (q', Z') to $\mathcal{S}$, where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$ unless there is $(q', Z'') \in \mathcal{S}$ s.t. $Z' \subseteq Z''$

Inclusion test
Can be made in $\mathcal{O}(|X|^2)$

Three properties

- Soundness: for every $(q, Z) \in \mathcal{S}$, there is $v \in Z$ s.t. (q, v) reachable
- Completeness: for every reachable (q, v), there is $(q, Z) \in \mathcal{S}$ s.t. $v \in Z$

Standard forward computation

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add (q', Z') to $\mathcal{S}$, where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$ unless there is $(q', Z'') \in \mathcal{S}$ s.t. $Z' \subseteq Z''$

Inclusion test
Can be made in $\mathcal{O}(|X|^2)$

Three properties

- Soundness: for every $(q, Z) \in \mathcal{S}$, there is $v \in Z$ s.t. (q, v) reachable
- Completeness: for every reachable (q, v), there is $(q, Z) \in \mathcal{S}$ s.t. $v \in Z$
- Termination: saturation eventually happens

Standard forward computation

- Initialize $\mathcal{S}$ with $(q_0, \vec{0})$
- Repeat until saturation:
  - If $(q, Z$

    where

    unless

The computation does not terminate in general

inclusion test
Can be made in $\mathcal{O}(|X|^2)$

Three properties

- Soundness: for every $(q, Z) \in \mathcal{S}$, there is $v \in Z$ s.t. (q, v) reachable
- Completeness: for every reachable (q, v), there is $(q, Z) \in \mathcal{S}$ s.t. $v \in Z$
- Termination: saturation eventually happens

Two approaches

- Extrapolation
- Simulation

The extrapolation approach

The extrapolation approach

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add $(q', \text{extra}(Z'))$ to $\mathcal{S}$, where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$ unless there is $(q', Z'') \in \mathcal{S}$ s.t. extra$(Z') \subseteq Z''$

The extrapolation approach

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add $(q', \text{extra}(Z'))$ to $\mathcal{S}$, where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$ unless there is $(q', Z'') \in \mathcal{S}$ s.t. extra$(Z') \subseteq Z''$

NF after extrapolation can be computed in $\mathcal{O}(|X|^3)$

Inclusion can be decided in $\mathcal{O}(|X|^2)$

The extrapolation approach

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add $(q', \text{extra}(Z'))$ to $\mathcal{S}$, where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$ unless there is $(q', Z'') \in \mathcal{S}$ s.t. extra$(Z') \subseteq Z''$

NF after extrapolation can be computed in $\mathcal{O}(|X|^3)$

Inclusion can be decided in $\mathcal{O}(|X|^2)$

Operator extra defined s.t.

- Termination is ensured (extra has finite range)
- Completeness is obvious
- *Soundness is challenging*

Extrapolation

Remove « irrelevant » constants w.r.t. the automaton
⇝ syntactic on the DBM

$$\begin{array}{cc} & \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \end{array} & \begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix} \end{array}$$

Extrapolation

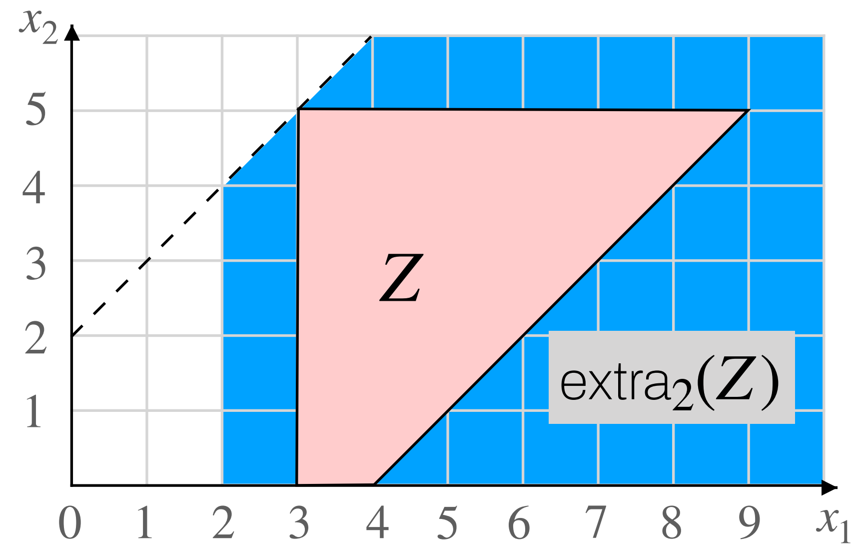
Remove « irrelevant » constants w.r.t. the automaton
$\rightsquigarrow$ syntactic on the DBM

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{pmatrix} \end{array} \rightsquigarrow \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} 0 & -2 & 0 \\ +\infty & 0 & +\infty \\ +\infty & 2 & 0 \end{pmatrix} \end{array}$$

Extrapolation

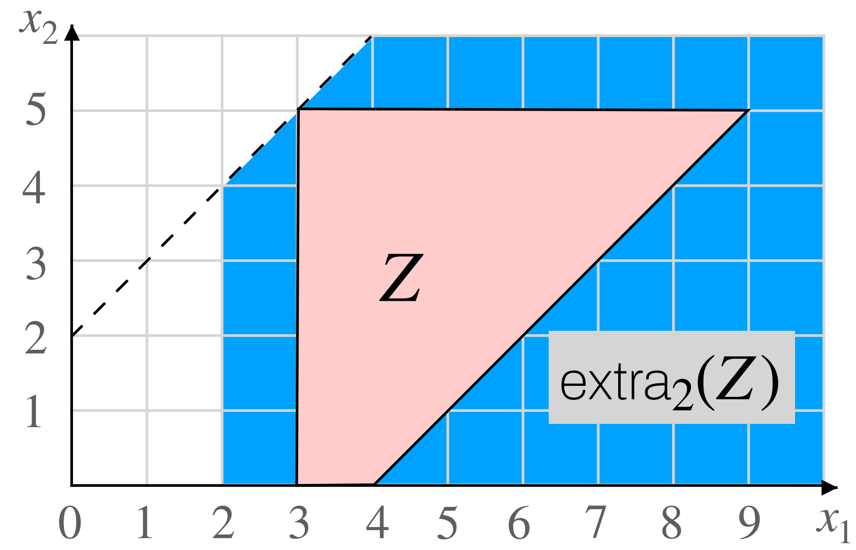
Remove « irrelevant » constants w.r.t. the automaton

⇝ syntactic on the DBM



Extrapolation

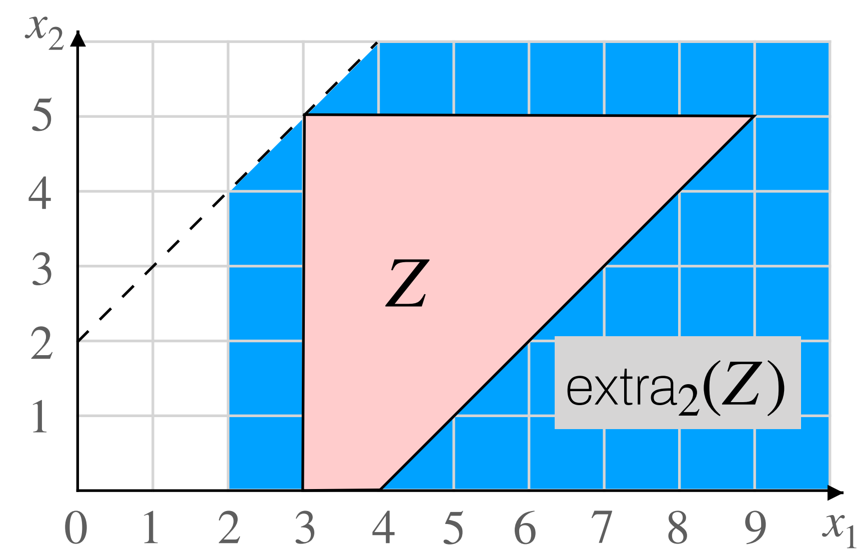
Remove « irrelevant » constants w.r.t. the automaton
⇝ syntactic on the DBM



- Extrapolation [DT98,Bou03,Bou04]
- State-dependent extrapolation [BBFL03]
- LU-extrapolation [BBLP04,BBLP06]

Extrapolation

Remove « irrelevant » constants w.r.t. the automaton
⇝ syntactic on the DBM

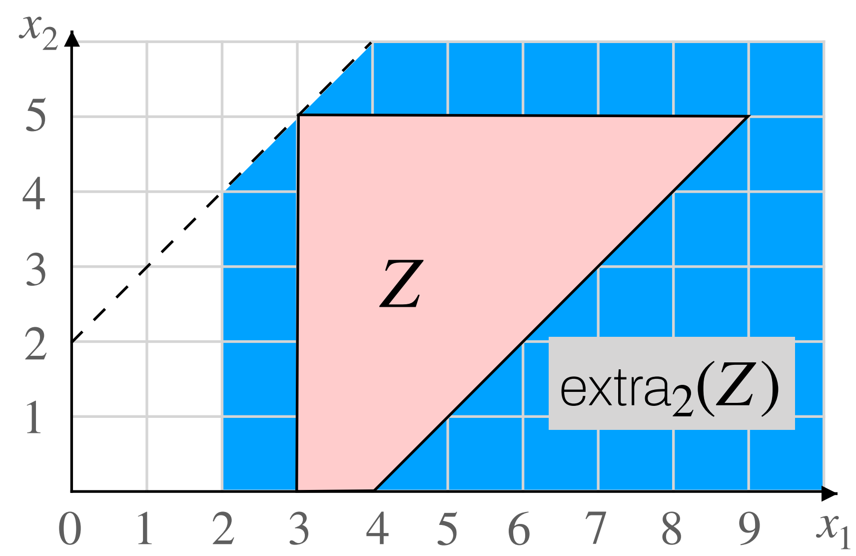


- Extrapolation [DT98,Bou03,Bou04]
- State-dependent extrapolation [BBFL03]
- LU-extrapolation [BBLP04,BBLP06]

➡ Soundness requires to show that extra is a simulation-based abstraction: $\forall v' \in \text{extra}(Z)\ \exists v \in Z$ s.t. $v' \preceq v$

Extrapolation

Remove « irrelevant »
constants w.r.t. the automaton
⇝ syntactic on the DBM

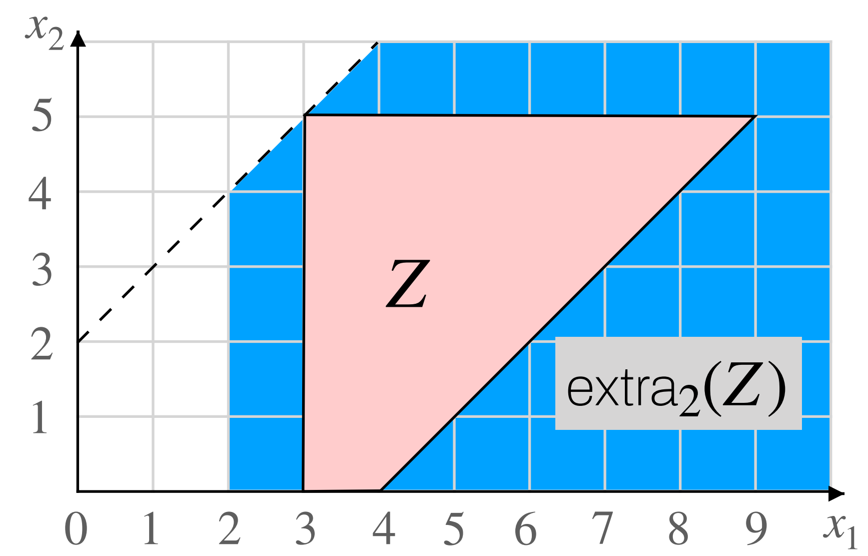


- Extrapolation [DT98,Bou03,Bou04]
- State-dependent extrapolation [BBFL03]
- LU-extrapolation [BBLP04,BBLP06]

➡ Soundness requires to show that extra is a simulation-based abstraction: $\forall v' \in \mathrm{extra}(Z)\ \exists v \in Z$ s.t. $v' \preceq v$

Extrapolation

Remove « irrelevant »
constants w.r.t. the automaton
⇝ syntactic on the DBM



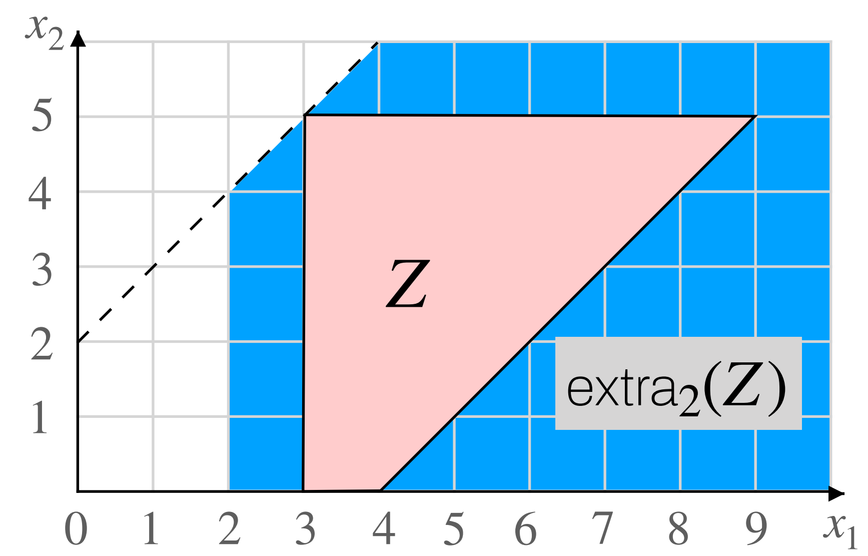
- Extrapolation [DT98,Bou03,Bou04]
- State-dependent extrapolation [BBFL03]
- LU-extrapolation [BBLP04,BBLP06]

Standard (resp. LU-)extrapolation is sound thanks to the region equivalence (resp. LU-simulation)

➡ Soundness requires to show that extra is a simulation-based abstraction: $\forall v' \in \text{extra}(Z)\ \exists v \in Z$ s.t. $v' \preceq v$

Extrapolation

Remove « irrelevant »
constants w.r.t. the automaton
⇝ syntactic on the DBM



- Extrapolation [DT98,Bou03,Bou04]
- State-dependent extrapolation [BBFL03]
- LU-extrapolation [BBLP04,BBLP06]

Standard (resp. LU-)extrapolation is sound thanks to the region equivalence (resp. LU-simulation)

➡ Soundness requires to show that extra is a simulation-based abstraction: $\forall v' \in \text{extra}(Z)\ \exists v \in Z$ s.t. $v' \preceq v$

[DT98] Daws, Tripakis: Model-checking of real-time reachability properties using abstractions (TACAS'98)
[Bou03] Bouyer: Untameable timed automata! (STACS'03)
[Bou04] Bouyer: Forward analysis of updatable timed automata (FMSD)
[BBFL03] Behrmann, Bouyer, Fleury, Larsen: Static guard analysis in timed automata verification (TACAS'03)
[BBLP04] Behrmann, Bouyer, Larsen, Pelánek: Lower and upper bounds in zone based abstractions of timed automata (TACAS'04)
[BBLP06] Behrmann, Bouyer, Larsen, Pelánek: Zone-based abstractions for timed automata exploiting lower and upper bounds (STTT)

Limits of the extrapolation approach

[Bou03] Bouyer: Untameable timed automata! (STACS'03)
[BBLP06] Behrmann, Bouyer, Larsen, Pelánek: Zone-based abstractions for timed automata exploiting lower and upper bounds (STTT)
[HKSW11] Herbreteau, Kini, Srivathsan, Walukiewicz: Using non-convex approximations for efficient analysis of timed automata (FSTTCS'11)

Limits of the extrapolation approach

➡ The extrapolation is required to transform a zone into a zone

[Bou03] Bouyer: Untameable timed automata! (STACS'03)
[BBLP06] Behrmann, Bouyer, Larsen, Pelánek: Zone-based abstractions for timed automata exploiting lower and upper bounds (STTT)
[HKSW11] Herbreteau, Kini, Srivathsan, Walukiewicz: Using non-convex approximations for efficient analysis of timed automata (FSTTCS'11)

Limits of the extrapolation approach

- ➡ The extrapolation is required to transform a zone into a zone
- ➡ It does not benefit from the coarsest abstractions of zones [HKSW11]
  - The region closure would in principle be sound, but it is not convex
  - The LU-abstraction $\mathfrak{a}_{LU}(Z) = \{v' \mid \exists v \in Z \text{ s.t. } v' \preceq_{LU} v\}$ would in principle be sound [BBLP06], but it is not convex

[Bou03] Bouyer: Untameable timed automata! (STACS'03)
[BBLP06] Behrmann, Bouyer, Larsen, Pelánek: Zone-based abstractions for timed automata exploiting lower and upper bounds (STTT)
[HKSW11] Herbreteau, Kini, Srivathsan, Walukiewicz: Using non-convex approximations for efficient analysis of timed automata (FSTTCS'11)

Limits of the extrapolation approach

➡ The extrapolation is required to transform a zone into a zone

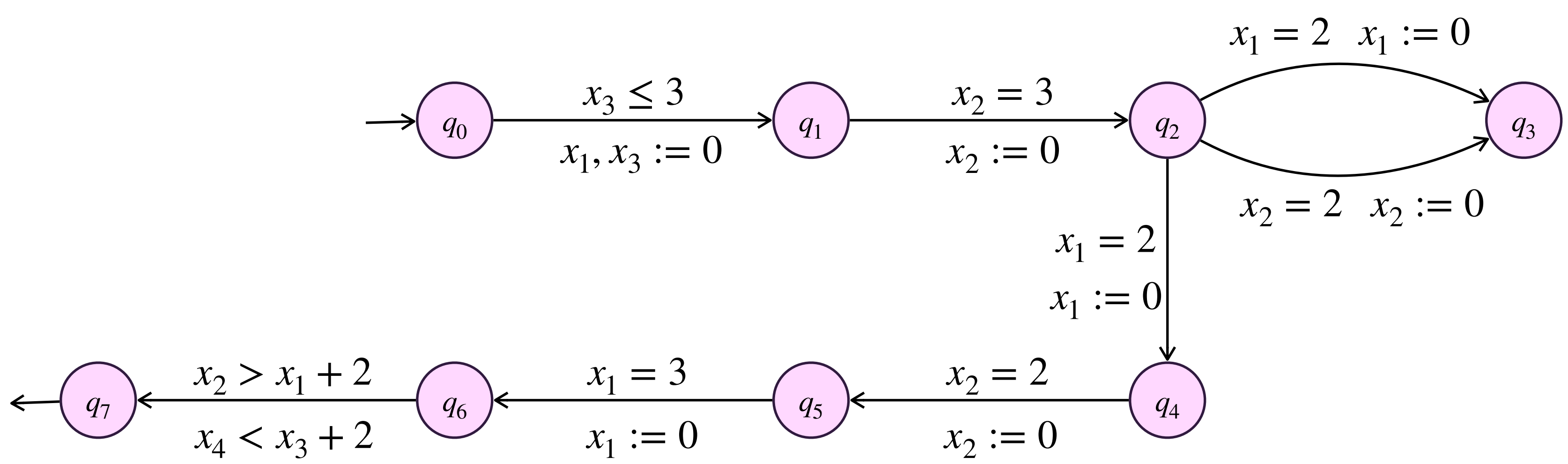
➡ It does not benefit from the coarsest abstractions of zones [HKSW11]

- The region closure would in principle be sound, but it is not convex
- The LU-abstraction $\mathfrak{a}_{LU}(Z) = \{v' \mid \exists v \in Z \text{ s.t. } v' \preceq_{LU} v\}$ would in principle be sound [BBLP06], but it is not convex

➡ The approach does not apply to timed automata with diagonal constraints [Bou03]

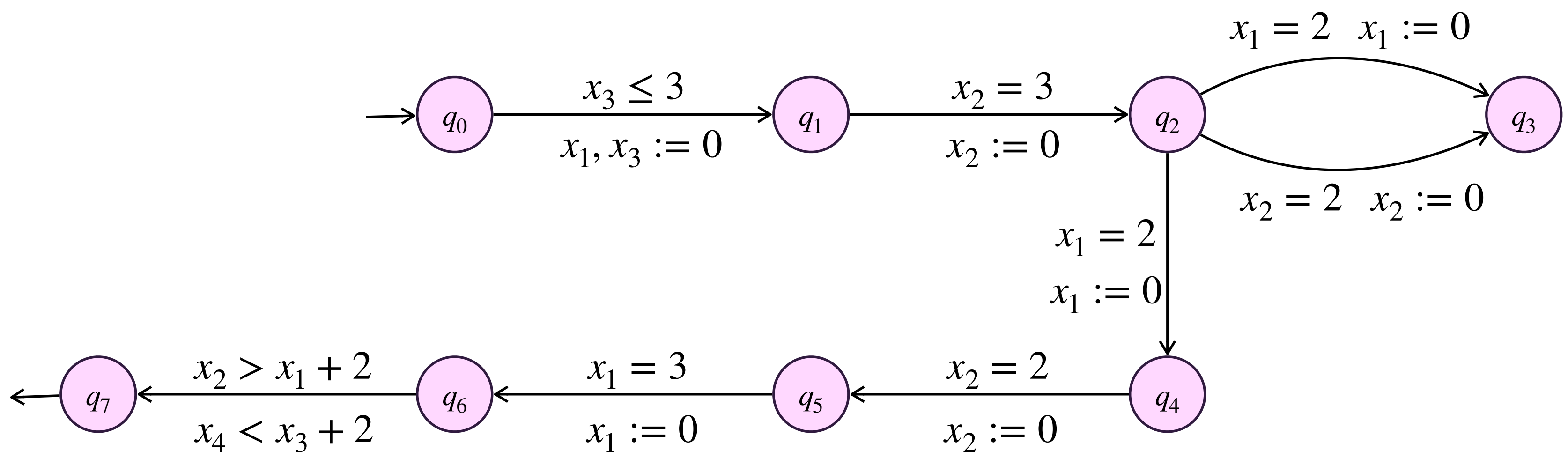
[Bou03] Bouyer: Untameable timed automata! (STACS'03)
[BBLP06] Behrmann, Bouyer, Larsen, Pelánek: Zone-based abstractions for timed automata exploiting lower and upper bounds (STTT)
[HKSW11] Herbreteau, Kini, Srivathsan, Walukiewicz: Using non-convex approximations for efficient analysis of timed automata (FSTTCS'11)

The buggy automaton



[Bou03] Bouyer. Untameable timed automata! (STACS'03).
[Bou04] Bouyer. Forward analysis of updatable timed automata (Formal Methods in System Design).

The buggy automaton

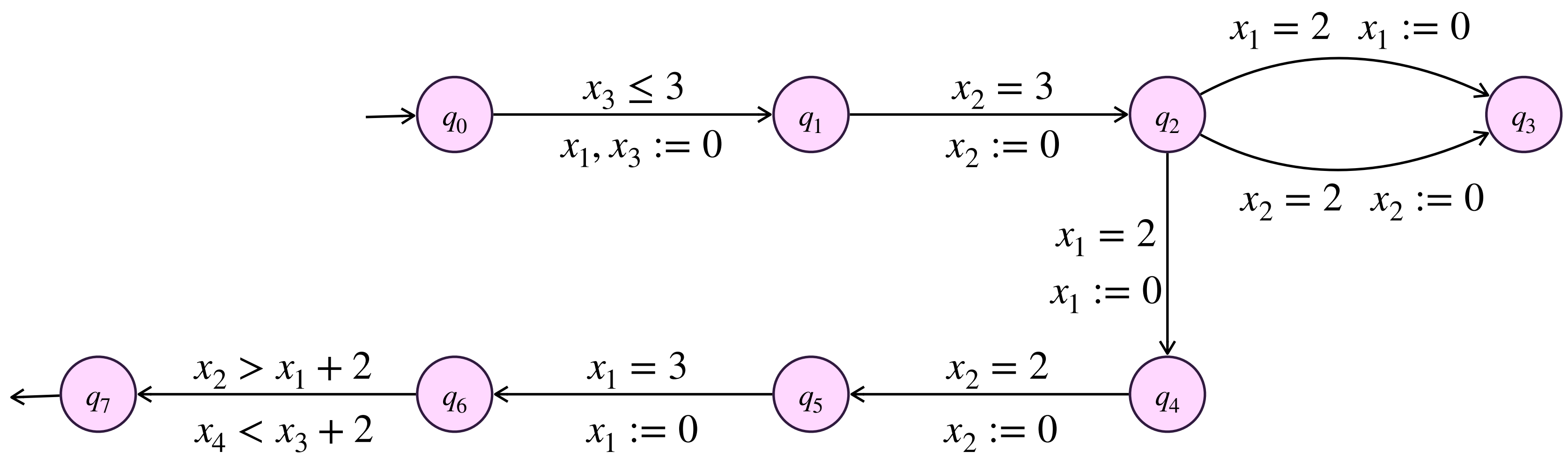


After α loops, the zone which is reached at q_6 is

$$Z_\alpha \ := \ (1 \leq x_2 - x_1 \leq 3) \wedge (1 \leq x_4 - x_3 \leq 3) \wedge (x_4 - x_2 = x_3 - x_1 = 2\alpha + 5)$$

[Bou03] Bouyer. Untameable timed automata! (STACS'03).
[Bou04] Bouyer. Forward analysis of updatable timed automata (Formal Methods in System Design).

The buggy automaton



After α loops, the zone which is reached at q_6 is

$$Z_\alpha \;:=\; (1 \leq x_2 - x_1 \leq 3) \wedge (1 \leq x_4 - x_3 \leq 3) \wedge (x_4 - x_2 = x_3 - x_1 = 2\alpha + 5)$$

- There is no extrapolation, which preserves zones, which is sound and finite for this timed automaton with diagonal constraints.

[Bou03] Bouyer. Untameable timed automata! (STACS'03).
[Bou04] Bouyer. Forward analysis of updatable timed automata (Formal Methods in System Design).

The simulation approach

The simulation approach

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add (q', Z') to $\mathcal{S}$, where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$ unless there is $(q', Z'') \in \mathcal{S}$ s.t. $Z' \preceq Z''$

The simulation approach

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add (q', Z') to $\mathcal{S}$,
    where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$
    unless there is $(q', Z'') \in \mathcal{S}$ s.t. $Z' \preceq Z''$

Properties

- *Termination is ensured if we require $\preceq$ has a finite-chain property*
- Soundness is obvious
- *Completeness relies on a simulation property for $\preceq$*

The simulation approach

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add (q', Z') to $\mathcal{S}$,
    where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$
    unless there is $(q', Z'') \in \mathcal{S}$ s.t. $Z' \preceq Z''$

Inclusion « up-to » simulation

Properties

- *Termination is ensured if we require $\preceq$ has a finite-chain property*
- Soundness is obvious
- *Completeness relies on a simulation property for $\preceq$*

The simulation approach

- Initialize $\mathcal{S}$ with $(q_0, \overrightarrow{0})$
- Repeat until saturation:
  - If $(q, Z) \in \mathcal{S}$, then add (q', Z') to $\mathcal{S}$, where $Z' = \overrightarrow{[Y](Z \cap g)}$ is the successor via $q \xrightarrow{g,Y} q'$ unless there is $(q', Z'') \in \mathcal{S}$ s.t. $Z' \preceq Z''$

Should be computationally efficient!

Inclusion « up-to » simulation

Properties

- *Termination is ensured if we require $\preceq$ has a finite-chain property*
- Soundness is obvious
- *Completeness relies on a simulation property for $\preceq$*

Simulation $\preceq$

If

$$\begin{tikzcd} (q, v_1) \arrow[d, "\delta"'] & \preceq & (q, v_2) \\ (q, v_1 + \delta) & & \end{tikzcd}$$

Simulation $\preceq$

$$\begin{tikzcd} \text{If} & (q, v_1) \arrow[d, "\delta"'] & \preceq & (q, v_2) \arrow[d, "\delta"] \\ \text{then} & (q, v_1 + \delta) & \preceq & (q, v_2 + \delta) \end{tikzcd}$$

Simulation $\preceq$

If $(q, v_1) \quad \preceq \quad (q, v_2)$

then $(q, v_1 + \delta) \quad \preceq \quad (q, v_2 + \delta)$

$$\begin{tikzcd}
(q, v_1) \arrow[d, "\delta"'] & \preceq & (q, v_2) \arrow[d, "\delta"] \\
(q, v_1 + \delta) & \preceq & (q, v_2 + \delta)
\end{tikzcd}$$

If $(q, v_1) \quad \preceq \quad (q, v_2)$

$$\begin{tikzcd}
(q, v_1) \arrow[d, "t"'] & \preceq & (q, v_2) \\
(q', v_1') & &
\end{tikzcd}$$

Simulation $\preceq$

If $(q, v_1) \preceq (q, v_2)$

$$\Big\downarrow \delta \qquad\qquad \Big\downarrow \delta$$

then $(q, v_1 + \delta) \preceq (q, v_2 + \delta)$

If $(q, v_1) \preceq (q, v_2)$

$$\Big\downarrow t \qquad\qquad \Big\downarrow t$$

then $(q', v_1') \preceq (q', v_2')$

Simulation $\preceq$

$$\begin{tikzcd} \text{If} & (q, v_1) \arrow[d, "\delta"'] & \preceq & (q, v_2) \arrow[d, "\delta"] \\ \text{then} & (q, v_1 + \delta) & \preceq & (q, v_2 + \delta) \end{tikzcd}$$

$$\begin{tikzcd} \text{If} & (q, v_1) \arrow[d, "t"'] & \preceq & (q, v_2) \arrow[d, "t"] \\ \text{then} & (q', v_1') & \preceq & (q', v_2') \end{tikzcd}$$

Inclusion « up-to » simulation

$(q, Z_1) \preceq (q, Z_2)$ iff $\forall v_1 \in Z_1, \exists v_2 \in Z_2$ s.t. $(q, v_1) \preceq (q, v_2)$

Simulation $\preceq$

If $(q, v_1) \quad \preceq \quad (q, v_2)$

$$\begin{tikzcd} (q, v_1) \arrow[d, "\delta"'] & \preceq & (q, v_2) \arrow[d, "\delta"] \\ (q, v_1 + \delta) & \preceq & (q, v_2 + \delta) \end{tikzcd}$$

then $(q, v_1 + \delta) \quad \preceq \quad (q, v_2 + \delta)$

If $(q, v_1) \quad \preceq \quad (q, v_2)$

$$\begin{tikzcd} (q, v_1) \arrow[d, "t"'] & \preceq & (q, v_2) \arrow[d, "t"] \\ (q', v_1') & \preceq & (q', v_2') \end{tikzcd}$$

then $(q', v_1') \quad \preceq \quad (q', v_2')$

Inclusion « up-to » simulation

$(q, Z_1) \preceq (q, Z_2)$ iff $\forall v_1 \in Z_1, \exists v_2 \in Z_2$ s.t. $(q, v_1) \preceq (q, v_2)$

- Note: $(q, Z_1) \preceq (q, Z_2)$ iff $Z_1 \subseteq$ Closure$_{\preceq}(Z_2)$
  iff Closure$_{\preceq}(Z_1) \subseteq$ Closure$_{\preceq}(Z_2)$

And concretely?

And concretely?

It has the finite-chain property

- The region equivalence [HKSW11]

The corresponding inclusion « up-to » can be decided in $\mathcal{O}(|X|^2)$

And concretely?

It has the finite-chain property

- The region equivalence [HKSW11]
- The LU-simulation [HSW12]

The corresponding inclusion « up-to » can be decided in $\mathcal{O}(|X|^2)$

And concretely?

It has the finite-chain property

- The region equivalence [HKSW11]
- The LU-simulation [HSW12]
- The $\mathscr{G}$-simulation [GMS18,GMS19,GMS20]

The corresponding inclusion « up-to » can be decided in $\mathcal{O}(|X|^2)$

And concretely?

It has the finite-chain property

- The region equivalence [HKSW11]
- The LU-simulation [HSW12]

The corresponding inclusion « up-to » can be decided in $\mathcal{O}(|X|^2)$

- The $\mathcal{G}$-simulation [GMS18,GMS19,GMS20]
  - It is coarser than the LU-simulation for diagonal-free automata

And concretely?

It has the finite-chain property

- The region equivalence [HKSW11]
- The LU-simulation [HSW12]

The corresponding inclusion « up-to » can be decided in $\mathcal{O}(|X|^2)$

- The $\mathscr{G}$-simulation [GMS18,GMS19,GMS20]
  - It is coarser than the LU-simulation for diagonal-free automata
  - It is correct for timed automata with diagonal constraints!

The corresponding inclusion « up-to » is NP-complete

And concretely?

It has the finite-chain property

- The region equivalence [HKSW11]
- The LU-simulation [HSW12]
- The $\mathscr{G}$-simulation [GMS18,GMS19,GMS20]
  - It is coarser than the LU-simulation for diagonal-free automata
  - It is correct for timed automata with diagonal constraints!
  - Adapts to (« decidable ») automata with updates

The corresponding inclusion « up-to » can be decided in $\mathcal{O}(|X|^2)$

The corresponding inclusion « up-to » is NP-complete

And concretely?

It has the finite-chain property

- The region equivalence [HKSW11]
- The LU-simulation [HSW12]

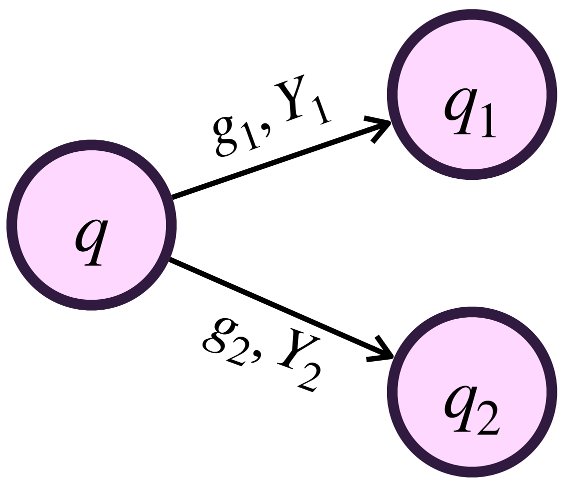
The corresponding inclusion « up-to » can be decided in $\mathcal{O}(|X|^2)$

- The $\mathcal{G}$-simulation [GMS18,GMS19,GMS20]
  - It is coarser than the LU-simulation for diagonal-free automata
  - It is correct for timed automata with diagonal constraints!
  - Adapts to (« decidable ») automata with updates

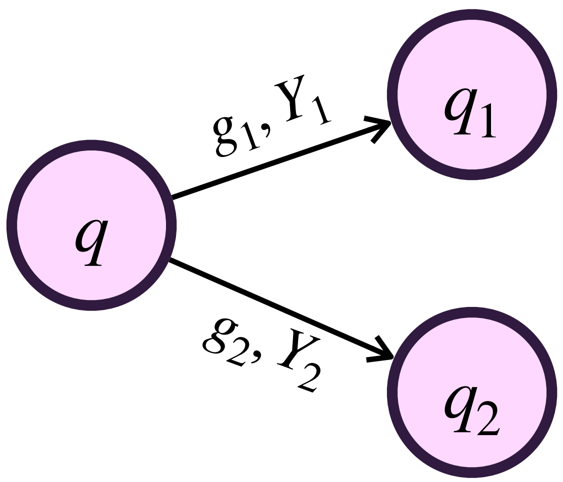
The corresponding inclusion « up-to » is NP-complete

[HKSW11] Herbreteau, Kini, Srivathsan, Walukiewicz: Using non-convex approximations for efficient analysis of timed automata (FSTTCS'11)
[HSW12] Herbreteau, Srivathsan, Walukiewicz: Better Abstractions for Timed Automata (LICS'12)
[GMS18] Gastin, Mukherjee, Srivathsan: Reachability in Timed Automata with Diagonal Constraints (CONCUR'18)
[GMS19] Gastin, Mukherjee, Srivathsan: Fast Algorithms for Handling Diagonal Constraints in Timed Automata (CAV'19)
[GMS20] Gastin, Mukherjee, Srivathsan: Reachability for Updatable Timed Automata Made Faster and More Effective (FSTTCS'20)

Constraints relevant at q: $\mathscr{G}(q)$

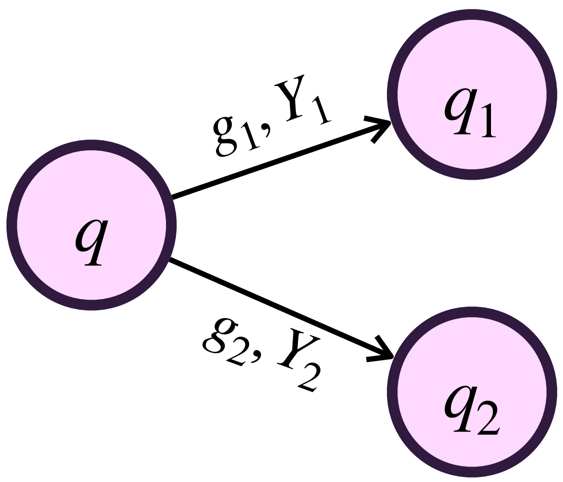


Constraints relevant at q: $\mathcal{G}(q)$



$$\begin{cases} \{g_1, g_2\} \subseteq \mathcal{G}(q) \\ \mathsf{pre}(\mathcal{G}(q_i), Y_i) \subseteq \mathcal{G}(q) \end{cases}$$

Constraints relevant at q: $\mathscr{G}(q)$

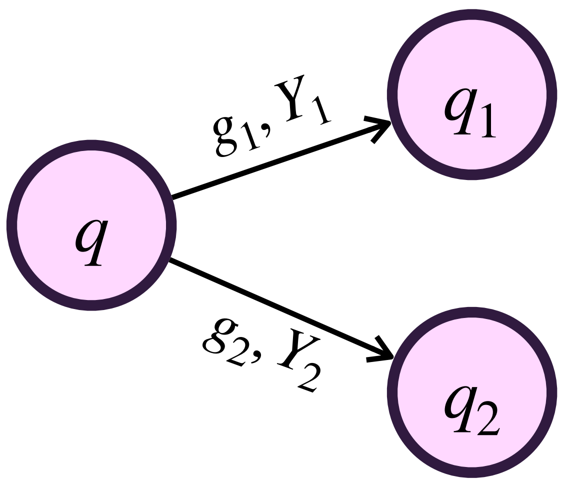


$$\begin{cases} \{g_1, g_2\} \subseteq \mathscr{G}(q) \\ \mathsf{pre}(\mathscr{G}(q_i), Y_i) \subseteq \mathscr{G}(q) \end{cases}$$

$$\mathsf{pre}(x \bowtie c, Y) = \begin{cases} \{x \bowtie c\} & \text{if } x \notin Y \\ \varnothing & \text{if } x \in Y \end{cases}$$

$$\mathsf{pre}(x - y \bowtie c, Y) = \begin{cases} \{x - y \bowtie c\} & \text{if } x, y \notin Y \\ \{x \bowtie c\} & \text{if } x \notin Y, y \in Y \\ \{-y \bowtie c\} & \text{if } x \in Y, y \notin Y \\ \varnothing & \text{if } x, y \in Y \end{cases}$$

Constraints relevant at q: $\mathscr{G}(q)$



$$\begin{cases} \{g_1, g_2\} \subseteq \mathscr{G}(q) \\ \mathrm{pre}(\mathscr{G}(q_i), Y_i) \subseteq \mathscr{G}(q) \end{cases}$$

$$\mathrm{pre}(x \bowtie c, Y) = \begin{cases} \{x \bowtie c\} & \text{if } x \notin Y \\ \varnothing & \text{if } x \in Y \end{cases}$$

$$\mathrm{pre}(x - y \bowtie c, Y) = \begin{cases} \{x - y \bowtie c\} & \text{if } x, y \notin Y \\ \{x \bowtie c\} & \text{if } x \notin Y, y \in Y \\ \{-y \bowtie c\} & \text{if } x \in Y, y \notin Y \\ \varnothing & \text{if } x, y \in Y \end{cases}$$

- Fixpoint computation terminates for timed automata; it also terminates for known decidable classes of updatable timed automata

The $\mathscr{G}$-simulation

The $\mathcal{G}$-simulation

Let $\mathcal{G}$ be the previous mapping

- We say that $(q, v) \preceq_{\mathcal{G}} (q, v')$ whenever for every $\varphi \in \mathcal{G}$, for every $\delta \geq 0$, $v + \delta \models \varphi$ implies $v' + \delta \models \varphi$

The $\mathcal{G}$-simulation

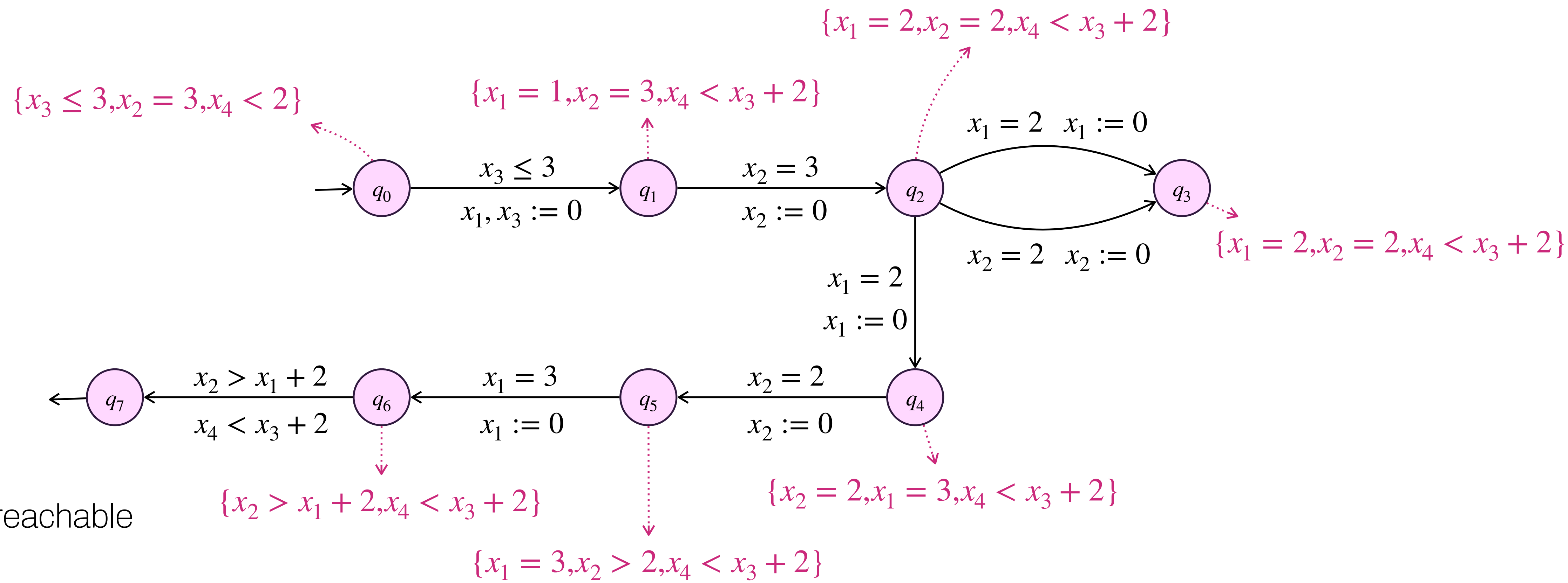
Let $\mathcal{G}$ be the previous mapping

- We say that $(q, v) \preceq_{\mathcal{G}} (q, v')$ whenever for every $\varphi \in \mathcal{G}$, for every $\delta \geq 0$, $v + \delta \models \varphi$ implies $v' + \delta \models \varphi$

Theorem

- $\preceq_{\mathcal{G}}$ is a simulation relation
- It satisfies the finite-chain property on zones

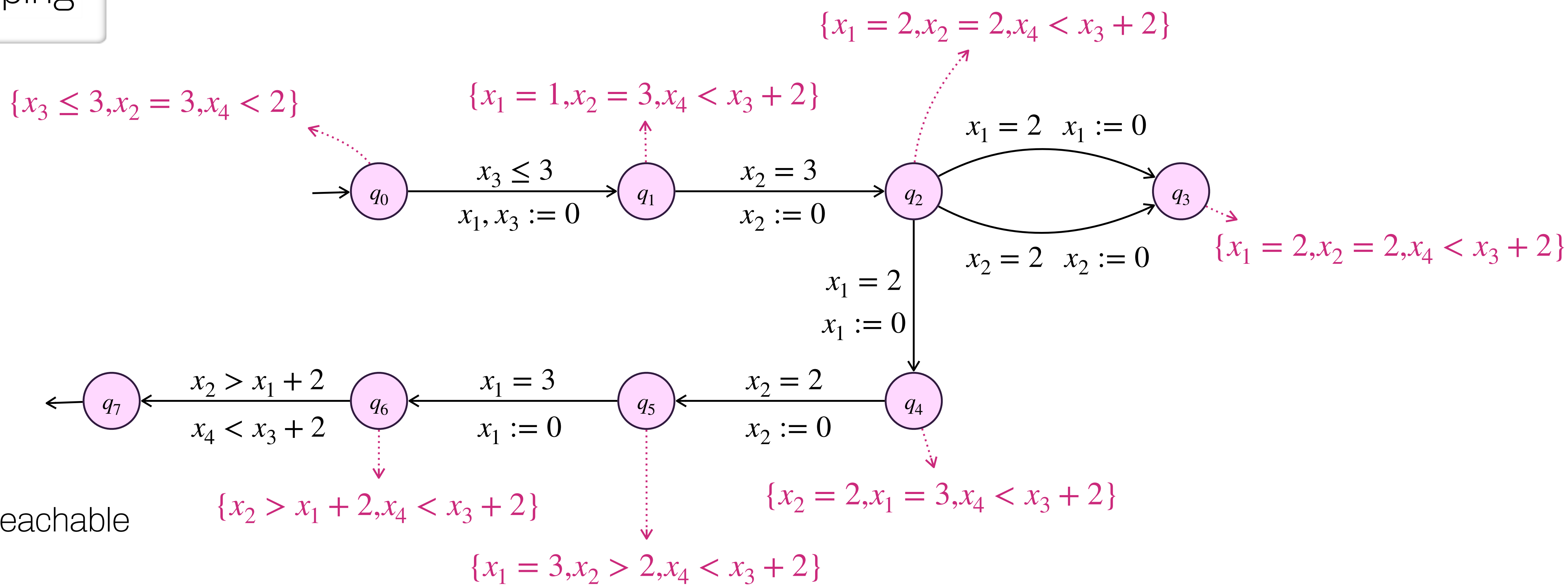
An example



q_7 is not reachable

An example

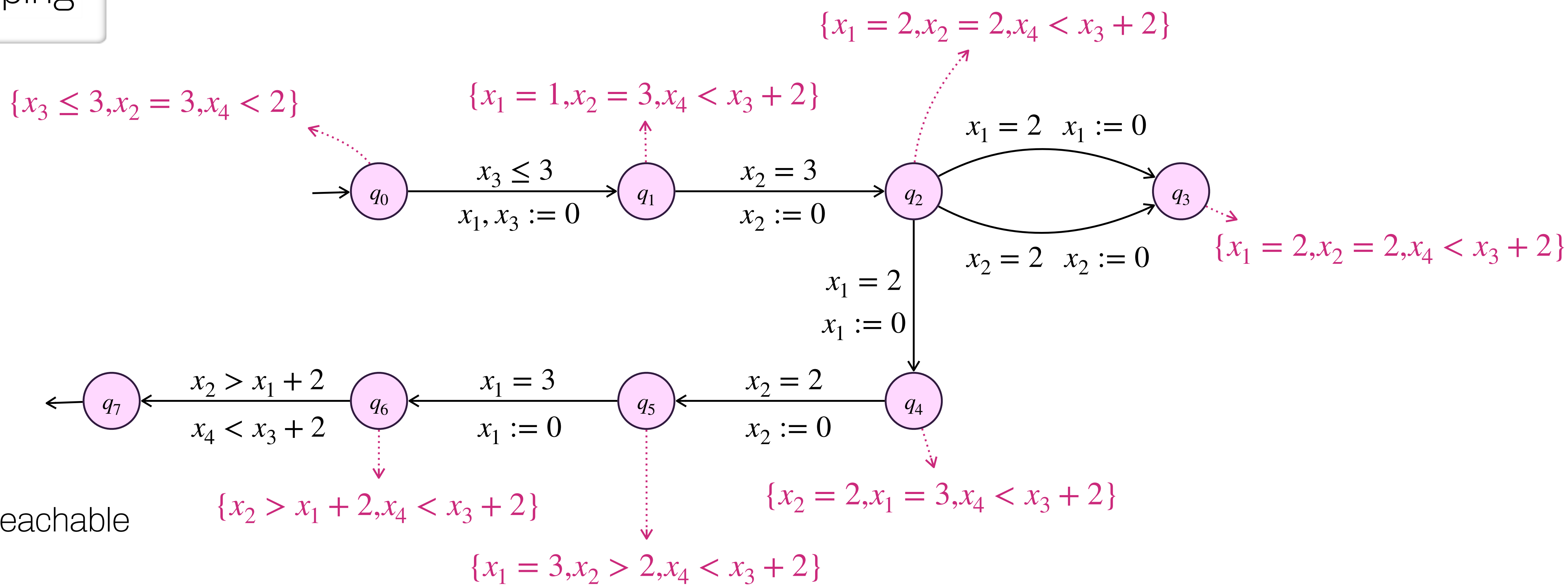
The $\mathscr{G}$ mapping



q_7 is not reachable

An example

The $\mathscr{G}$ mapping



q_7 is not reachable

- On this automaton, any extrapolation-based method fails [Bou04]
- The $\preceq_{\mathscr{G}}$-simulation approach terminates at the second iteration

Going further

Going further

- Liveness properties [HSWT16,HSWT20]

Going further

- Liveness properties [HSWT16,HSWT20]
- Weighted timed automata [BCM16]
- Pushdown timed automata [AGP21]
  (talk of Akshay at SNR)
- Event-clock automata [AGGS22]

Going further

- Liveness properties [HSWT16,HSWT20]
- Weighted timed automata [BCM16]
- Pushdown timed automata [AGP21]
  (talk of Akshay at SNR)
- Event-clock automata [AGGS22]

[BCM16] Bouyer, Colange, Markey: Symbolic Optimal Reachability in Weighted Timed Automata (CAV'16)
[HSTW20] Herbreteau, Srivathsan, Tran, Walukiewicz: Why Liveness for Timed Automata Is Hard, and What We Can Do About It (ACM Trans. Comput. Log)
[AGP21] Akshay, Gastin, Prakash: Fast Zone-Based Algorithms for Reachability in Pushdown Timed Automata (CAV'21)
[AGGS22] Akshay, Gastin, Govind, Srivathsan: Simulations for Event-Clock Automata (CONCUR'22)

Tools

- Uppaal https://uppaal.org
- Tchecker https://github.com/ticktac-project/tchecker
- Red https://sites.google.com/site/redlibtw/
- Pat https://pat.comp.nus.edu.sg
- Rabbit https://www.sosy-lab.org/people/beyer/Rabbit/
- MCTA http://gki.informatik.uni-freiburg.de/tools/mcta/
- ...

Tool UPPAAL

https://uppaal.org

- Developed since 1995
- Successfully used in the industry, with many case studies
- Many extensions:
  - games, weighted timed automata, testing, statistical model-checking, ...
- Implements extrapolation-based algorithms

Tool TChecker

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

- Developed since a couple of years, under development
- Fully open-source verification tool for timed automata
- Implements extrapolation and simulation-based algorithms
- Made also as a framework to develop new verification algorithms or data structures

Conclusion

What next?

What next?

- Much **algorithmic effort** has been made to reduce the impact of the timing aspects (reduce the number of zones to visit)
  - Need to push the ideas to larger classes of models
  - In each case, one of the the difficulties lies in the proof of efficiency of inclusion « up-to »

What next?

- Much **algorithmic effort** has been made to reduce the impact of the timing aspects (reduce the number of zones to visit)
  - Need to push the ideas to larger classes of models
  - In each case, one of the the difficulties lies in the proof of efficiency of inclusion « up-to »

- A major bottleneck: the **state explosion** due to control states
  - Use of BDD/SAT technics, bounded model-checking, …
    → No technics overwrites the other, they are useful and complementary
  - Local-time semantics + POR (talk of Sri at SNR) [GHSW22]

[GHSW22] Govind, Herbreteau, Srivathsan, Walukiewicz: Abstractions for the local-time semantics of timed automata: a foundation for partial-order methods (LICS'22)

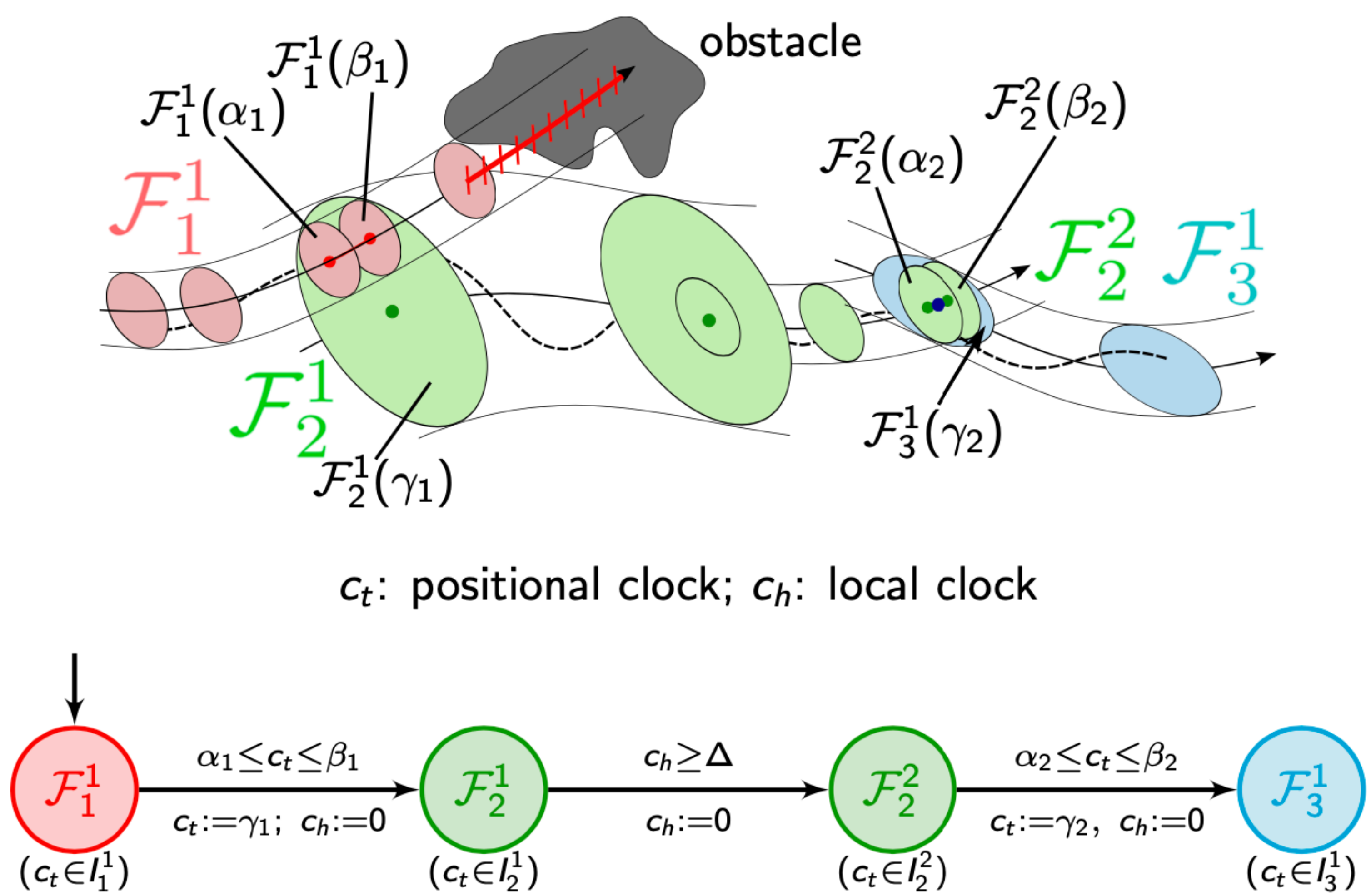
What next?

- **Domain-specific** algorithms:
  - Funnel automata for robotic systems [BMPS15,BMPS17]

[BMPS15] Bouyer, Markey, Perrin, Schlehuber-Caissier:Timed-Automata Abstraction of Switched Dynamical Systems Using Control Funnels (FORMATS'15)
[BMPS17] Bouyer, Markey, Perrin, Schlehuber-Caissier:Timed-automata abstraction of switched dynamical systems using control invariants (Real Time Syst.)

What next?

- **Domain-specific** algorithms:
  - Funnel automata for robotic systems [BMPS15,BMPS17]



[BMPS15] Bouyer, Markey, Perrin, Schlehuber-Caissier:Timed-Automata Abstraction of Switched Dynamical Systems Using Control Funnels (FORMATS’15)

[BMPS17] Bouyer, Markey, Perrin, Schlehuber-Caissier:Timed-automata abstraction of switched dynamical systems using control invariants (Real Time Syst.)

Thank you for your attention!