Shrinking Timed Automata

Ocan Sankur

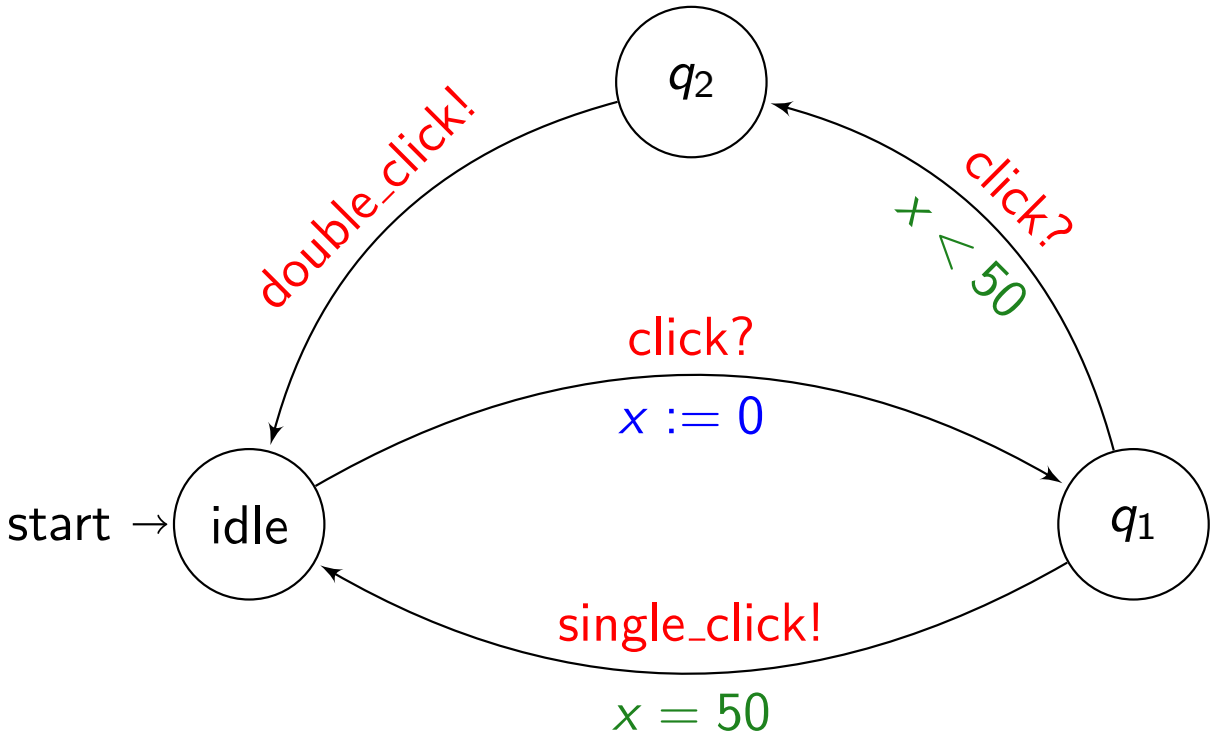
Joint with Patricia Bouyer and Nicolas Markey

ENS Cachan & CNRS

Mumbai, FSTTCS 2011

Timed Automata: **Exact** Semantics

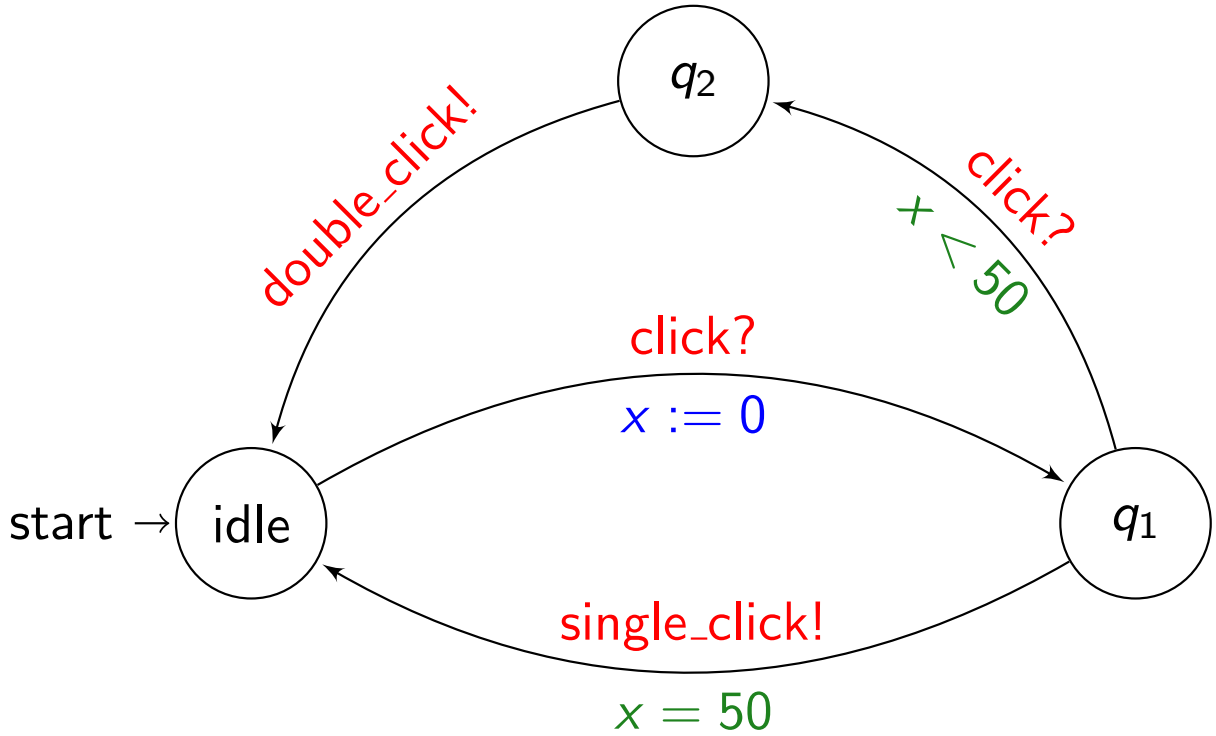
Timed automata = Finite automata + Analog clocks. [Alur and Dill 1994]



- Clocks cannot be stopped, all grow at the same rate.
- An edge is activated when its clock constraint holds.
- A clock can be reset by a transition.

Timed Automata: **Exact** Semantics

Timed automata = Finite automata + Analog clocks. [Alur and Dill 1994]



Runs of a timed automaton: $\mathcal{A}$

$$(\text{idle}, x = 0) \xrightarrow{23.7} (\text{idle}, x = 23.7) \xrightarrow{click?} (q_1, x = 0) \xrightarrow{10} (q_1, x = 10)$$
$$\xrightarrow{click?} (q_2, x = 10) \xrightarrow{double\_click} (\text{idle}, x = 10) \ \ldots$$

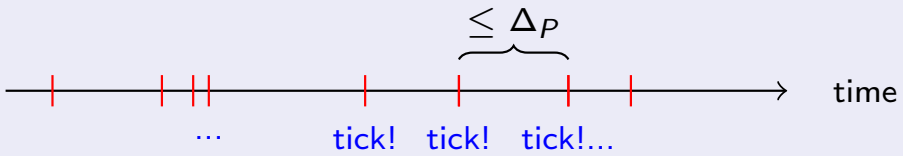
Timed Automata: **Program** Semantics

The semantics of timed automata is idealistic:

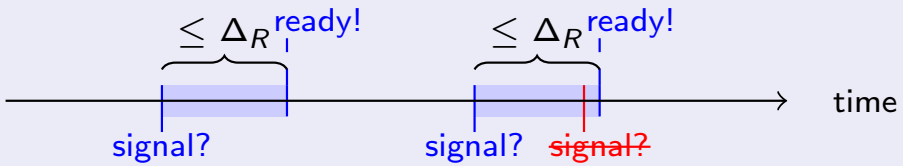
- No minimum delay between actions, $\xrightarrow{a}\xrightarrow{0.00001}\xrightarrow{b}$.
- clocks are infinitely precise. "$1 \leq x \leq 3$".

Real-world systems have

- **digital clocks** updated regularly:



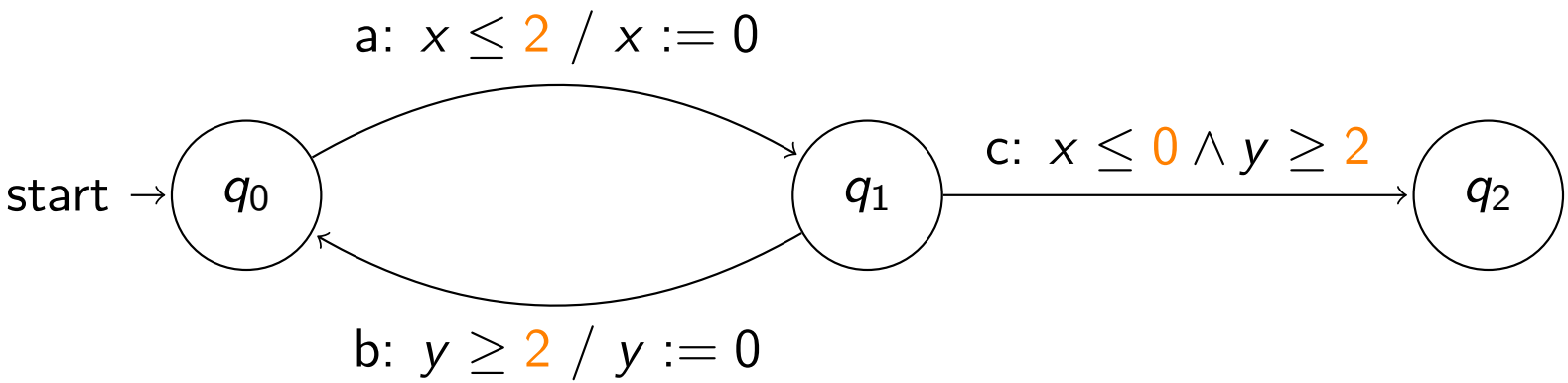
- nonzero **reaction time**:



We study a variant of the **program semantics** of [De Wulf, Doyen and Raskin 2004].

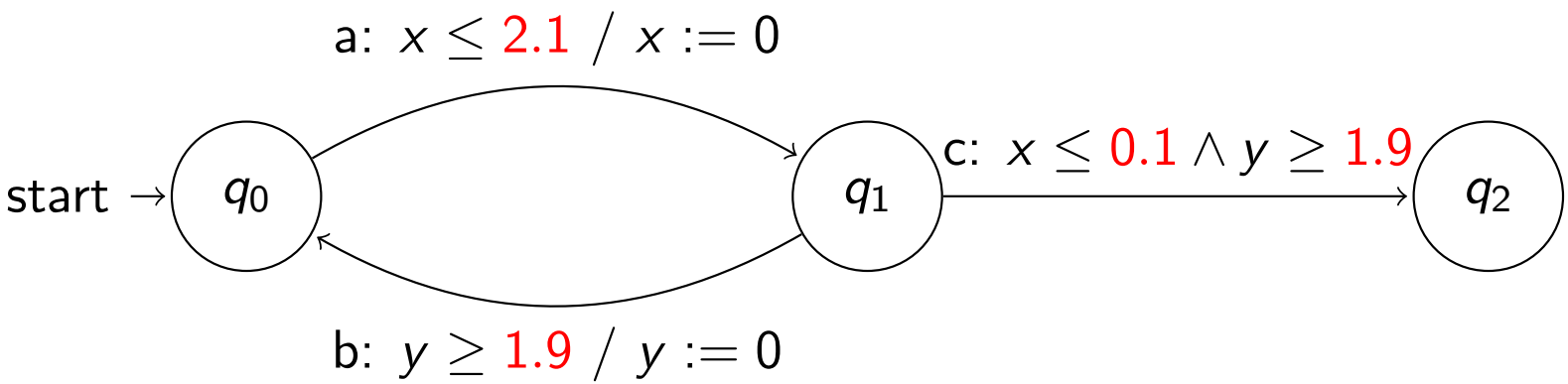
Timed Automata: **Enlarged** Semantics

Clock imprecisions can be modelled by **enlarging** the clock constraints.
Consider the timed automaton $\mathcal{A}$:



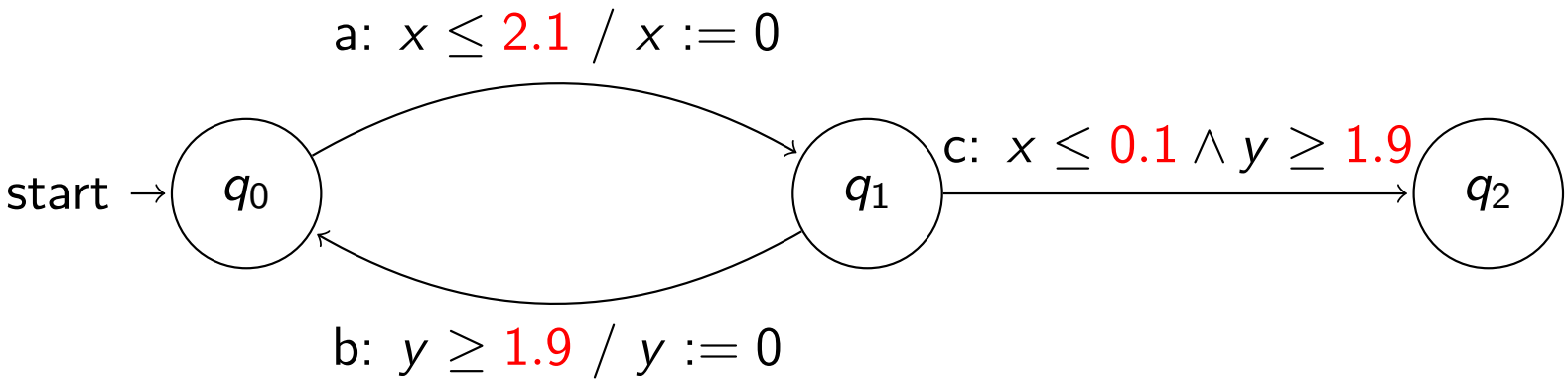
Timed Automata: **Enlarged** Semantics

Clock imprecisions can be modelled by **enlarging** the clock constraints.
For $\Delta = 0.1$, $\mathcal{A}_\Delta$ is defined by,



Timed Automata: **Enlarged** Semantics

Clock imprecisions can be modelled by **enlarging** the clock constraints.
For $\Delta = 0.1$, $\mathcal{A}_\Delta$ is defined by,



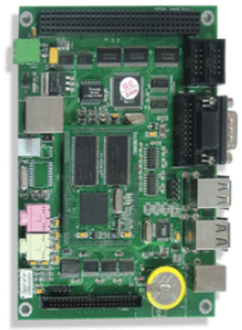
Relation between semantics

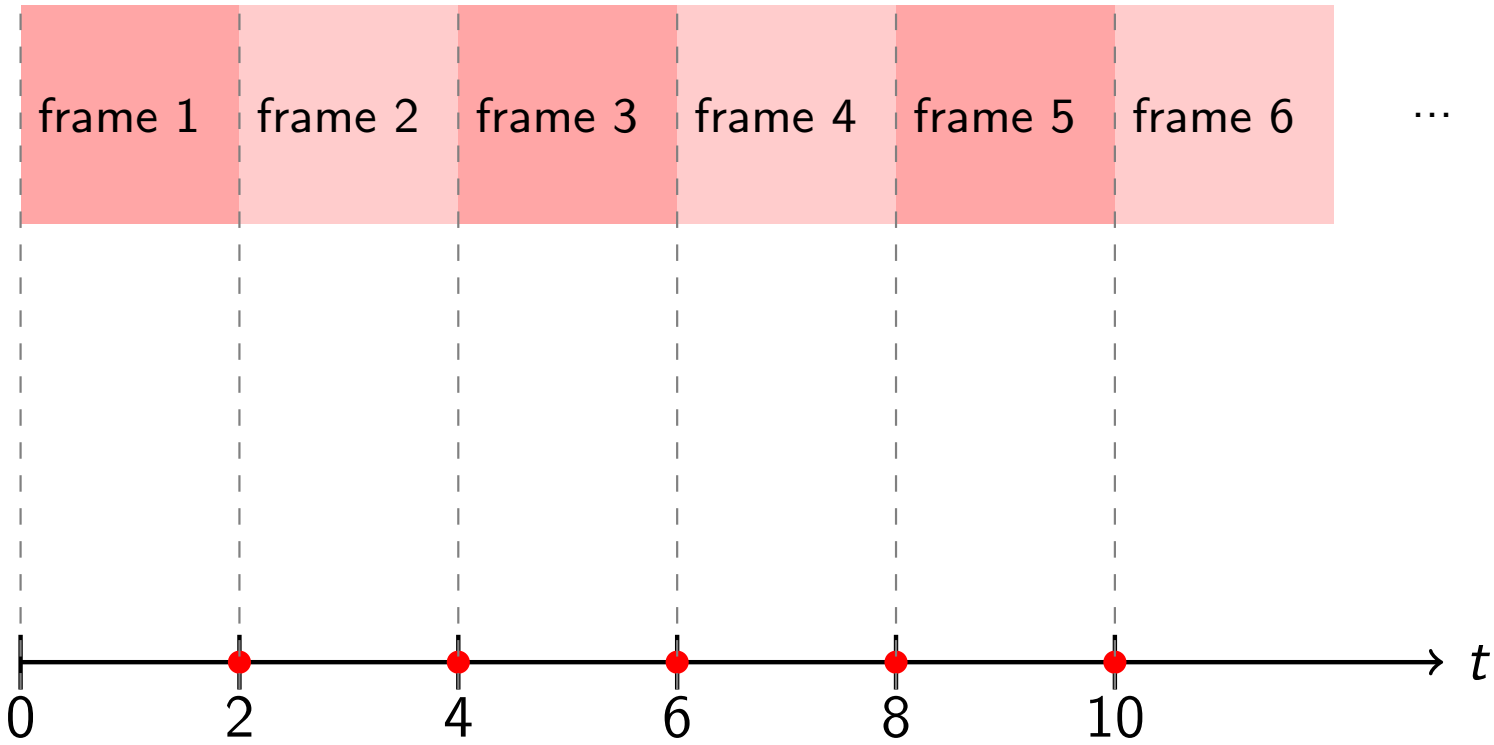
$$\mathcal{A} \quad \sqsubseteq \quad \text{program}(\mathcal{A}_\Delta) \quad \sqsubseteq \quad \mathcal{A}_{2\Delta}$$

for some $\Delta > 0$, **this work** + [De Wulf, Doyen, Raskin 2004].

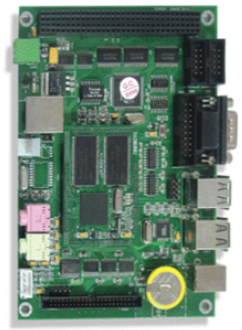
"The implementation has more behaviours than the exact semantics".

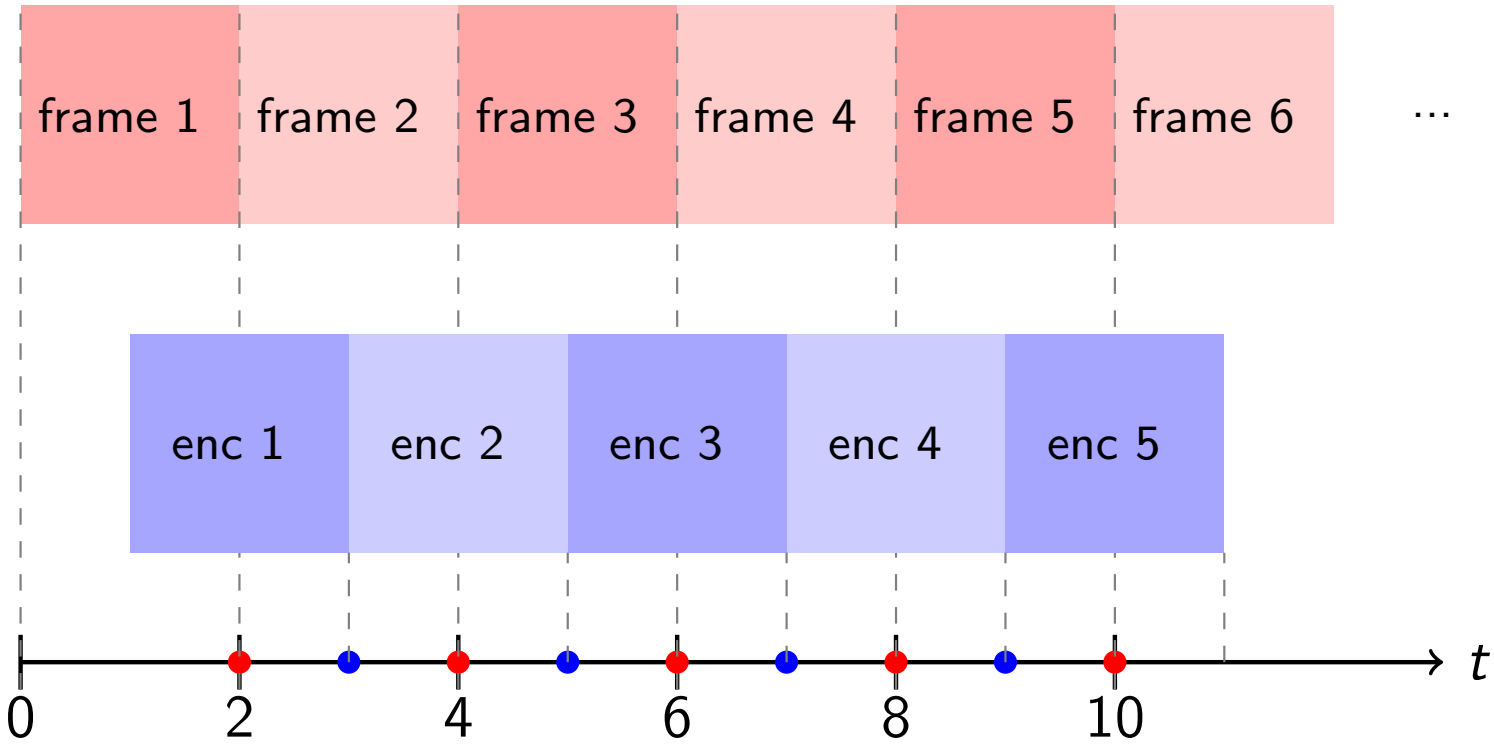
A **Non-Robust** Timed System



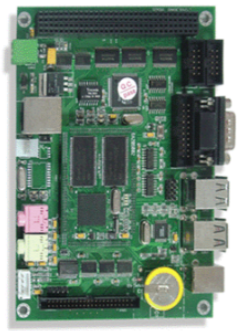


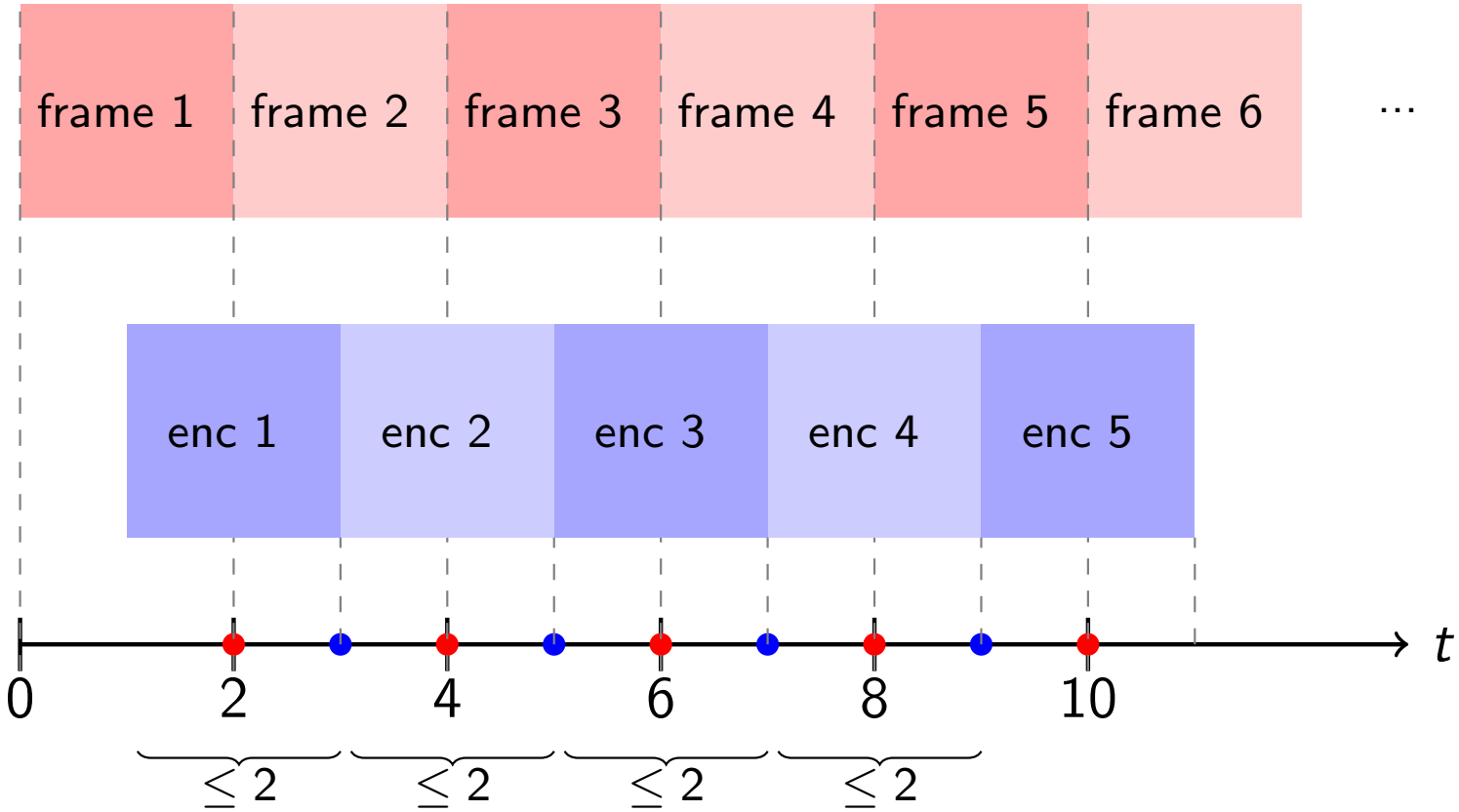
A **Non-Robust** Timed System



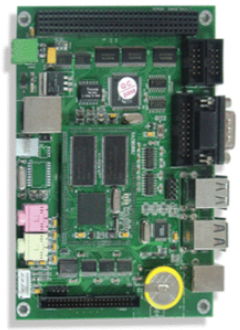


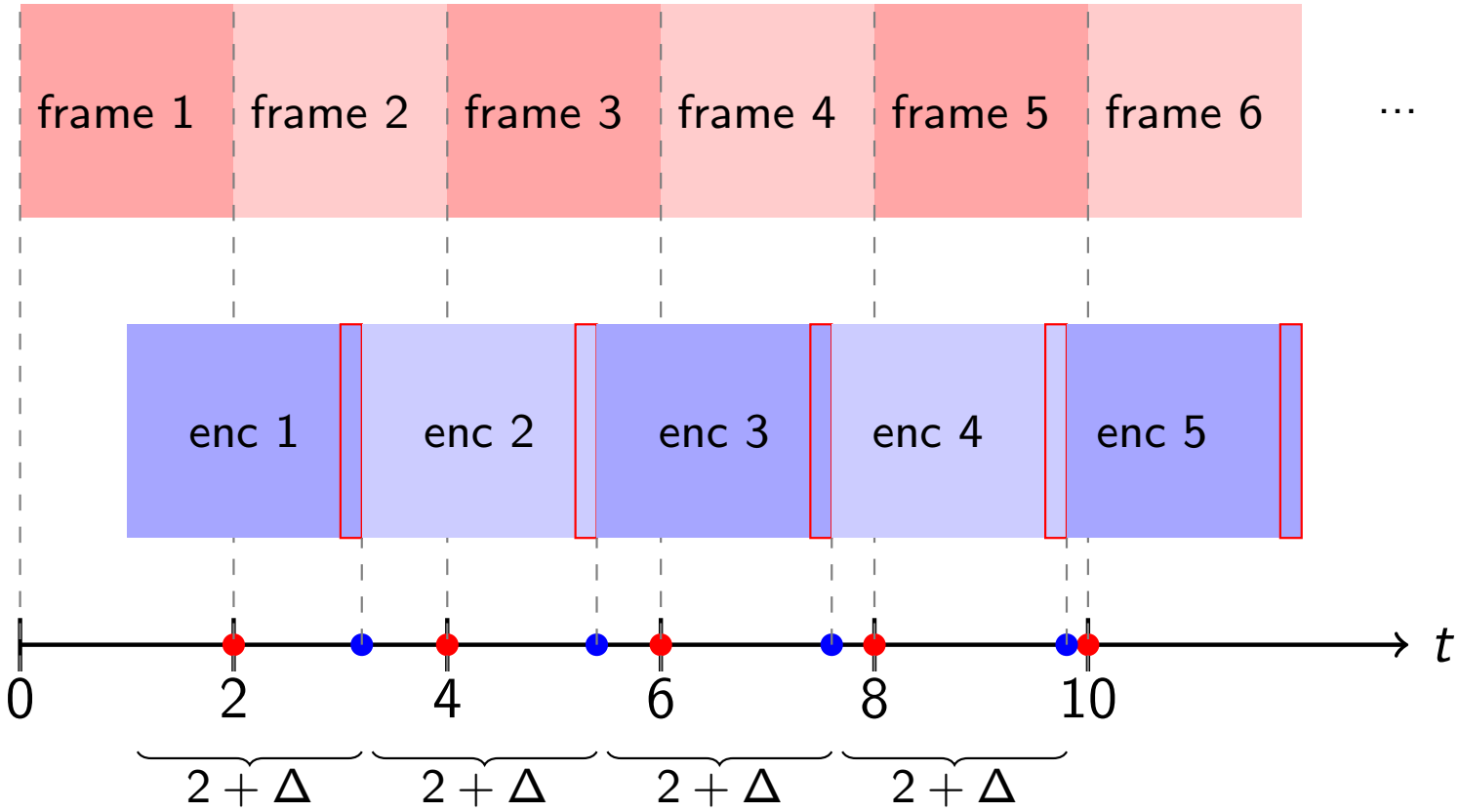
A **Non-Robust** Timed System



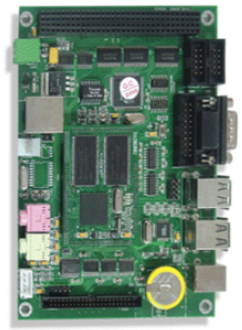


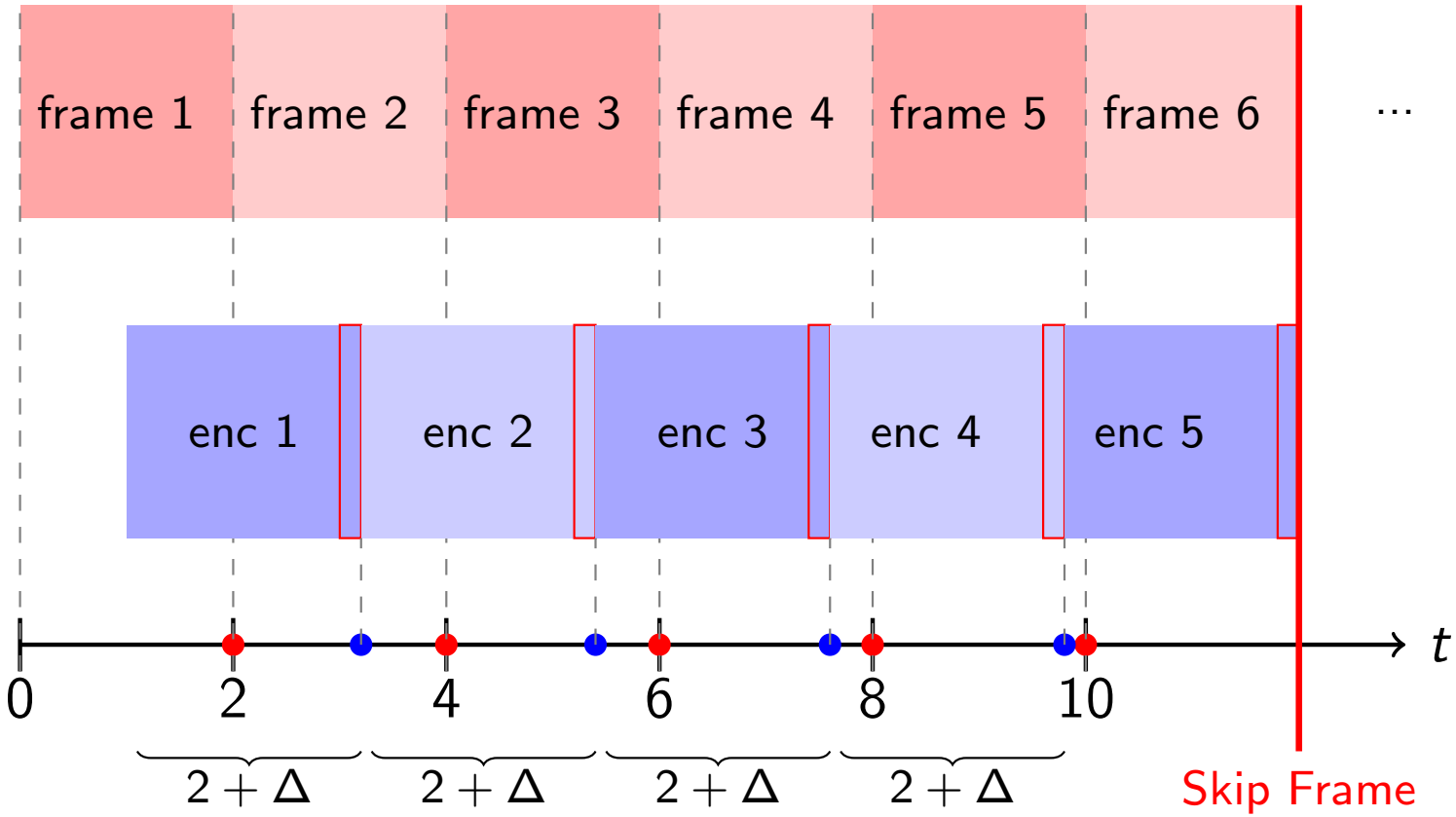
A **Non-Robust** Timed System



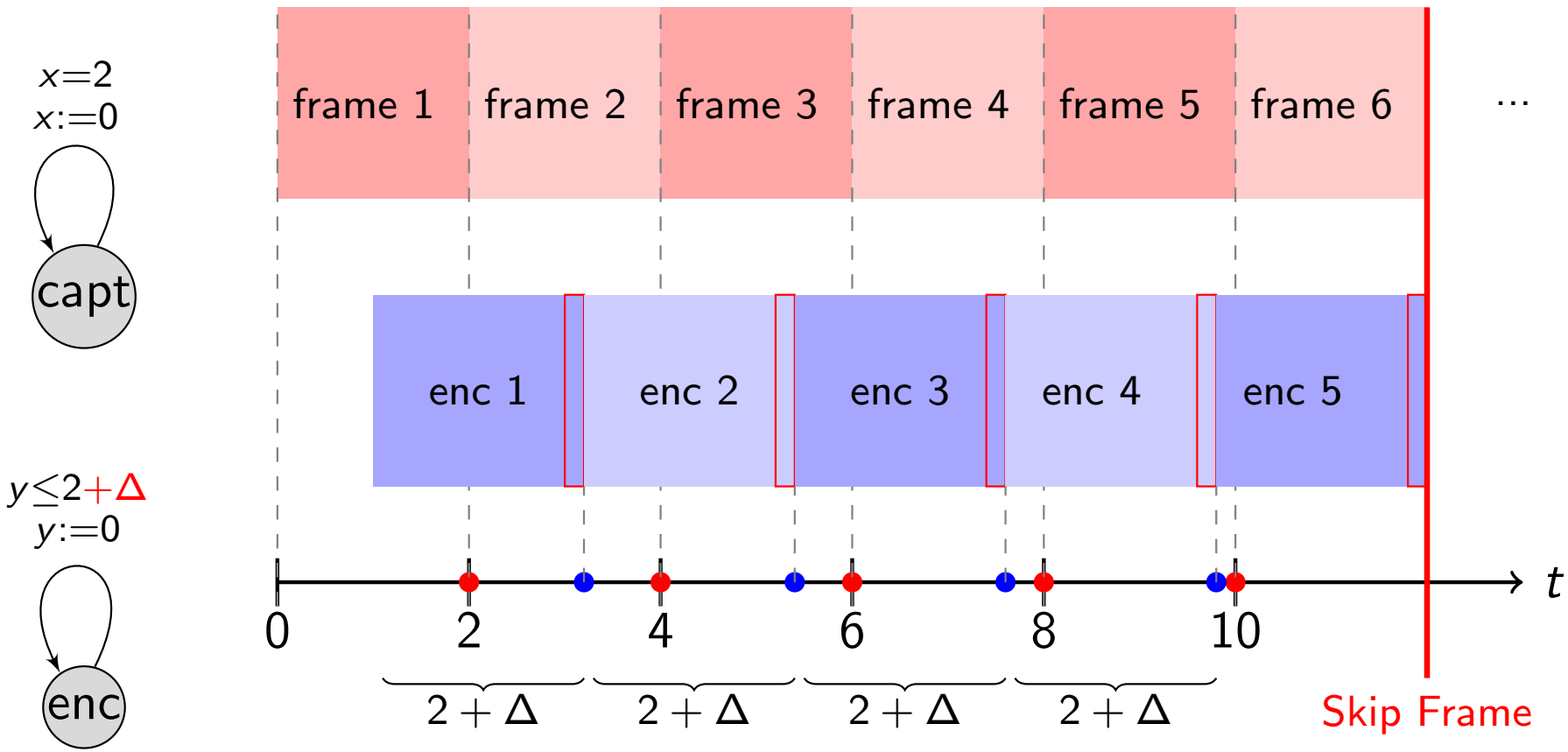


A **Non-Robust** Timed System





A **Non-Robust** Timed System



Background: Enlarged semantics

"Enlarged/Program semantics can **add undesired** behaviour to timed automata".

[Puri 1998, De Wulf, Doyen, Markey, Raskin 2004]

Parameterized Robust Model-Checking

Given TA $\mathcal{A}$ and property ϕ, decide if $\exists \Delta > 0$, $\mathcal{A}_\Delta \models \phi$.

Decidable for:

- Safety, [Puri 1998], [DDMR 2004] [Daws, Kordy 2006], [Jaubert, Reynier 2011]
- LTL, coFlat-MTL [Bouyer, Markey, Reynier 2006/08], [Bouyer, Markey, S. 2011].
- Untimed language equivalence $L(\mathcal{A}) = L(\mathcal{A}_\Delta)$ [S. 2011]

Background: Enlarged semantics

"Enlarged/Program semantics can **add undesired** behaviour to timed automata".

[Puri 1998, De Wulf, Doyen, Markey, Raskin 2004]

Parameterized Robust Model-Checking

Given TA $\mathcal{A}$ and property ϕ, decide if $\exists \Delta > 0$, $\mathcal{A}_\Delta \models \phi$.

Decidable for:

- Safety, [Puri 1998], [DDMR 2004] [Daws, Kordy 2006], [Jaubert, Reynier 2011]
- LTL, coFlat-MTL [Bouyer, Markey, Reynier 2006/08], [Bouyer, Markey, S. 2011].
- Untimed language equivalence $L(\mathcal{A}) = L(\mathcal{A}_\Delta)$ [S. 2011]

Idea of **previous** work

Check whether the additional behaviour introduced by imprecisions is OK.

for small enough $\Delta > 0$

Background: Enlarged semantics

"Enlarged/Program semantics can **add undesired** behaviour to timed automata".

[Puri 1998, De Wulf, Doyen, Markey, Raskin 2004]

Parameterized Robust Model-Checking

Given TA $\mathcal{A}$ and property ϕ, decide if $\exists \Delta > 0$, $\mathcal{A}_\Delta \models \phi$.

Decidable for:

- Safety, [Puri 1998], [DDMR 2004] [Daws, Kordy 2006], [Jaubert, Reynier 2011]
- LTL, coFlat-MTL [Bouyer, Markey, Reynier 2006/08], [Bouyer, Markey, S. 2011].
- Untimed language equivalence $L(\mathcal{A}) = L(\mathcal{A}_\Delta)$ [S. 2011]

Idea of **this work**

Modify the automaton to **exclude** any additional behaviour under imprecisions.

for small enough $\Delta > 0$

Main Tool: Shrinking

Abstract model	Real-world behaviour
$\ell \xrightarrow{1 \leq x \leq 2} \ell'$	$\ell \xrightarrow{1-\Delta \leq x \leq 2+\Delta} \ell'$

Main Tool: Shrinking

Abstract Model	Real-world behaviour
$\ell \xrightarrow{1 \leq x \leq 2} \ell'$	$\ell \xrightarrow{1-\Delta \leq x \leq 2+\Delta} \ell'$
$\ell \xrightarrow{1+\delta' \leq x \leq 2-\delta} \ell'$	$\ell \xrightarrow{1+\delta'-\Delta \leq x \leq 2-\delta+\Delta} \ell'$

Main Tool: Shrinking

Abstract Model	Real-world behaviour
$\ell \xrightarrow{1 \leq x \leq 2} \ell'$	$\ell \xrightarrow{1-\Delta \leq x \leq 2+\Delta} \ell'$
$\ell \xrightarrow{1+\delta' \leq x \leq 2-\delta} \ell'$	$\ell \xrightarrow{1+\delta'-\Delta \leq x \leq 2-\delta+\Delta} \ell'$

$$1 \leq 1 + \delta' - \Delta \leq x \leq 2 - \delta + \Delta \leq 2 \qquad \text{when } \delta, \delta' \geq \Delta.$$

"Shrink the clock constraints in the model, to prevent additional behaviour in implementation".

Main Tool: Shrinking

Abstract Model	Real-world behaviour
$\ell \xrightarrow{1\leq x\leq 2} \ell'$	$\ell \xrightarrow{1-\Delta\leq x\leq 2+\Delta} \ell'$
$\ell \xrightarrow{1+\delta'\leq x\leq 2-\delta} \ell'$	$\ell \xrightarrow{1+\delta'-\Delta\leq x\leq 2-\delta+\Delta} \ell'$

$$1 \leq 1+\delta'-\Delta \leq x \leq 2-\delta+\Delta \leq 2 \qquad \text{when } \delta, \delta' \geq \Delta.$$

We consider a separate **shrinking parameter** for each atomic clock constraint: $k_1\delta, k_2\delta, \ldots$ where $\delta > 0$ and $\vec{k} \in \mathbb{N}_{>0}$

Looking for $\vec{\delta} \in \mathbb{Q}^n_{>0}$ $\Leftrightarrow$ looking for $\delta\vec{k}$, where $\delta \in \mathbb{Q}_{>0}$ and $\vec{k} \in \mathbb{N}^n_{>0}$.

The **shrunk automaton** is written $\mathcal{A}_{-\delta\vec{k}}$.

Shrinking Timed Automata

We have

$$\text{program}(\mathcal{A}_{-\delta\vec{k}+\Delta}) \quad \sqsubseteq \quad \mathcal{A}.$$

for appropriate $0 < 2\Delta < \min \delta\vec{k}$.

▶ The behaviour of the **real-world system** $\text{program}(\mathcal{A}_{-\delta\vec{k}})$ is included in that of the **abstract model** $\mathcal{A}$.

Shrinking Timed Automata

We have

$$\text{program}(\mathcal{A}_{-\delta\vec{k}+\Delta}) \quad \sqsubseteq \quad \mathcal{A}.$$

for appropriate $0 < 2\Delta < \min \delta\vec{k}$.

Problem: Shrinkability

Find $\delta\vec{k}$ such that **program**$(\mathcal{A}_{-\delta\vec{k}+\Delta})$ satisfies:

- $\mathcal{A} \sqsubseteq_{\text{t.a.}} \text{program}(\mathcal{A}_{-\delta\vec{k}+\Delta})$,
- and it is non-blocking.

Shrinking Timed Automata

We have

$$\mathcal{A}_{-\delta\vec{k}} \quad \sqsubseteq \quad \text{program}(\mathcal{A}_{-\delta\vec{k}+\Delta}) \quad \sqsubseteq \quad \mathcal{A}.$$

for appropriate $0 < 2\Delta < \min \delta\vec{k}$.

Problem: Shrinkability

Find $\delta\vec{k}$ such that **program**$(\mathcal{A}_{-\delta\vec{k}+\Delta})$ satisfies:

- $\mathcal{A} \sqsubseteq_{\text{t.a.}} \text{program}(\mathcal{A}_{-\delta\vec{k}+\Delta})$,
- and it is non-blocking.

Shrinking Timed Automata

We have

$$\mathcal{A} \quad \sqsubseteq? \quad \mathcal{A}_{-\delta\vec{k}} \quad \sqsubseteq \quad \text{program}(\mathcal{A}_{-\delta\vec{k}+\Delta}) \quad \sqsubseteq \quad \mathcal{A}.$$

for appropriate $0 < 2\Delta < \min \delta\vec{k}$.

Main Result: Shrinkability

One can decide the existence of $\delta\vec{k}$, and compute the "least" solution, for which,

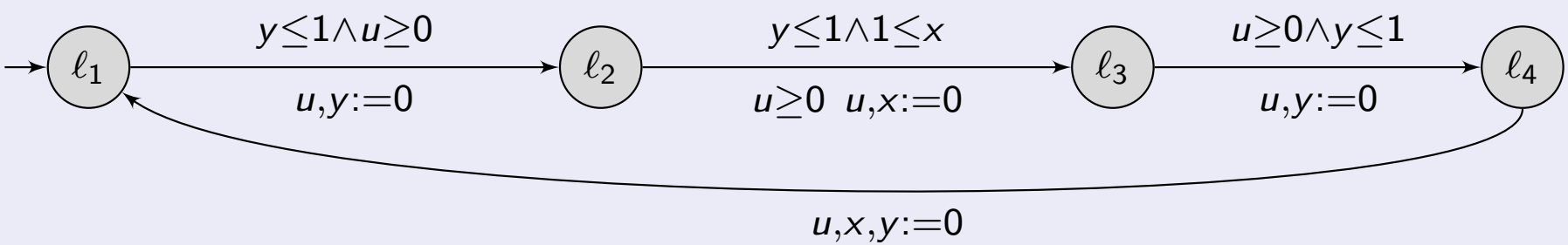
- $\mathcal{A} \quad \sqsubseteq_{\text{t.a.}} \quad \mathcal{A}_{-\delta\vec{k}}$, in **EXPTIME**,
- $\mathcal{A}_{-\delta\vec{k}}$ is non-blocking. in **PSPACE**, and **NP** for bounded-branching.

and both at the same time in **EXPTIME**.

$\Rightarrow$ **program**$(\mathcal{A}_{-\delta\vec{k}+\Delta})$ is non-blocking.

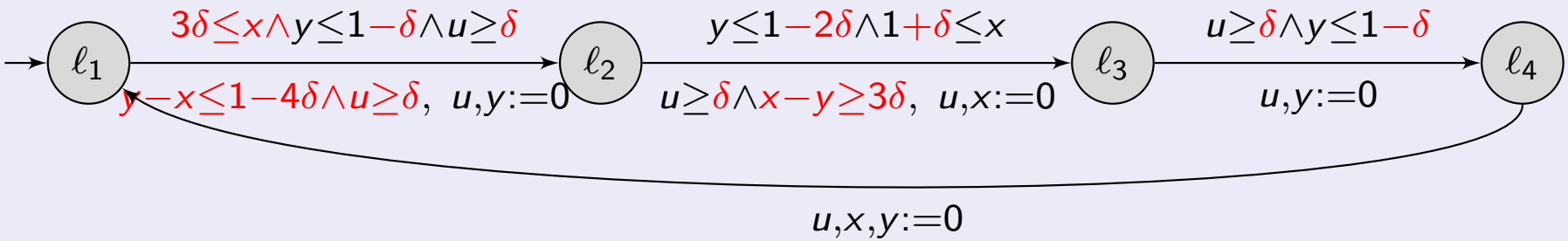
Example of Shrinking

A **shrinkable** automaton



Example of Shrinking

A **shrunk** automaton

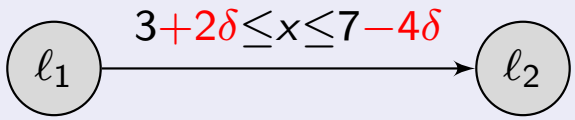


$$\mathcal{A} \quad \sqsubseteq_{\text{t.a.}} \quad \mathcal{A}_{-\delta\vec{k}} \quad \sqsubseteq \quad \mathcal{A}.$$

and non-blocking, for **all** $\delta \in [0, \frac{1}{4}]$

Interpretation of Shrinking

Developer's guide to shrinking

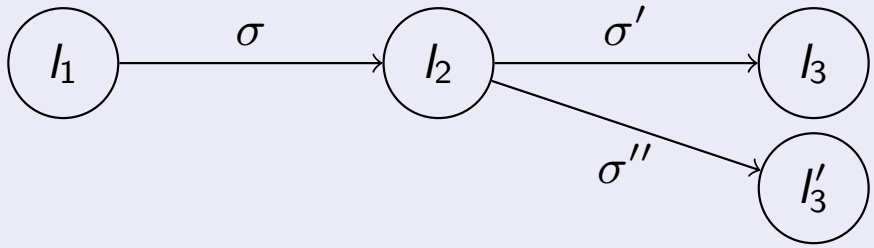


- If the edge is **controllable** by the system, do the action 2δ later than allowed, and 4δ before the deadline.

- If the edge is **uncontrollable** (e.g. execution of task), the guard corresponds to BCET $\leq x \leq$ WCET:
adjust your timing analysis to ensure $3+2\delta \leq x \leq 7-4\delta$.

Non-blocking Timed Automata

Definition: Non-blockingness



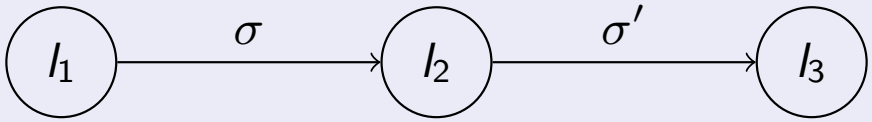
Whenever σ is taken, either σ' or σ'' are eventually firable.

Fix-point characterization

Let G_σ denote the **guards** of the timed automaton. It is non-blocking iff,

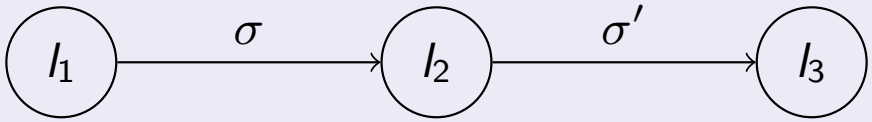
$$\forall \sigma, \quad [\![G_\sigma]\!] \subseteq \bigcup_{l_1 \xrightarrow{\sigma} l_2 \xrightarrow{\sigma'} l_3} \mathrm{Unreset}_{R_\sigma}(\mathrm{Pre}_{\mathrm{time}}([\![G_{\sigma'}]\!])).$$

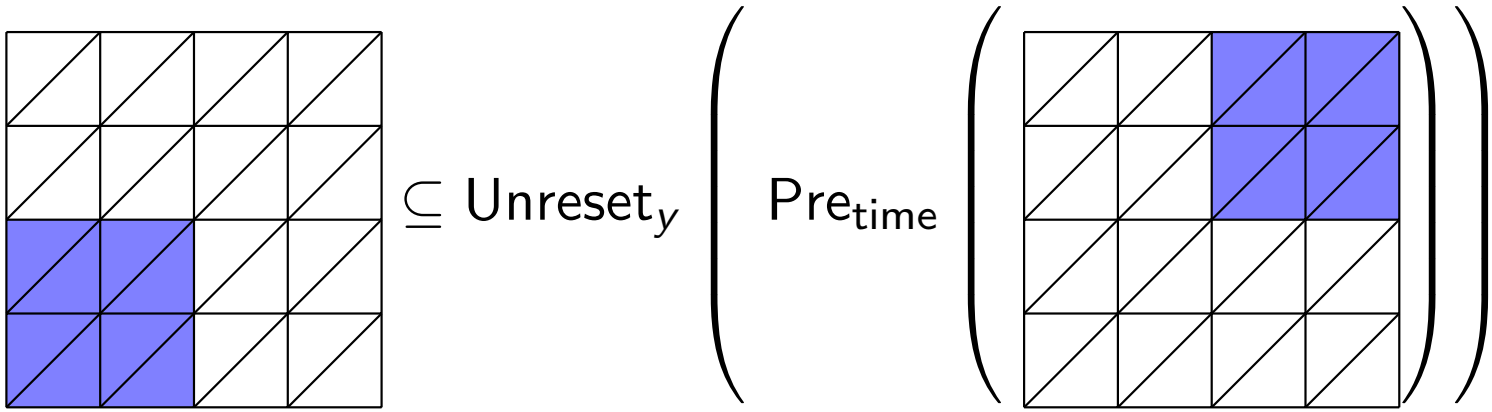
Technique for Computing Shrinking Parameters



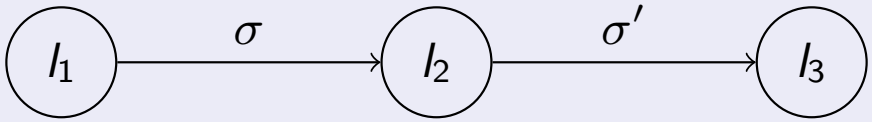
$$[\![G_\sigma]\!] \subseteq \mathsf{Unreset}_{R_\sigma}(\mathsf{Pre}_{\mathsf{time}}([\![G_{\sigma'}]\!])).$$

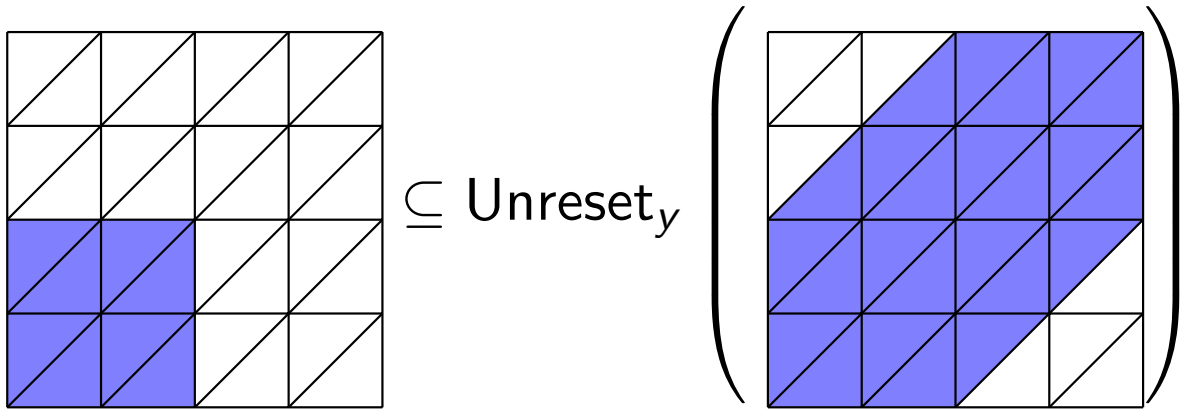
Technique for Computing Shrinking Parameters



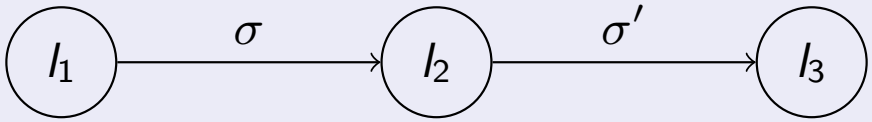


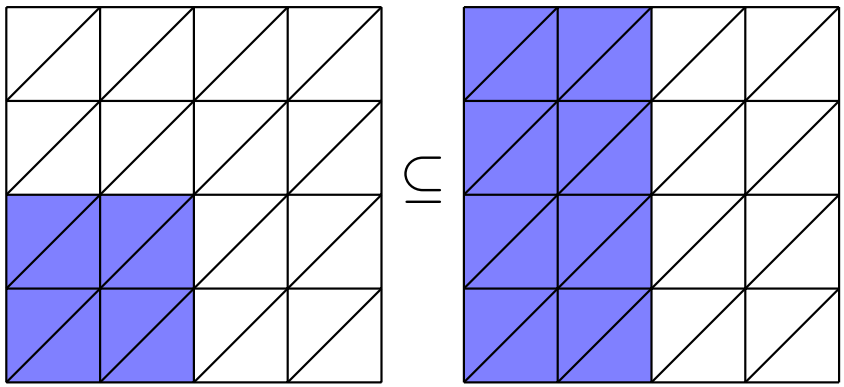
Technique for Computing Shrinking Parameters



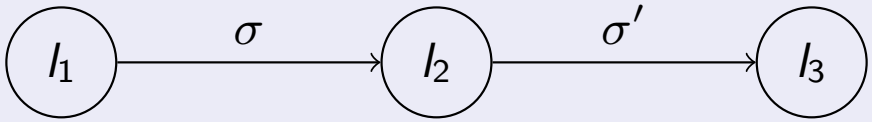


Technique for Computing Shrinking Parameters





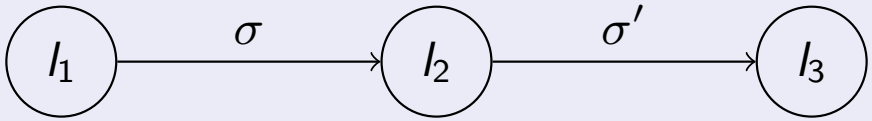
Technique for Computing Shrinking Parameters

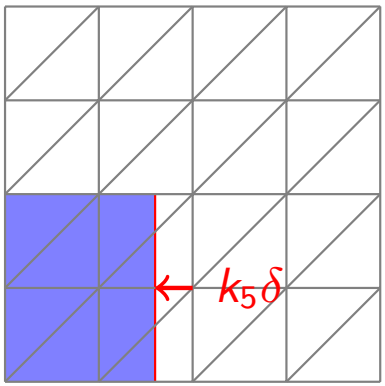


$$[\![\langle G_\sigma \rangle_{-\vec{k}\delta}]\!] \subseteq \mathsf{Unreset}_{R_\sigma}(\mathsf{Pre}_{\mathsf{time}}([\![\langle G_{\sigma'} \rangle_{-\vec{k}\delta}]\!])) \quad ?$$

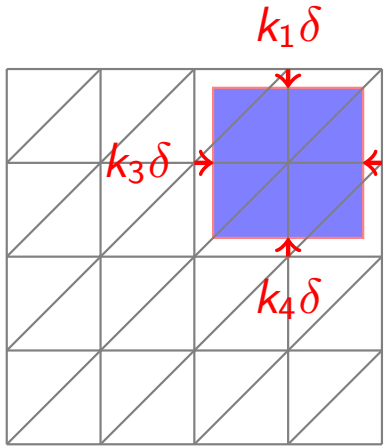
Determine $\vec{k}$

Technique for Computing Shrinking Parameters



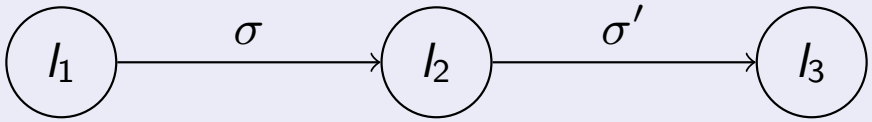


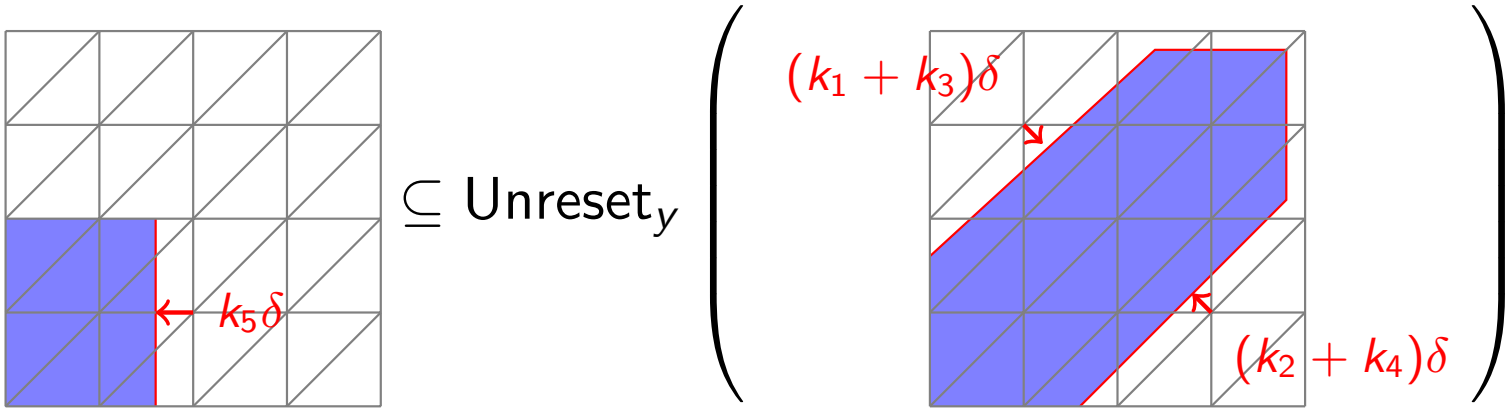
$\subseteq \mathsf{Unreset}_y \left(\mathsf{Pre}_{\mathsf{time}} \left(\right. \right.$



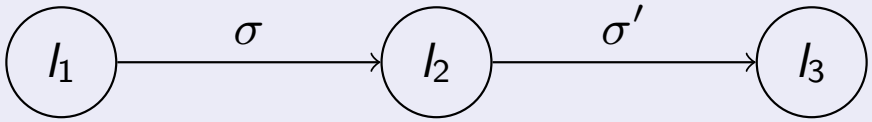
$\left. \right) \right)$

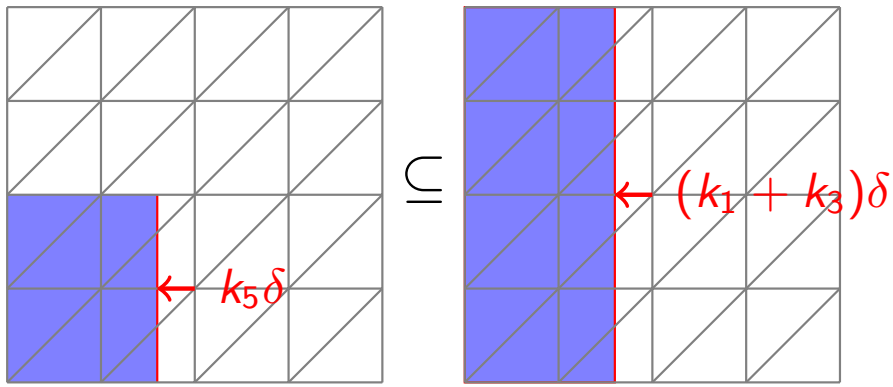
Technique for Computing Shrinking Parameters



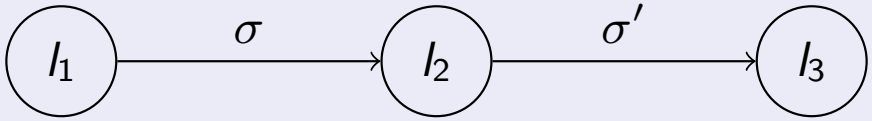


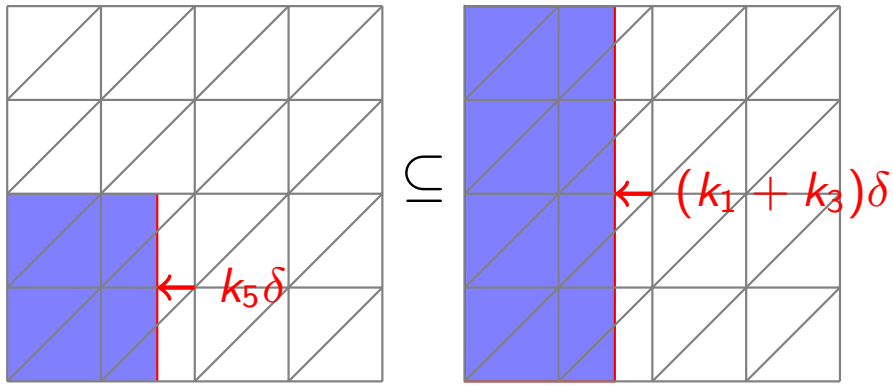
Technique for Computing Shrinking Parameters





Technique for Computing Shrinking Parameters





Then, $\vec{k}$ should satisfy

$$k_5 = \max(k_5, k_1 + k_3).$$

$$[\![\langle G_\sigma \rangle_{-\vec{k}\delta}]\!] \subseteq \mathsf{Unreset}_{R_\sigma}(\mathsf{Pre}_{\mathsf{time}}([\![\langle G_{\sigma'} \rangle_{-\vec{k}\delta}]\!]))$$

$$\Leftrightarrow$$

$$k_5 = \max(k_5, k_1 + k_3).$$

$$[\![\langle G_\sigma \rangle_{-\vec{k}\delta}]\!] \subseteq \mathsf{Unreset}_{R_\sigma}(\mathsf{Pre}_{\mathsf{time}}([\![\langle G_{\sigma'} \rangle_{-\vec{k}\delta}]\!]))$$
$$\Leftrightarrow$$
$$k_5 = \max(k_5, k_1 + k_3).$$

Key Theorem

Let $\vec{M} = f(\vec{M})$ be a **fixpoint equation on zones**, and $\vec{M}$ a solution.

f uses $\mathsf{Pre}_{\mathsf{time}}()$, $\cap$, $\mathsf{Unreset}_{\cdot}()$.

For any $\vec{k} \in \mathbb{N}^n_{>0}$,

$$\langle \vec{M} \rangle_{-\vec{k}\delta} = f(\langle \vec{M} \rangle_{-\vec{k}\delta}) \qquad \forall \text{ small } \delta > 0$$
$$\Leftrightarrow$$
$$\vec{k} = \phi(\vec{k}),$$

where ϕ is a **max-plus expression**.

$$[\![\langle G_\sigma \rangle_{-\vec{k}\delta}]\!] \subseteq \mathsf{Unreset}_{R_\sigma}(\mathsf{Pre}_{\mathsf{time}}([\![\langle G_{\sigma'} \rangle_{-\vec{k}\delta}]\!]))$$
$$\Leftrightarrow$$
$$k_5 = \max(k_5, k_1 + k_3).$$

Key Theorem

Let $\vec{M} = f(\vec{M})$ be a **fixpoint equation on zones**, and $\vec{M}$ a solution.

f uses $\mathsf{Pre}_{\mathsf{time}}()$, $\cap$, $\mathsf{Unreset}_{\cdot}()$.

For any $\vec{k} \in \mathbb{N}^n_{>0}$,

$$\langle \vec{M} \rangle_{-\vec{k}\delta} = f(\langle \vec{M} \rangle_{-\vec{k}\delta}) \qquad \forall \text{ small } \delta > 0$$
$$\Leftrightarrow$$
$$\vec{k} = \phi(\vec{k}),$$

where ϕ is a **max-plus expression**.

▶ **Max-plus algebra:** We prove that such fixpoint equations can be solved in poly-time.

Conclusion

Summary

- Shrinking always ensures Imp $\sqsubseteq$ Spec.
- Shrinking parameters can be **synthesized automatically** so that Spec $\sqsubseteq$ Imp, and Imp non-blocking.
- Complexity: NP, PSPACE, EXPTIME.
- These properties preserved in the **program semantics**.

$\rightarrow$ Can be used to define the implementation or in the timing analysis.

Technics

- Zones + parameters $\leftrightarrow$ max-plus algebra.
- New data structure: **DBMs w/ parameterized max-plus exp.**.
  Application to other problems: e.g. robust controller synthesis.