Fast algorithms for handling diagonal constraints in Timed Automata

Paul Gastin
LSV

Sayan Mukherjee
CMI

B. Srivathsan
CMI

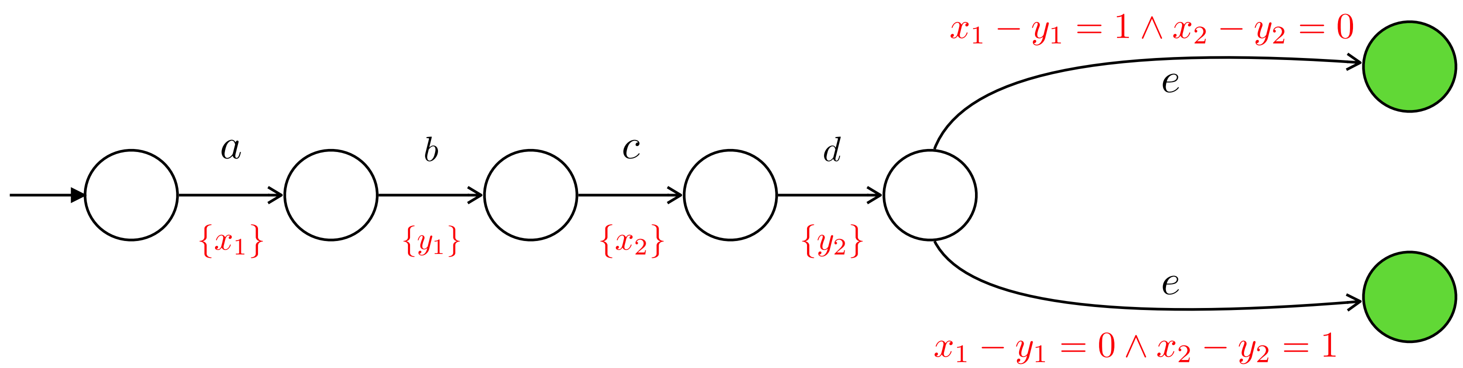
Supported by UMI ReLaX and ANR TickTac

Diagonal constraints in Timed Automata

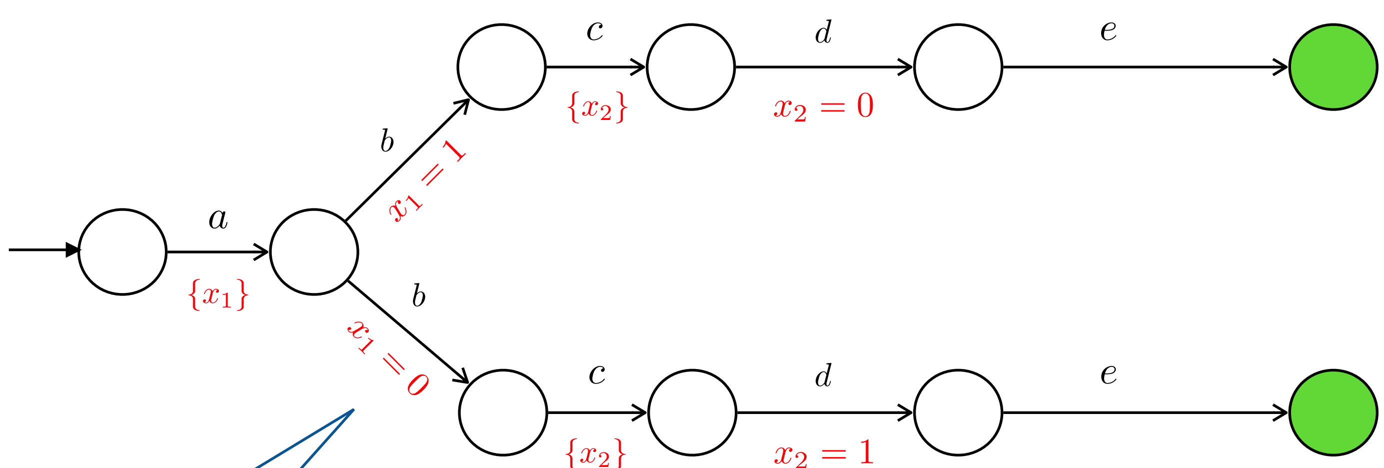
$g := x \lhd c \mid c \lhd x \mid x - y \lhd c \mid g \land g$

diagonal constraint

Timed automaton
with
diagonal constraints



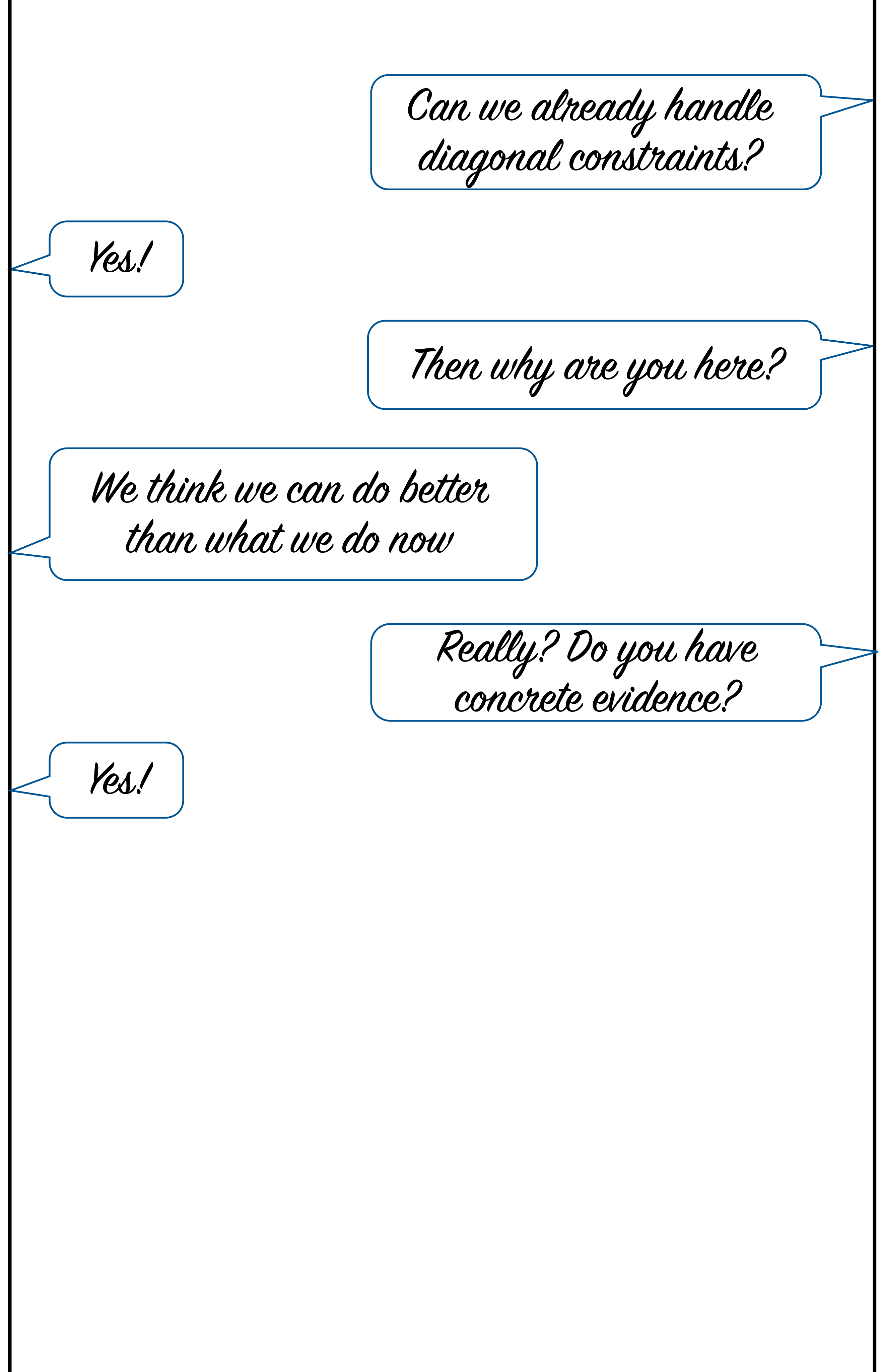
These two timed automata are language equivalent



Timed automaton
without
diagonal constraints

non-diagonal constraint

$g := x \lhd c \mid c \lhd x \mid g \land g$


Can we already handle diagonal constraints?
Yes!
Then why are you here?
We think we can do better than what we do now
Really? Do you have concrete evidence?
Yes!

Experimental Results

Based on TChecker (older version) Thanks Frédéric!

zone splitting + $Extra_M$

state blowup + $Extra_{LU}$

		$\mathcal{A}$: contains diagonals				$\mathcal{A}_{df}$: diagonal-free *euivalent* of $\mathcal{A}$			
		TChecker with $\sqsubseteq_{\mathcal{G}}^{LU}$		UPPAAL		UPPAAL		TChecker	
Model	$\#\mathcal{D}$	Time	Nodes count	Time	Nodes count	Time	Nodes count	Time	Nodes count
Unreachable									
Cex 1	2	0.004	7	0.001	26	0.001	17	0.006	17
Cex 2	4	0.047	241	0.026	2180	0.005	1039	0.067	1039
Cex 3	6	7.399	7111	111.168	182394	1.028	60982	40.092	60982
Cex 4	8	857.662	185209	timeout	-	734.543	3447119	timeout	-
Fischer 3	3	0.007	104	0.087	4272	0.001	268	0.013	268
Fischer 4	4	0.032	452	307.836	357687	0.009	1815	0.100	1815
Fischer 5	5	0.257	1842	timeout	-	0.116	12511	1.856	12511
Fischer 7	7	15.032	26812	timeout	-	174.560	693603	timeout	-
Job Shop 3	12	0.420	278	23.093	31711	0.003	845	0.312	845
Job Shop 5	20	285.421	10592	timeout	-	4.633	179607	150.811	179607
Job Shop 7	28	timeout	-	timeout	-	timeout	-	timeout	-
Reachable									
Job Shop 3	12	0.019	38	22.435	31607	0.004	839	0.013	29
Job Shop 5	20	0.279	98	timeout	-	4.552	179597	0.012	67
Job Shop 7	28	1.754	192	timeout	-	timeout	-	0.036	121
Job Shop 9	36	7.040	318	timeout	-	timeout	-	0.056	191

Exponential blowup (Unreachable rows)

#states almost the same (Reachable rows)

Sources:

- Cex → Bouyer '03 , Reynier '07 (tool report)
- Fischer → Reynier '07 (tool report)
- Job Shop → Abdeddaim, Asarin, Maler '06 (*added diagonal constraints*)

Can we already handle diagonal constraints?

Yes!

Then why are you here?

We think we can do better than what we do now

Really? Do you have concrete evidence?

Yes!

What are the existing methods?

Overview

Problem we are interested in: Practically efficient algorithms for reachability

Diagonal-free Timed Automata

Zone-based *forward analysis*

BY04, BBLP06, BBFL03, . . .
UPPAAL

HSW12, HT15
TChecker

Our contribution

"Correct" zone-based *forward analysis*, that works directly on diagonal automata

Motivation 1

Automata with diagonal constraints are exponentially more succinct [BC05]

Timed Automata with diagonal constraints

"Naive" zone-based forward analysis is wrong [Bou04]

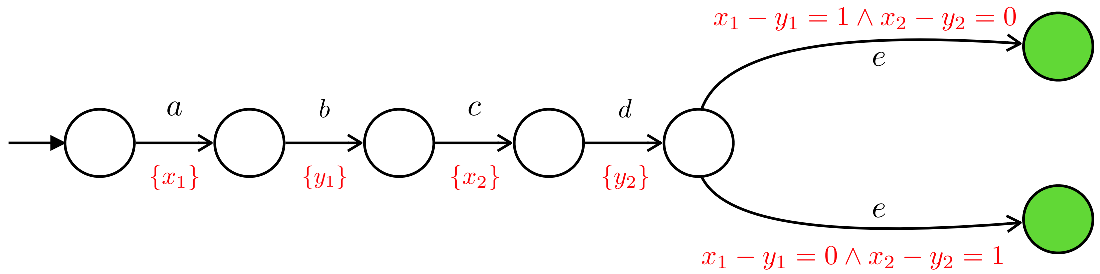
Current algorithms

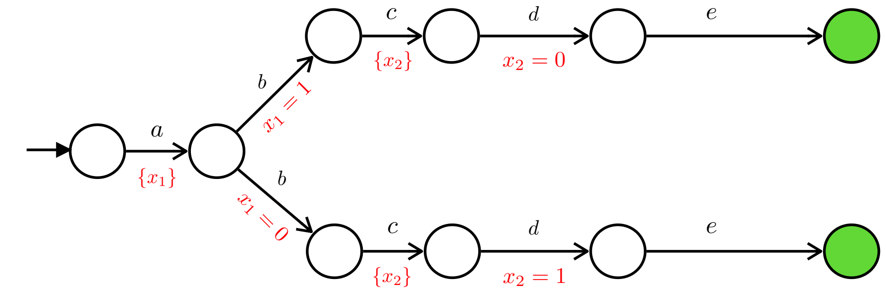
1. Convert *diagonal* to *diagonal-free* + apply zone-based forward analysis on *diagonal-free*
2. [BY04] Forward analysis with *Zone-splitting*
3. [BLR05] Abstraction refinement framework, zone-based *"forward-backward" analysis*

Motivation 2

Lift existing optimizations for diagonal-free automata to automata with diagonal constraints

Method 1 : Removing *all* diagonals





Timed automaton
with
diagonal constraints

Exponential blowup →

Timed automaton
without
diagonal constraints

Bérard, Diekert, Gastin, Petit '98 — This translation is always possible

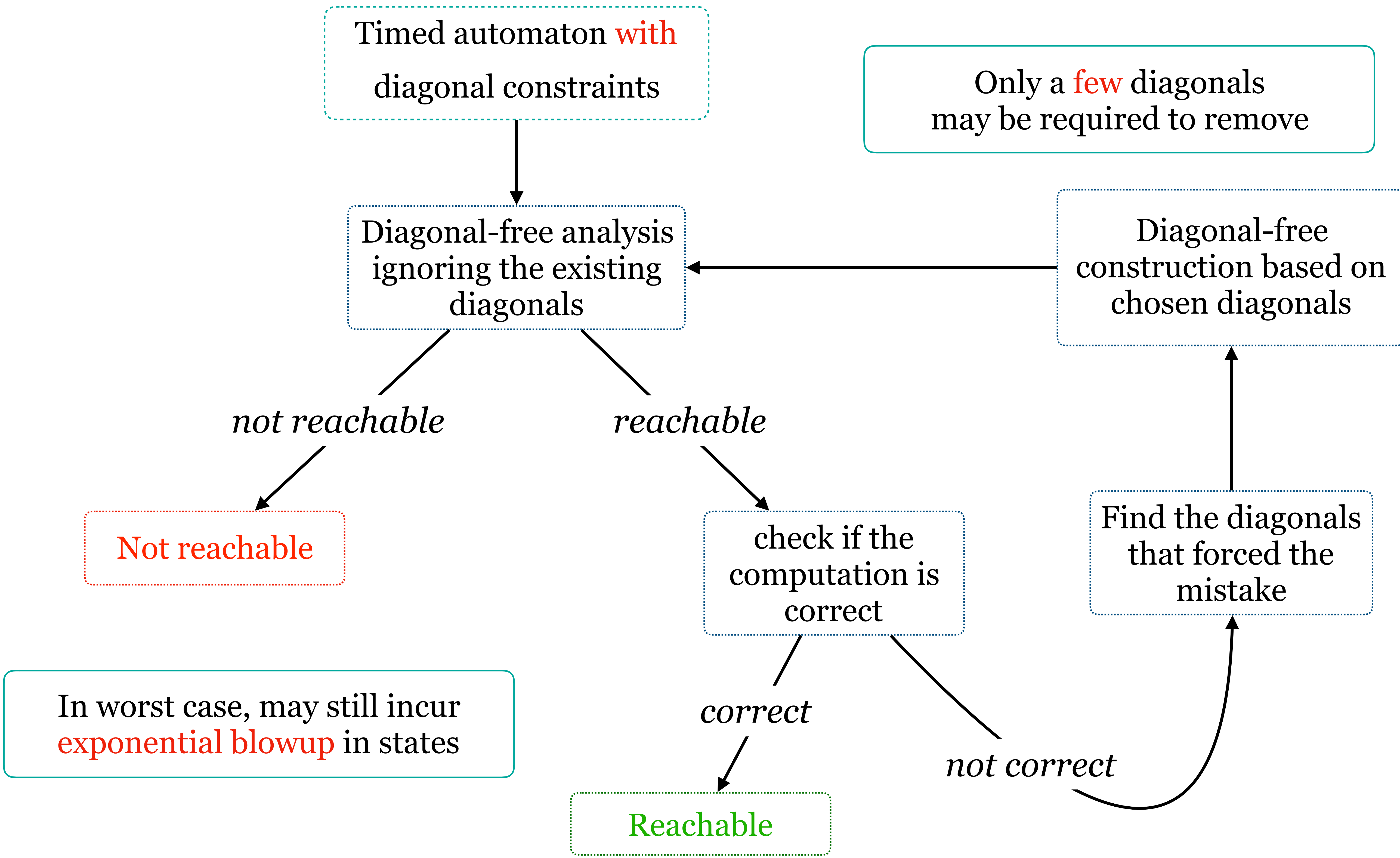
Bouyer, Chevalier '05 — In worst case, exponential blowup

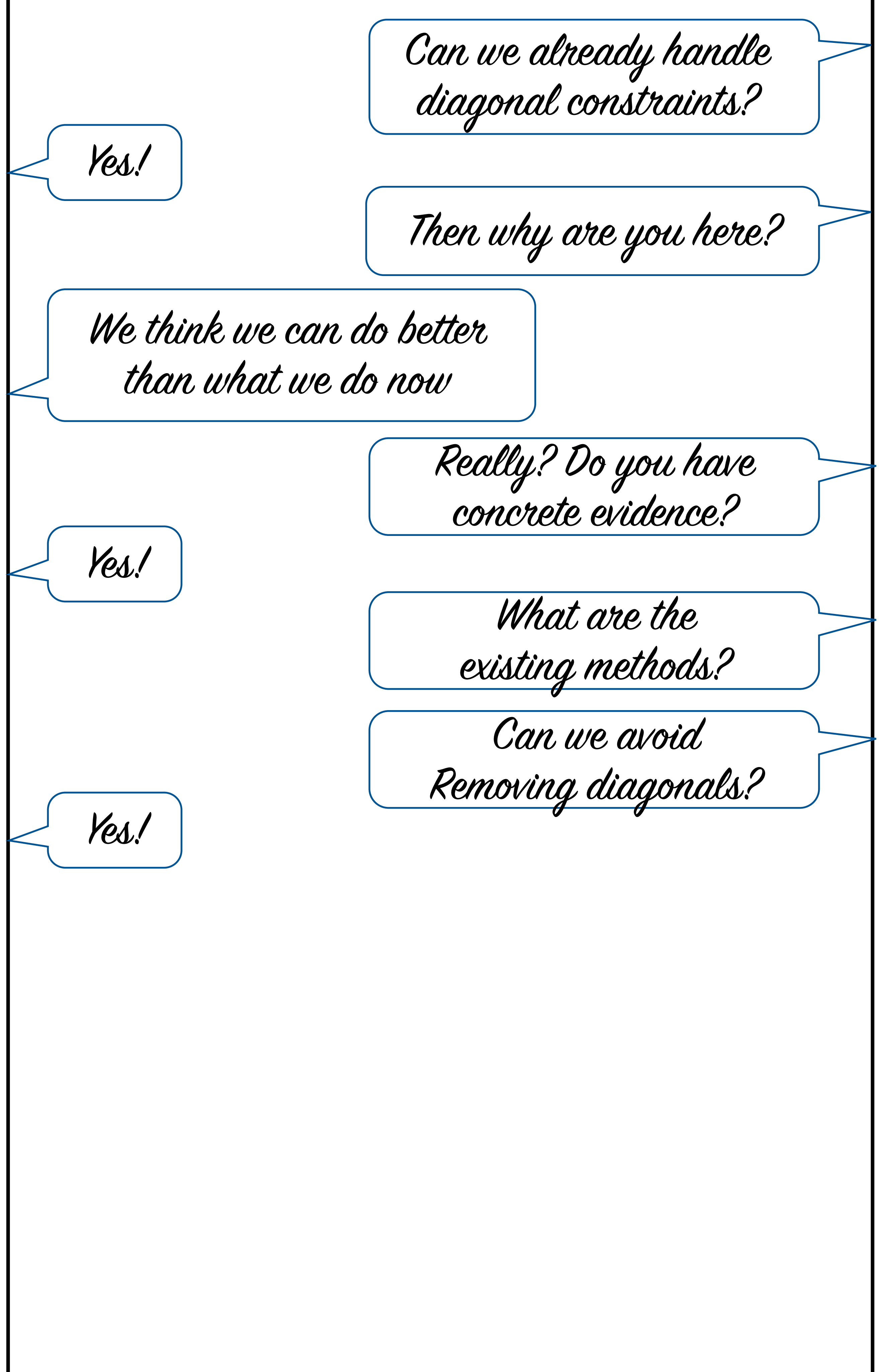
Algorithm is well studied

Zone based *forward analysis* on diagonal-free automata

Bouyer, Laroussinie, Reynier '05

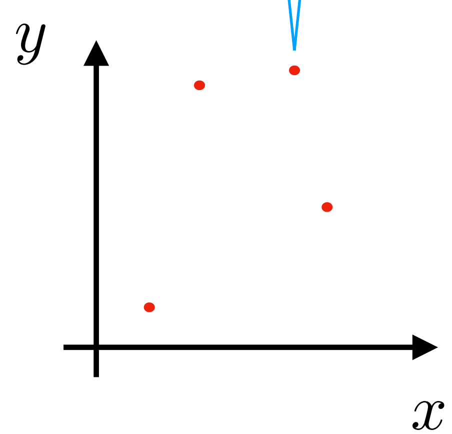
Method 2 : Removing *relevant* diagonals

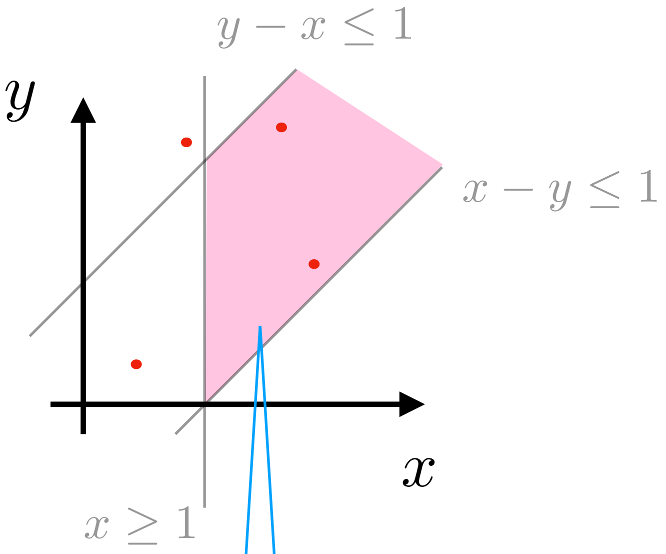



Can we already handle diagonal constraints?
Yes!
Then why are you here?
We think we can do better than what we do now
Really? Do you have concrete evidence?
Yes!
What are the existing methods?
Can we avoid Removing diagonals?
Yes!

Zones

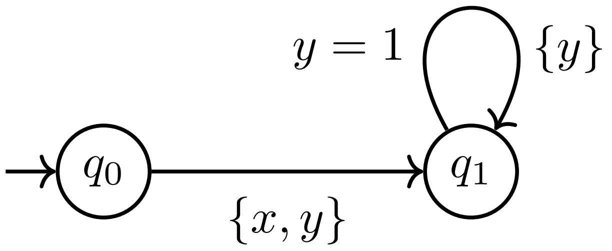
A *valuation* is a function $v : X \to \mathbb{R}_{\geq 0}$



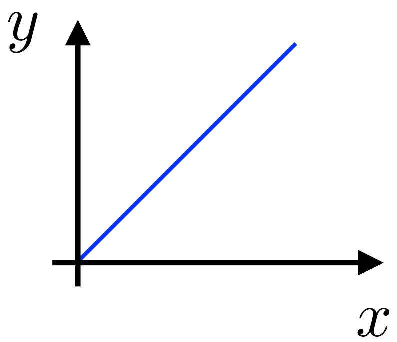


A *zone* is a collection of valuations described by a conjunction of constraints of the form $x - y \sim c$ or $x \sim c$, $\sim \in \{<, \leq, \geq, >\}$

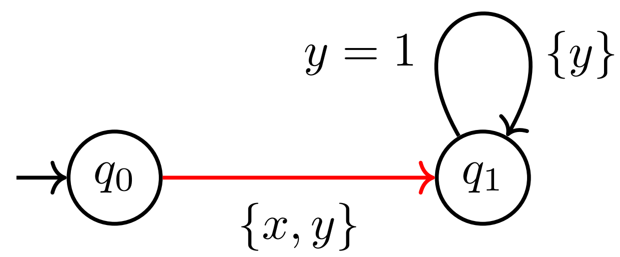
Zone based forward analysis



(q_0, Z_0)



Zone based forward analysis



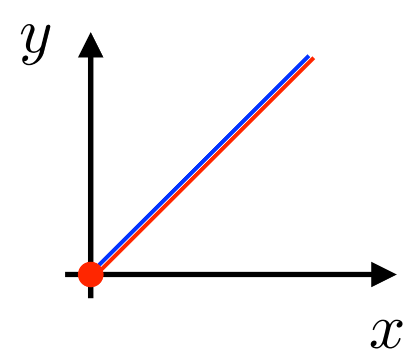
$$(q_0, {\color{blue}Z_0}) \longrightarrow (q_1, {\color{red}Z_1})$$

Successor computation

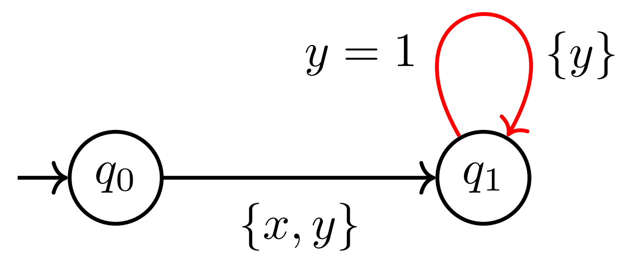
$$q \xrightarrow[R]{g} q'$$

$$(q, Z) \rightarrow (q', Z')$$

where $Z' = \overrightarrow{[R](Z \cap g)}$



Zone based forward analysis



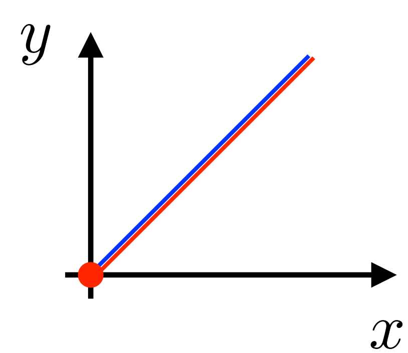
$$(q_0, Z_0) \longrightarrow (q_1, Z_1) \longrightarrow (q_1, Z_2)$$

Successor computation

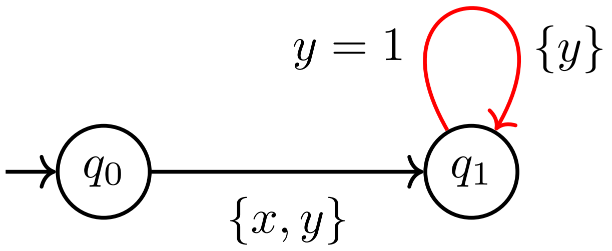
$$q \xrightarrow[R]{g} q'$$

$$(q, Z) \rightarrow (q', Z')$$

where $Z' = \overrightarrow{[R](Z \cap g)}$



Zone based forward analysis



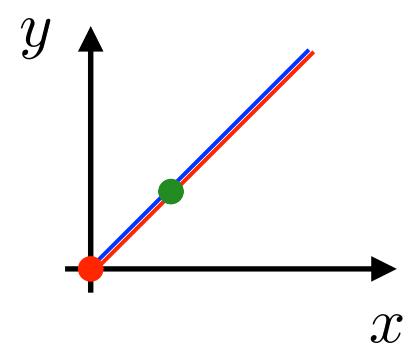
$$(q_0, Z_0) \longrightarrow (q_1, Z_1) \longrightarrow (q_1, Z_2)$$

Successor computation

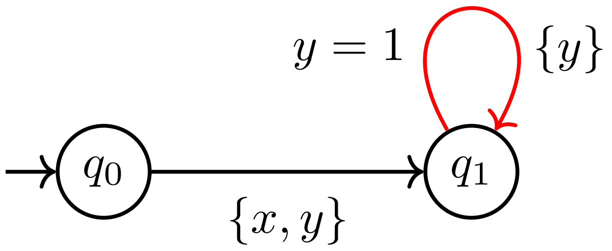
$$q \xrightarrow[R]{g} q'$$

$$(q, Z) \to (q', Z')$$

where $Z' = \overrightarrow{[R](Z \cap g)}$



Zone based forward analysis



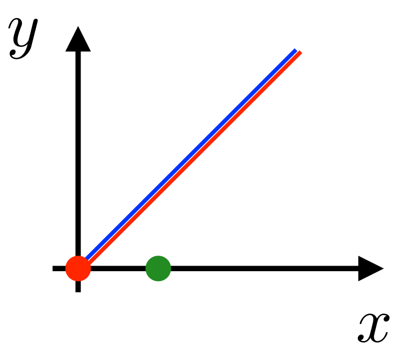
$$(q_0, Z_0) \longrightarrow (q_1, Z_1) \longrightarrow (q_1, Z_2)$$

Successor computation

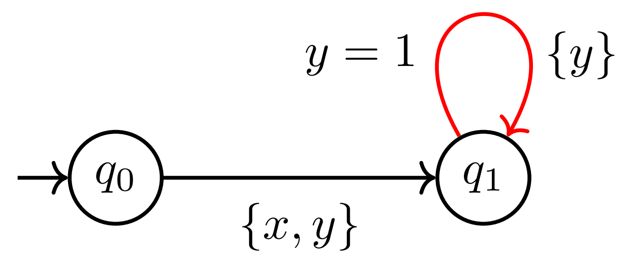
$$q \xrightarrow[R]{g} q'$$

$$(q, Z) \rightarrow (q', Z')$$

where $Z' = \overrightarrow{[R](Z \cap g)}$



Zone based forward analysis



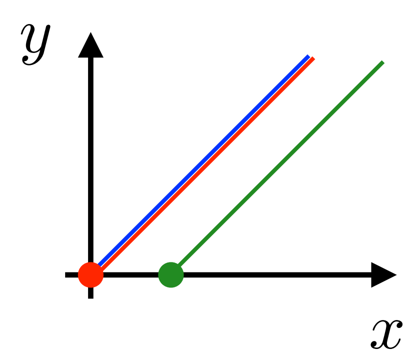
$$(q_0, {\color{blue}Z_0}) \longrightarrow (q_1, {\color{red}Z_1}) \longrightarrow (q_1, {\color{green}Z_2})$$

Successor computation

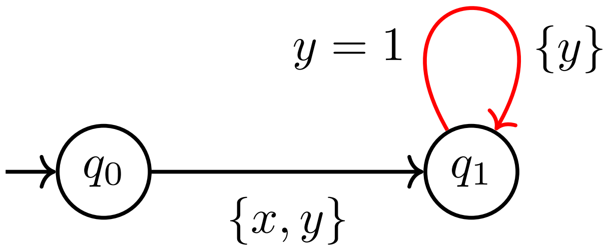
$$q \xrightarrow[R]{g} q'$$

$$(q, Z) \to (q', Z')$$

where $Z' = \overrightarrow{[R](Z \cap g)}$



Zone based forward analysis



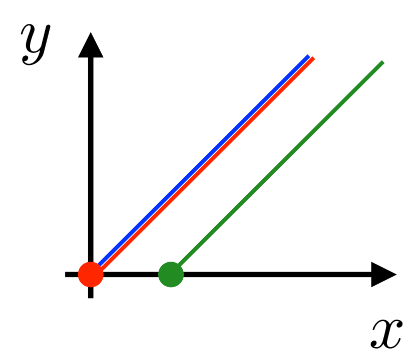
$$(q_0, Z_0) \longrightarrow (q_1, Z_1) \longrightarrow (q_1, Z_2)$$

Successor computation

$$q \xrightarrow[R]{g} q'$$

$$(q, Z) \rightarrow (q', Z')$$

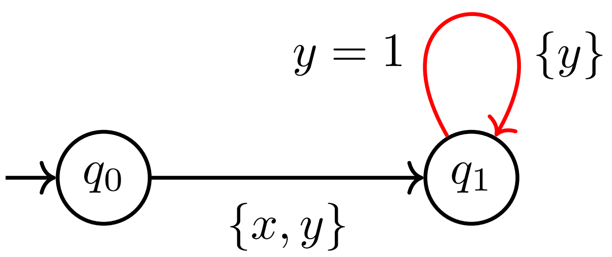
where $Z' = \overrightarrow{[R](Z \cap g)}$



Is $Z_2 \subseteq Z_1$?

No

Zone based forward analysis



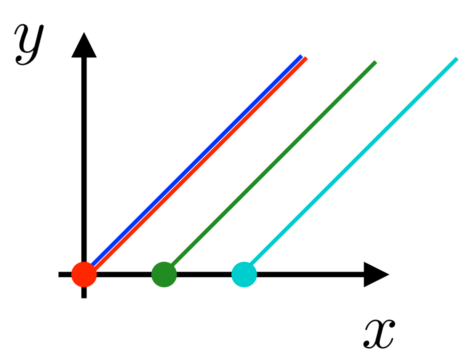
$$(q_0, Z_0) \longrightarrow (q_1, Z_1) \longrightarrow (q_1, Z_2) \longrightarrow (q_1, Z_3)$$

Successor computation

$$q \xrightarrow[R]{g} q'$$

$$(q, Z) \rightarrow (q', Z')$$

where $Z' = \overrightarrow{[R](Z \cap g)}$



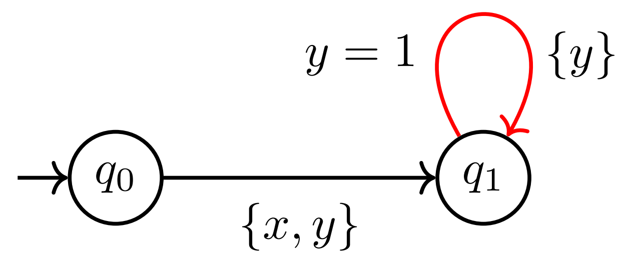
Is $Z_3 \subseteq Z_2$?

No

Is $Z_3 \subseteq Z_1$?

No

Zone based forward analysis



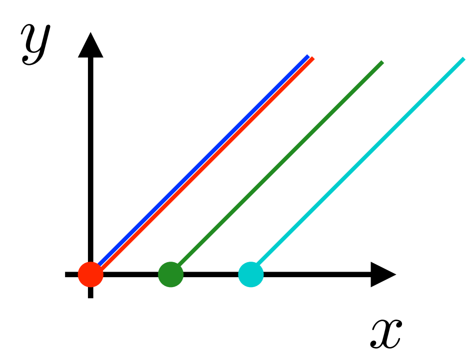
$$(q_0, Z_0) \longrightarrow (q_1, Z_1) \longrightarrow (q_1, Z_2) \longrightarrow (q_1, Z_3) \longrightarrow \cdots$$

Successor computation

$$q \xrightarrow[R]{g} q'$$

$$(q, Z) \rightarrow (q', Z')$$

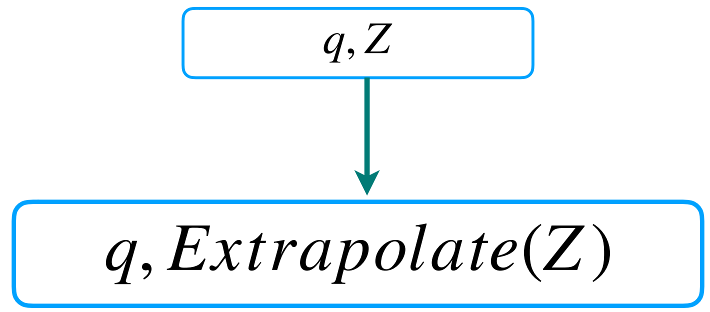
where $Z' = \overrightarrow{[R](Z \cap g)}$



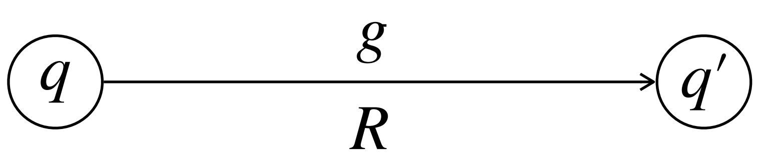
This method may not terminate

Two solutions: Extrapolate or Simulate!

Extrapolation



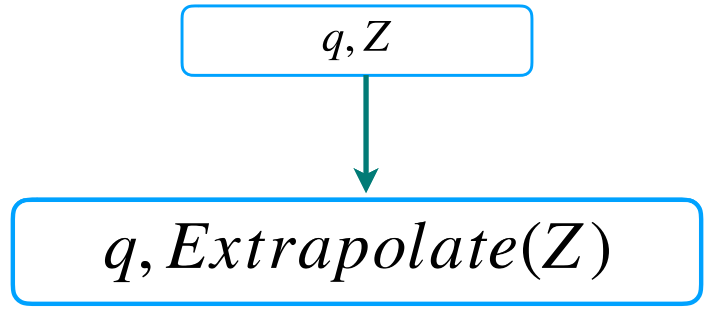
Updated successor computation



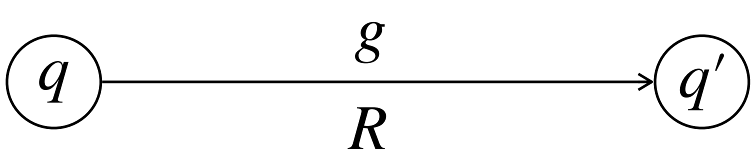
$$(q, Z) \longrightarrow (q', Z')$$

$$Z' = Extrapolate\left(\overrightarrow{[R](Z \cap g)} \right)$$

Extrapolation



Updated successor computation



$$(q, Z) \longrightarrow (q', Z')$$

$$Z' = Extrapolate\left(\overrightarrow{[R](Z \cap g)} \right)$$

Extrapolation operators for diagonal-free

Daws, Tripakis '98	$Extra_M$
Behrmann, Bouyer, Larsen, Pelanek '06	$Extra_{LU}$

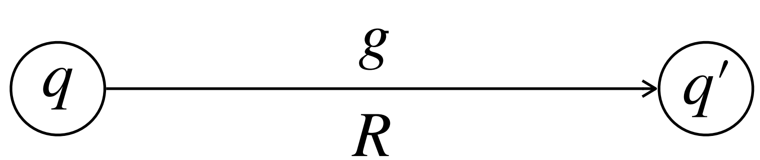
Extrapolation

q, Z

↓

$q, Extrapolate(Z)$

Updated successor computation



$$(q, Z) \longrightarrow (q', Z')$$

$$Z' = Extrapolate\left(\overrightarrow{[R](Z \cap g)} \right)$$

Extrapolation operators for diagonal-free

Daws, Tripakis '98 — $Extra_M$

Behrmann, Bouyer, Larsen, Pelanek '06 — $Extra_{LU}$

Bouyer '03 — No extrapolation operator is correct when the underlying TA has diagonal constraints

Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

→ q_0, Z_0

Extrapolation

Splitting

Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

→

q_0, Z_0

Extrapolation

Splitting

Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

$q_0, Z_0 \cap g_1 \cap g_2$

$q_0, Z_0 \cap g_1 \cap \neg g_2$

$q_0, Z_0 \cap \neg g_1 \cap g_2$

$q_0, Z_0 \cap \neg g_1 \cap \neg g_2$

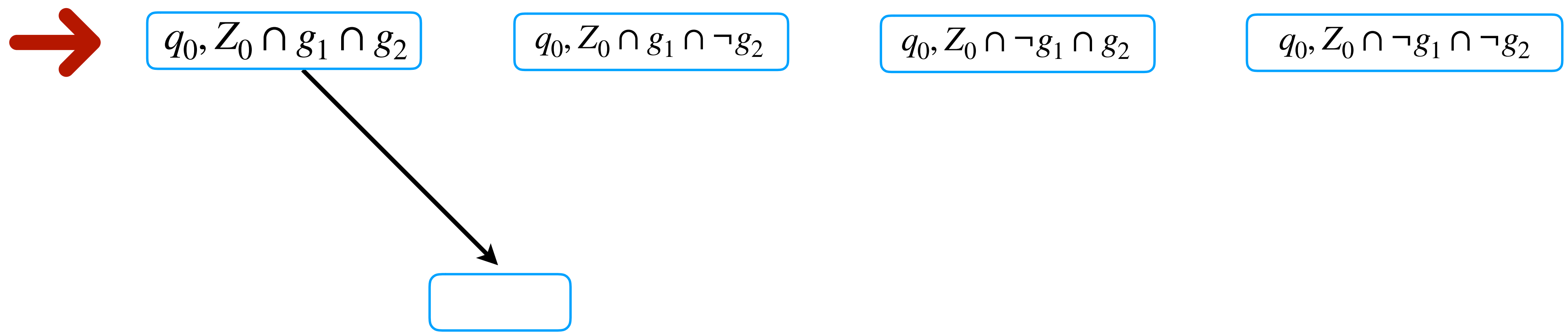
Extrapolation

Splitting

Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2



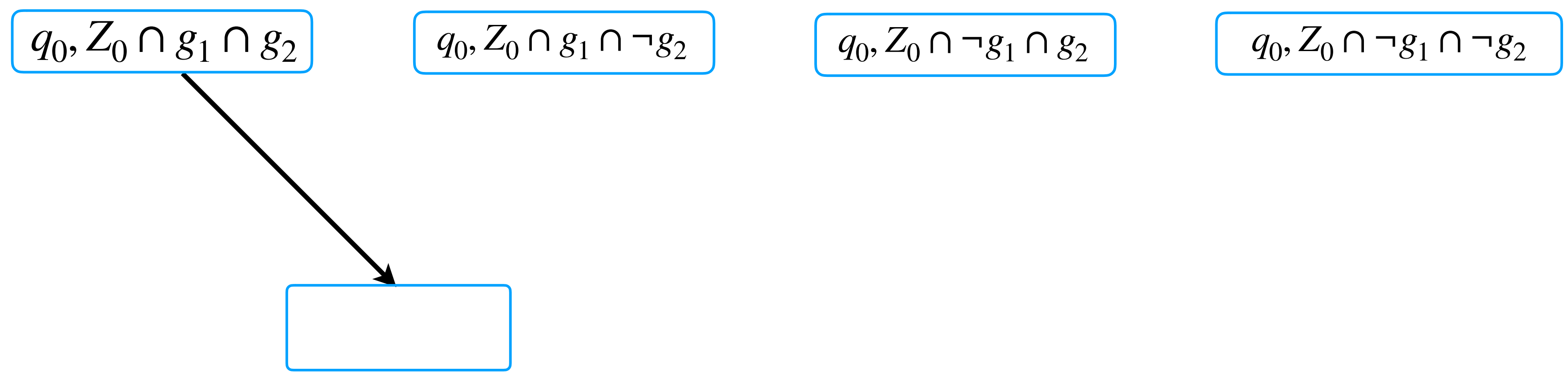
Extrapolation

Splitting

Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

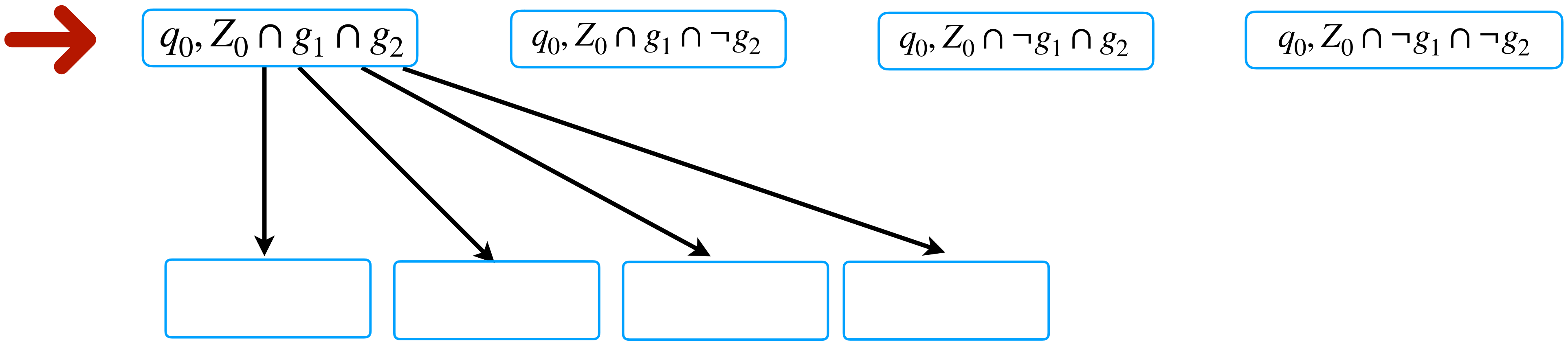


Extrapolation

Splitting

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2



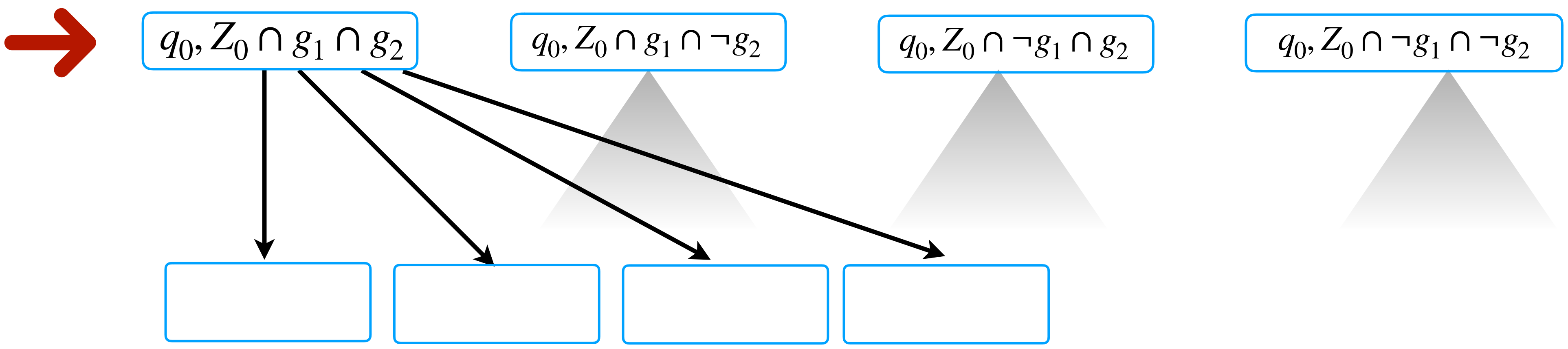
Extrapolation

Splitting

Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2



Extrapolation

Splitting

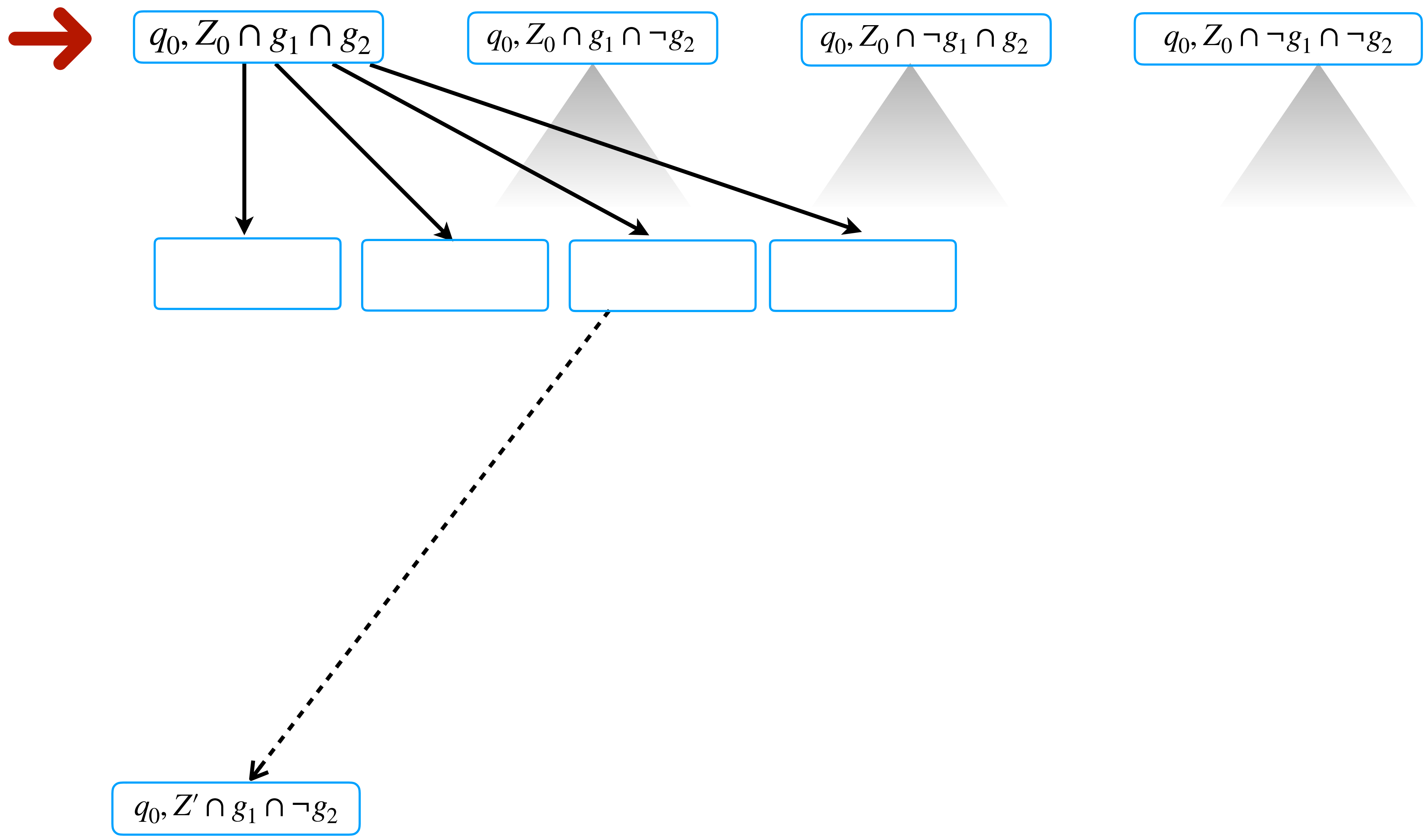
Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

Extrapolation

Splitting



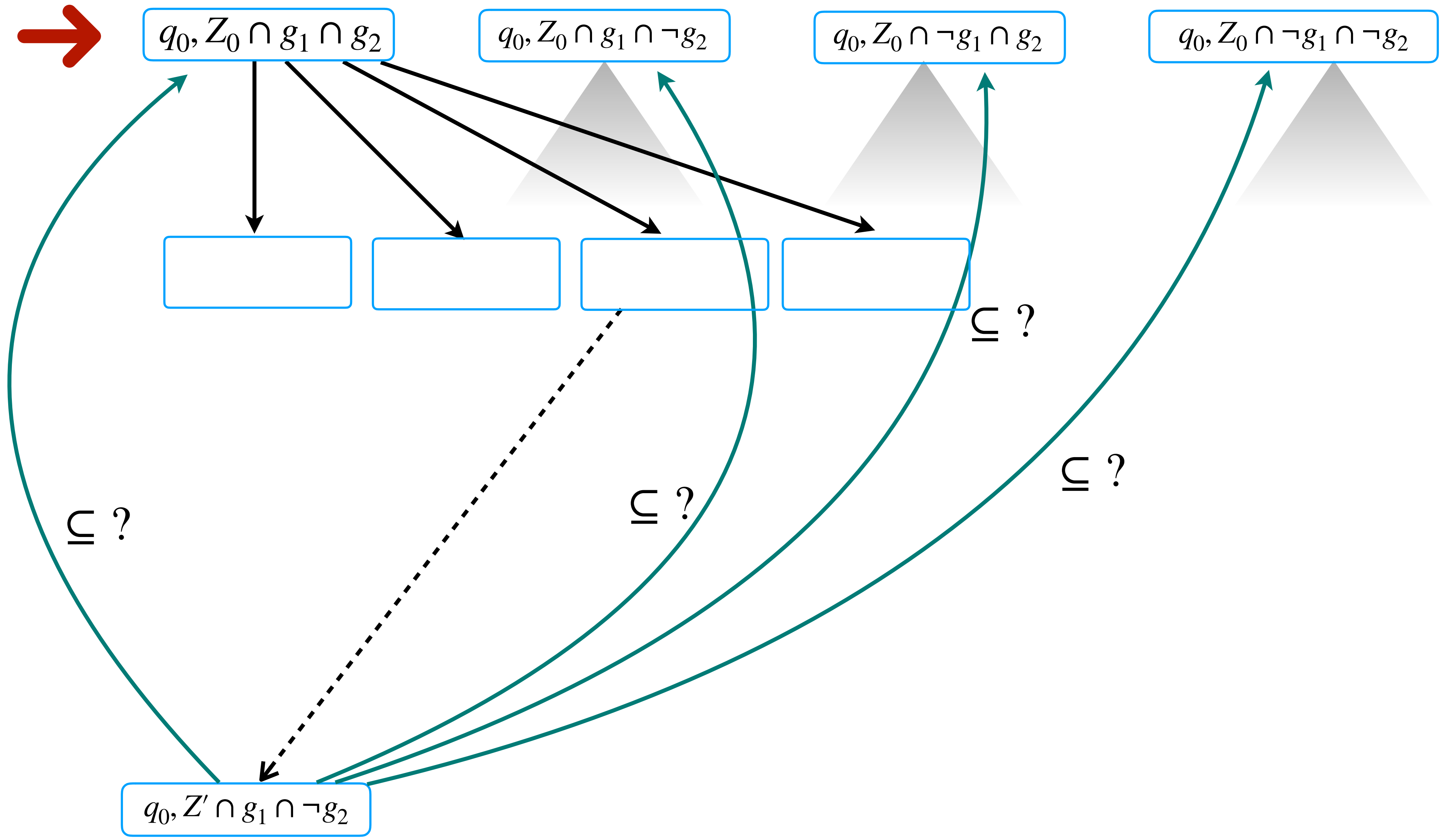
Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

Extrapolation

Splitting



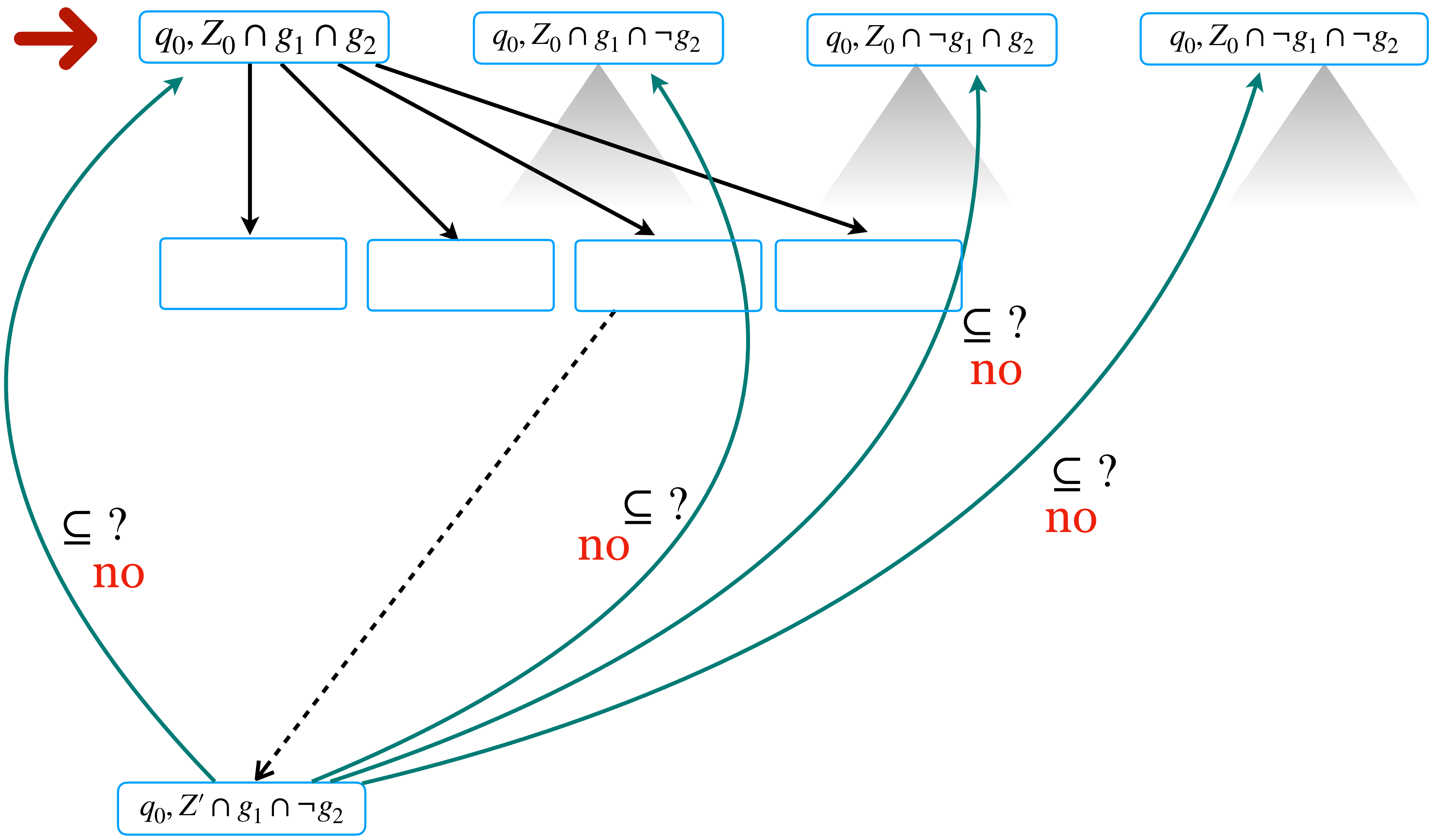
Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

Extrapolation

Splitting



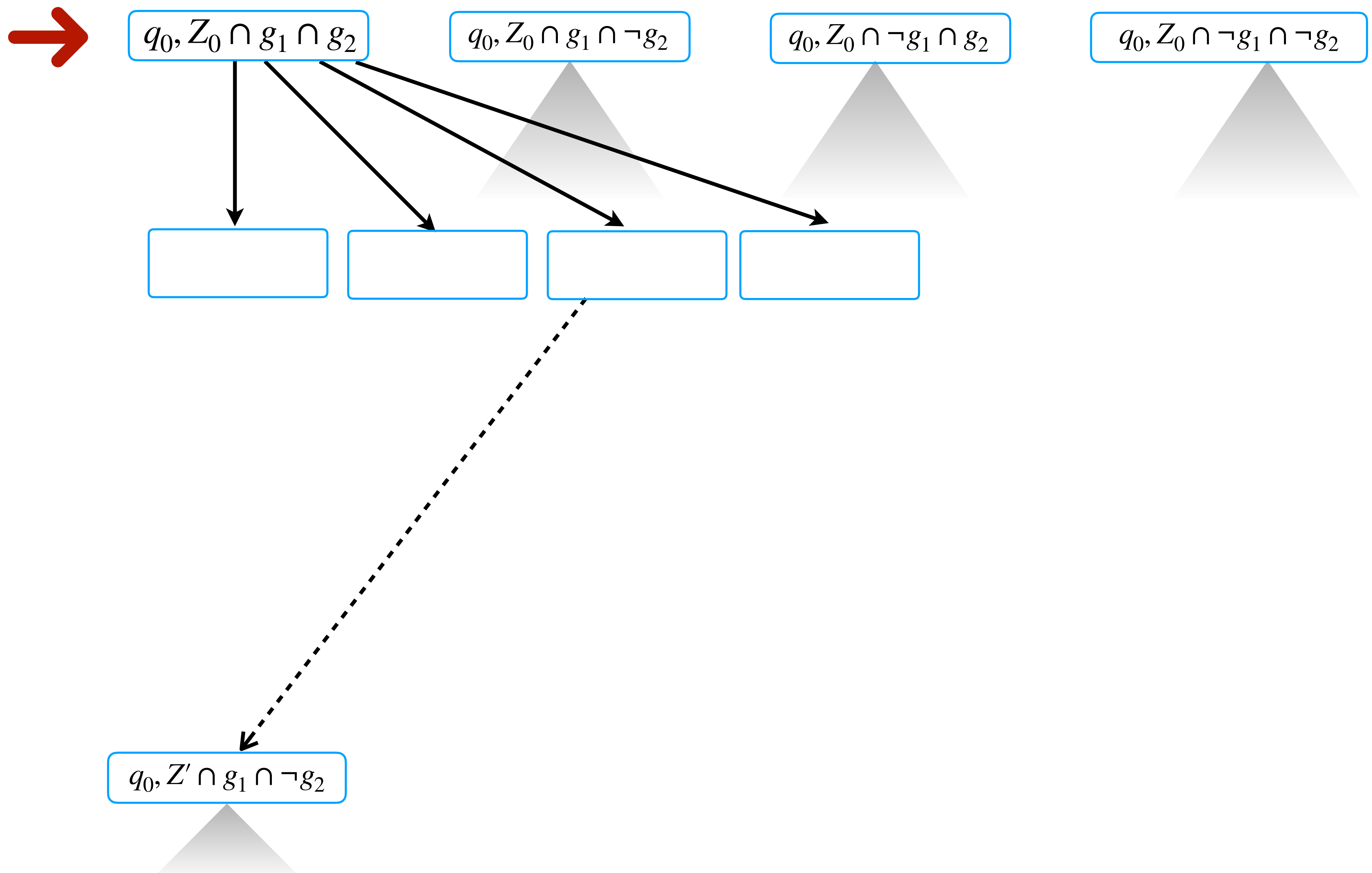
Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

Extrapolation

Splitting



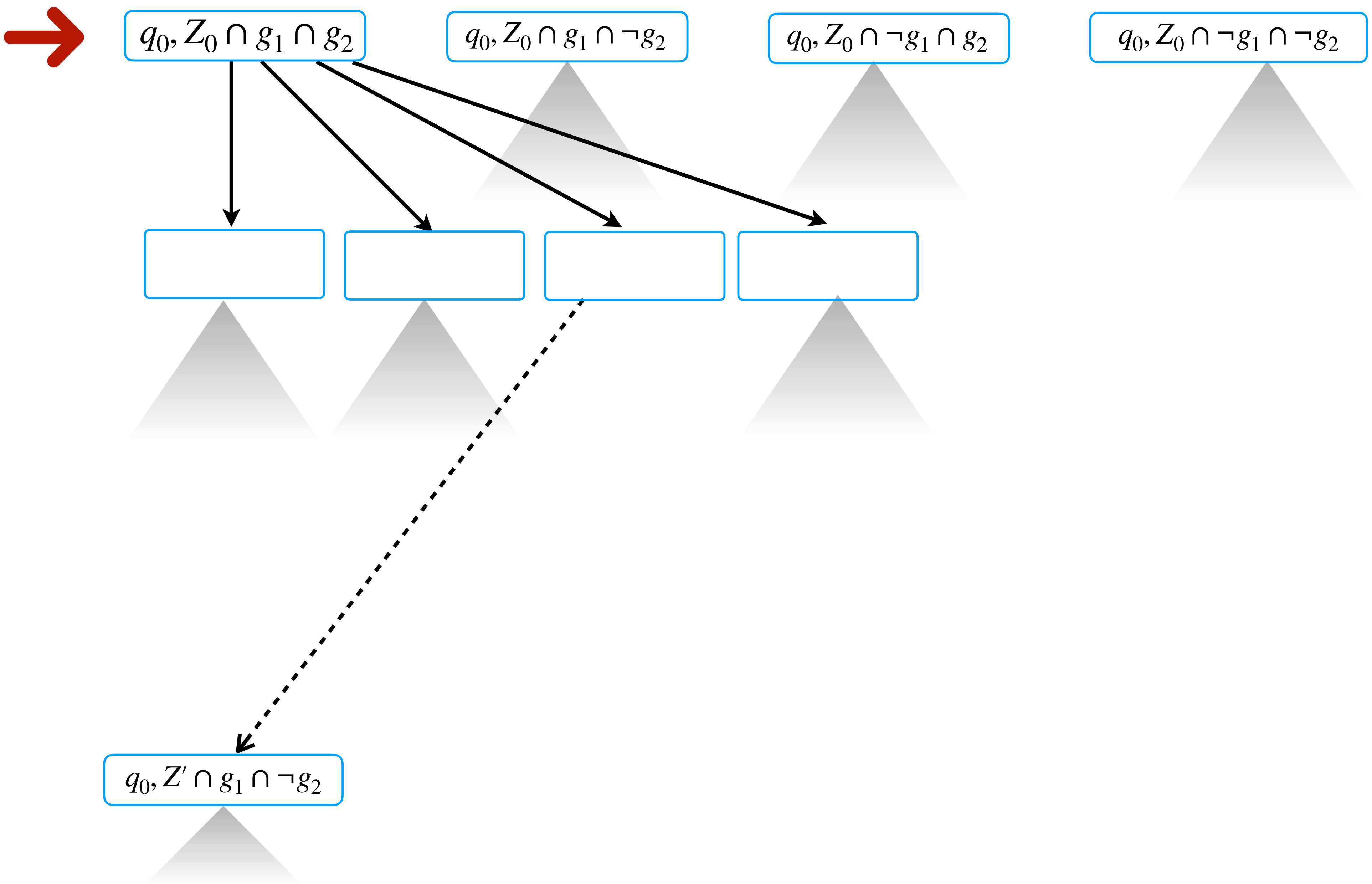
Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

Extrapolation

Splitting



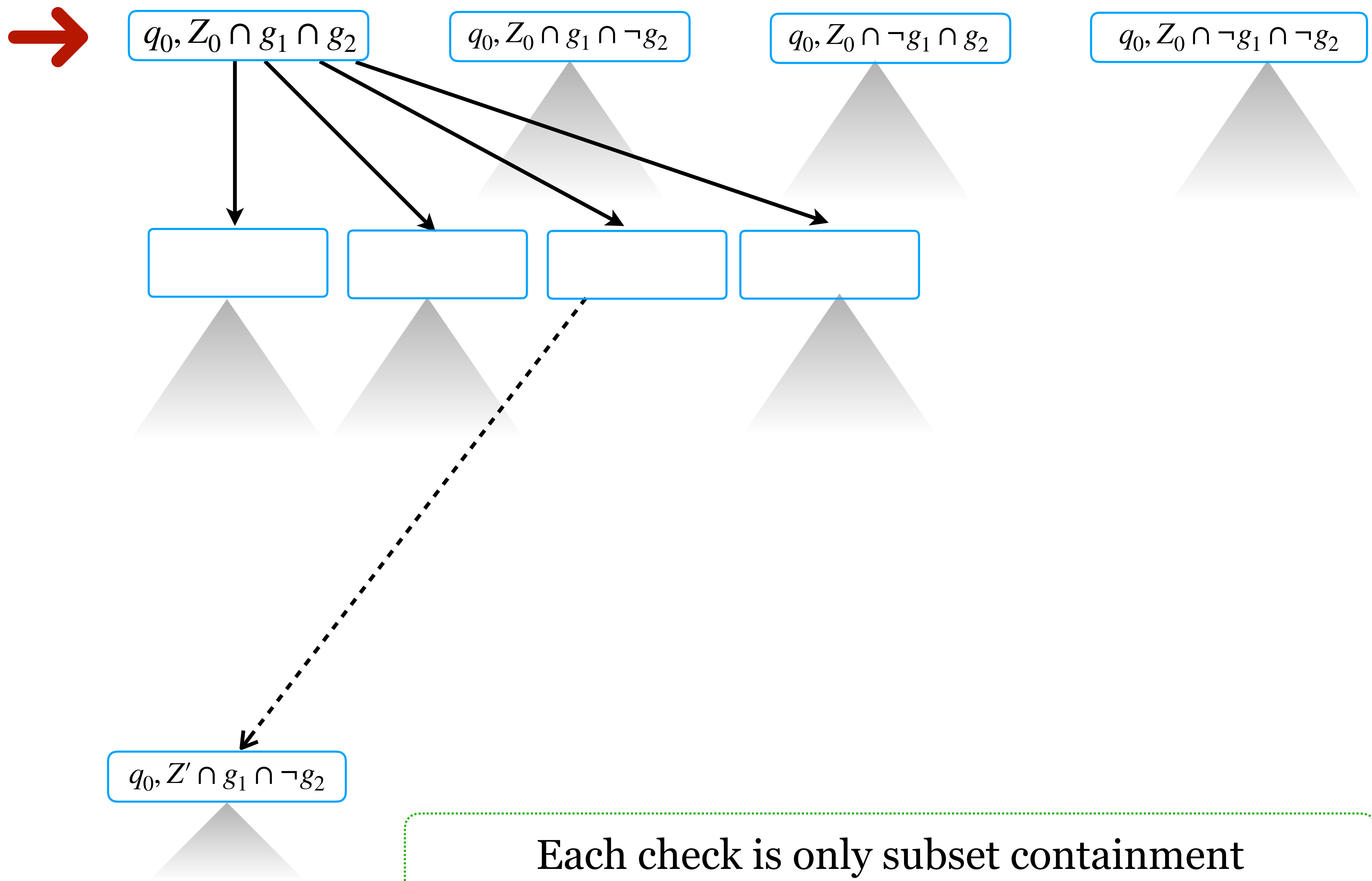
Bengtsson, Yi '04

Method 3 : Zone Splitting

Timed automaton
with
diagonal constraints
g_1, g_2

Extrapolation

Splitting



Each check is only subset containment

Each zone is split into exponentially many zones

Zone graph incurs exponential blowup!

Can we already handle diagonal constraints?

Yes!

Then why are you here?

We think we can do better than what we do now

Really? Do you have concrete evidence?

Yes!

What are the existing methods?

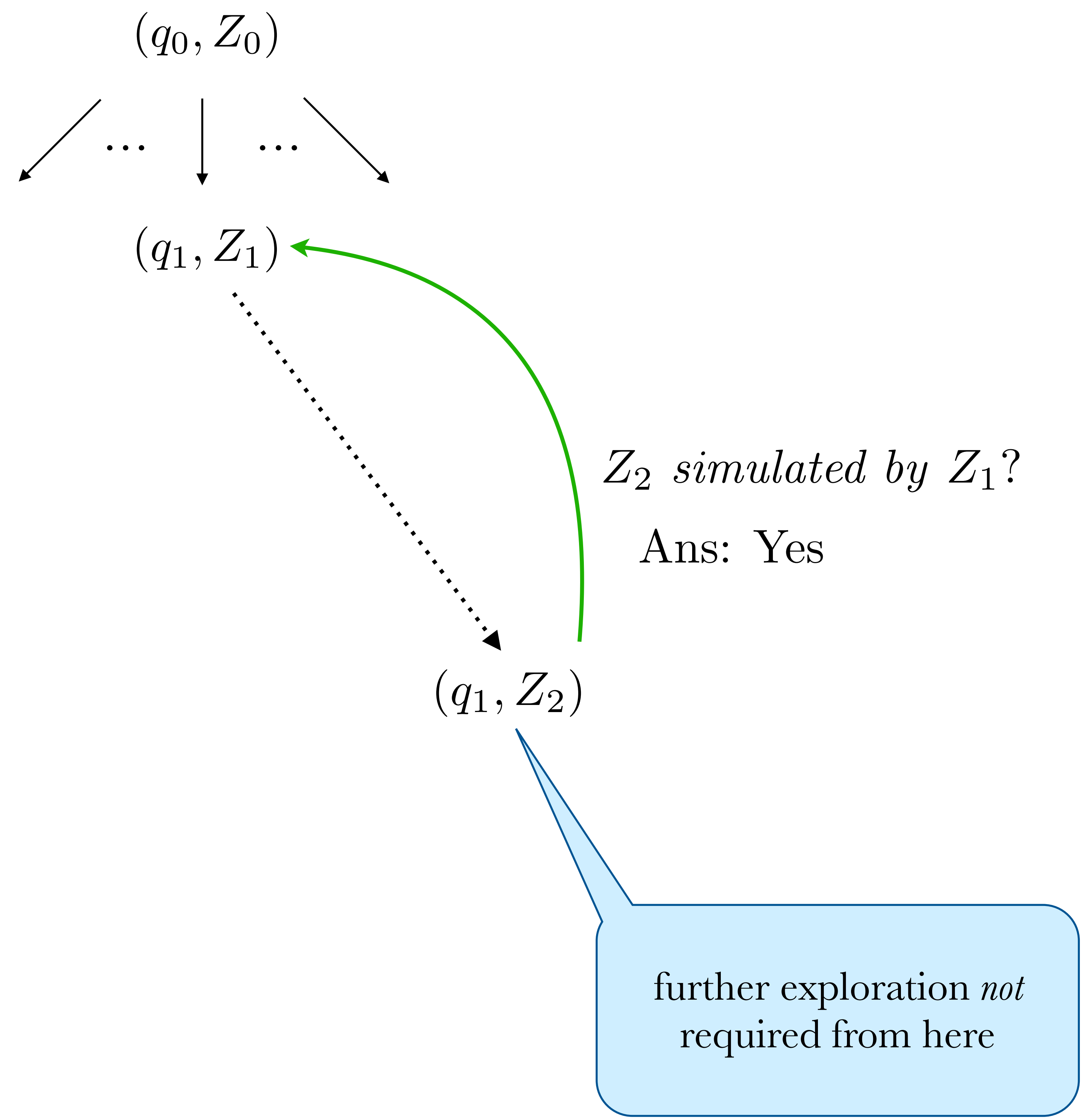
Can we avoid Removing diagonals?

Yes!

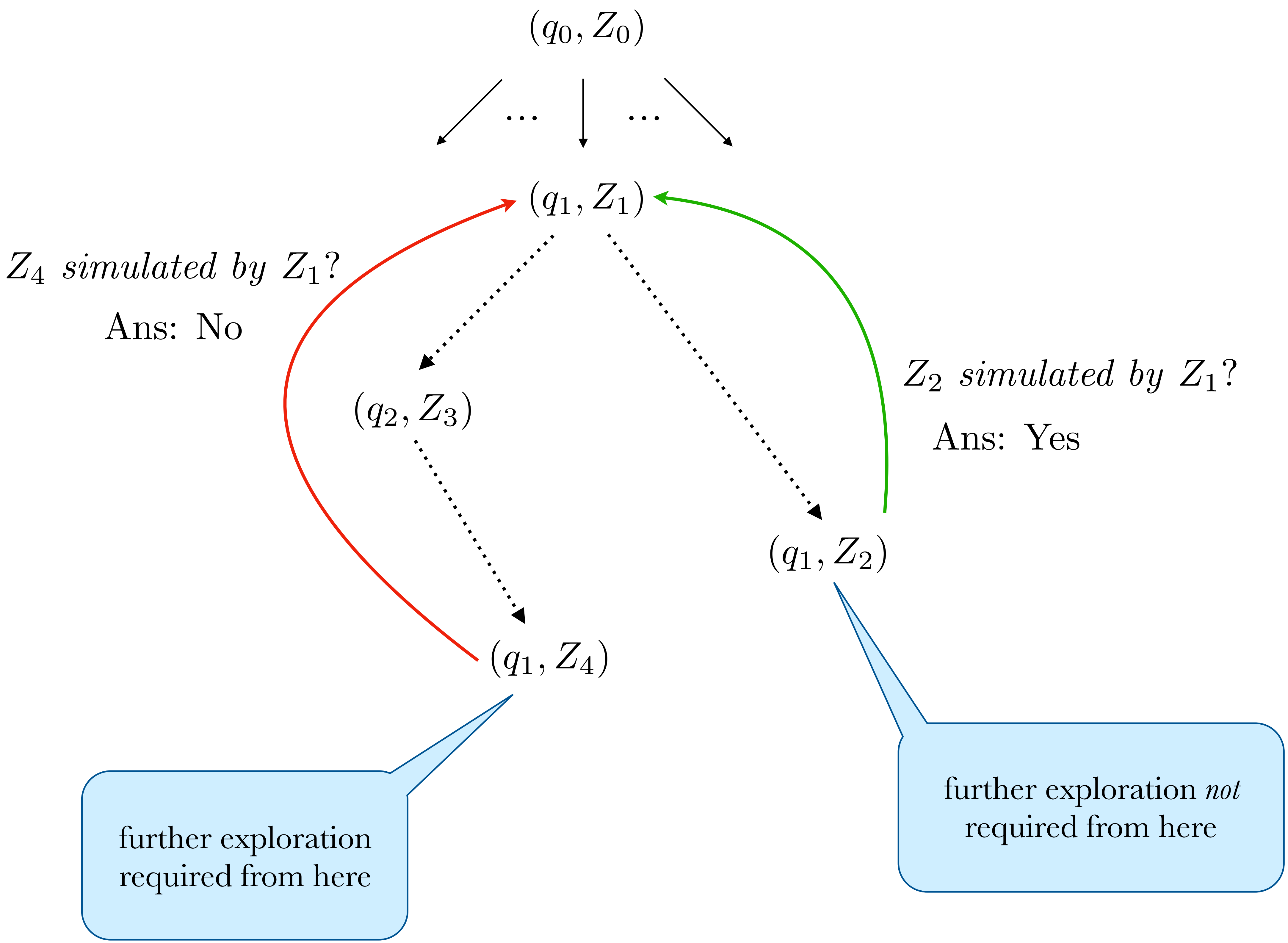
What is your method?

Avoid blowups in both states and zones

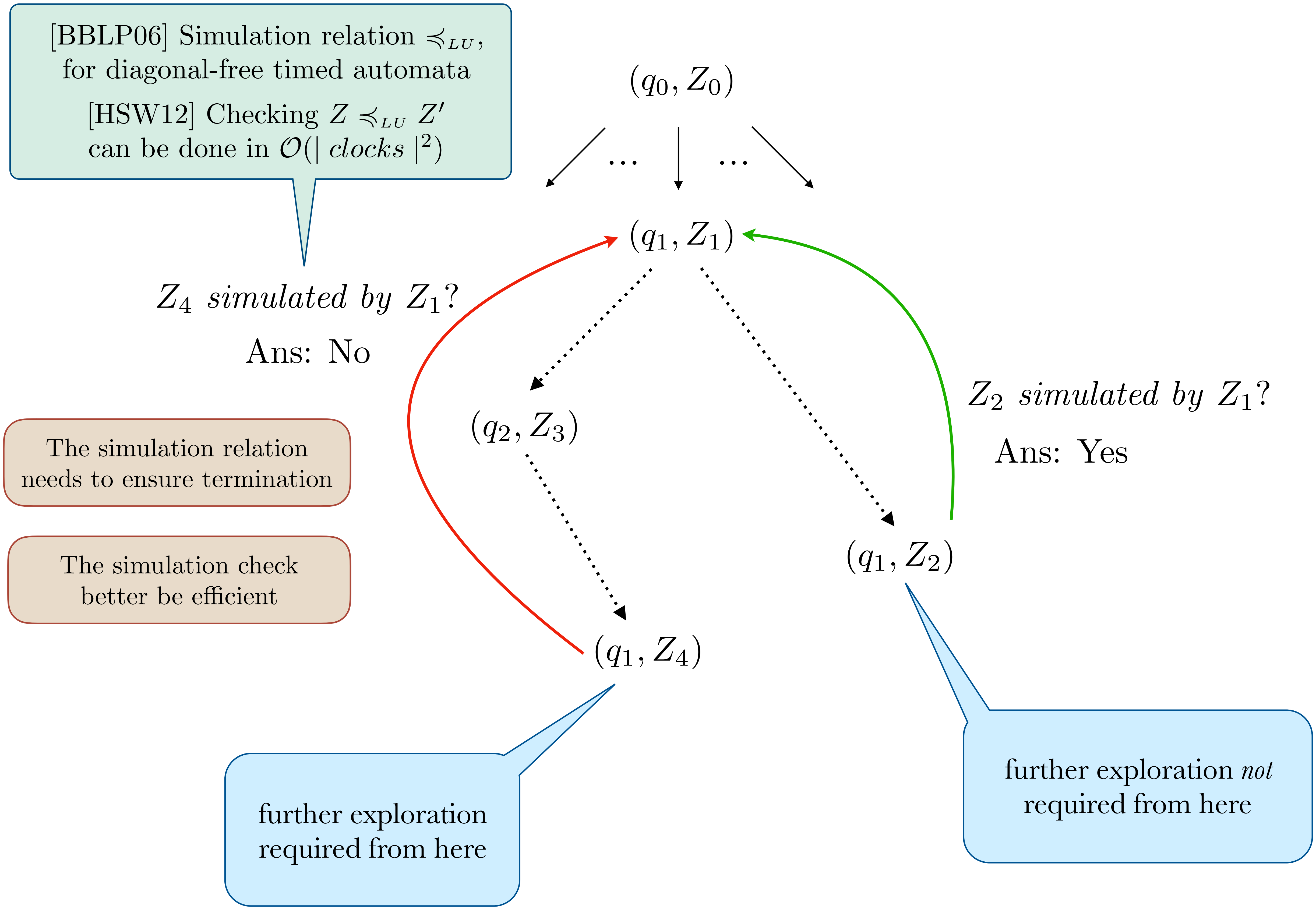
Reachability algorithm using simulation



Reachability algorithm using simulation



Reachability algorithm using simulation



Reachability algorithm using simulation

[BBLP06] Simulation relation $\preccurlyeq_{LU}$, for diagonal-free timed automata

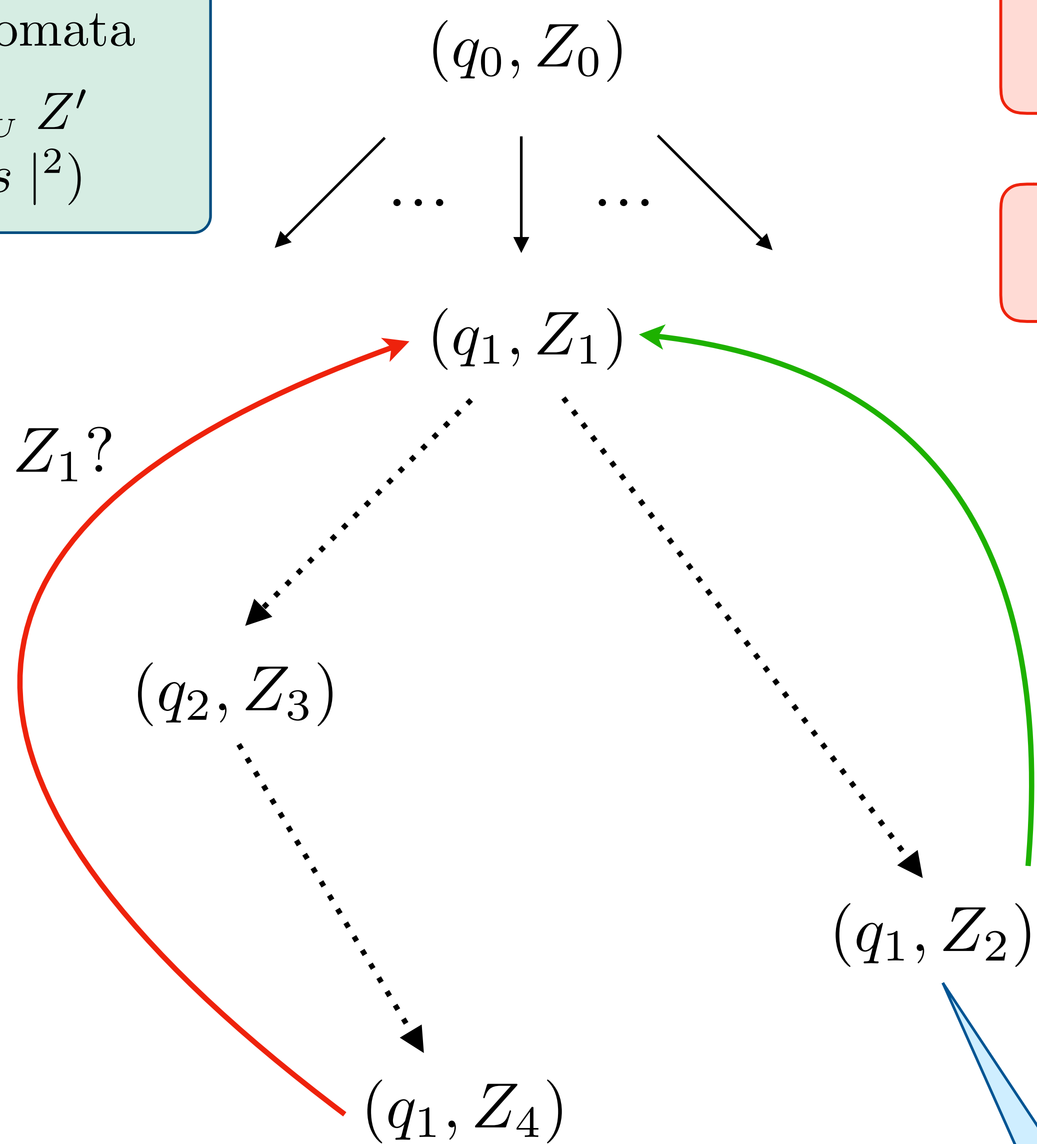
[HSW12] Checking $Z \preccurlyeq_{LU} Z'$ can be done in $\mathcal{O}(|\ clocks\ |^2)$

Our goal 1: Define a simulation relation in the presence of diagonal constraints

Our goal 2: Algorithm for Z *simulated by* Z', for that simulation relation

Z_4 simulated by Z_1?

Ans: No



Z_2 simulated by Z_1?

Ans: Yes

The simulation relation needs to ensure termination

The simulation check better be efficient

further exploration required from here

further exploration *not* required from here

What is a simulation relation?

$(q, v) \preccurlyeq (q, v')$ $\qquad$ $q \xrightarrow{\varphi, R} q'$

$$\begin{array}{ccc} (q,v) & \preccurlyeq & (q,v') \\ \forall\delta \downarrow & & \downarrow \exists\delta' \\ (q, v+\delta) & \preccurlyeq & (q, v'+\delta') \end{array} \qquad \begin{array}{ccc} (q,v) & \preccurlyeq & (q,v') \\ \varphi \downarrow & & \downarrow \varphi \\ (q,v) & \preccurlyeq & (q,v') \end{array} \qquad \begin{array}{ccc} (q,v) & \preccurlyeq & (q,v') \\ R \downarrow & & \downarrow R \\ (q',[R]v) & \preccurlyeq & (q',[R]v') \end{array}$$

What is a simulation relation?

$$(q,v) \preccurlyeq (q,v') \qquad\qquad q \xrightarrow{\varphi, R} q'$$

Time abstract

$$\begin{array}{ccc} (q,v) & \preccurlyeq & (q,v') \\ \forall\delta \big\downarrow & & \big\downarrow \exists\delta' \\ (q,v+\delta) & \preccurlyeq & (q,v'+\delta') \end{array} \qquad \begin{array}{ccc} (q,v) & \preccurlyeq & (q,v') \\ \varphi \big\downarrow & & \big\downarrow \varphi \\ (q,v) & \preccurlyeq & (q,v') \end{array} \qquad \begin{array}{ccc} (q,v) & \preccurlyeq & (q,v') \\ R \big\downarrow & & \big\downarrow R \\ (q',[R]v) & \preccurlyeq & (q',[R]v') \end{array}$$

Precise time

$$\begin{array}{ccc} (q,v) & \preccurlyeq & (q,v') \\ \forall\delta \big\downarrow & & \big\downarrow \delta \\ (q,v+\delta) & \preccurlyeq & (q,v'+\delta) \end{array}$$

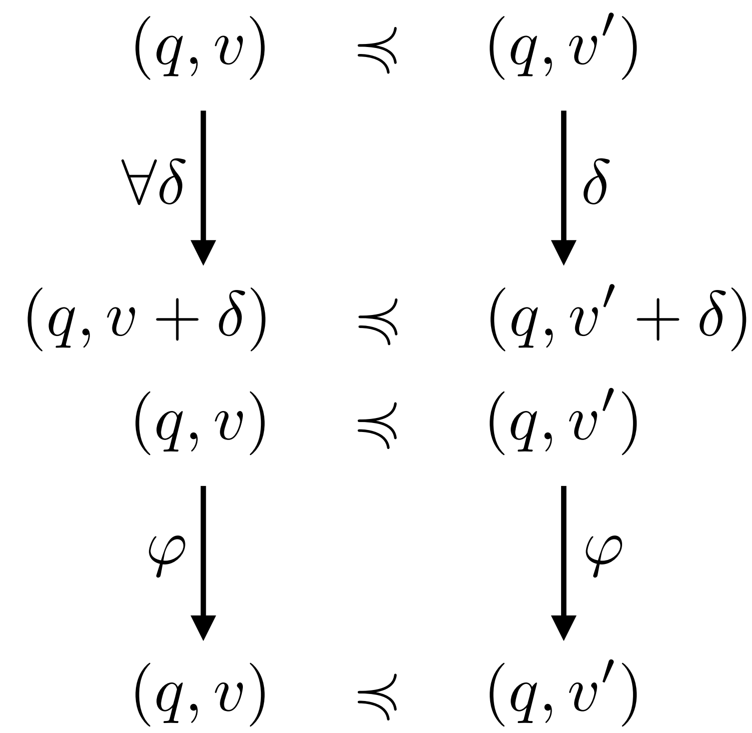
Simulation over valuations

$\mathcal{G}$ = set of constraints

$$v \sqsubseteq_{\mathcal{G}} v'$$

if

$$\forall \varphi \in \mathcal{G} \quad \forall \delta \geq 0 \qquad v + \delta \models \varphi \implies v' + \delta \models \varphi$$



Example

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

$$v = (1.3, 4)$$

$$v' = (3, 1)$$

$$\mathcal{G} = \{1 < x,\ x - y \leq 2\}$$

$$v \sqsubseteq_{\mathcal{G}} v'$$

$$\mathcal{G} = \{x < 4,\ y - x \leq 1\}$$

$$v + 2 \models x < 4$$

$$v' + 2 \not\models x < 4$$

$$v \not\sqsubseteq_{\mathcal{G}} v'$$

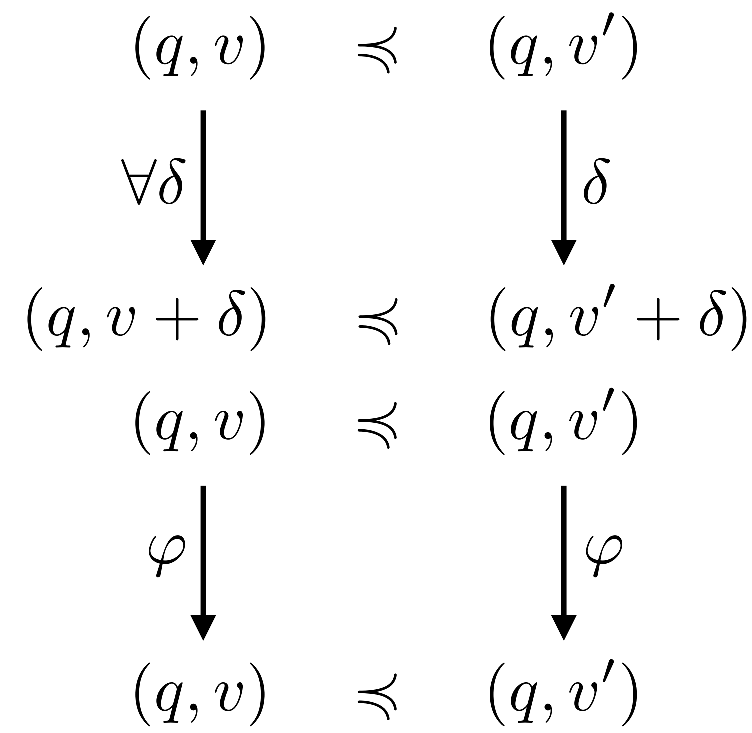
Simulation over valuations

$\mathcal{G}$ = set of constraints

$$v \sqsubseteq_{\mathcal{G}} v'$$

if

$$\forall \varphi \in \mathcal{G} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$



Can we get rid of $\forall \delta \geq 0$?

Example

$X = \{x, y\}$	$\mathcal{G} = \{1 < x,\; x - y \leq 2\}$	$\mathcal{G} = \{x < 4,\; y - x \leq 1\}$
$v = (1.3, 4)$ $v' = (3, 1)$	$v \sqsubseteq_{\mathcal{G}} v'$	$v + 2 \models x < 4$ $v' + 2 \not\models x < 4$ $v \not\sqsubseteq_{\mathcal{G}} v'$

A useful characterisation

$\mathcal{G}$ = set of constraints $\qquad v \sqsubseteq_{\mathcal{G}} v'$

if

$$\forall \varphi \in \mathcal{G} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

$\mathcal{G}^d$ = set of diagonal constraints of $\mathcal{G}$

$\mathcal{G}^{df}$ = set of non-diagonal constraints of $\mathcal{G}$

$$\forall \varphi^{df} \in \mathcal{G}^{df} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi^{df} \implies v' + \delta \models \varphi^{df}$$

and

$$\forall \varphi^{d} \in \mathcal{G}^{d} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi^{d} \implies v' + \delta \models \varphi^{d}$$

A useful characterisation

$\mathcal{G}$ = set of constraints $\qquad v \sqsubseteq_{\mathcal{G}} v'$

if

$$\forall \varphi \in \mathcal{G} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

$\mathcal{G}^d$ = set of diagonal constraints of $\mathcal{G}$

$\mathcal{G}^{df}$ = set of non-diagonal constraints of $\mathcal{G}$

$$\forall \varphi^{df} \in \mathcal{G}^{df} \;\; \forall \delta \geq 0$$
$$v + \delta \models \varphi^{df} \implies v' + \delta \models \varphi^{df}$$

and

$$\forall \varphi^{d} \in \mathcal{G}^{d} \;\; \forall \delta \geq 0$$
$$v + \delta \models \varphi^{d} \implies v' + \delta \models \varphi^{d}$$

This is same as

$$v \models \varphi^{d} \implies v' \models \varphi^{d}$$

A useful characterisation

$\mathcal{G}$ = set of constraints

$$v \sqsubseteq_{\mathcal{G}} v'$$

if

$$\forall \varphi \in \mathcal{G} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

$\mathcal{G}^d$ = set of diagonal constraints of $\mathcal{G}$

$\mathcal{G}^{df}$ = set of non-diagonal constraints of $\mathcal{G}$

$$\forall \varphi^{df} \in \mathcal{G}^{df} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi^{df} \implies v' + \delta \models \varphi^{df}$$

and

$$\forall \varphi^{d} \in \mathcal{G}^{d} \quad v \models \varphi^{d} \implies v' \models \varphi^{d}$$

A useful characterisation

$\mathcal{G}$ = set of constraints $\qquad v \sqsubseteq_{\mathcal{G}} v'$

if

$$\forall \varphi \in \mathcal{G} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

$\mathcal{G}^d$ = set of diagonal constraints of $\mathcal{G}$

$\mathcal{G}^{df}$ = set of non-diagonal constraints of $\mathcal{G}$

$$\begin{gathered} \forall \varphi^{df} \in \mathcal{G}^{df} \;\; \forall \delta \geq 0 \\ v + \delta \models \varphi^{df} \implies v' + \delta \models \varphi^{df} \\ v \models \varphi^{df} \implies v' \models \varphi^{df} \end{gathered} \quad \text{and} \quad \begin{gathered} \forall \varphi^{d} \in \mathcal{G}^{d} \\ v \models \varphi^{d} \implies v' \models \varphi^{d} \end{gathered}$$

This is not equivalent

A useful characterisation

$\mathcal{G}$ = set of constraints $\qquad v \sqsubseteq_{\mathcal{G}} v'$

if

$$\forall \varphi \in \mathcal{G} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

$\mathcal{G}^d$ = set of diagonal constraints of $\mathcal{G}$

$\mathcal{G}^{df}$ = set of non-diagonal constraints of $\mathcal{G}$

$$\forall \varphi^{df} \in \mathcal{G}^{df} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi^{df} \implies v' + \delta \models \varphi^{df}$$

and

$$\forall \varphi^{d} \in \mathcal{G}^{d} \quad v \models \varphi^{d} \implies v' \models \varphi^{d}$$

$\Uparrow$ $\quad$ $\Downarrow$ almost

$v \preccurlyeq_{LU} v'$

[BBLP06] Simulation relation $\preccurlyeq_{LU}$, for diagonal-free timed automata

[HSW12] Checking $Z \preccurlyeq_{LU} Z'$ can be done in $\mathcal{O}(|\,clocks\,|^2)$

A useful characterisation

$\mathcal{G}$ = set of constraints

$$v \sqsubseteq_{\mathcal{G}} v'$$

if

$$\forall \varphi \in \mathcal{G} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

$\mathcal{G}^d$ = set of diagonal constraints of $\mathcal{G}$

$\mathcal{G}^{df}$ = set of non-diagonal constraints of $\mathcal{G}$

$$\forall \varphi^{df} \in \mathcal{G}^{df} \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi^{df} \implies v' + \delta \models \varphi^{df}$$

and

$$\forall \varphi^{d} \in \mathcal{G}^{d} \quad v \models \varphi^{d} \implies v' \models \varphi^{d}$$

$\Uparrow$ $\quad$ $\Downarrow$ almost

$$v \preccurlyeq_{LU} v'$$

[BBLP06] Simulation relation $\preccurlyeq_{LU}$, for diagonal-free timed automata

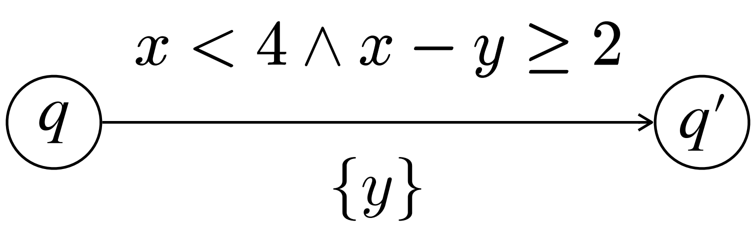
[HSW12] Checking $Z \preccurlyeq_{LU} Z'$ can be done in $\mathcal{O}(|\ clocks\ |^2)$

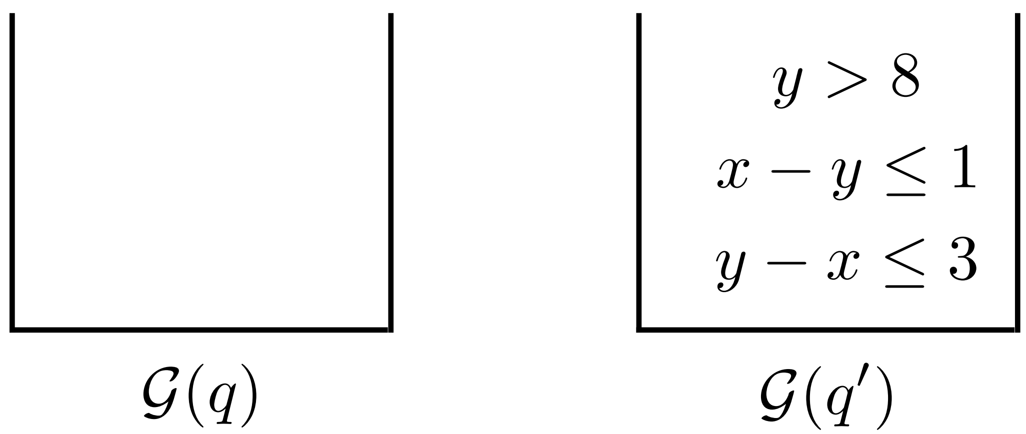
$$v \sqsubseteq_{\mathcal{G}}^{LU} v' \text{ if } v \preccurlyeq_{LU} v' \text{ and } \forall \text{ diagonal } \varphi \in \mathcal{G},\; v \models \varphi \implies v' \models \varphi$$

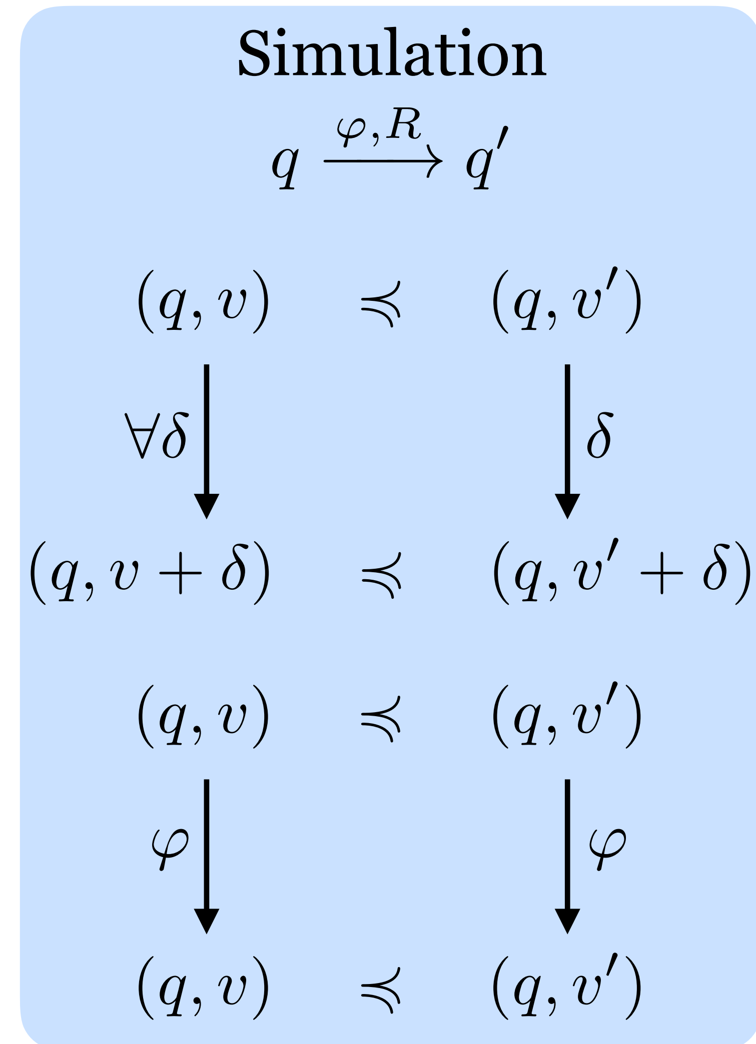
State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$







State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

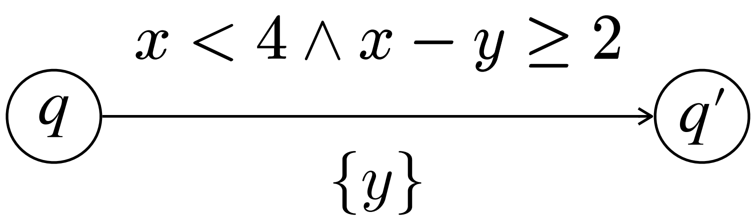
$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$

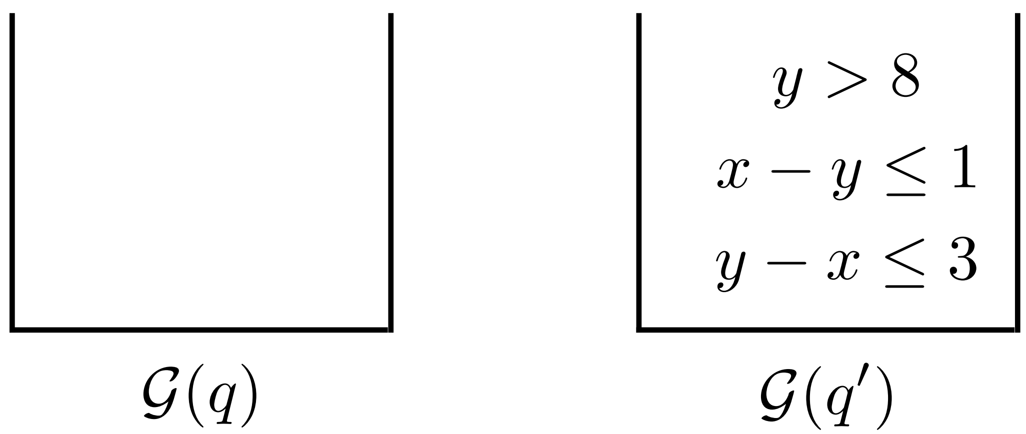
Simulation

$$q \xrightarrow{\varphi, R} q'$$

$$(q, v) \preccurlyeq (q, v') \quad \xrightarrow{\forall\delta} \quad (q, v+\delta) \preccurlyeq (q, v'+\delta)$$

$$(q, v) \preccurlyeq (q, v') \quad \xrightarrow{\varphi} \quad (q, v) \preccurlyeq (q, v')$$



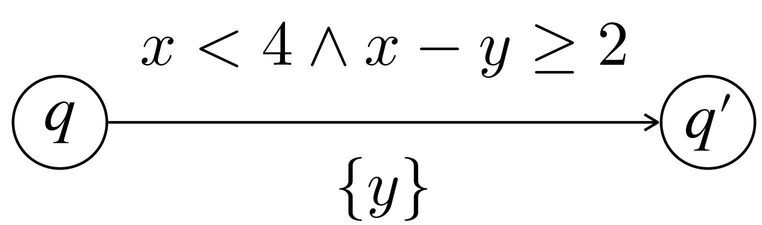


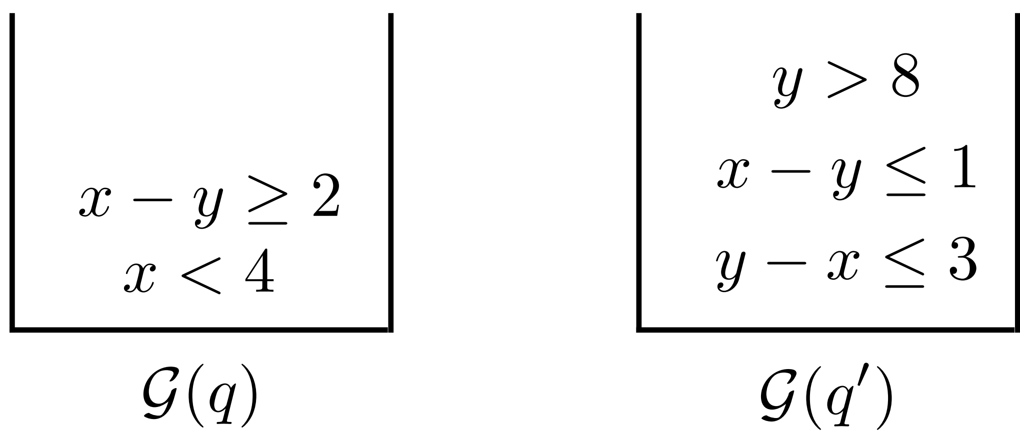
$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \vDash \varphi \implies v' + \delta \vDash \varphi$$

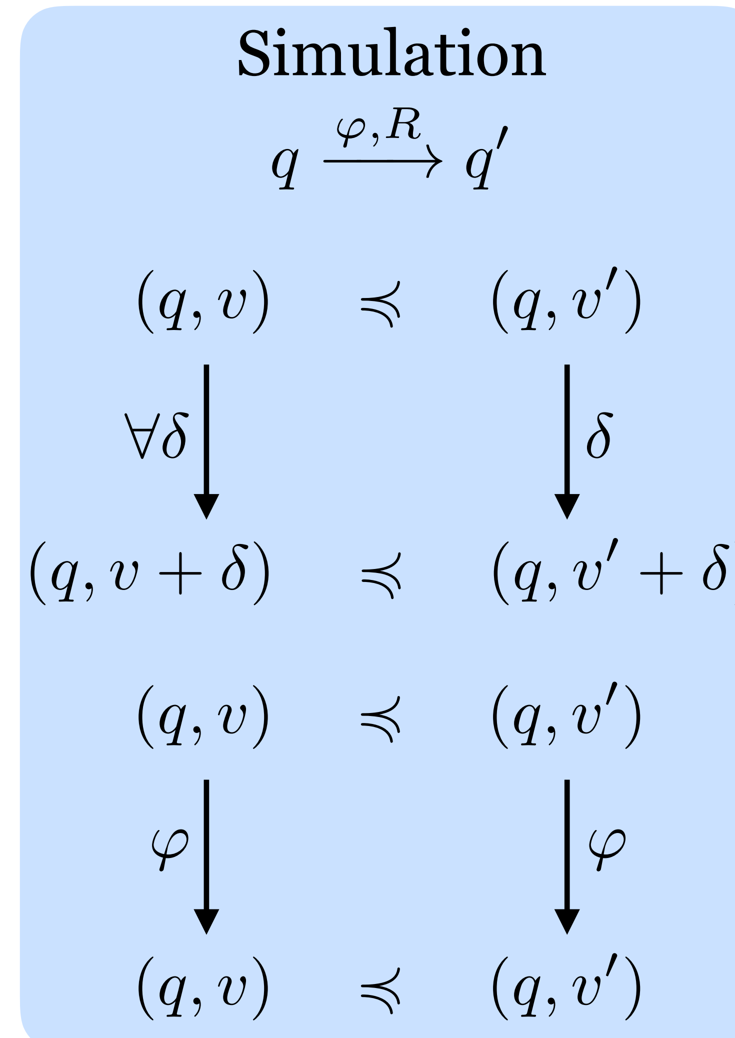
State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q,v) \preccurlyeq_{\mathcal{A}} (q,v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$





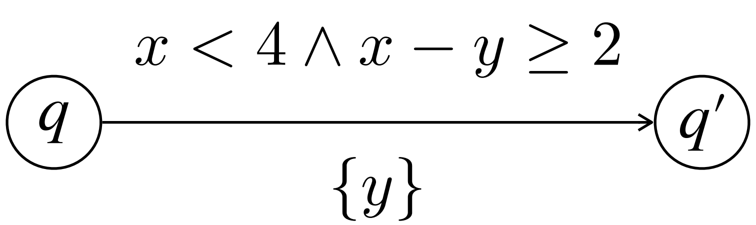


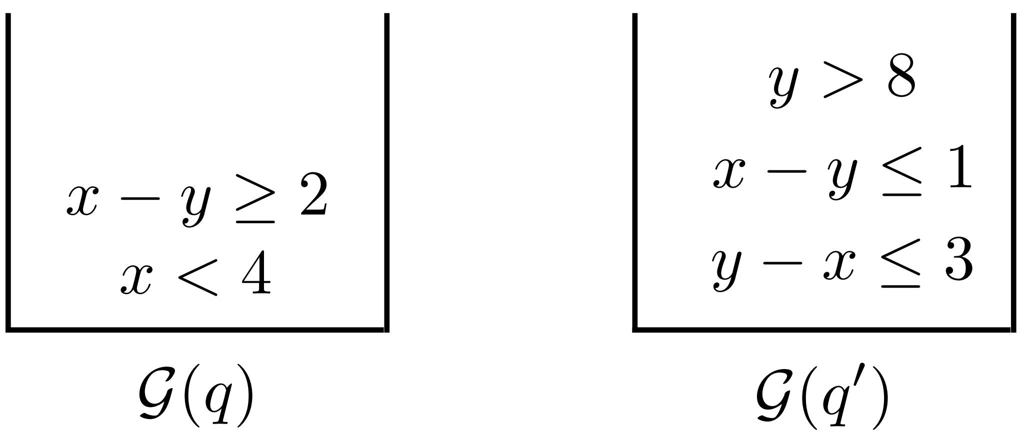
$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

State based guards

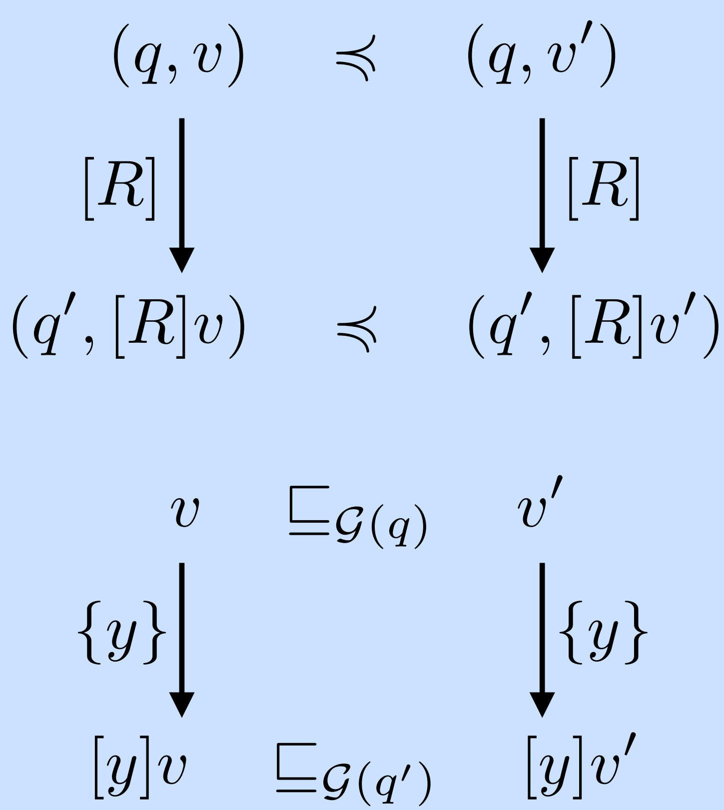
Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$





Simulation

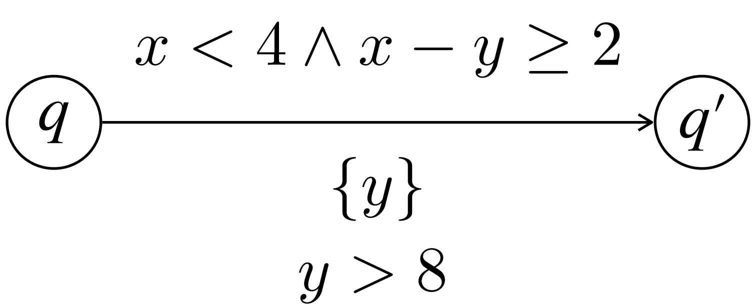


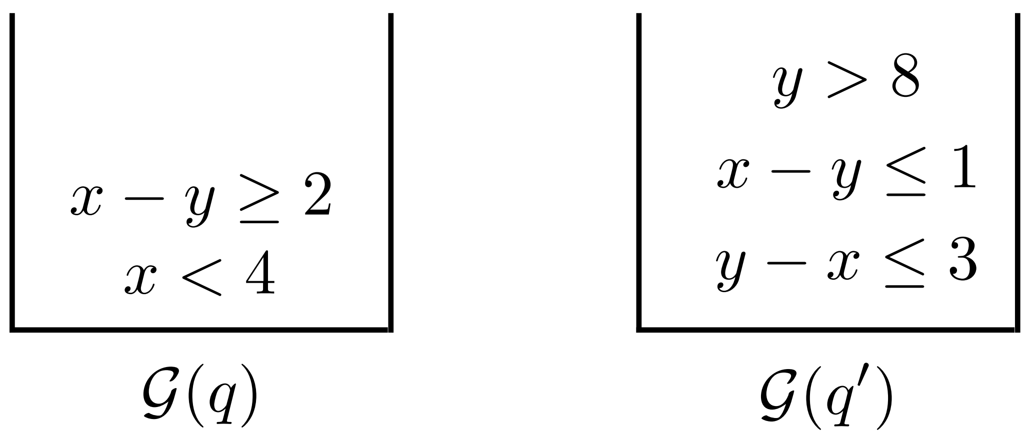
$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \ \ \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

State based guards

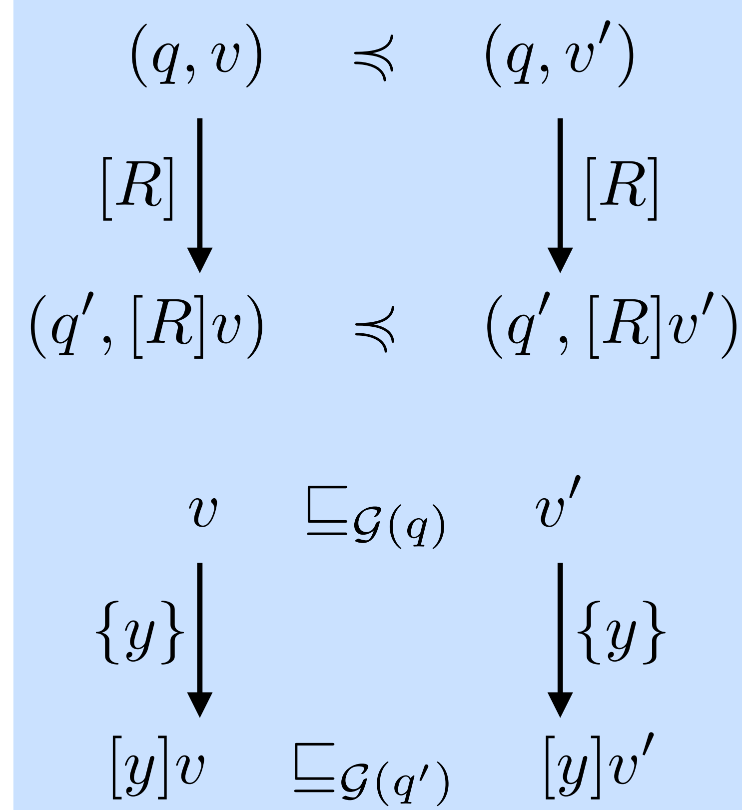
Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$





Simulation



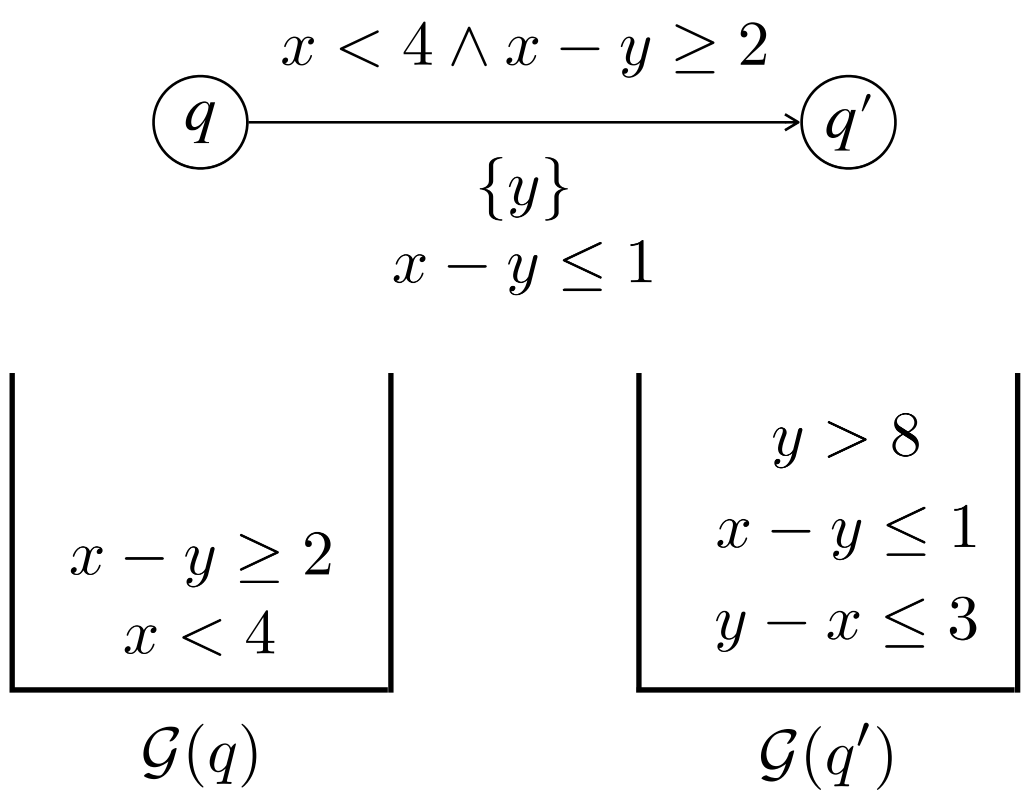
$$\forall \delta \geq 0 \quad [y]v + \delta \models y > 8 \implies [y]v' + \delta \models y > 8$$

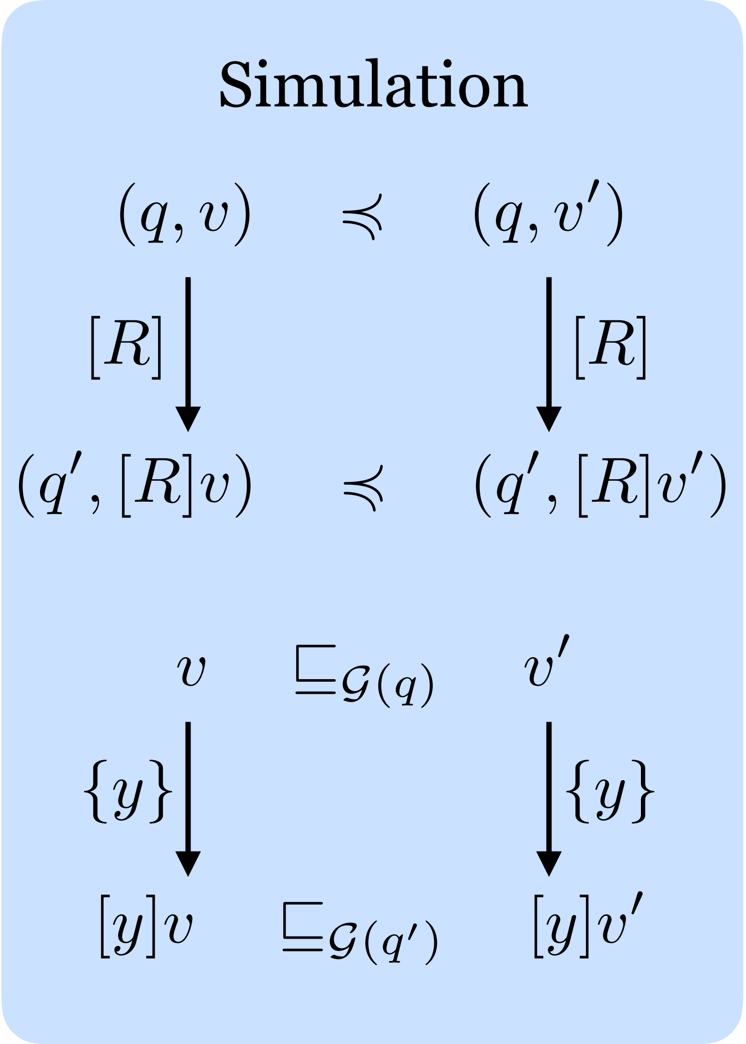
$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$





$$\forall \delta \geq 0 \quad [y]v + \delta \models x - y \leq 1 \implies [y]v' + \delta \models x - y \leq 1$$

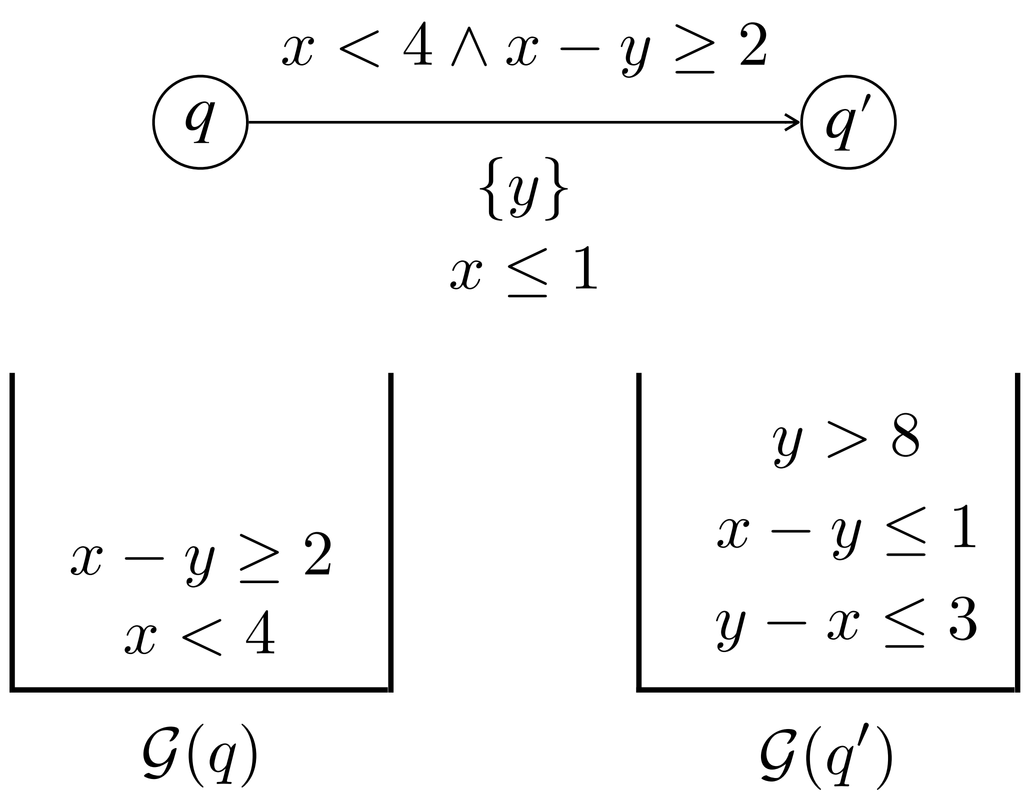
$$\forall \delta \geq 0 \quad [y]v + \delta \models y > 8 \implies [y]v' + \delta \models y > 8$$

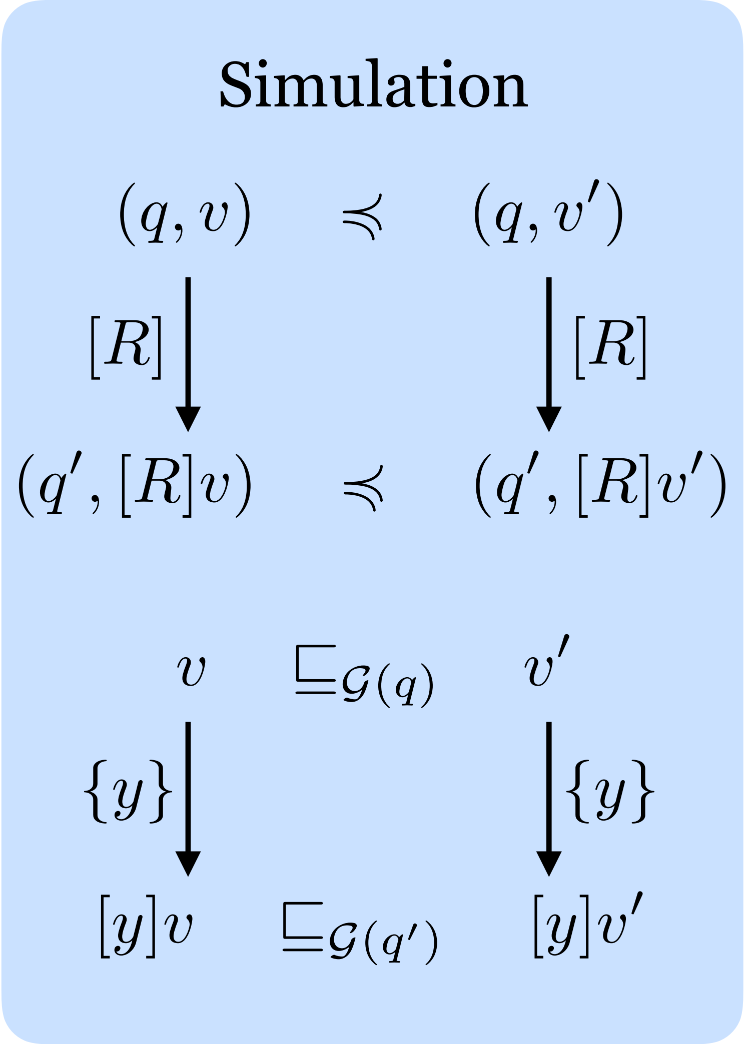
$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$





$$\forall \delta \geq 0 \quad [y]v + \delta \models x - y \leq 1 \implies [y]v' + \delta \models x - y \leq 1$$

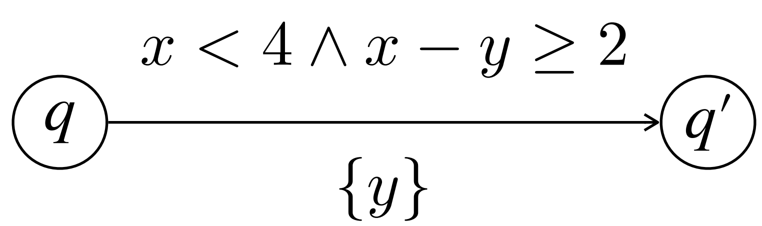
$$\forall \delta \geq 0 \quad [y]v + \delta \models y > 8 \implies [y]v' + \delta \models y > 8$$

$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \vDash \varphi \implies v' + \delta \vDash \varphi$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$

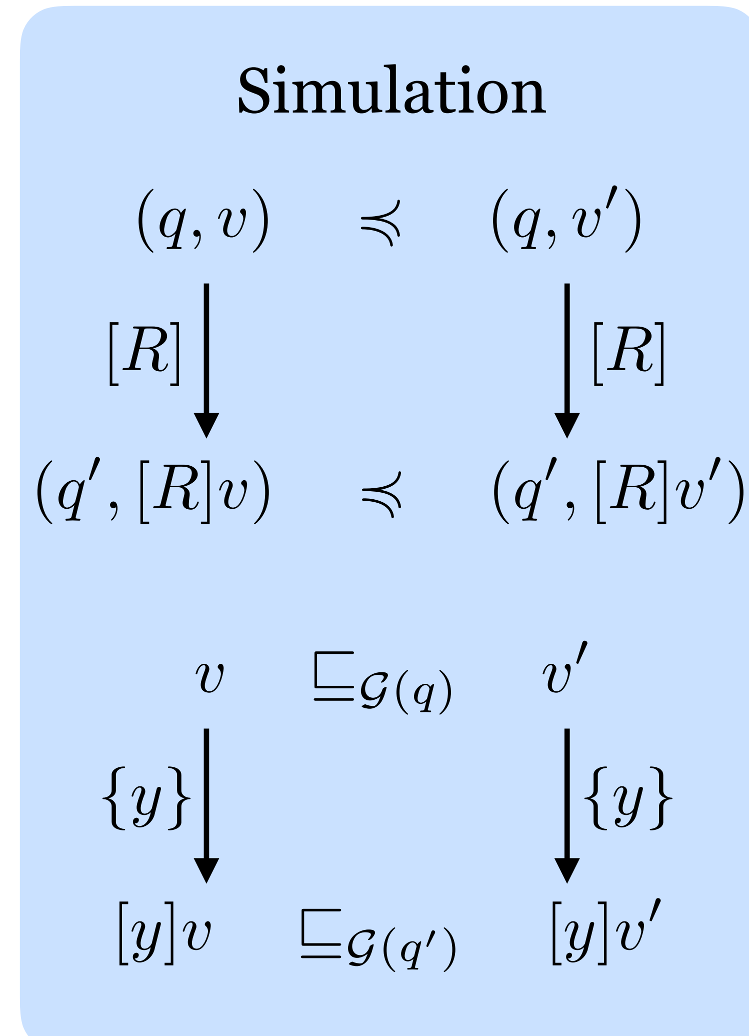


$$x \leq 1 \qquad x - y \geq 2 \qquad x < 4$$

$\mathcal{G}(q)$

$$y > 8 \qquad x - y \leq 1 \qquad y - x \leq 3$$

$\mathcal{G}(q')$



$$\forall \delta \geq 0 \quad [y]v + \delta \models x - y \leq 1 \implies [y]v' + \delta \models x - y \leq 1$$

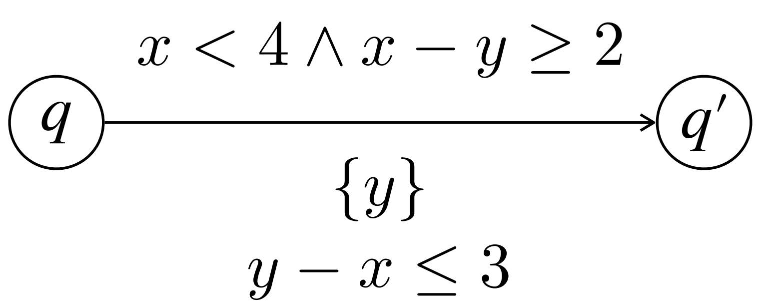
$$\forall \delta \geq 0 \quad [y]v + \delta \models y > 8 \implies [y]v' + \delta \models y > 8$$

$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

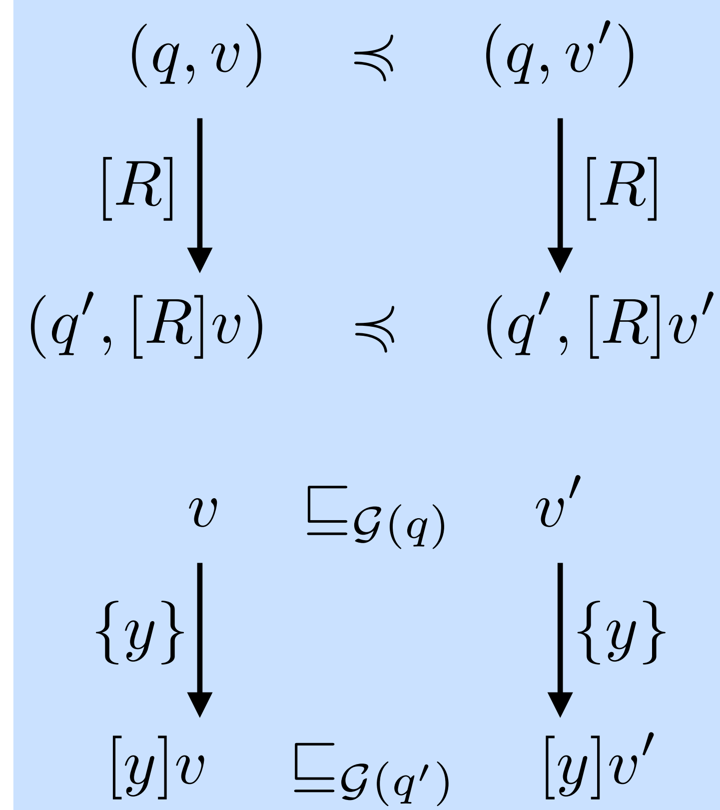
$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



$\mathcal{G}(q)$: $x \leq 1$, $x - y \geq 2$, $x < 4$

$\mathcal{G}(q')$: $y > 8$, $x - y \leq 1$, $y - x \leq 3$

Simulation



$$\forall \delta \geq 0 \quad [y]v + \delta \models x - y \leq 1 \implies [y]v' + \delta \models x - y \leq 1$$

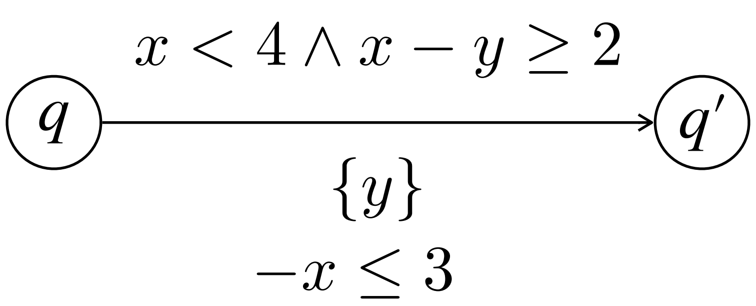
$$\forall \delta \geq 0 \quad [y]v + \delta \models y > 8 \implies [y]v' + \delta \models y > 8$$

$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

State based guards

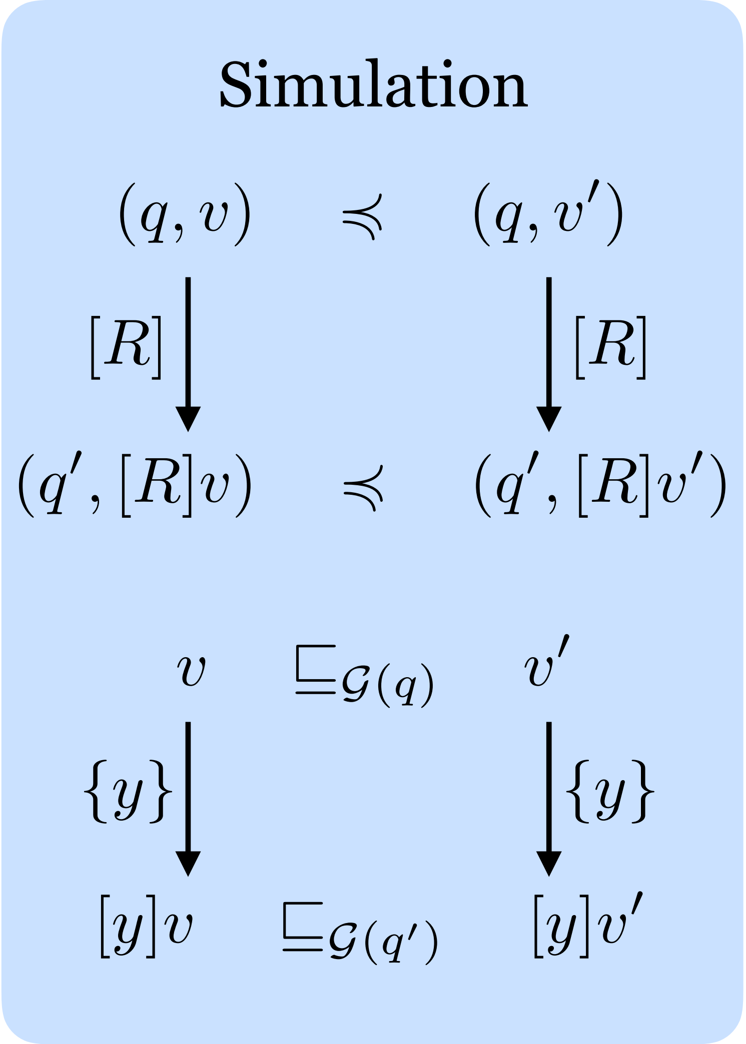
Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



$\mathcal{G}(q)$: $x \leq 1$, $x - y \geq 2$, $x < 4$

$\mathcal{G}(q')$: $y > 8$, $x - y \leq 1$, $y - x \leq 3$



$$\forall \delta \geq 0 \quad [y]v + \delta \models x - y \leq 1 \implies [y]v' + \delta \models x - y \leq 1$$

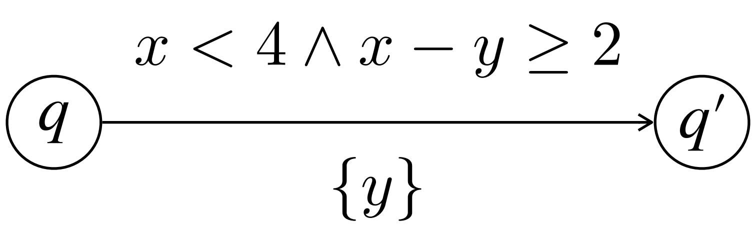
$$\forall \delta \geq 0 \quad [y]v + \delta \models y > 8 \implies [y]v' + \delta \models y > 8$$

$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q,v) \preccurlyeq_{\mathcal{A}} (q,v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



$$-x \leq 3 \qquad x \leq 1 \qquad x - y \geq 2 \qquad x < 4$$

$\mathcal{G}(q)$

$$y > 8 \qquad x - y \leq 1 \qquad y - x \leq 3$$

$\mathcal{G}(q')$

Simulation

$$(q,v) \preccurlyeq (q,v')$$

$$[R] \downarrow \qquad \downarrow [R]$$

$$(q',[R]v) \preccurlyeq (q',[R]v')$$

$$v \sqsubseteq_{\mathcal{G}(q)} v'$$

$$\{y\} \downarrow \qquad \downarrow \{y\}$$

$$[y]v \sqsubseteq_{\mathcal{G}(q')} [y]v'$$

$$\forall \delta \geq 0 \quad [y]v + \delta \models x - y \leq 1 \implies [y]v' + \delta \models x - y \leq 1$$

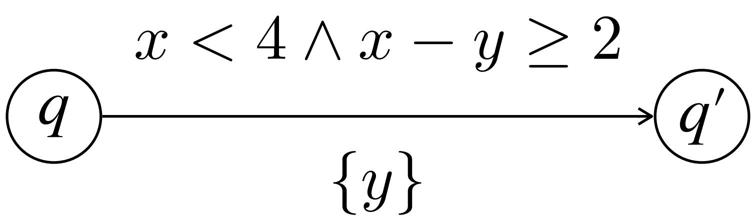
$$\forall \delta \geq 0 \quad [y]v + \delta \models y > 8 \implies [y]v' + \delta \models y > 8$$

$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \ \ \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

$-x \leq 3$
$x \leq 1$
$x - y \geq 2$
$x < 4$

$\mathcal{G}(q)$

$y > 8$
$x - y \leq 1$
$y - x \leq 3$

$\mathcal{G}(q')$

No new constant gets generated

This least fixed-point computation always terminates

$\preccurlyeq_{\mathcal{A}}$ is a finite (time-concrete) simulation relation

What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G},\ v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z\ \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$

What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G},\ v \models \varphi \implies v' \models \varphi$

$Z \sqsubseteq_{\mathcal{G}} Z'$ if $\forall v \in Z\ \exists v' \in Z'$ such that $v \sqsubseteq_{\mathcal{G}} v'$

Check $Z \preccurlyeq_{LU} Z'$

What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G},\ v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z\ \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$

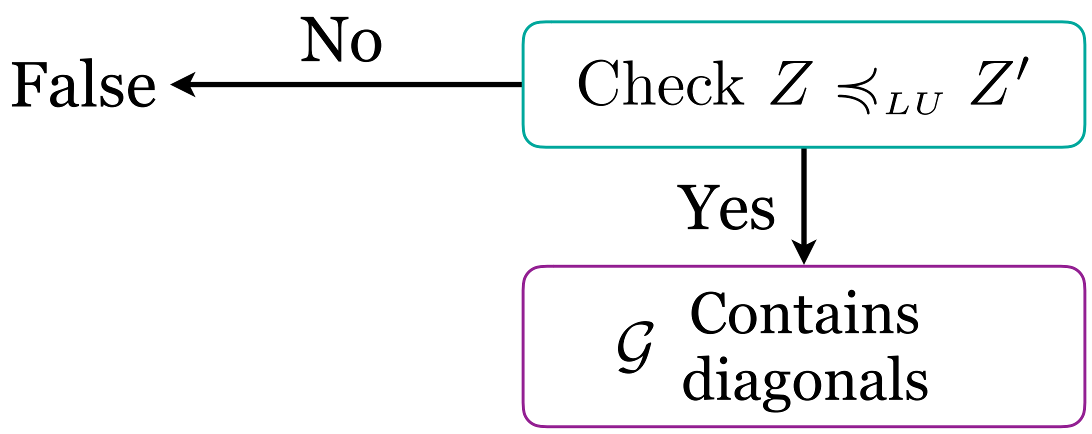
False ← No — Check $Z \preccurlyeq_{LU} Z'$

What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G},\ v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z\ \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$

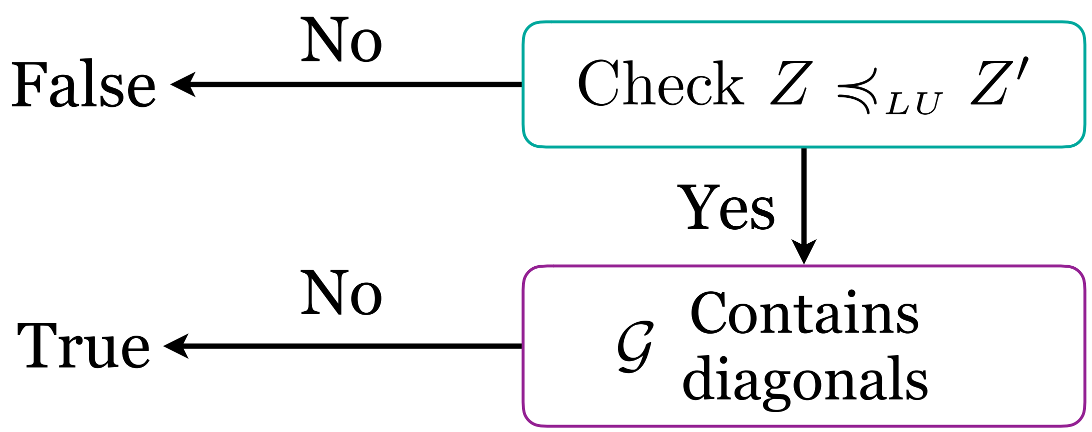


What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G}$, $v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z \; \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$

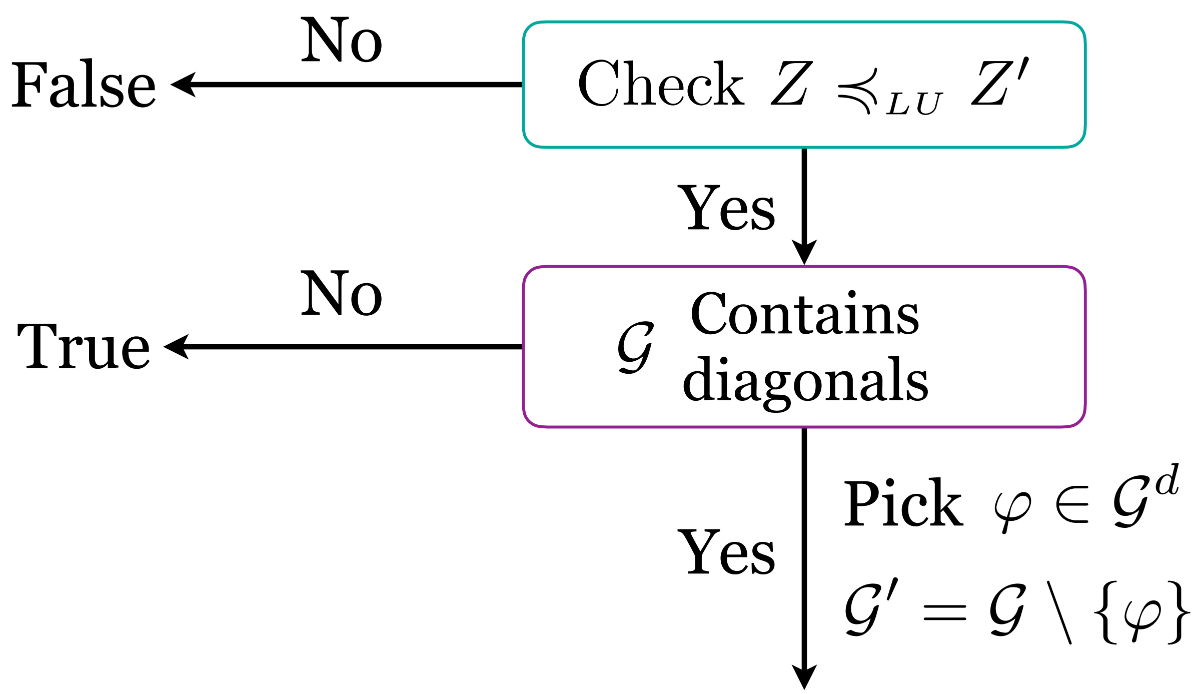


What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G},\ v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z\ \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$

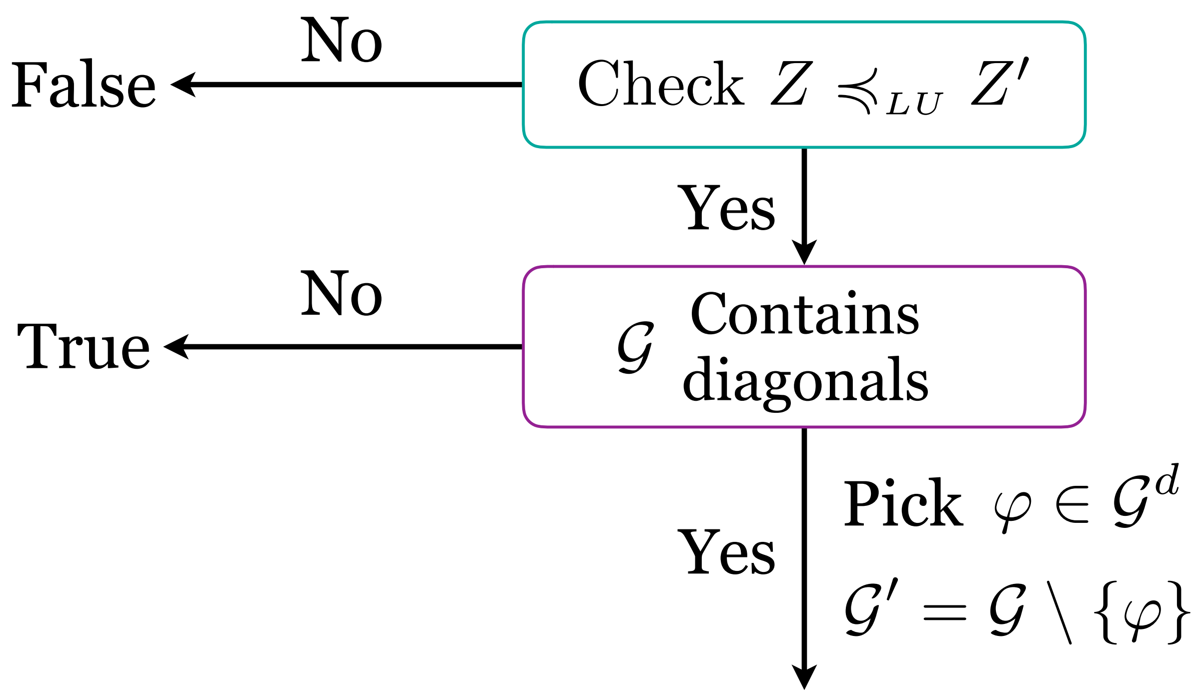


What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G}$, $v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z\ \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$



$$Z \sqsubseteq_{\mathcal{G}} Z'$$

if and only if

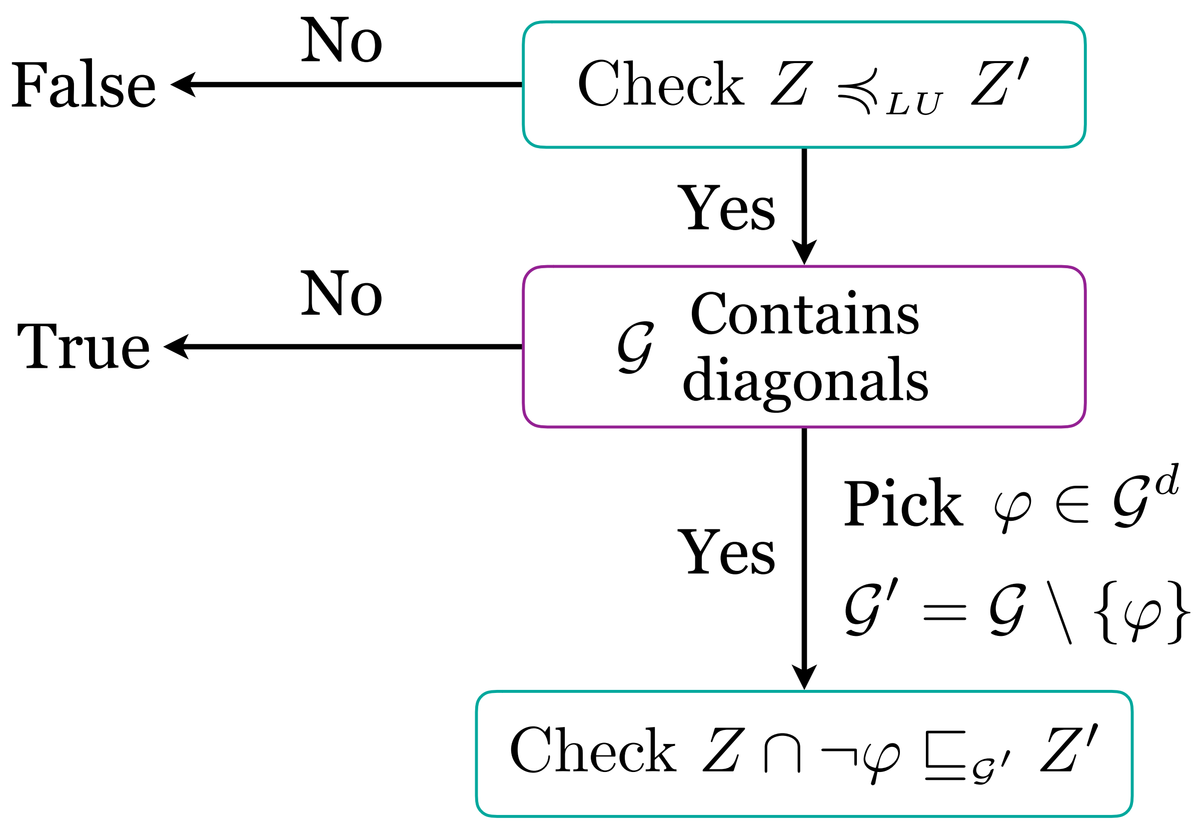
$$Z \cap \neg\varphi \sqsubseteq_{\mathcal{G}'} Z' \text{ and}$$
$$Z \cap \varphi \sqsubseteq_{\mathcal{G}'} Z' \cap \varphi$$

What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G}$, $v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z \, \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$



$Z \sqsubseteq_{\mathcal{G}} Z'$

if and only if

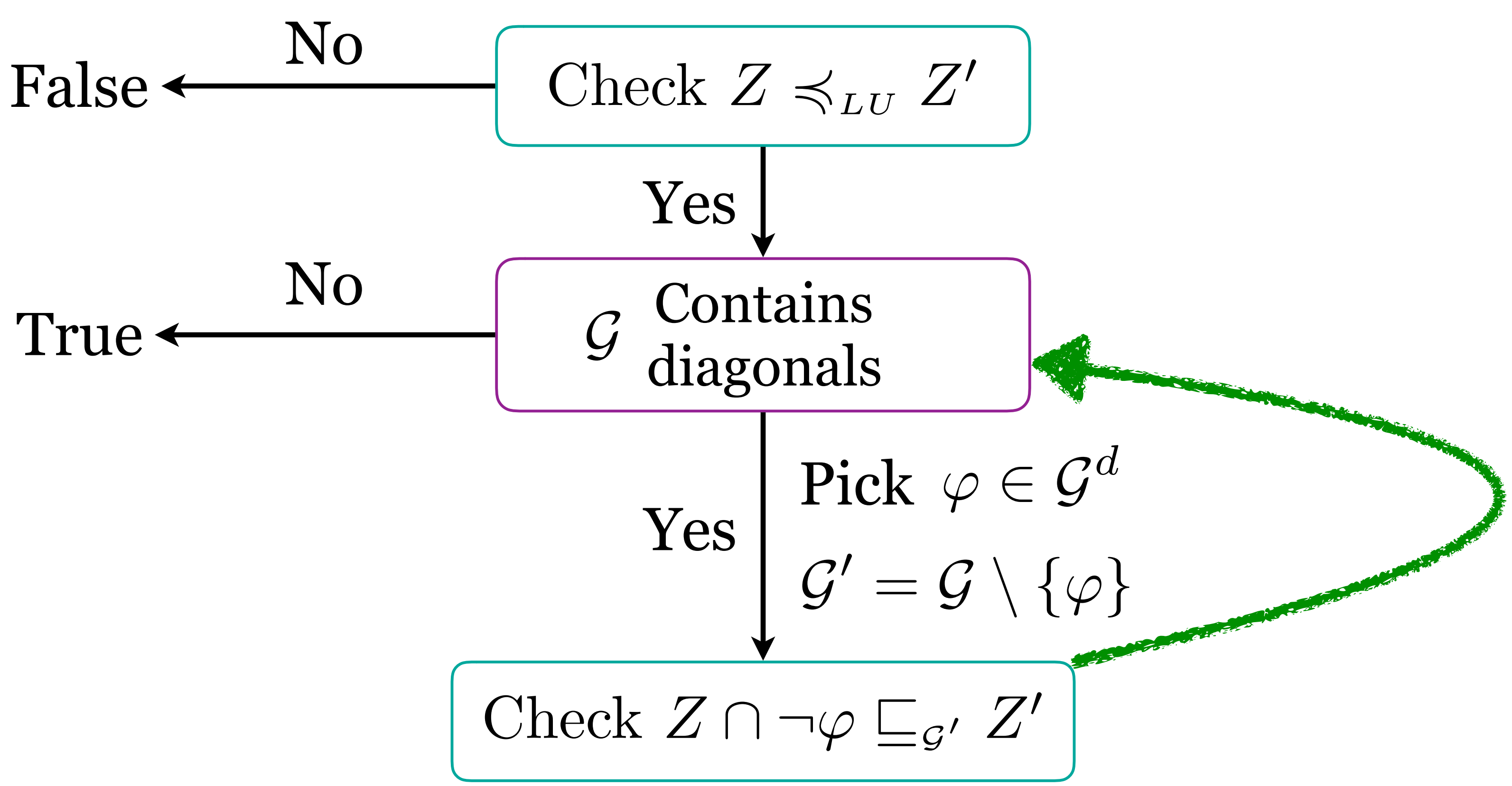
$Z \cap \neg\varphi \sqsubseteq_{\mathcal{G}'} Z'$ and
$Z \cap \varphi \sqsubseteq_{\mathcal{G}'} Z' \cap \varphi$

What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G},\ v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z\ \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$



$Z \sqsubseteq_{\mathcal{G}} Z'$

if and only if

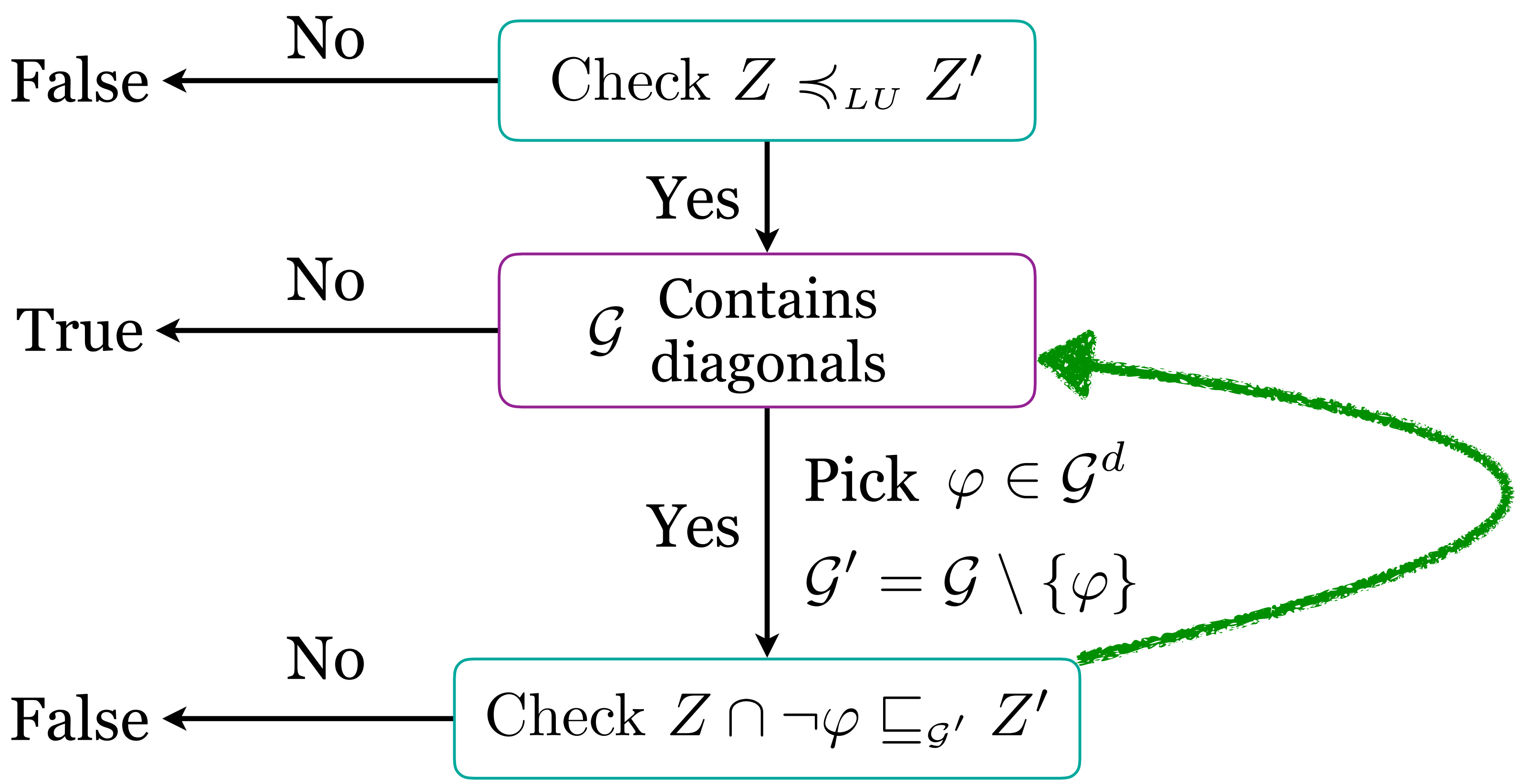
$Z \cap \neg\varphi \sqsubseteq_{\mathcal{G}'} Z'$ and
$Z \cap \varphi \sqsubseteq_{\mathcal{G}'} Z' \cap \varphi$

What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G}$, $v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z \; \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$



$Z \sqsubseteq_{\mathcal{G}} Z'$

if and only if

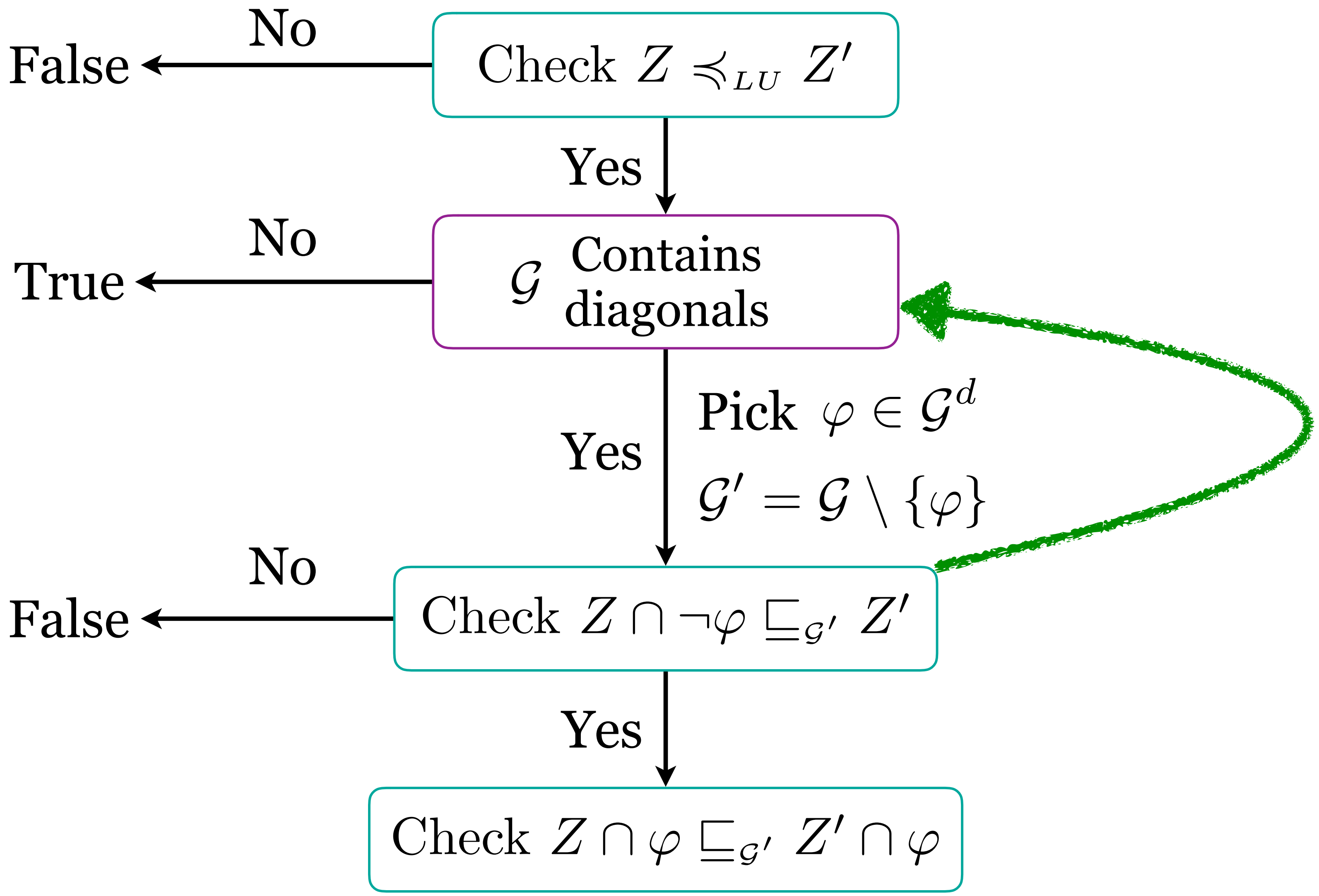
$Z \cap \neg\varphi \sqsubseteq_{\mathcal{G}'} Z'$ and
$Z \cap \varphi \sqsubseteq_{\mathcal{G}'} Z' \cap \varphi$

What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G},\ v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z\ \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$



$Z \sqsubseteq_{\mathcal{G}} Z'$

if and only if

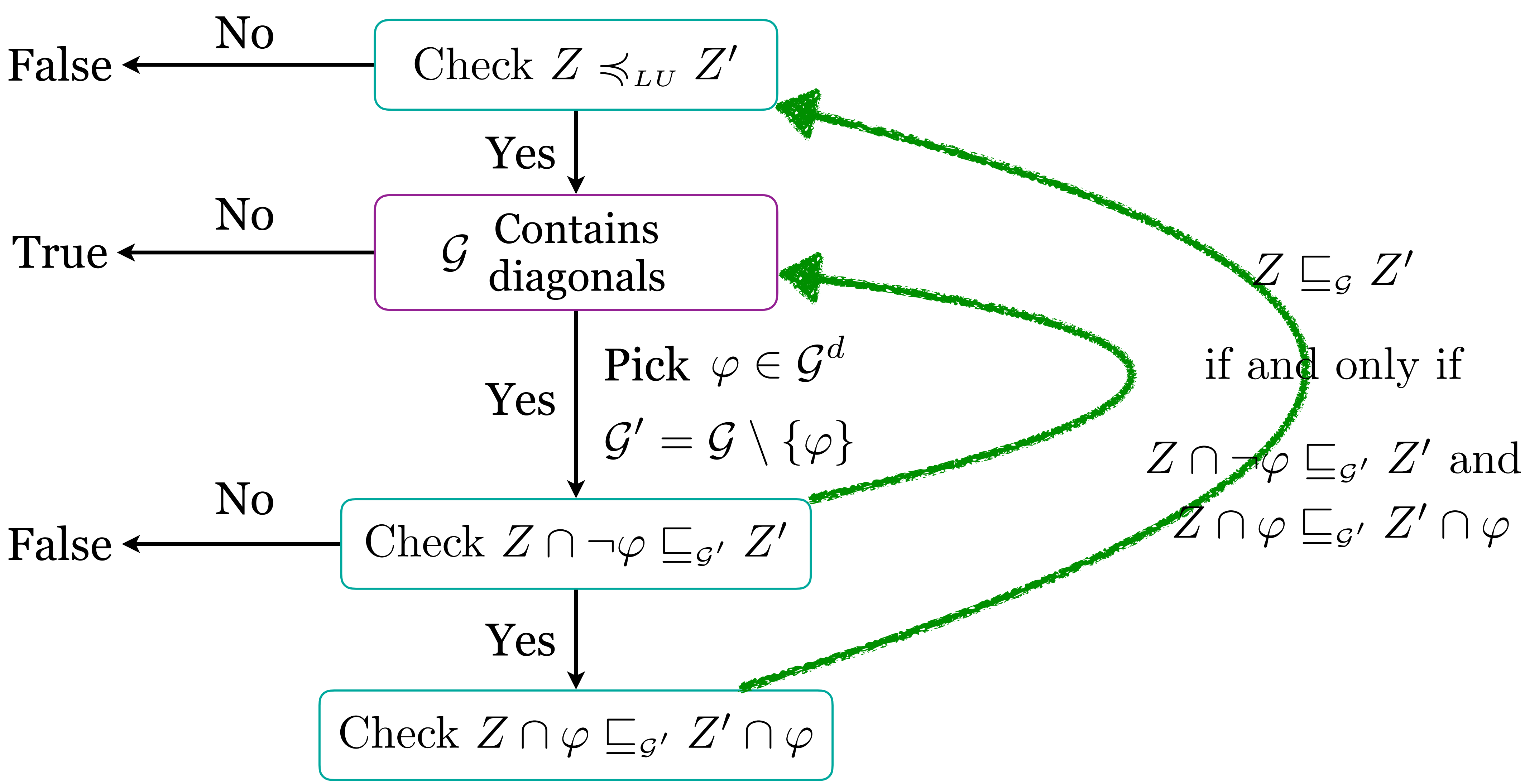
$Z \cap \neg\varphi \sqsubseteq_{\mathcal{G}'} Z'$ and
$Z \cap \varphi \sqsubseteq_{\mathcal{G}'} Z' \cap \varphi$

What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G}$, $v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z\ \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$

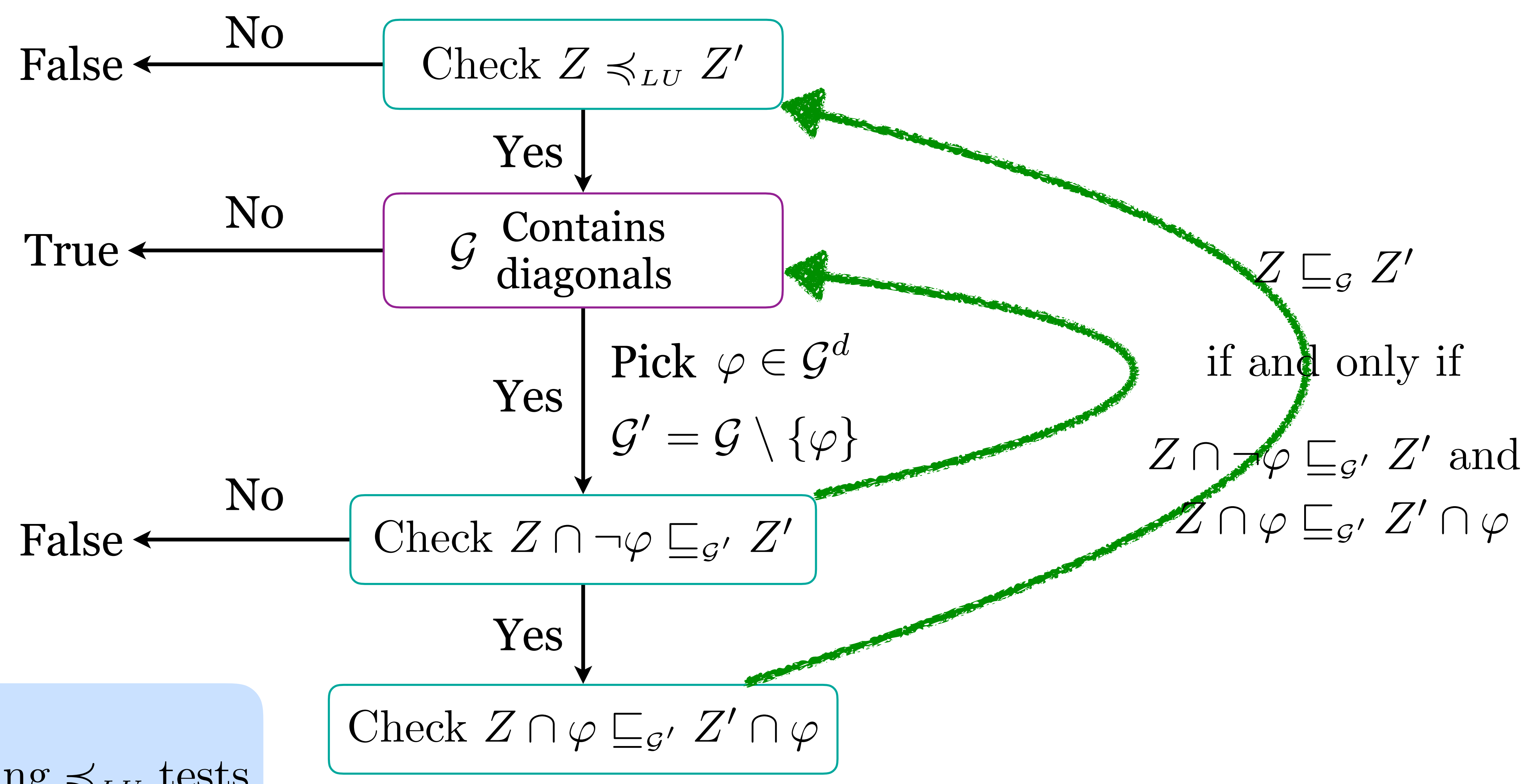


What about zones?

This check is NP-hard

$v \preccurlyeq_{LU} v'$ and $\forall$ diagonal $\varphi \in \mathcal{G}$, $v \models \varphi \implies v' \models \varphi$

$$Z \sqsubseteq_{\mathcal{G}} Z' \text{ if } \forall v \in Z\ \exists v' \in Z' \text{ such that } v \sqsubseteq_{\mathcal{G}} v'$$



We are only doing $\preccurlyeq_{LU}$ tests

Can we already handle diagonal constraints?

Yes!

Then why are you here?

We think we can do better than what we do now

Really? Do you have concrete evidence?

Yes!

What are the existing methods?

Can we avoid Removing diagonals?

Yes!

What is your method?

Avoid blowups in *both* states and zones

What about updates?

Bouyer, Dufourd, Fleury, Petit '04

Updatable Timed Automata

Timed Automata

+

Updates

Update ::= $x := c \mid x := y + d \qquad y \in X, d \in \mathbb{Z}, c \in \mathbb{N}$

<u>Use cases</u>

UPPAAL-TIMES (Tool for Scheduling) $x := x - c,\ x := c$

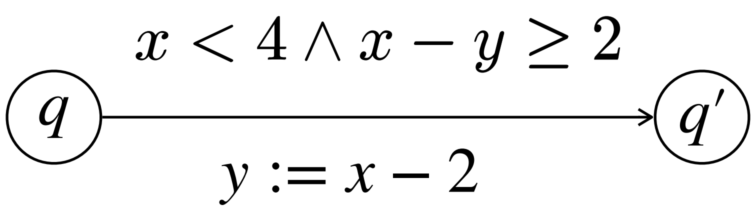
Adaptive Task Automata with EDF Scheduling $x := y$

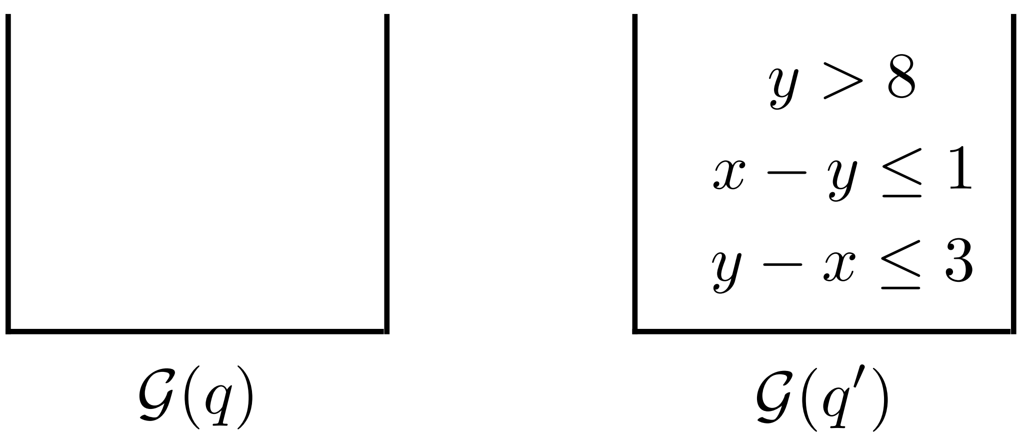
[Hatvani, David, Seceleanu, Pettersson AVoCS '14]

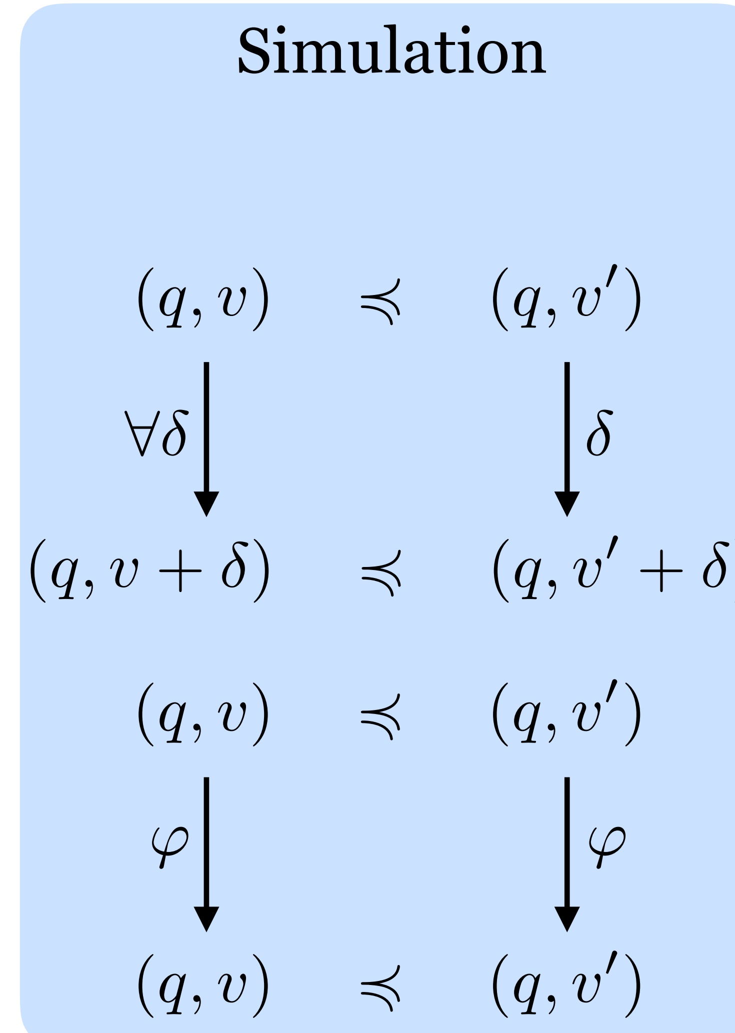
State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$





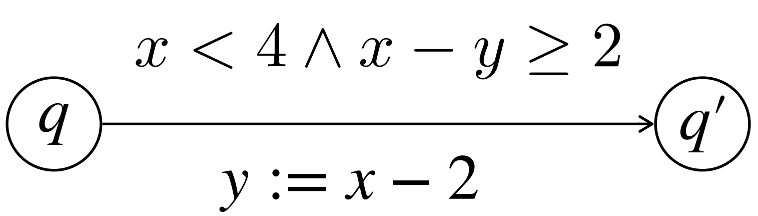


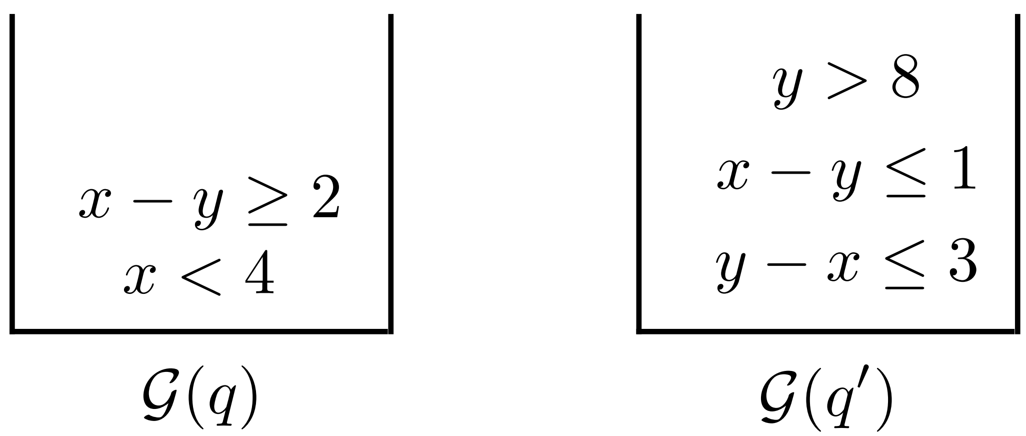
$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \vDash \varphi \implies v' + \delta \vDash \varphi$$

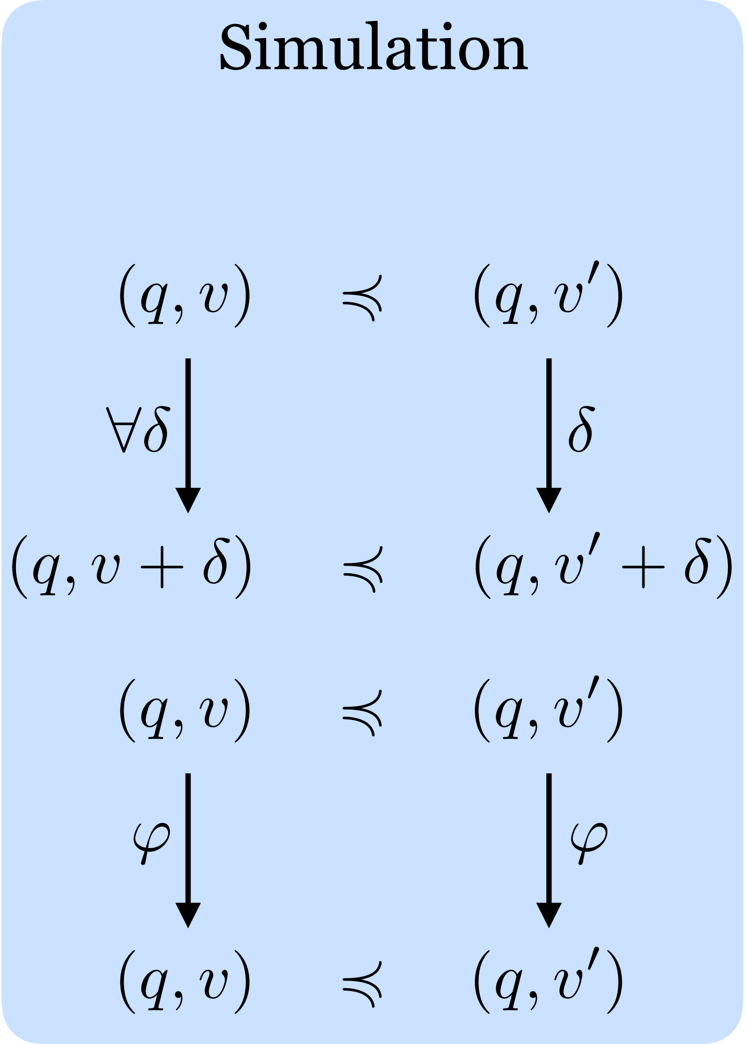
State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$





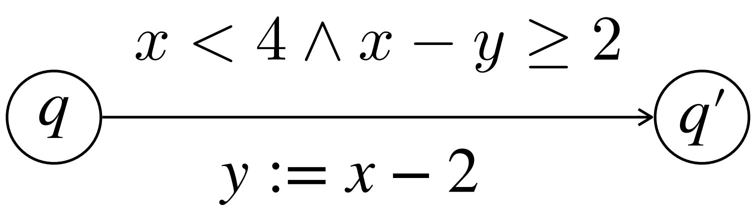


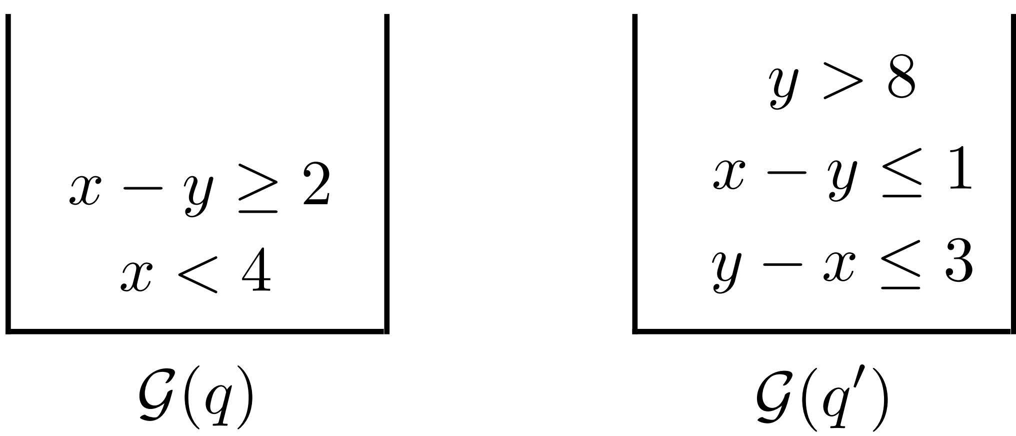
$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \text{if} \quad \forall \varphi \in \mathcal{G}(q) \ \ \forall \delta \geq 0 \quad v + \delta \vDash \varphi \implies v' + \delta \vDash \varphi$$

State based guards

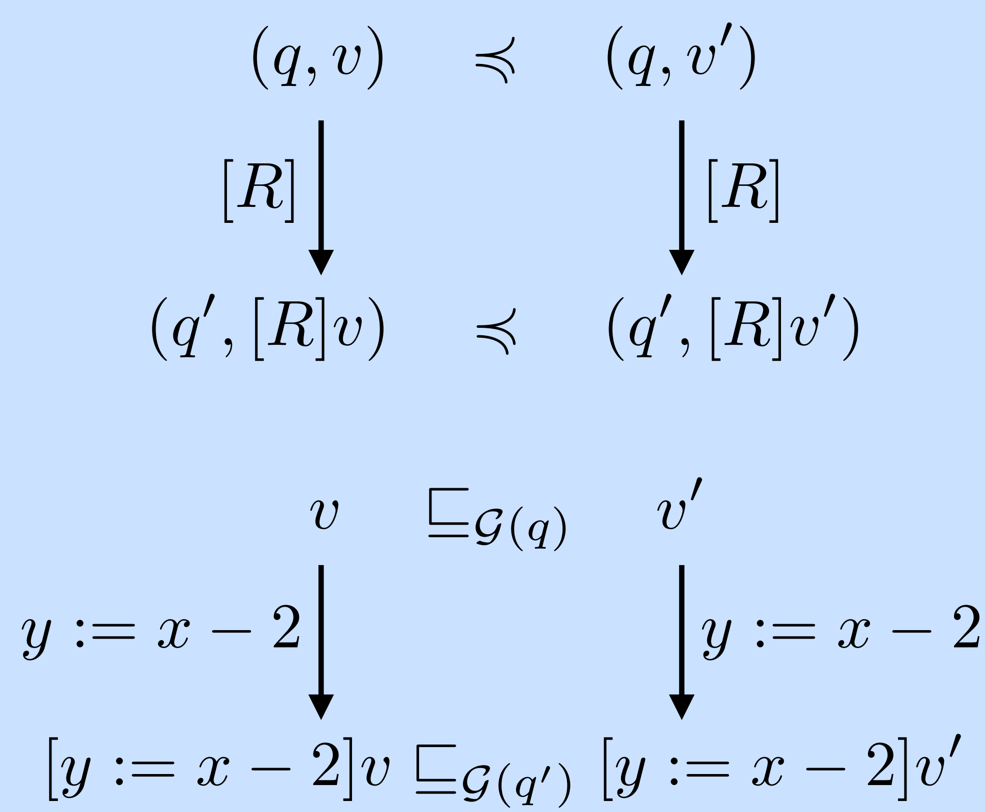
Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$





Simulation



$$\forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \vDash \varphi \implies v' + \delta \vDash \varphi$$

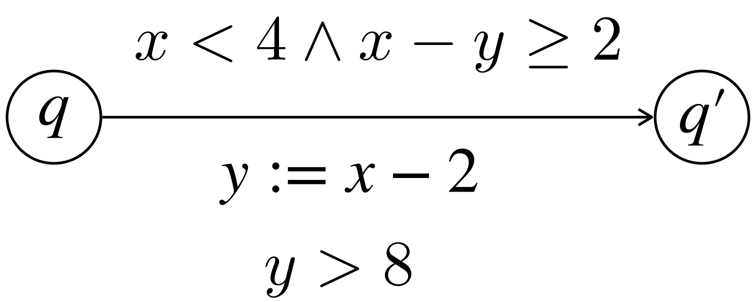
For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



$\mathcal{G}(q)$: $x - y \geq 2$, $x < 4$

$\mathcal{G}(q')$: $y > 8$, $x - y \leq 1$, $y - x \leq 3$

$$\forall \varphi \in \mathcal{G}(q) \ \ \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

Simulation

$$(q, v) \preccurlyeq (q, v') \quad \xrightarrow{[R]} \quad (q', [R]v) \preccurlyeq (q', [R]v')$$

$$v \sqsubseteq_{\mathcal{G}(q)} v' \quad \xrightarrow{y := x - 2} \quad [y := x - 2]v \sqsubseteq_{\mathcal{G}(q')} [y := x - 2]v'$$

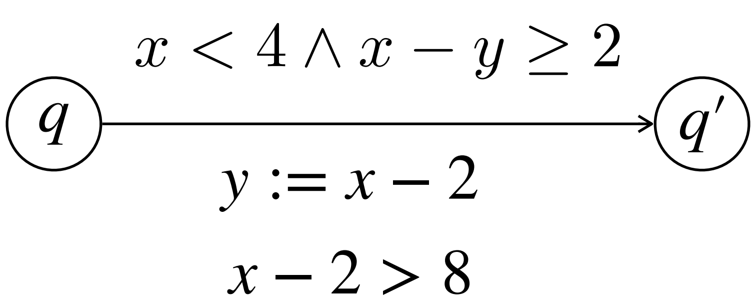
For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



$x - y \geq 2$

$x < 4$

$\mathcal{G}(q)$

$y > 8$

$x - y \leq 1$

$y - x \leq 3$

$\mathcal{G}(q')$

Simulation

$$(q, v) \preccurlyeq (q, v')$$

$$\downarrow [R] \qquad \downarrow [R]$$

$$(q', [R]v) \preccurlyeq (q', [R]v')$$

$$v \sqsubseteq_{\mathcal{G}(q)} v'$$

$$\downarrow y := x - 2 \qquad \downarrow y := x - 2$$

$$[y := x - 2]v \sqsubseteq_{\mathcal{G}(q')} [y := x - 2]v'$$

$$\forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

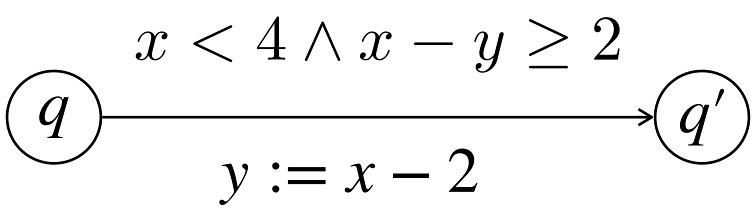
For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



$\mathcal{G}(q)$: $x - 2 > 8$, $x - y \geq 2$, $x < 4$

$\mathcal{G}(q')$: $y > 8$, $x - y \leq 1$, $y - x \leq 3$

Simulation

$$(q, v) \preccurlyeq (q, v')$$

$$\downarrow [R] \qquad \downarrow [R]$$

$$(q', [R]v) \preccurlyeq (q', [R]v')$$

$$v \sqsubseteq_{\mathcal{G}(q)} v'$$

$$\downarrow y := x - 2 \qquad \downarrow y := x - 2$$

$$[y := x - 2]v \sqsubseteq_{\mathcal{G}(q')} [y := x - 2]v'$$

$$\forall \varphi \in \mathcal{G}(q) \quad \forall \delta \geq 0 \quad v + \delta \vDash \varphi \implies v' + \delta \vDash \varphi$$

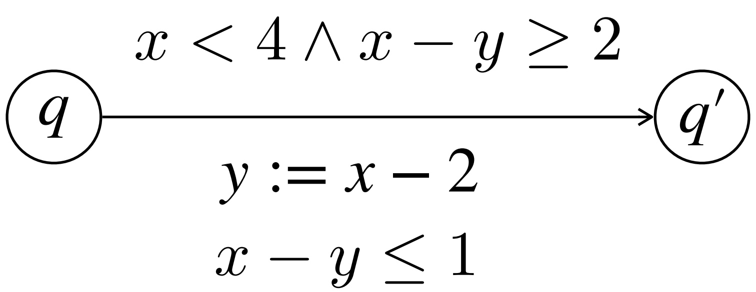
For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



$\mathcal{G}(q)$: $x - 2 > 8$, $x - y \geq 2$, $x < 4$

$\mathcal{G}(q')$: $y > 8$, $x - y \leq 1$, $y - x \leq 3$

Simulation

$$\begin{array}{ccc} (q, v) & \preccurlyeq & (q, v') \\ \downarrow [R] & & \downarrow [R] \\ (q', [R]v) & \preccurlyeq & (q', [R]v') \end{array}$$

$$\begin{array}{ccc} v & \sqsubseteq_{\mathcal{G}(q)} & v' \\ y := x - 2 \downarrow & & \downarrow y := x - 2 \\ [y := x - 2]v & \sqsubseteq_{\mathcal{G}(q')} & [y := x - 2]v' \end{array}$$

$$\forall \varphi \in \mathcal{G}(q) \ \ \forall \delta \geq 0 \quad v + \delta \vDash \varphi \implies v' + \delta \vDash \varphi$$

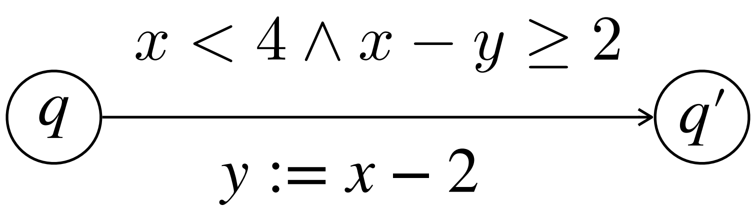
For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



$x - 2 > 8$
$x - y \geq 2$
$x < 4$

$\mathcal{G}(q)$

$y > 8$
$x - y \leq 1$
$y - x \leq 3$

$\mathcal{G}(q')$

Simulation

$$\begin{array}{ccc} (q, v) & \preccurlyeq & (q, v') \\ \downarrow [R] & & \downarrow [R] \\ (q', [R]v) & \preccurlyeq & (q', [R]v') \end{array}$$

$$\begin{array}{ccc} v & \sqsubseteq_{\mathcal{G}(q)} & v' \\ \downarrow y := x - 2 & & \downarrow y := x - 2 \\ [y := x - 2]v & \sqsubseteq_{\mathcal{G}(q')} & [y := x - 2]v' \end{array}$$

$$\forall \varphi \in \mathcal{G}(q) \quad \forall \delta \geq 0 \quad v + \delta \vDash \varphi \implies v' + \delta \vDash \varphi$$

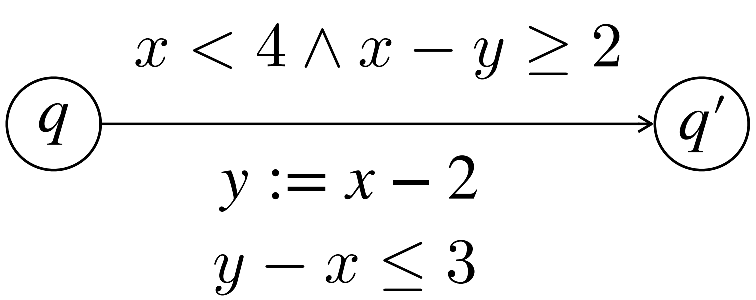
For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



$\mathcal{G}(q)$: $x - 2 > 8$, $x - y \geq 2$, $x < 4$

$\mathcal{G}(q')$: $y > 8$, $x - y \leq 1$, $y - x \leq 3$

Simulation

$$(q, v) \preccurlyeq (q, v')$$
$$\downarrow [R] \qquad \downarrow [R]$$
$$(q', [R]v) \preccurlyeq (q', [R]v')$$

$$v \sqsubseteq_{\mathcal{G}(q)} v'$$
$$\downarrow y := x - 2 \qquad \downarrow y := x - 2$$
$$[y := x - 2]v \sqsubseteq_{\mathcal{G}(q')} [y := x - 2]v'$$

$$\forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \vDash \varphi \implies v' + \delta \vDash \varphi$$

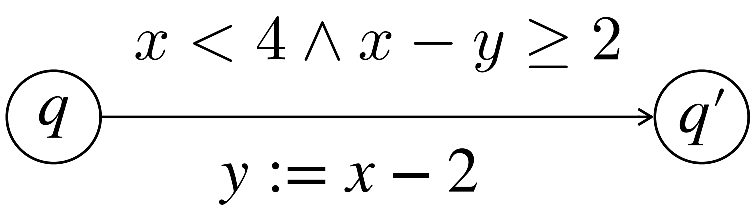
For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



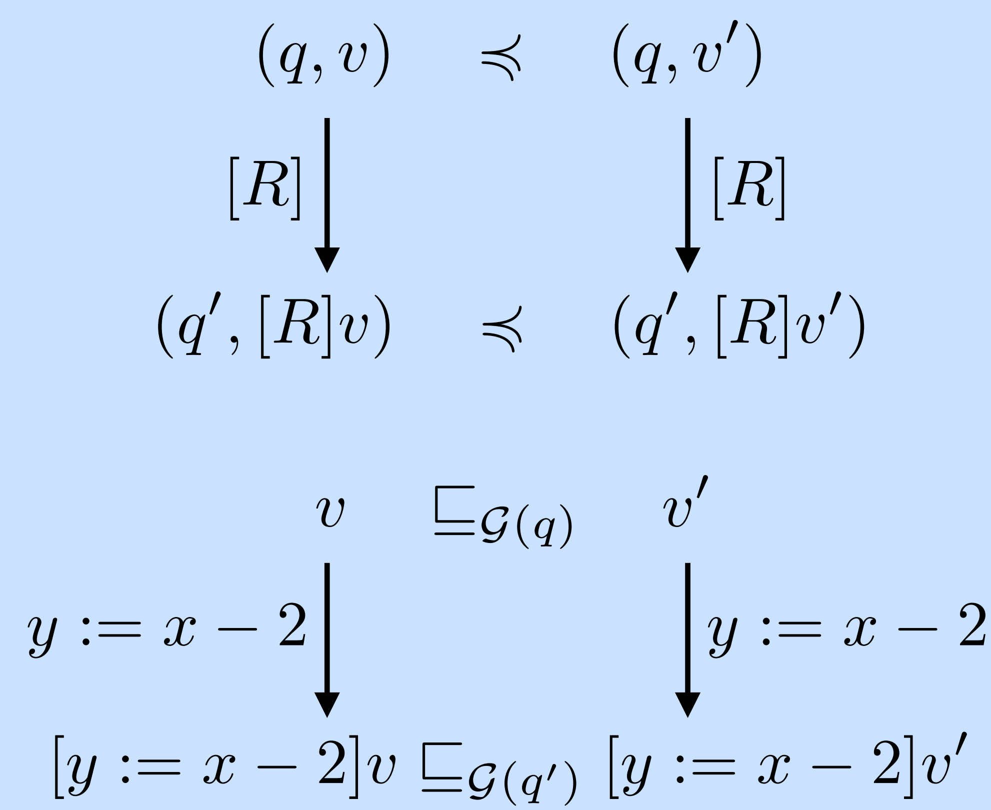
$$x - 2 > 8 \qquad x - y \geq 2 \qquad x < 4$$

$\mathcal{G}(q)$

$$y > 8 \qquad x - y \leq 1 \qquad y - x \leq 3$$

$\mathcal{G}(q')$

Simulation



$$\forall \varphi \in \mathcal{G}(q) \;\; \forall \delta \geq 0 \quad v + \delta \models \varphi \implies v' + \delta \models \varphi$$

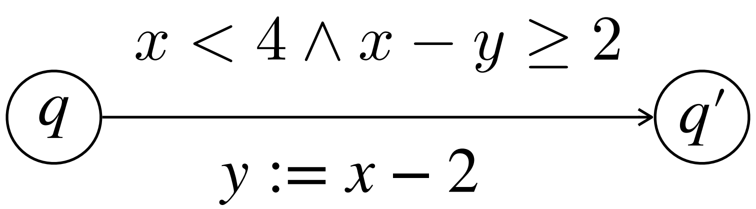
For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

State based guards

Given $\mathcal{A}$ associate $\mathcal{G}(q)$ to every state q of $\mathcal{A}$ so that $\preccurlyeq_{\mathcal{A}}$ is a simulation where

$$(q, v) \preccurlyeq_{\mathcal{A}} (q, v') \quad \text{if} \quad v \sqsubseteq_{\mathcal{G}(q)} v'$$



For all $q \xrightarrow{\varphi, R} q'$

$$\mathcal{G}(q) \supseteq \{\varphi\} \cup wp(\sqsubseteq_{\mathcal{G}(q')}, R)$$

$\mathcal{G}(q)$:

$x - 2 > 8$

$x - y \geq 2$

$x < 4$

$\mathcal{G}(q')$:

$y > 8$

$x - y \leq 1$

$y - x \leq 3$

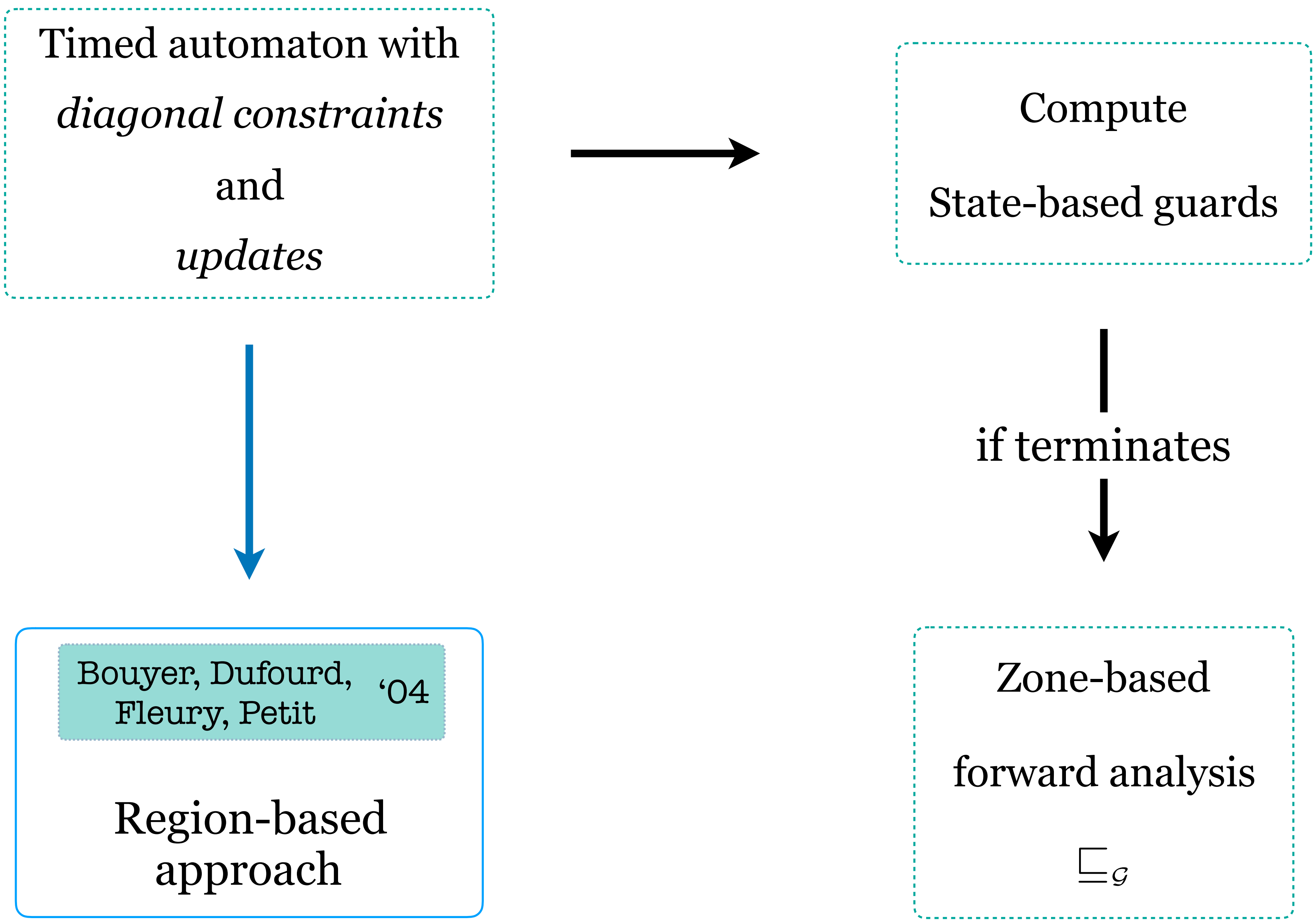
New constants may get generated

This least fixed-point computation may not always terminate

If number of iterations crosses a fixed number, we can conclude that it will never terminate

If this fixed-point computation terminates reachability can be checked

Algorithm


Timed automaton with
diagonal constraints
and
updates
Compute
State-based guards
if terminates
Bouyer, Dufourd, Fleury, Petit
'04
Region-based approach
Zone-based
forward analysis
$\sqsubseteq_{\mathcal{G}}$

Conclusions and future work

Simulation approach avoids exponential blowups
in both state and zone levels

Difficulty transferred to the Simulation check (NP-hard)

Optimized simulation test

Encouraging experiments

Incorporating *Updates* in the implementation and
applying it to Scheduling

Further optimizations to the simulation based algorithm

Can we already handle diagonal constraints?

Yes!

Then why are you here?

We think we can do better than what we do now

Really? Do you have concrete evidence?

Yes!

What are the existing methods?

Can we avoid Removing diagonals?

Yes!

What is your method?

Avoid blowups in *both* states and zones

What about updates?

More questions?