Timed Models for Concurrent Systems

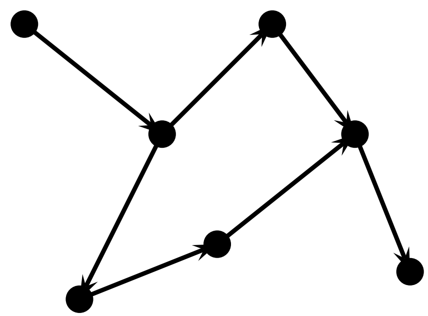
Patricia Bouyer

LSV – CNRS & ENS de Cachan

Model-Checking

Does the system satisfy the property?

Modelization

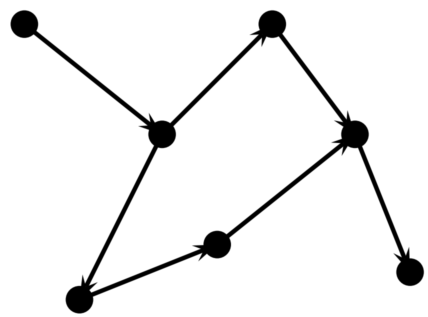


φ

Model-Checking

Does the system satisfy the property?

Modelization



$\models$ φ

Model-checking algorithm

Time!

Context: verification of embedded critical systems

Time

- ✔ naturally appears in real systems
- ✔ appears in properties (for ex. bounded response time)

➜ Need of models and specification languages integrating timing aspects

➜ Challenge: Integrate time in concurrent models

Roadmap

- ✔ About time semantics
- ✔ Timed specification languages
- ✔ Some possible timed models
- ✔ Timed automata
- ✔ Networks of TA, discussion
- ✔ Verification methods
- ✔ Conclusion remarks

About Time Semantics

[Alur's PhD Thesis 1991]

Adding Timing Informations

Which semantics?

- ✔ **Untimed case:** sequence of observable events

  a: send message b: receive message

  $$a\ b\ a\ b\ a\ b\ a\ b\ a\ b\ \cdots = (a\ b)^\omega$$

- ✔ **Timed case:** sequence of dated observable events

  $$(a, d_1)\ (b, d_2)\ (a, d_3)\ (b, d_4)\ (a, d_5)\ (b, d_6)\ \cdots$$

  d_1: date at which the first a occurs
  d_2: date at which the first b occurs
  ...

Process: set of such (un)timed sequences

Three Propositions

- ✔ **Discrete-time semantics:**

  dates are taken in $\mathbb{N}$, the set of integers

  Ex: (a, 1).(b, 3).(c, 4).(a, 6)

- ✔ **Dense-time semantics:**

  dates are taken in $\mathbb{Q}^+$, the set of positive rationals,
  or in $\mathbb{R}^+$, the set of positive reals

  Ex: (a, 1.28).(b, 3.1).(c, 3.98)(a, 6.13)

- ✔ **Fictitious-clock semantics:**

  "tick" action denoting each unit of time

  Ex: tick.a.tick.tick.b.c.tick.tick.tick.a
  or alternatively (a, 1).(b, 3).(c, 3).(a, 6)

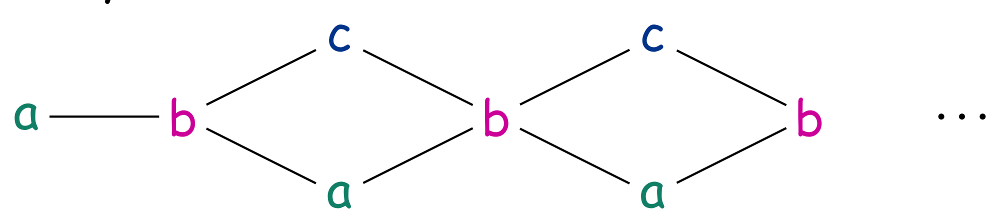
Synchronization of Processes

- ✔ **Untimed case:** Synchronization on common events, interleaving of causally independent events

Example: P = ({a, b}, (a b)$^\omega$) || Q = ({b, c}, (b c)$^\omega$)

a b c a b {a, c} b a c b

Can be represented by:



- ✔ **Timed case:** No interleaving possible; time orders events

> **Hyp:** All components are driven by a common clock

The Discrete-Time Semantics

- ✔ the simplest one
- ✔ equivalent to the untimed semantics (if no action, say action $\emptyset$)

Ex: the timed sequence

$$(a,1)\,.\,(b,2)\,.\,(\{a,b\},4)\,.\,(b,5)\ \ldots$$

is represented by the untimed sequence

$$\{a\}\,.\,\{b\}\,.\,\emptyset\,.\,\{a,b\}\,.\,\{b\}\ \ldots$$

➜ no really new technique needed

The Dense-Time Semantics

- ✔ **a more realistic model:** causally independent events may appear arbitrarly close to each other

  Ex: (a, 1) . (b, 2) . (c, 3.93) . (a, 3.98) . (b, 5) . (c, 6.02)

- ✔ a system and its **environment**: no constraint on the timing of signals from the environment

- ✔ if strange behaviours are not wished (e.g. zeno behaviours), one can simply avoid them

➜ new techniques needed

The Fictitous-Clock Model

Much different from the two previous models, more uncertainty.

The Fictitous-Clock Model

Much different from the two previous models, more uncertainty.

- ✔ **Ex:** the sequence tick . a . tick . tick . b . c . tick . tick . tick . a represents a timed sequence of events

$$(a, d_1) . (b, d_2) . (c, d_3) . (a, d_4)$$

where $1 \leq d_1 < 2$, $3 \leq d_2 \leq d_3 < 4$ and $6 \leq d_4 < 7$.

The Fictitous-Clock Model

Much different from the two previous models, more uncertainty.

- **Ex:** the sequence tick . a . tick . tick . b . c . tick . tick . tick . a represents a timed sequence of events

$$(a, d_1) . (b, d_2) . (c, d_3) . (a, d_4)$$

where $1 \le d_1 < 2$, $3 \le d_2 \le d_3 < 4$ and $6 \le d_4 < 7$.

- **Parallel composition:** almost as in the untimed case, but synchronization of all "tick" actions (it is thus more constrained)

➜ $\sim$ untimed case, use same techniques

The Fictitous-Clock Model

Much different from the two previous models, more uncertainty.

- ✔ **Ex:** the sequence tick . a . tick . tick . b . c . tick . tick . tick . a represents a timed sequence of events

$$(a, d_1) . (b, d_2) . (c, d_3) . (a, d_4)$$

where $1 \leq d_1 < 2$, $3 \leq d_2 \leq d_3 < 4$ and $6 \leq d_4 < 7$.

- ✔ **Parallel composition:** almost as in the untimed case, but synchronization of all "tick" actions (it is thus more constrained)

➜ $\sim$ untimed case, use same techniques

- ✔ Can also be viewed as an approximation of the dense-time semantics

The Fictitous-Clock Model

Much different from the two previous models, more uncertainty.

- **Ex:** the sequence tick . a . tick . tick . b . c . tick . tick . tick . a represents a timed sequence of events

$$(a, d_1) . (b, d_2) . (c, d_3) . (a, d_4)$$

where $1 \leq d_1 < 2$, $3 \leq d_2 \leq d_3 < 4$ and $6 \leq d_4 < 7$.

- **Parallel composition:** almost as in the untimed case, but synchronization of all "tick" actions (it is thus more constrained)

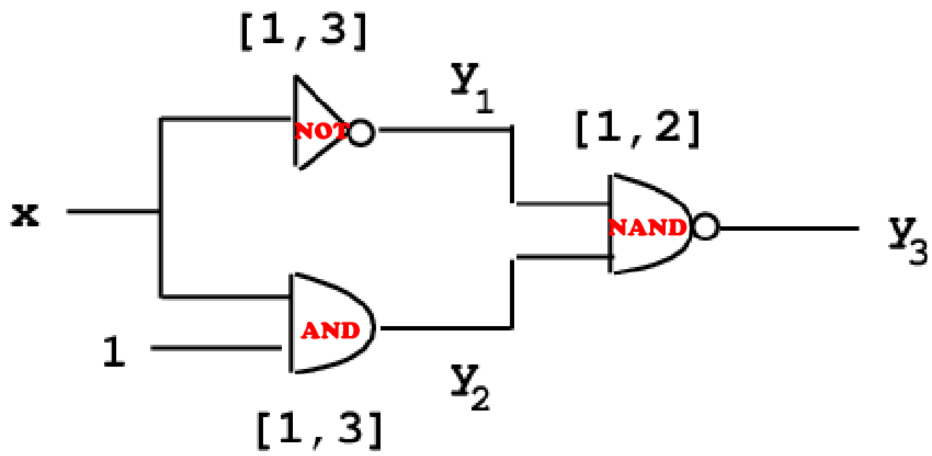
➜ $\sim$ untimed case, use same techniques

- Can also be viewed as an approximation of the dense-time semantics
- **Pb:** no precise timing informations (if k ticks in between two actions, it means that these two actions are separated by some delay in $[k-1, k+1[$)

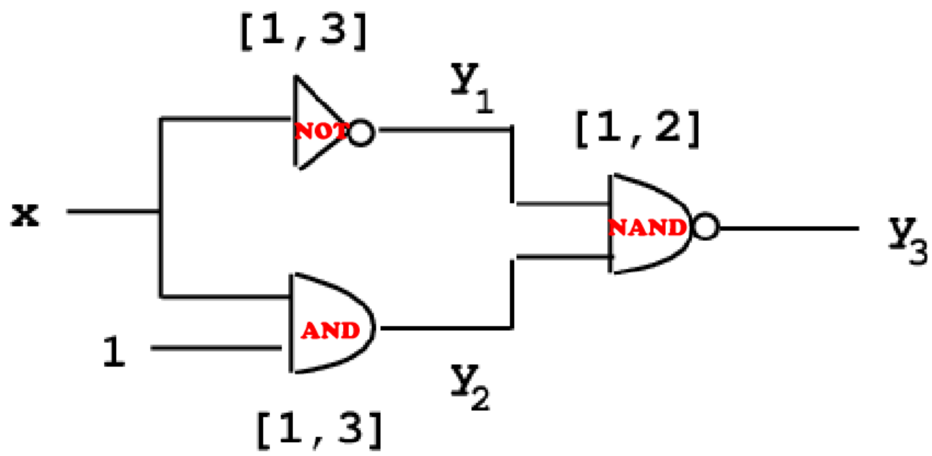
A Case for Dense-Time [Alur 1991]

- ✔ **Correctness:** discussion in the context of reachability problems for asynchronous digital circuits **[Brzozowski, Seger 1991]**

A Digital Circuit [BS91]



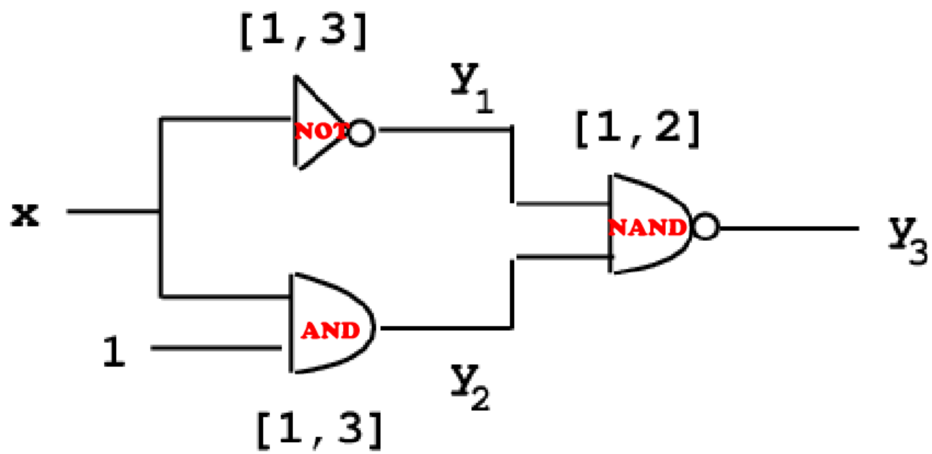
A Digital Circuit [BS91]



Start with x=0 and y=[101] (stable configuration)

A Digital Circuit

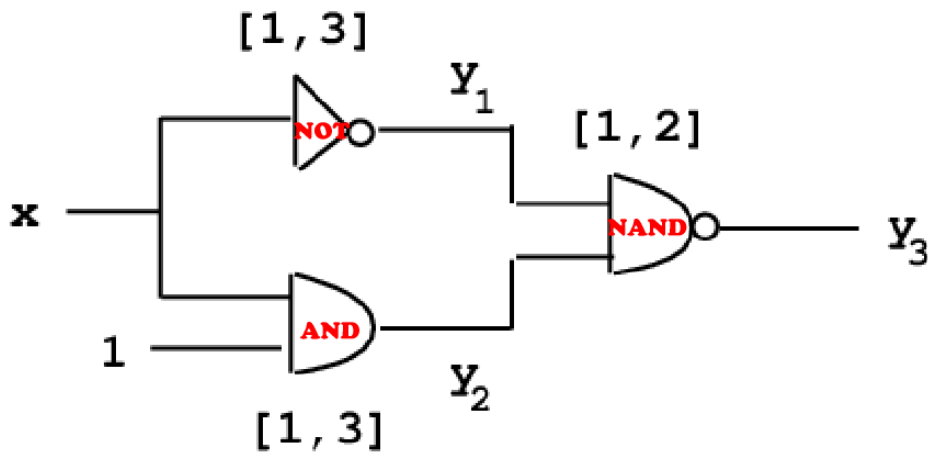
[BS91]



Start with x=0 and y=[101] (stable configuration)

The input x changes to 1. The corresponding stable state is y=[011]

A Digital Circuit [BS91]



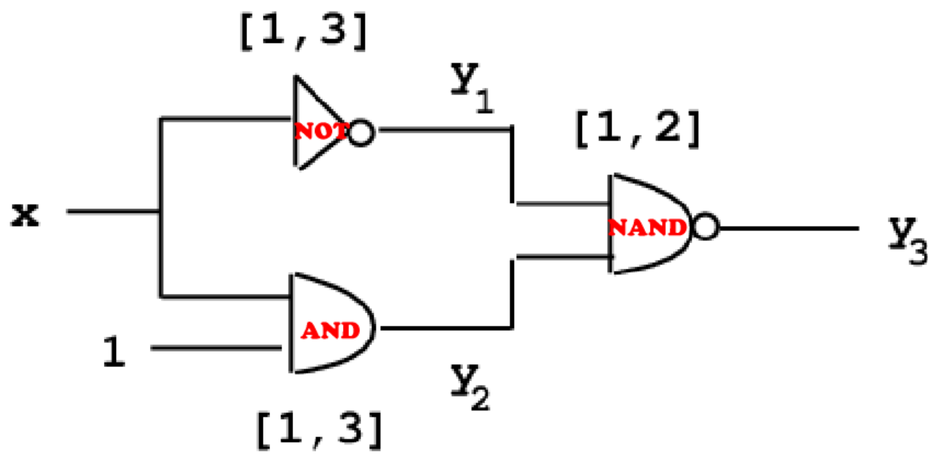
Start with x=0 and y=[101] (stable configuration)

The input x changes to 1. The corresponding stable state is y=[011]

However, many possible behaviours, e.g.

$$[101] \xrightarrow[1.2]{y_2} [111] \xrightarrow[2.5]{y_3} [110] \xrightarrow[2.8]{y_1} [010] \xrightarrow[4.5]{y_3} [011]$$

A Digital Circuit [BS91]



Start with x=0 and y=[101] (stable configuration)

The input x changes to 1. The corresponding stable state is y=[011]

However, many possible behaviours, e.g.

$$[101] \xrightarrow[1.2]{y_2} [111] \xrightarrow[2.5]{y_3} [110] \xrightarrow[2.8]{y_1} [010] \xrightarrow[4.5]{y_3} [011]$$

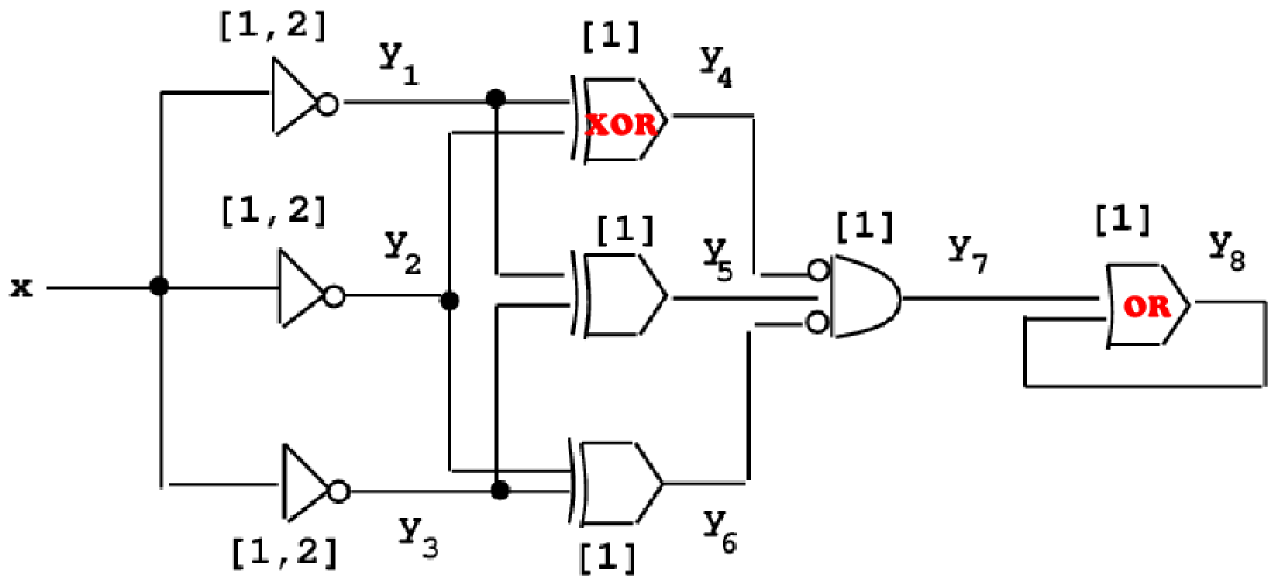
Reachable configurations: {[101], [111], [110], [010], [011], [001]}

Discretizing is Not Sufficient

[Brzozowski Seger 1991]

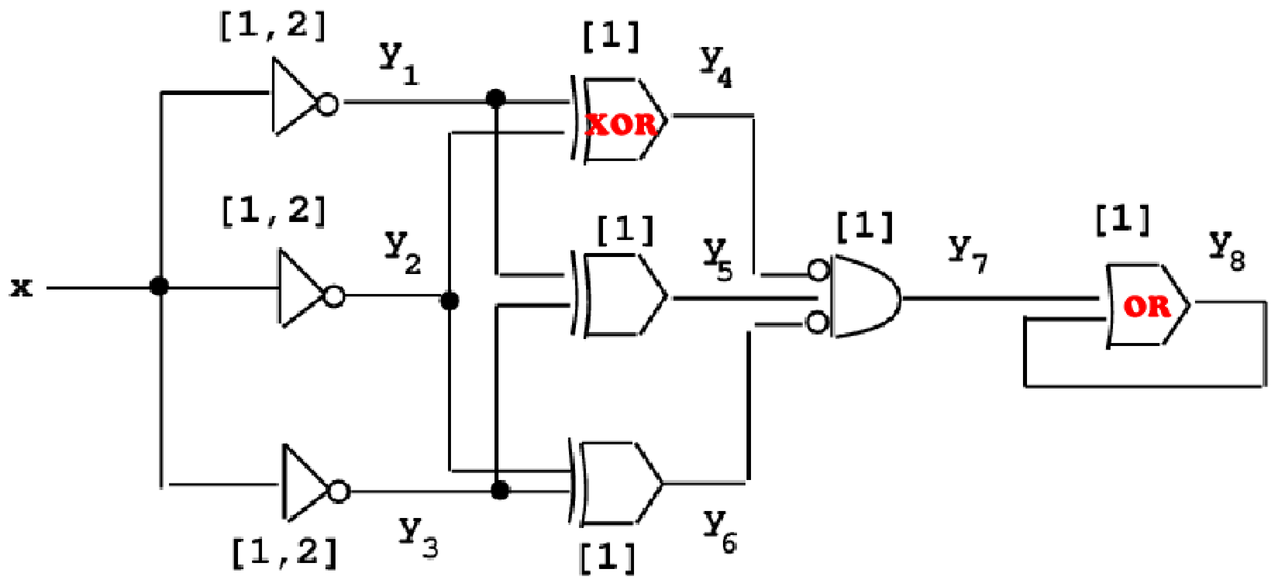
Theorem: for every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

Discretizing is Not Sufficient – Example



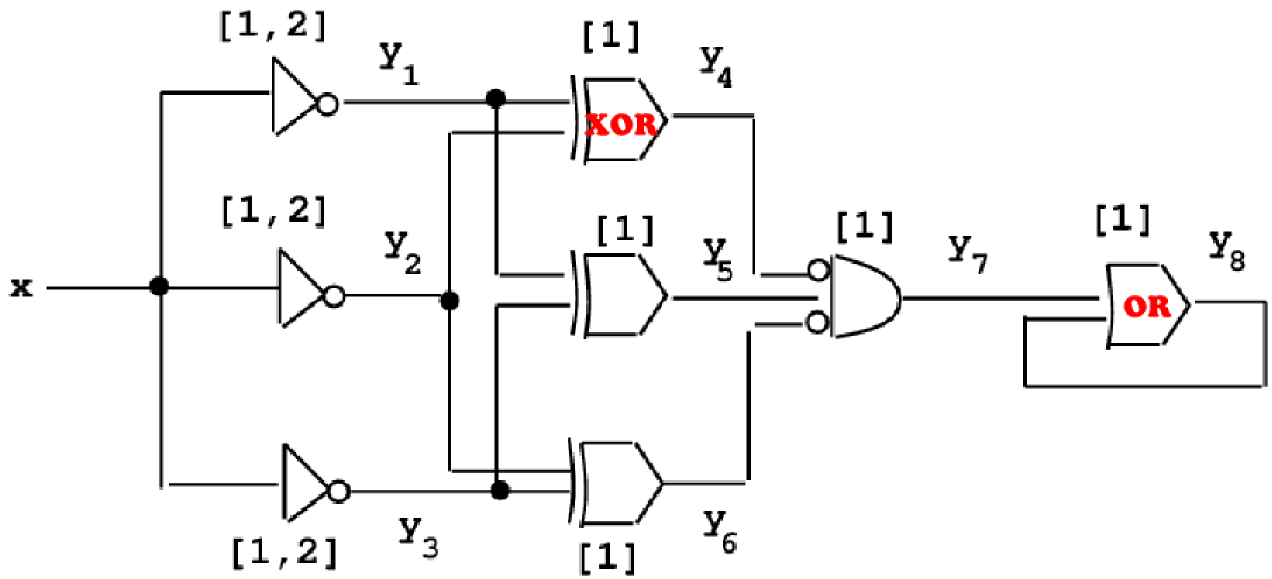
✔ This digital circuit **is not** 1-discretizable.

Discretizing is Not Sufficient – Example



- ✔ This digital circuit **is not** 1-discretizable.
- ✔ Why that? (initially x = 0 and y = [11100000], x is set to 1)

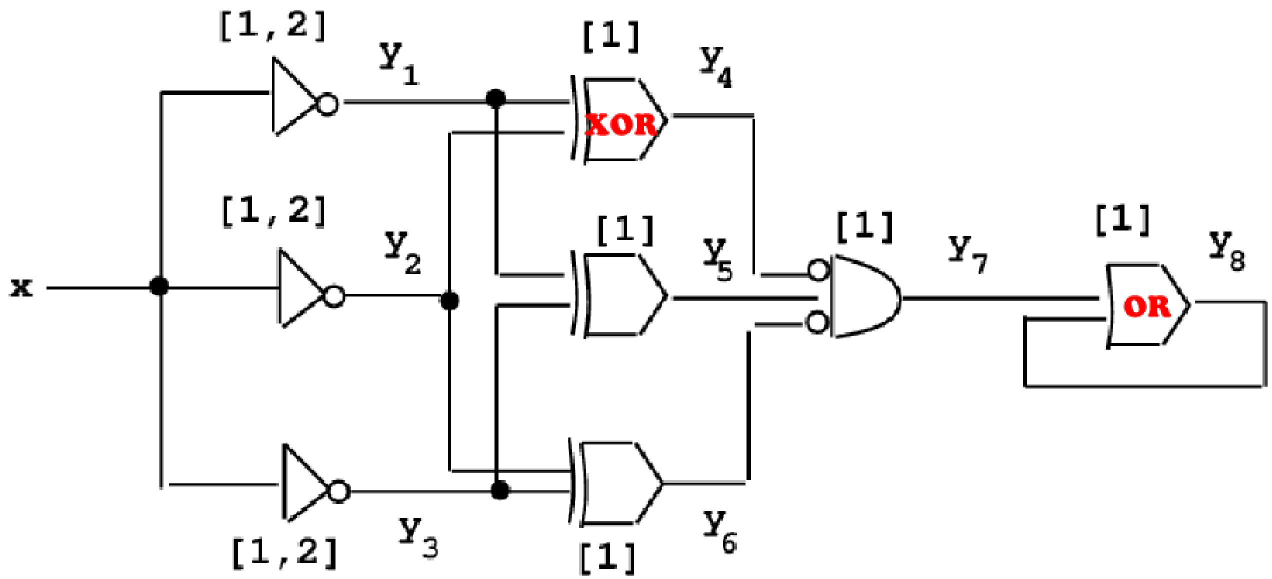
Discretizing is Not Sufficient – Example



- ✔ This digital circuit **is not** 1-discretizable.
- ✔ Why that? (initially x = 0 and y = [11100000], x is set to 1)

- $[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$

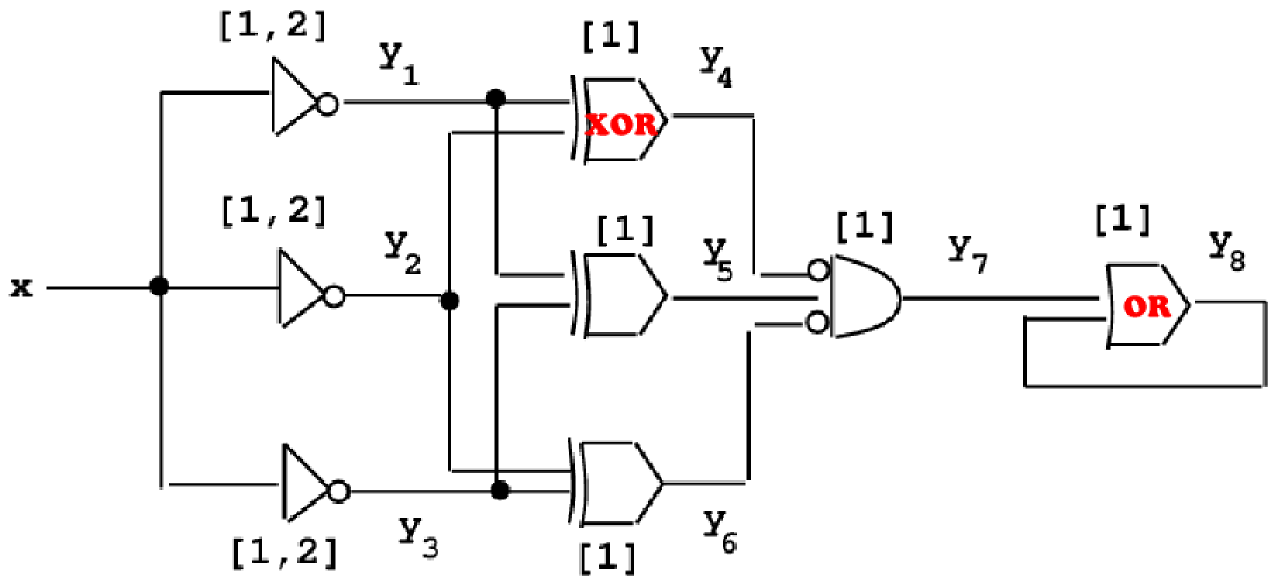
Discretizing is Not Sufficient – Example



- ✔ This digital circuit **is not** 1-discretizable.
- ✔ Why that? (initially x = 0 and y = [11100000], x is set to 1)

- $[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$
- $[11100000] \xrightarrow[1]{y_1, y_2, y_3} [00000000]$

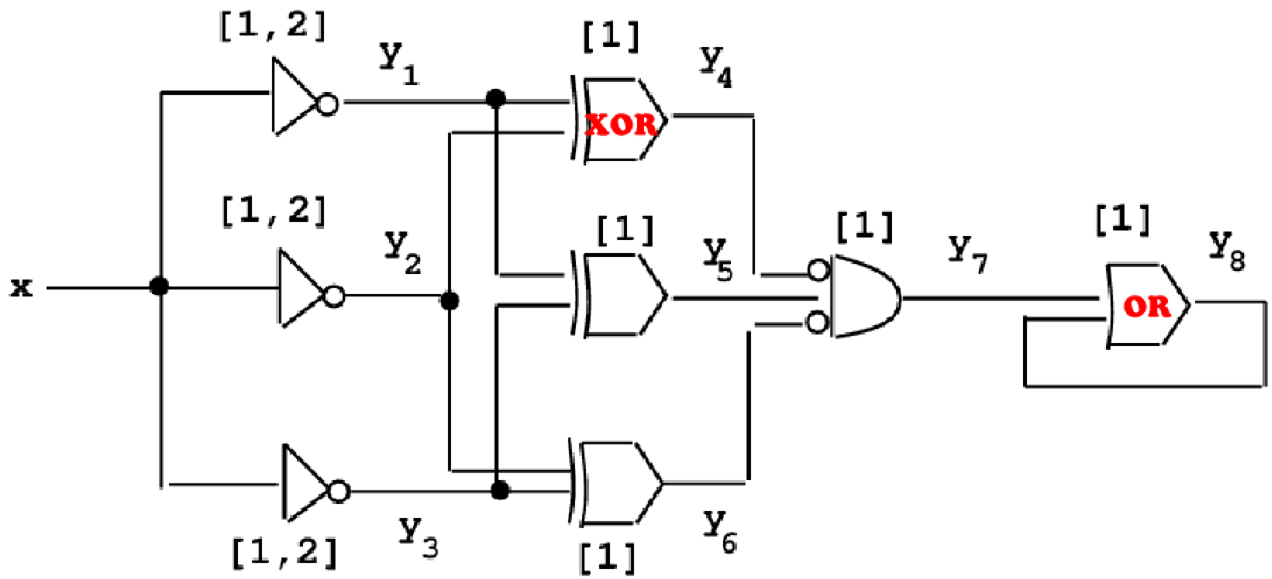
Discretizing is Not Sufficient – Example



- ✔ This digital circuit **is not** 1-discretizable.
- ✔ Why that? (initially x = 0 and y = [11100000], x is set to 1)

- $[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$
- $[11100000] \xrightarrow[1]{y_1, y_2, y_3} [00000000]$
- $[11100000] \xrightarrow[1]{y_1} [01111000] \xrightarrow[2]{y_2, y_3, y_4, y_5} [00000000]$

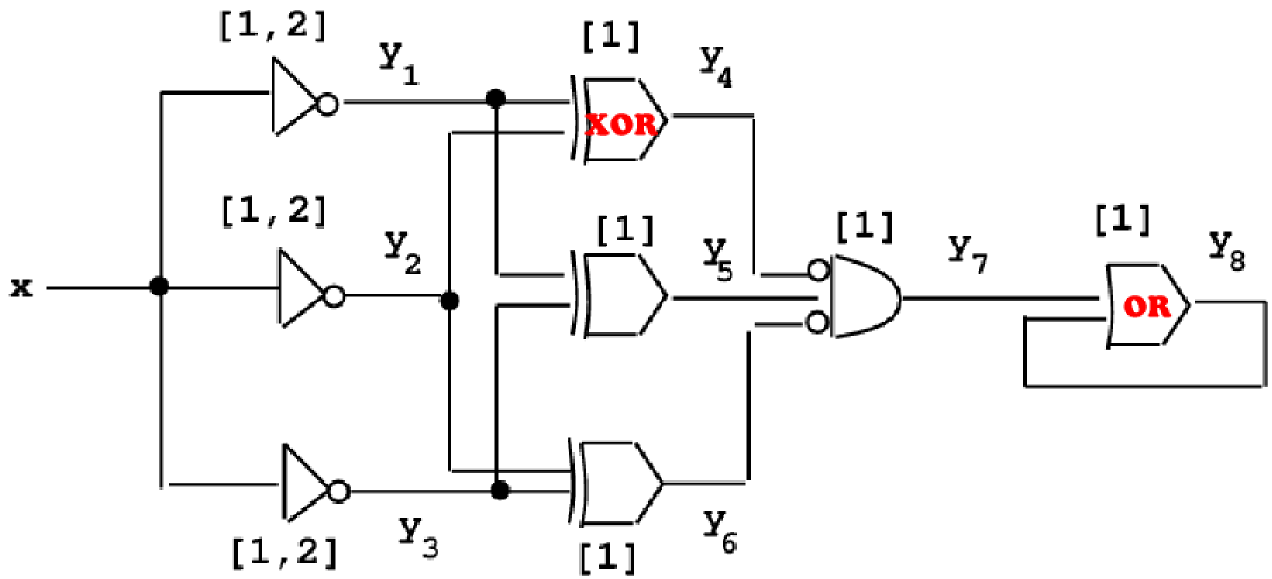
Discretizing is Not Sufficient – Example



- ✔ This digital circuit **is not** 1-discretizable.
- ✔ Why that? (initially $x = 0$ and $y = [11100000]$, x is set to 1)

- $[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} [00000001]$
- $[11100000] \xrightarrow[1]{y_1, y_2, y_3} [00000000]$
- $[11100000] \xrightarrow[1]{y_1} [01111000] \xrightarrow[2]{y_2, y_3, y_4, y_5} [00000000]$
- $[11100000] \xrightarrow[1]{y_1, y_2} [00100000] \xrightarrow[2]{y_3, y_5, y_6} [00001100] \xrightarrow[3]{y_5, y_6} [00000000]$

Discretizing is Not Sufficient - Example



- ✔ This digital circuit **is not** 1-discretizable.
- ✔ Why that? (initially x = 0 and y = [11100000], x is set to 1)

- $[11100000] \xrightarrow[1]{y_1} [01100000] \xrightarrow[1.5]{y_2} [00100000] \xrightarrow[2]{y_3, y_5} [00001000] \xrightarrow[3]{y_5, y_7} [00000010] \xrightarrow[4]{y_7, y_8} \boxed{[00000001]}$
- $[11100000] \xrightarrow[1]{y_1, y_2, y_3} \boxed{[00000000]}$
- $[11100000] \xrightarrow[1]{y_1} [01111000] \xrightarrow[2]{y_2, y_3, y_4, y_5} \boxed{[00000000]}$
- $[11100000] \xrightarrow[1]{y_1, y_2} [00100000] \xrightarrow[2]{y_3, y_5, y_6} [00001100] \xrightarrow[3]{y_5, y_6} \boxed{[00000000]}$

Discretizing is Not Sufficient

[Brzozowski Seger 1991]

Theorem: for every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

Discretizing is Not Sufficient

[Brzozowski Seger 1991]

Theorem: for every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

Claim: finding a correct granularity is as difficult as computing the set of reachable states in dense-time

Discretizing is Not Sufficient

[Brzozowski Seger 1991]

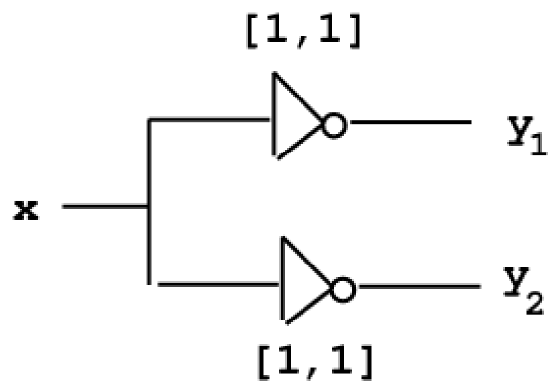
Theorem: for every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$).

Claim: finding a correct granularity is as difficult as computing the set of reachable states in dense-time

Further counter-example: there exist systems for which no granularity exists

(see later)

Fictitious-Clock Model: Too Large



- ✔ Dense-time: {[11], [00]}
- ✔ Fictitious-clock: {[11], [10], [01], [00]}

$$(tick.y_1) \parallel (tick.y_2) \quad = \quad \{tick.y_1.y_2, tick.y_2.y_1\}$$

➜ over-approximation of the set of reachable states

A Case for Dense-Time

- ✔ **Correctness**
- ✔ **Expressiveness:** discrete-time and fictitious-clock models can be expressed by dense-time models

A Case for Dense-Time

- ✔ **Correctness**
- ✔ **Expressiveness**
- ✔ **Compositionality:** the semantics of one component depends on the granularity of the whole system and of the property we want to check

Ex: P: process such that a and b strictly alternate and each b is exactly one unit of time later than a
Q process such that a and b strictly alternate, each b is exactly one unit of time later than a, and each a is at least one unit of time later than each b

- If the granularity is 1,

A Case for Dense-Time

- ✔ **Correctness**
- ✔ **Expressiveness**
- ✔ **Compositionality:** the semantics of one component depends on the granularity of the whole system and of the property we want to check

Ex: P: process such that a and b strictly alternate and each b is exactly one unit of time later than a
Q process such that a and b strictly alternate, each b is exactly one unit of time later than a, and each a is at least one unit of time later than each b

- If the granularity is 1, T(P)=T(Q)
- If the granularity is $\frac{1}{2}$,

A Case for Dense-Time

- ✔ **Correctness**
- ✔ **Expressiveness**
- ✔ **Compositionality:** the semantics of one component depends on the granularity of the whole system and of the property we want to check

  Ex: P: process such that a and b strictly alternate and each b is exactly one unit of time later than a
  Q process such that a and b strictly alternate, each b is exactly one unit of time later than a, and each a is at least one unit of time later than each b

  - If the granularity is 1, T(P)=T(Q)
  - If the granularity is $\frac{1}{2}$, T(P) ≠ T(Q)

A Case for Dense-Time

- ✔ **Correctness**
- ✔ **Expressiveness**
- ✔ **Compositionality:** the semantics of one component depends on the granularity of the whole system and of the property we want to check

  Ex: P: process such that a and b strictly alternate and each b is exactly one unit of time later than a
  Q process such that a and b strictly alternate, each b is exactly one unit of time later than a, and each a is at least one unit of time later than each b

  - If the granularity is 1, T(P)=T(Q)
  - If the granularity is $\frac{1}{2}$, T(P) $\neq$ T(Q)

➜ Dense-time: a good alternative to have a compositional semantics.

A Case for Dense-Time

- ✔ **Correctness**
- ✔ **Expressiveness**
- ✔ **Compositionality**
- ✔ **Complexity:** dense-time more complex than the two other semantics (ex: inclusion)

However: refining the granularity increases the complexity...

A Case for Dense-Time

- ✔ **Correctness**
- ✔ **Expressiveness**
- ✔ **Compositionality**
- ✔ **Complexity**

In the following we choose the **dense-time** semantics

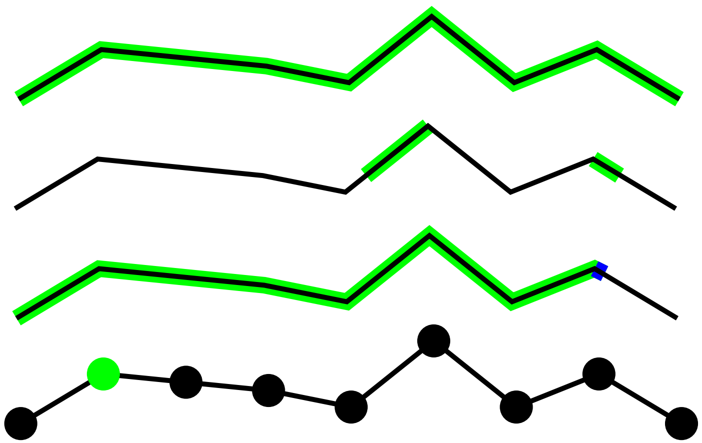
Timed Specification Languages

Classical Verification Problems

- reachability of a control state
- $\mathcal{S} \sim \mathcal{S}'$: bisimulation, etc...
- $L(\mathcal{S}) \subseteq L(\mathcal{S}')$: language inclusion
- $\mathcal{S} \models \varphi$ for some formula φ: model-checking
- $\mathcal{S} \parallel A_T$ + reachability: testing automata
- ...

Classical Temporal Logics

Path formulas:



$G\varphi$ « Always »

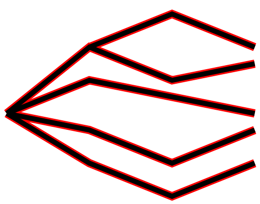
$F\varphi$ « Eventually »

$\varphi U \varphi'$ « Until »

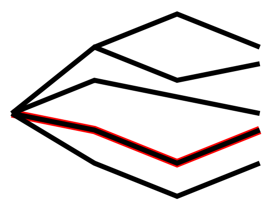
$X\varphi$ « Next »

State formulas:

$A\psi$



$E\psi$



➜ LTL: Linear Temporal Logic **[Pnueli 1977]**,
CTL: Computation Tree Logic **[Emerson, Clarke 1982]**

Adding Time to Temporal Logics

Classical temporal logics allow us to express that

"any problem is followed by an alarm"

Adding Time to Temporal Logics

Classical temporal logics allow us to express that

"any problem is followed by an alarm"

With CTL:

$$AG(\text{problem} \Rightarrow AF\ \text{alarm})$$

Adding Time to Temporal Logics

Classical temporal logics allow us to express that

"any problem is followed by an alarm"

With CTL:

$$AG(\text{problem} \Rightarrow AF\ \text{alarm})$$

How can we express:

"any problem is followed by an alarm **in at most 20 time units**"

Adding Time to Temporal Logics

Classical temporal logics allow us to express that

"any problem is followed by an alarm"

With CTL:

$$AG(\text{problem} \Rightarrow AF\ \text{alarm})$$

How can we express:

"any problem is followed by an alarm **in at most 20 time units**"

✔ Temporal logics with **subscripts**.

ex: CTL + $\left|\begin{array}{l} E\varphi U_{\sim k}\psi \\ A\varphi U_{\sim k}\psi \end{array}\right.$

$$AG(\text{problem} \Rightarrow AF_{\leq 20}\ \text{alarm})$$

Adding Time to Temporal Logics

Classical temporal logics allow us to express that

"any problem is followed by an alarm"

With CTL:

$$AG(\text{problem} \Rightarrow AF\ \text{alarm})$$

How can we express:

"any problem is followed by an alarm **in at most 20 time units**"

- ✔ Temporal logics with **subscripts**.

$$AG(\text{problem} \Rightarrow AF_{\leq 20}\ \text{alarm})$$

- ✔ Temporal logics with **clocks**.

$$AG(\text{problem} \Rightarrow (x\ \mathbf{in}\ AF(x \leq 20 \wedge \text{alarm})))$$

Adding Time to Temporal Logics

Classical temporal logics allow us to express that

"any problem is followed by an alarm"

With CTL:

$$AG(\text{problem} \Rightarrow AF\ \text{alarm})$$

How can we express:

"any problem is followed by an alarm **in at most 20 time units**"

- ✔ Temporal logics with **subscripts**.

$$AG(\text{problem} \Rightarrow AF_{\leq 20}\ \text{alarm})$$

- ✔ Temporal logics with **clocks**.

$$AG(\text{problem} \Rightarrow (x\ \mathbf{in}\ AF(x \leq 20 \wedge \text{alarm})))$$

→ **TCTL**: Timed CTL [ACD90,ACD93,HNSY94]

An Other Specification Language

Timed modal logics or timed μ-calculus with **clocks** e.g. [LLW95]

- ✔ prop, boolean combinators
- ✔ $< a > \varphi$, $[a]\,\varphi$
- ✔ $\exists\!\!\exists\ \varphi$, $\forall\!\!\forall\ \varphi$
- ✔ min(X, φ), max(X, φ)
- ✔ $x \leq cte$, x in φ

An Other Specification Language

Timed modal logics or timed μ-calculus with **clocks** e.g. [LLW95]

- prop, boolean combinators
- $\langle a \rangle \varphi$, $[a]\varphi$
- $\exists \varphi$, $\forall \varphi$
- $\min(X, \varphi)$, $\max(X, \varphi)$
- $x \leq cte$, x in φ

Examples: "$AG\,\varphi$": $\max(X, \varphi \wedge \bigwedge_a [a]X \wedge \forall \varphi)$

An Other Specification Language

Timed modal logics or timed μ-calculus with **clocks** e.g. [LLW95]

- prop, boolean combinators
- $\langle a \rangle \varphi$, $[a]\,\varphi$
- $\exists\!\!\!\exists\ \varphi$, $\forall\!\!\!\forall\ \varphi$
- $\min(X, \varphi)$, $\max(X, \varphi)$
- $x \leq cte$, x in φ

Examples: "$AG\,\varphi$": $\max(X, \varphi \wedge \bigwedge_a [a]X \wedge \forall\!\!\!\forall \varphi)$

"$A(\varphi W \psi)$": $\max(X, \psi \vee (\varphi \wedge \bigwedge_a [a]X \wedge \forall\!\!\!\forall X))$

An Other Specification Language

Timed modal logics or timed μ-calculus with **clocks** e.g. [LLW95]

- ✔ prop, boolean combinators
- ✔ $\langle a \rangle \varphi$, $[a]\varphi$
- ✔ $\exists \varphi$, $\forall \varphi$
- ✔ $\min(X,\varphi)$, $\max(X,\varphi)$
- ✔ $x \leq cte$, x in φ

Examples: "$AG\,\varphi$": $\max(X, \varphi \wedge \bigwedge_a [a]X \wedge \forall \varphi)$

"$A(\varphi W \psi)$": $\max(X, \psi \vee (\varphi \wedge \bigwedge_a [a]X \wedge \forall X))$

"non-zenoness (action and time)": x in $\max(X, x \leq 1 \wedge \forall X \wedge \bigwedge_a [a]X)$

An Other Specification Language

Timed modal logics or timed μ-calculus with **clocks** e.g. [LLW95]

- ✔ prop, boolean combinators
- ✔ $<a>\varphi$, $[a]\varphi$
- ✔ $\exists\!\!\!\!\!\exists\, \varphi$, $\forall\!\!\!\!\forall\, \varphi$
- ✔ $min(X, \varphi)$, $max(X, \varphi)$
- ✔ $x \leq cte$, x in φ

Examples: "$AG\,\varphi$": $max(X, \varphi \wedge \bigwedge_a [a]X \wedge \forall\!\!\!\!\forall \varphi)$

"$A(\varphi W \psi)$": $max(X, \psi \vee (\varphi \wedge \bigwedge_a [a]X \wedge \forall\!\!\!\!\forall X))$

"non-zenoness (action and time)": x in $max(X, x \leq 1 \wedge \forall\!\!\!\!\forall X \wedge \bigwedge_a [a]X)$

$$\Phi = \text{problem} \Rightarrow x \text{ in } max(Z, alarme \vee (x \leq 20 \wedge \bigwedge_a [a]Z \wedge \forall\!\!\!\!\forall Z))$$

$$max(Y, \Phi \wedge \bigwedge_a [a]Y \wedge \forall\!\!\!\!\forall Y)$$

An Other Example

- the bell rings every 15 minutes

An Other Example

- the bell rings every 15 minutes

$$\text{bell} \wedge \qquad\qquad AG(\text{bell} \Rightarrow AF_{=15}\ \text{bell})$$

An Other Example

- the bell rings every 15 minutes

$$\text{bell} \wedge AG_{0<.<15} \neg\text{bell} \wedge AG(\text{bell} \Rightarrow AF_{=15}\ \text{bell}) \wedge AG(\neg\text{bell} \Rightarrow AG_{=15} \neg\text{bell})$$

An Other Example

- the bell rings every 15 minutes

$$\text{bell} \wedge AG_{0<.<15}\ \neg\text{bell} \wedge AG(\text{bell} \Rightarrow AF_{=15}\ \text{bell}) \wedge AG(\neg\text{bell} \Rightarrow AG_{=15}\ \neg\text{bell})$$

$\text{bell}\wedge$ $(0 < x < 15 \wedge \neg\text{bell}) \vee ((x = 15 \vee x = 0) \wedge \text{bell})$

An Other Example

- the bell rings every 15 minutes

$$\text{bell} \wedge AG_{0<.<15}\, \neg\text{bell} \wedge AG(\text{bell} \Rightarrow AF_{=15}\, \text{bell}) \wedge AG(\neg\text{bell} \Rightarrow AG_{=15}\, \neg\text{bell})$$

$$\text{bell} \wedge x \text{ in } \max(X, x \leq 15 \Rightarrow ((0 < x < 15 \wedge \neg\text{bell}) \vee ((x = 15 \vee x = 0) \wedge \text{bell} \wedge x \text{ in } (\forall X \wedge \bigwedge_a [a]X)$$

Some Possible Timed Models

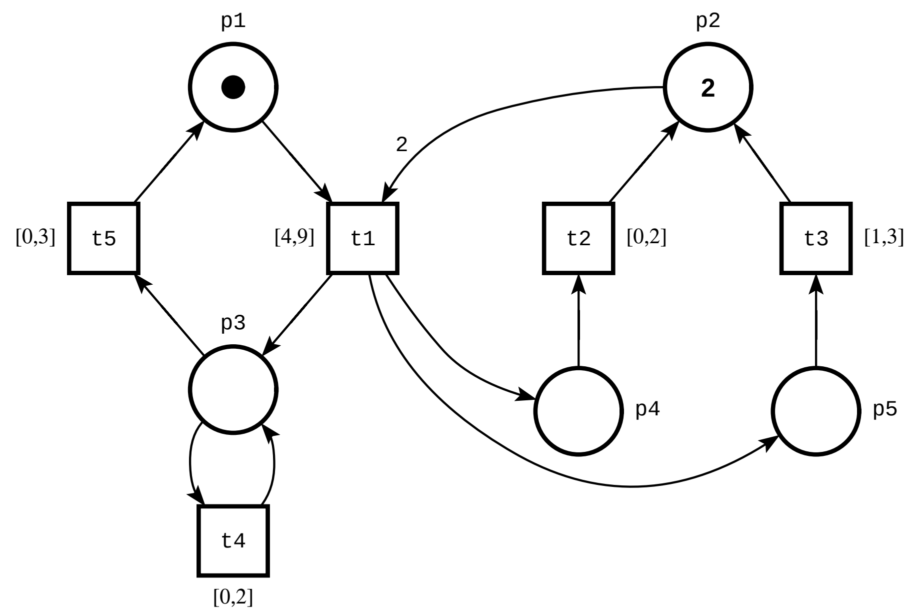
- Time Petri nets
- Timed process algebra
- Timed MSCs
- Graphs with durations
- Timed automata
- ...

Some Possible Timed Models

- Time Petri nets
- Timed process algebra
- Timed MSCs
- Graphs with durations
- Timed automata
- ...

Time Petri Nets

[Merlin 1974, Berthomieu & Diaz 1991]



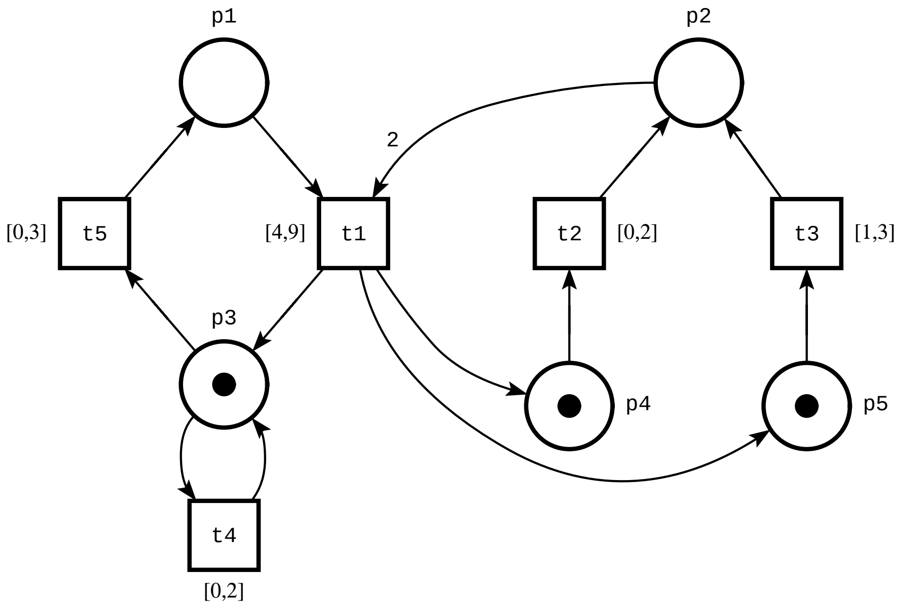
Initial marking: $(p_1, p_2(2))$

Only t_1 can be fired: $4 \leq t_1 \leq 9$

➜ t_1 is fired after θ_1

Time Petri Nets

[Merlin 1974, Berthomieu & Diaz 1991]



Initial marking: $(p_1, p_2(2))$

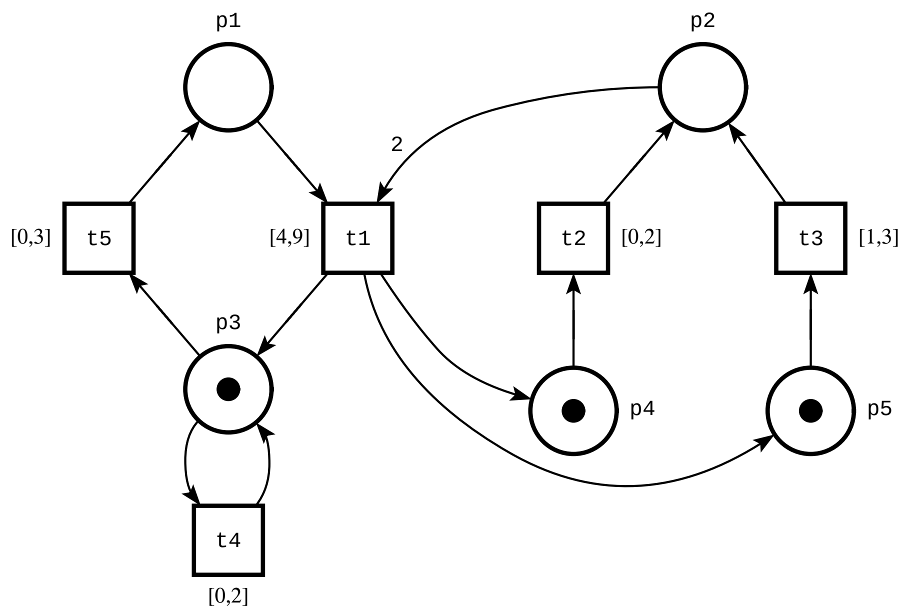
Only t_1 can be fired: $4 \leq t_1 \leq 9$

➜ t_1 is fired after θ_1

Marking: (p_3, p_4, p_5)

Time Petri Nets

[Merlin 1974, Berthomieu & Diaz 1991]



Initial marking: $(p_1, p_2(2))$

Only t_1 can be fired: $4 \leq t_1 \leq 9$

$\rightarrow$ t_1 is fired after θ_1

Marking: (p_3, p_4, p_5)

t_2, t_3, t_4 and t_5 can be fired:

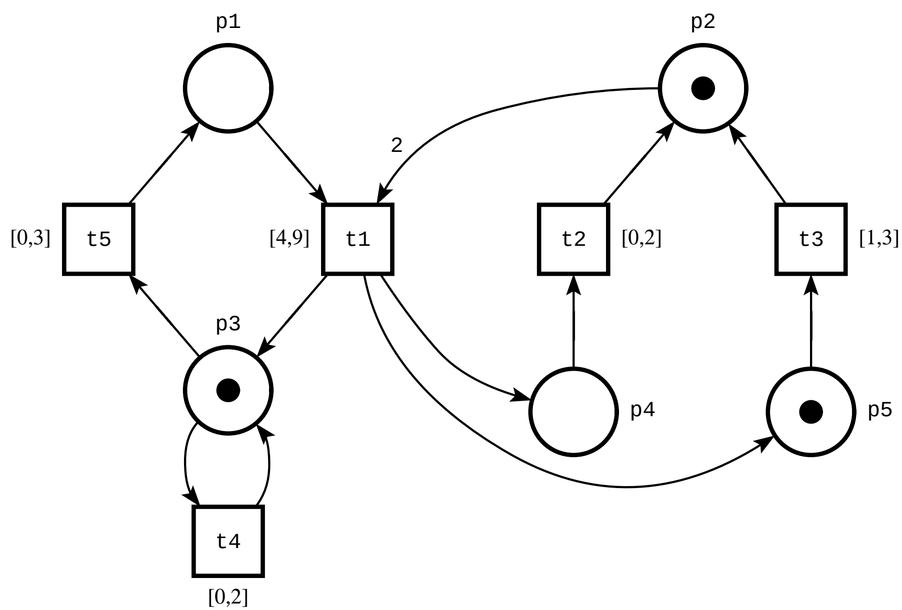
$0 \leq t_2 \leq 2$ $\quad$ $1 \leq t_3 \leq 3$

$0 \leq t_4 \leq 2$ $\quad$ $0 \leq t_5 \leq 3$

$\rightarrow$ t_2 is fired after θ_2

Time Petri Nets

[Merlin 1974, Berthomieu & Diaz 1991]



Initial marking: $(p_1, p_2(2))$

Only t_1 can be fired: $4 \leq t_1 \leq 9$

➜ t_1 is fired after θ_1

Marking: (p_3, p_4, p_5)

t_2, t_3, t_4 and t_5 can be fired:

$0 \leq t_2 \leq 2$ $\quad$ $1 \leq t_3 \leq 3$

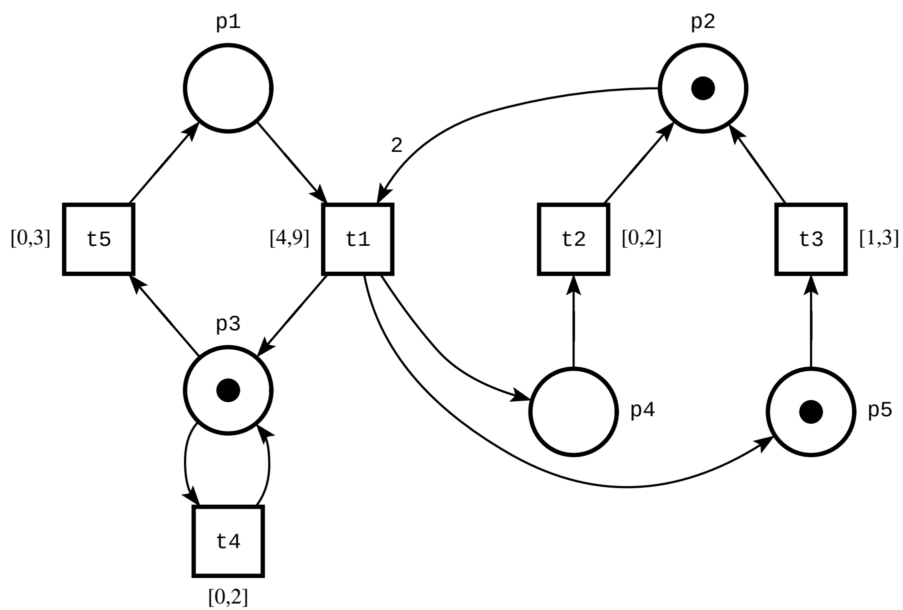
$0 \leq t_4 \leq 2$ $\quad$ $0 \leq t_5 \leq 3$

➜ t_2 is fired after θ_2

Marking: (p_2, p_3, p_5)

Time Petri Nets

[Merlin 1974, Berthomieu & Diaz 1991]



Initial marking: $(p_1, p_2(2))$

Only t_1 can be fired: $4 \leq t_1 \leq 9$

➜ t_1 is fired after θ_1

Marking: (p_3, p_4, p_5)

t_2, t_3, t_4 and t_5 can be fired:

$0 \leq t_2 \leq 2$ $\quad$ $1 \leq t_3 \leq 3$

$0 \leq t_4 \leq 2$ $\quad$ $0 \leq t_5 \leq 3$

➜ t_2 is fired after θ_2

Marking: (p_2, p_3, p_5)

t_3, t_4 and t_5 can be fired:

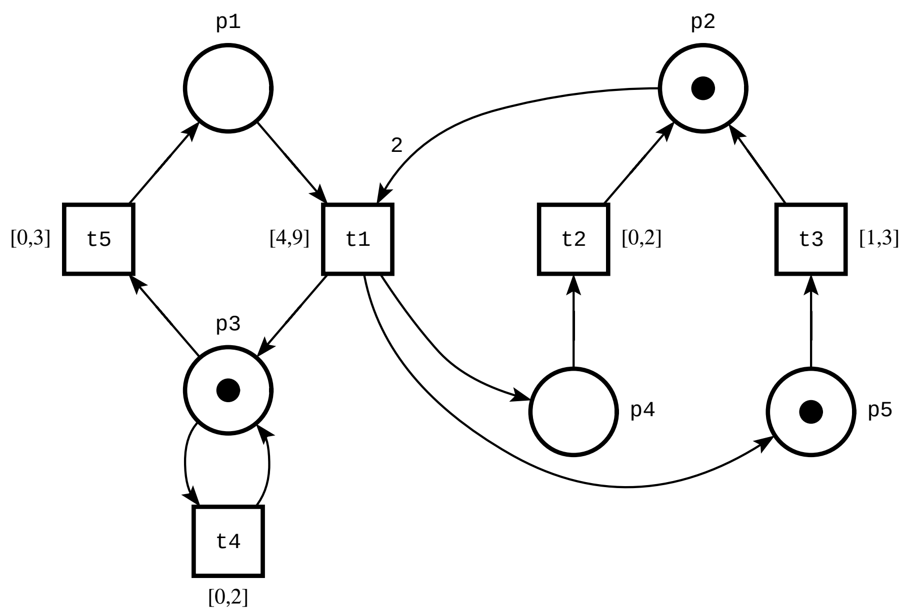
$\max(0, 1 - \theta_2) \leq t_3 \leq 3 - \theta_2$

$0 \leq t_4 \leq 2 - \theta_2$

$0 \leq t_5 \leq 3 - \theta_2$

Time Petri Nets

[Merlin 1974, Berthomieu & Diaz 1991]



Initial marking: $(p_1, p_2(2))$

Only t_1 can be fired: $4 \leq t_1 \leq 9$

$\rightarrow$ t_1 is fired after θ_1

Marking: (p_3, p_4, p_5)

t_2, t_3, t_4 and t_5 can be fired:

$0 \leq t_2 \leq 2$ $\quad$ $1 \leq t_3 \leq 3$

$0 \leq t_4 \leq 2$ $\quad$ $0 \leq t_5 \leq 3$

$\rightarrow$ t_2 is fired after θ_2

Marking: (p_2, p_3, p_5)

t_3, t_4 and t_5 can be fired:

$$\max(0, 1 - \theta_2) \leq t_3 \leq 3 - \theta_2$$

$$0 \leq t_4 \leq 2 - \theta_2$$

$$0 \leq t_5 \leq 3 - \theta_2$$

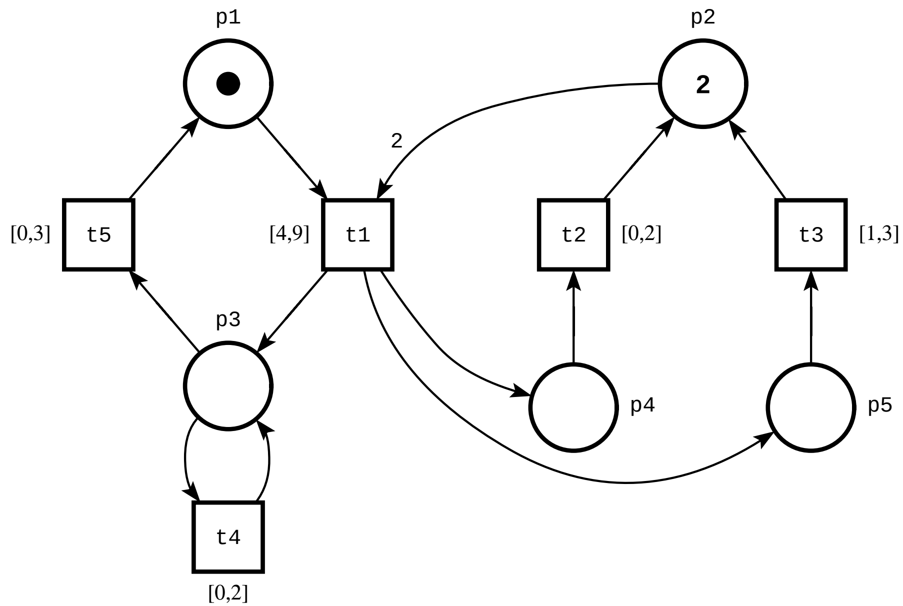
The scheduling $(t_1.t_2, \theta_1 = 5.\theta_2 = 0)$ is realizable

Time Petri Nets - Symbolic Analysis

Initial marking: $(p_1, p_2(2))$

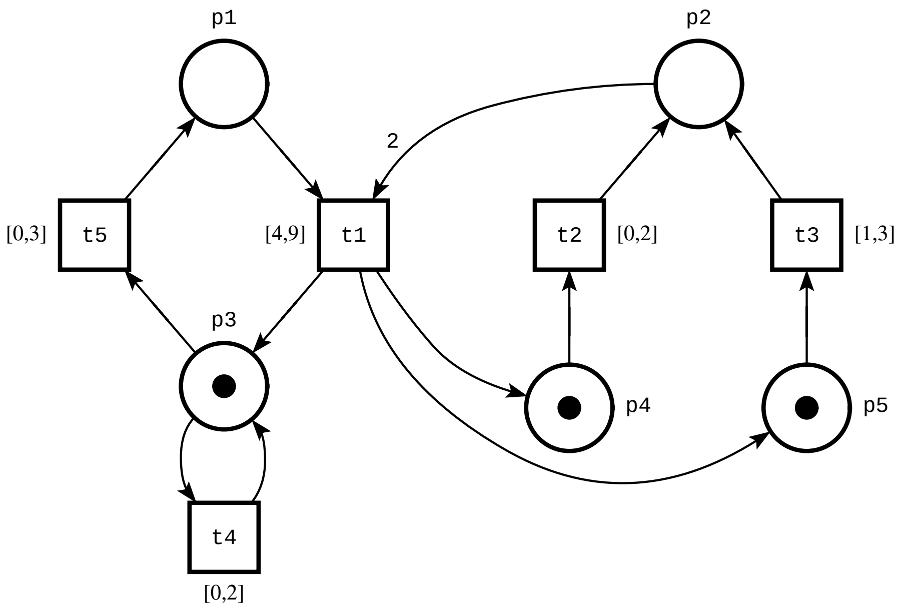
Initial class: $4 \leq t_1 \leq 9$

Graph of State Classes



Time Petri Nets - Symbolic Analysis

Graph of State Classes



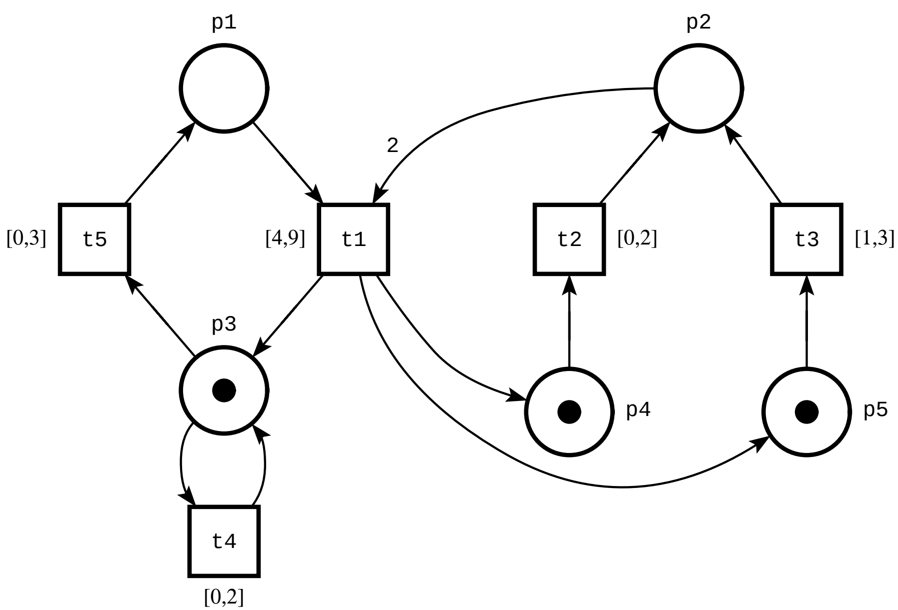
Initial marking: $(p_1, p_2(2))$

Initial class: $4 \leq t_1 \leq 9$

Marking: (p_3, p_4, p_5)

Time Petri Nets - Symbolic Analysis

Graph of State Classes



Initial marking: $(p_1, p_2(2))$

Initial class: $4 \leq t_1 \leq 9$

Marking: (p_3, p_4, p_5)

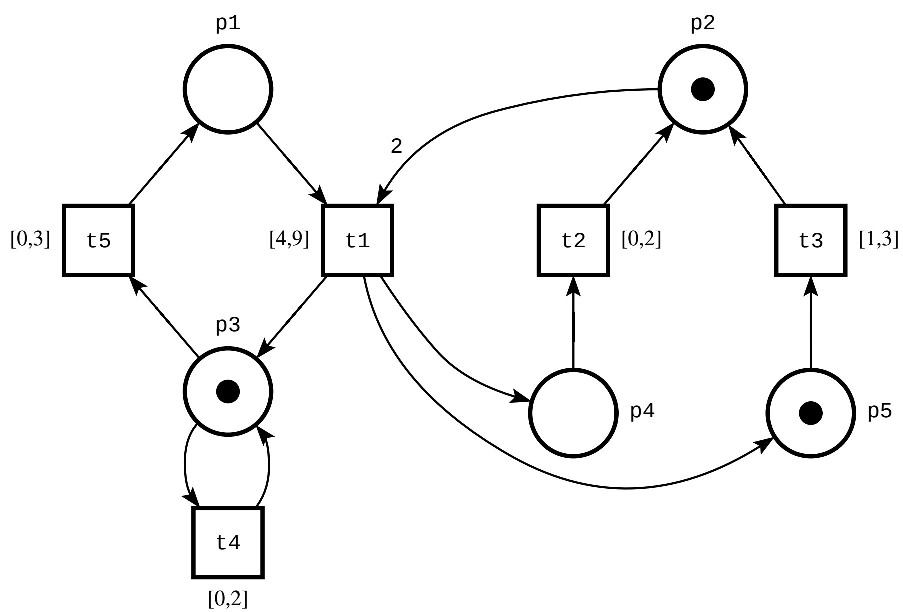
Class: $0 \leq t_2 \leq 2$ $\quad 1 \leq t_3 \leq 3$

$0 \leq t_4 \leq 2$ $\quad 0 \leq t_5 \leq 3$

➜ t_2 is fired after θ_2

Time Petri Nets - Symbolic Analysis

Graph of State Classes



Initial marking: $(p_1, p_2(2))$

Initial class: $4 \leq t_1 \leq 9$

Marking: (p_3, p_4, p_5)

Class: $0 \leq t_2 \leq 2$ $\quad 1 \leq t_3 \leq 3$

$0 \leq t_4 \leq 2$ $\quad 0 \leq t_5 \leq 3$

$\rightarrow$ t_2 is fired after θ_2

Marking: (p_2, p_3, p_5)

$\max(0, 1-\theta_2) \leq t_3 \leq 3-\theta_2$

$0 \leq t_4 \leq 2-\theta_2$

$0 \leq t_5 \leq 3-\theta_2$

Class: (eliminate θ_2)

$0 \leq t_3 \leq 3$ $\quad t_4 - t_3 \leq 1$

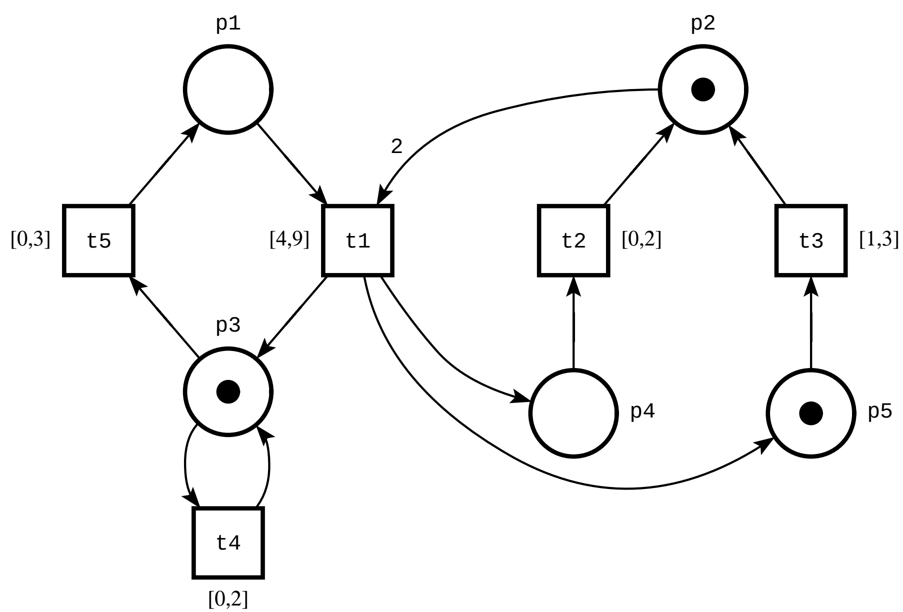
$0 \leq t_4 \leq 2$ $\quad t_5 - t_3 \leq 2$

$0 \leq t_5 \leq 3$

and so on...

Time Petri Nets - Symbolic Analysis

Graph of State Classes



potentially infinite graph...

Initial marking: $(p_1, p_2(2))$

Initial class: $4 \leq t_1 \leq 9$

Marking: (p_3, p_4, p_5)

Class: $0 \leq t_2 \leq 2$ $\quad 1 \leq t_3 \leq 3$

$0 \leq t_4 \leq 2$ $\quad 0 \leq t_5 \leq 3$

$\rightarrow t_2$ is fired after θ_2

Marking: (p_2, p_3, p_5)

$\max(0, 1 - \theta_2) \leq t_3 \leq 3 - \theta_2$

$0 \leq t_4 \leq 2 - \theta_2$

$0 \leq t_5 \leq 3 - \theta_2$

Class: (eliminate θ_2)

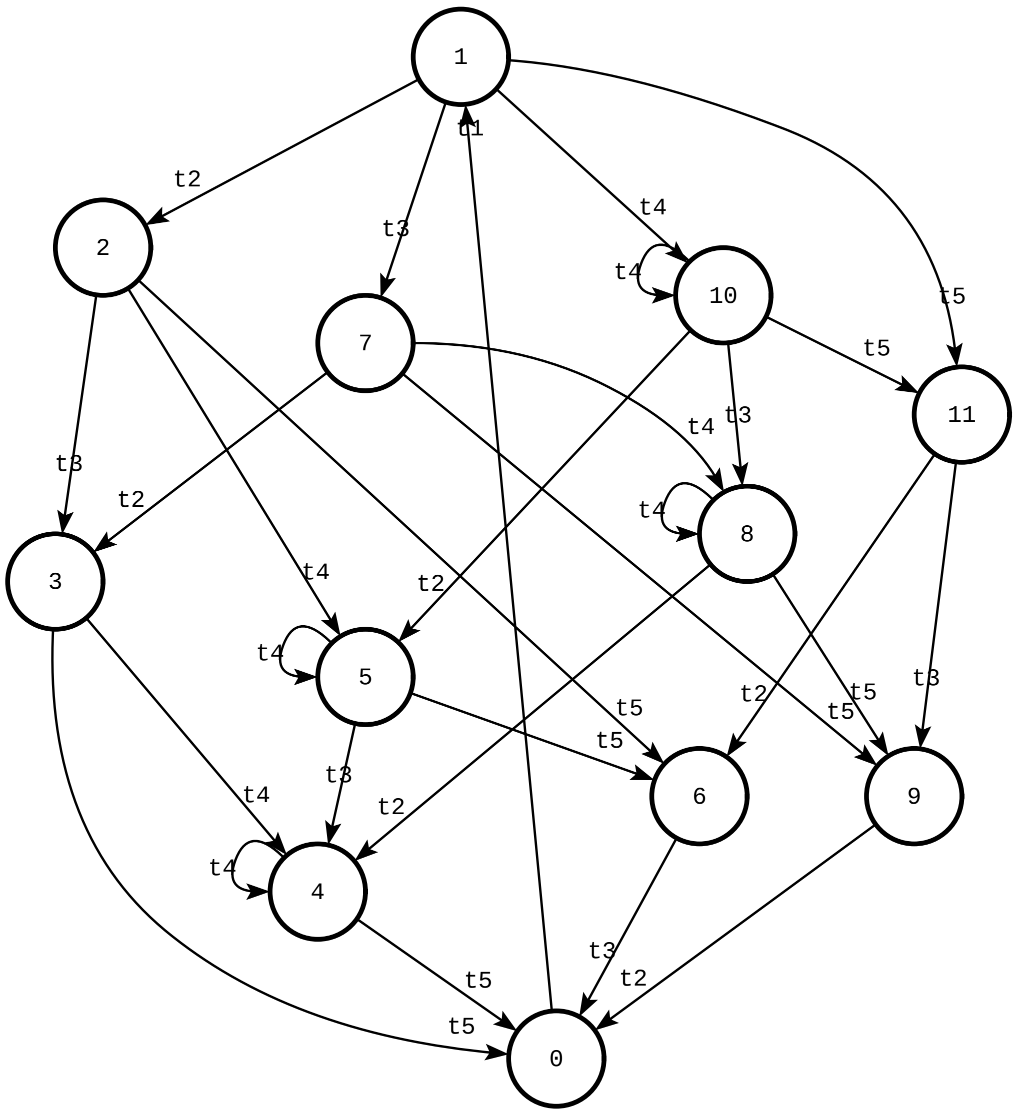
$0 \leq t_3 \leq 3$ $\quad t_4 - t_3 \leq 1$

$0 \leq t_4 \leq 2$ $\quad t_5 - t_3 \leq 2$

$0 \leq t_5 \leq 3$

and so on...

Graph of State Classes for the Example



Time Petri Nets - Properties

[Berthomieu & Menasche 1983] [Berthomieu & Diaz 1991]

Theorem: The number of state classes of a T-safe TPN is bounded if and only if the net is bounded.

Time Petri Nets - Properties

[Berthomieu & Menasche 1983] [Berthomieu & Diaz 1991]

Theorem: The number of state classes of a T-safe TPN is bounded if and only if the net is bounded.

Theorem: Boundedness and reachability of TPNs are undecidable.

Time Petri Nets - Properties

[Berthomieu & Menasche 1983] [Berthomieu & Diaz 1991]

Theorem: The number of state classes of a T-safe TPN is bounded if and only if the net is bounded.

Theorem: Boundedness and reachability of TPNs are undecidable.

Theorem: Reachability and LTL model-checking of bounded TPNs are decidable.

Time Petri Nets - Properties

[Berthomieu & Menasche 1983] [Berthomieu & Diaz 1991]

Theorem: The number of state classes of a T-safe TPN is bounded if and only if the net is bounded.

Theorem: Boundedness and reachability of TPNs are undecidable.

Theorem: Reachability and LTL model-checking of bounded TPNs are decidable.

A more involved construction is needed for branching-time model-checking
[Yoneda & Ryuba 1998]

Time Petri Nets – Properties

[Berthomieu & Menasche 1983] [Berthomieu & Diaz 1991]

Theorem: The number of state classes of a T-safe TPN is bounded if and only if the net is bounded.

Theorem: Boundedness and reachability of TPNs are undecidable.

Theorem: Reachability and LTL model-checking of bounded TPNs are decidable.

A more involved construction is needed for branching-time model-checking

[Yoneda & Ryuba 1998]

[Cassez & Roux 2003] Any bounded TPN can be transformed into an equivalent (for strong bisimulation) **network of timed automata**. [one transition = one component]

Time Petri Nets – Properties

[Berthomieu & Menasche 1983] [Berthomieu & Diaz 1991]

Theorem: The number of state classes of a T-safe TPN is bounded if and only if the net is bounded.

Theorem: Boundedness and reachability of TPNs are undecidable.

Theorem: Reachability and LTL model-checking of bounded TPNs are decidable.

A more involved construction is needed for branching-time model-checking
[Yoneda & Ryuba 1998]

[Cassez & Roux 2003] Any bounded TPN can be transformed into an equivalent (for strong bisimulation) **network of timed automata**. [one transition = one component]

➜ formal proof of the decidability of TCTL for bounded TPNs

Time Petri Nets – Properties

[Berthomieu & Menasche 1983] [Berthomieu & Diaz 1991]

Theorem: The number of state classes of a T-safe TPN is bounded if and only if the net is bounded.

Theorem: Boundedness and reachability of TPNs are undecidable.

Theorem: Reachability and LTL model-checking of bounded TPNs are decidable.

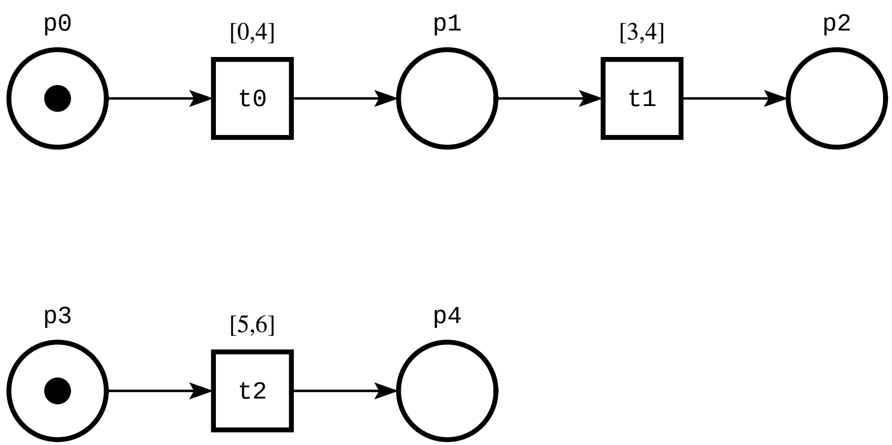
A more involved construction is needed for branching-time model-checking [Yoneda & Ryuba 1998]

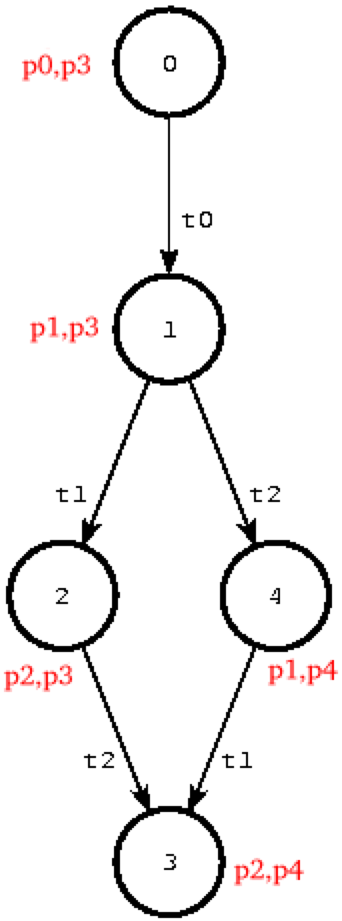
[Cassez & Roux 2003] Any bounded TPN can be transformed into an equivalent (for strong bisimulation) **network of timed automata**. [one transition = one component]

→ formal proof of the decidability of TCTL for bounded TPNs

[Gardey, Roux & Roux 2003] Zone-based algorithm for checking reachability

Problem with Branching Time





CTL formula: $EF(p1 \wedge p3 \wedge AF(p2 \wedge p3))$

Graphs with Durations

✔ $\ell \xrightarrow{[n,m]} \ell'$: "moving from ℓ to ℓ' takes some duration d in $[n, m]$"

Graphs with Durations

✔ $\ell \xrightarrow{[n,m]} \ell'$: "moving from ℓ to ℓ' takes some duration d in $[n, m]$"

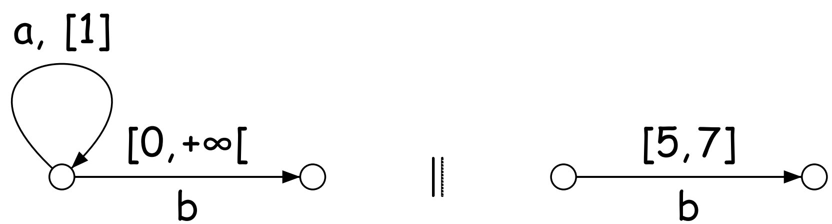
➜ reachability is NLOGSPACE-complete
[Laroussinie, Markey & Schnoebelen 2004]

Graphs with Durations

- ✔ $\ell \xrightarrow{[n,m]} \ell'$: "moving from ℓ to ℓ' takes some duration d in [n, m]"

  ➜ reachability is NLOGSPACE-complete
  [Laroussinie, Markey & Schnoebelen 2004]

- ✔ **Problem with parallel composition**

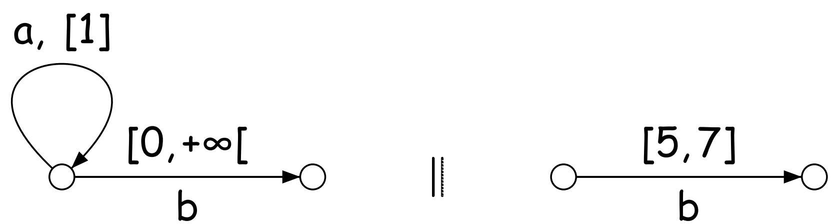


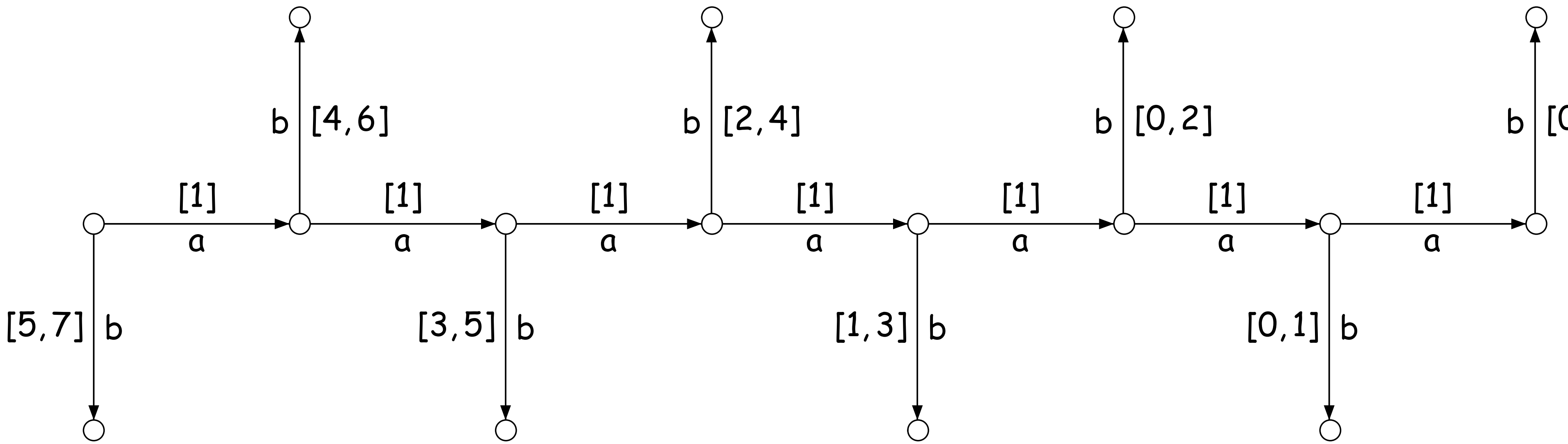
Graphs with Durations

- ✔ $\ell \xrightarrow{[n,m]} \ell'$: "moving from ℓ to ℓ' takes some duration d in $[n,m]$"

➜ reachability is NLOGSPACE-complete
[Laroussinie, Markey & Schnoebelen 2004]

- ✔ **Problem with parallel composition**





Graphs with Durations

- ✔ $\ell \xrightarrow{[n,m]} \ell'$: "moving from ℓ to ℓ' takes some duration d in $[n, m]$"

  ➜ reachability is NLOGSPACE-complete
  [Laroussinie, Markey & Schnoebelen 2004]

- ✔ **Problem with parallel composition**

  ➜ Even if low complexity bounds, not convenient for modelling concurrency

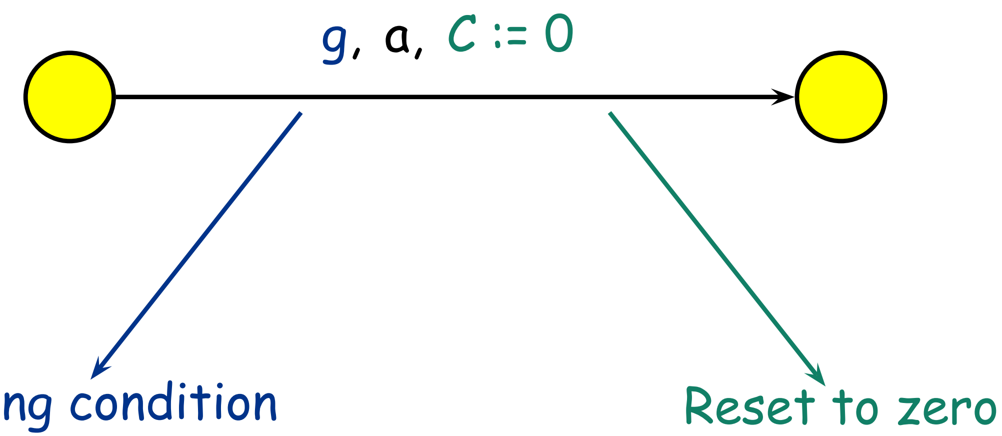
Some Comments

- ✔ the most-accepted timed model is **timed automata**
- ✔ the techniques used for analyzing TPNs and timed automata are very similar
  - state class graph $\leftrightarrow$ zone automaton
  - strong state class graph **[Berthomieu & Vernadat 2003]**
    $\leftrightarrow$ minimal graph **[Bouajjani, Fernandez, Halbwachs & Raymond 1992]**

Timed Automata

Timed Automata [Alur & Dill 90's]

- ✔ A finite control structure + variables (clocks)
- ✔ A transition is of the form:



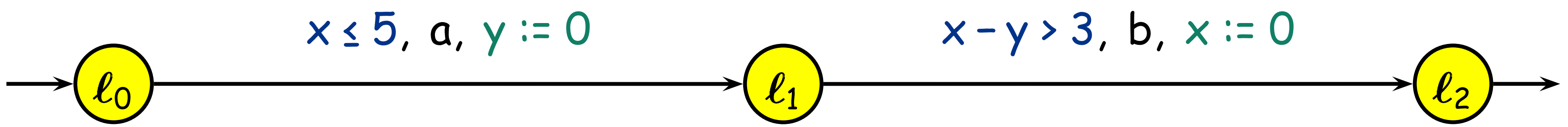
- ✔ An enabling condition (or guard) is:

$$g \ ::= \ x \sim c \ \mid \ x - y \sim c \ \mid \ g \wedge g$$

where $\sim \in \{<, \leq, =, \geq, >\}$

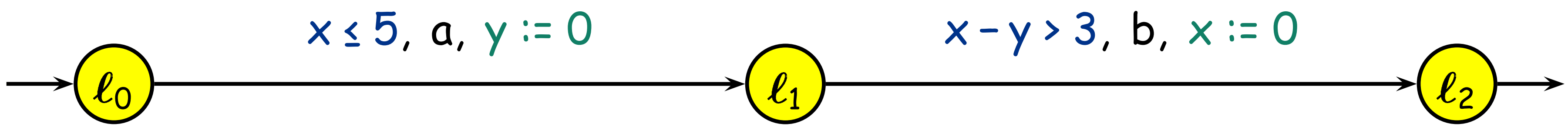
Timed Automata, an Example

x, y : clocks



Timed Automata, an Example

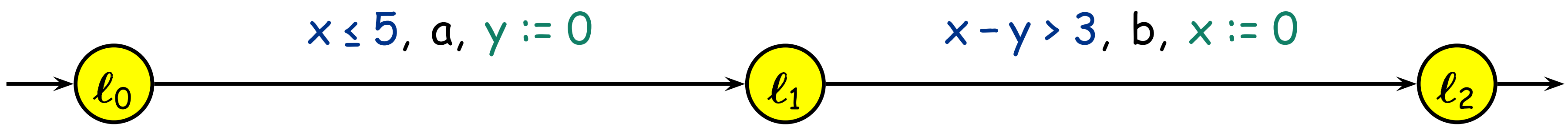
x, y : clocks



	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

Timed Automata, an Example

x, y : clocks

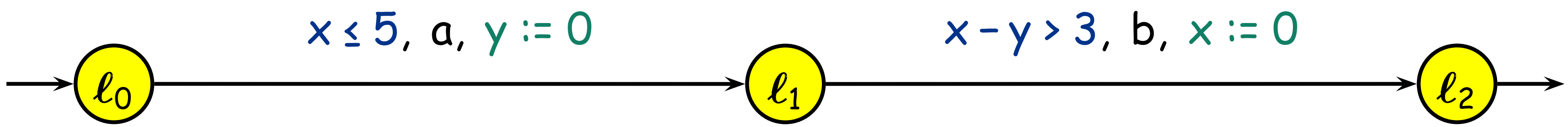


	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

(clock) valuation

Timed Automata, an Example

x, y : clocks



	ℓ_0	$\xrightarrow{\delta(4.1)}$	ℓ_0	$\xrightarrow{a}$	ℓ_1	$\xrightarrow{\delta(1.4)}$	ℓ_1	$\xrightarrow{b}$	ℓ_2
x	0		4.1		4.1		5.5		0
y	0		4.1		0		1.4		1.4

(clock) valuation

→ timed word $(a, 4.1)(b, 5.5)$

TA Semantics

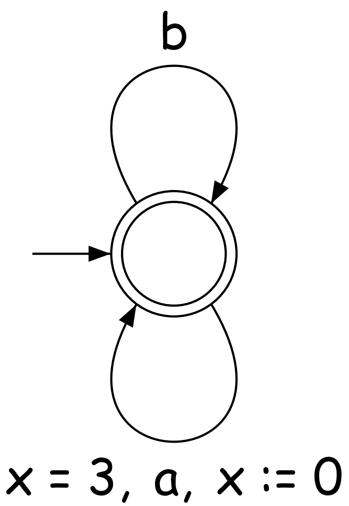
- $\mathcal{A} = (\Sigma, L, X, \longrightarrow)$ is a TA
- **Configurations:** $(\ell, v) \in L \times T^X$ where T is the time domain
- **Timed Transition System:**
  - **action transition:** $(\ell, v) \xrightarrow{a} (\ell', v')$ if $\exists \ell \xrightarrow{g,a,r} \ell' \in \mathcal{A}$ s.t. $\begin{array}{l} v \models g \\ v' = v[r \leftarrow 0] \end{array}$
  - **delay transition:** $(q, v) \xrightarrow{\delta(d)} (q, v + d)$ if $d \in T$

Some Exercices

What do the following TA recognize?

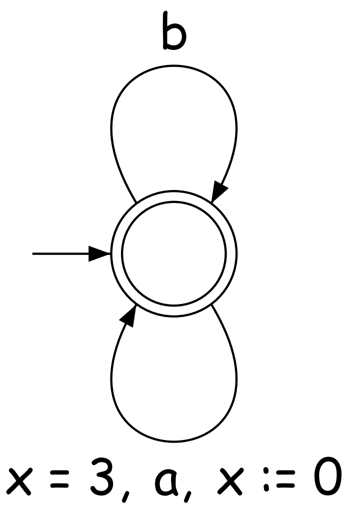
Some Exercices

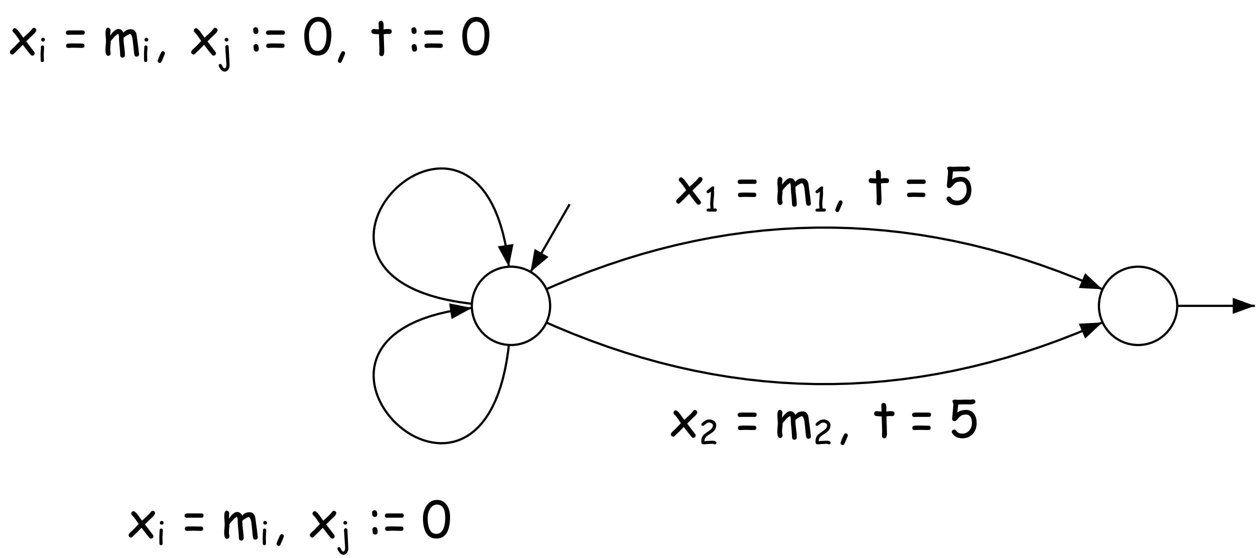
What do the following TA recognize?



Some Exercices

What do the following TA recognize?





Composition of TA

To model concurrent systems: several communicating components

➜ n-ary synchronization function

(combine synchronization rules and interleaving rules)

Composition of TA

To model concurrent systems: several communicating components

➜ n-ary synchronization function

(combine synchronization rules and interleaving rules)

Note: e.g. in **Uppaal**: binary synchronization, in **HyTech**: binary synchronization, in **Kronos**: binary synchronization, **(H)CMC**: n-ary synchronization

Composition of TA

To model concurrent systems: several communicating components

➜ n-ary synchronization function

(combine synchronization rules and interleaving rules)

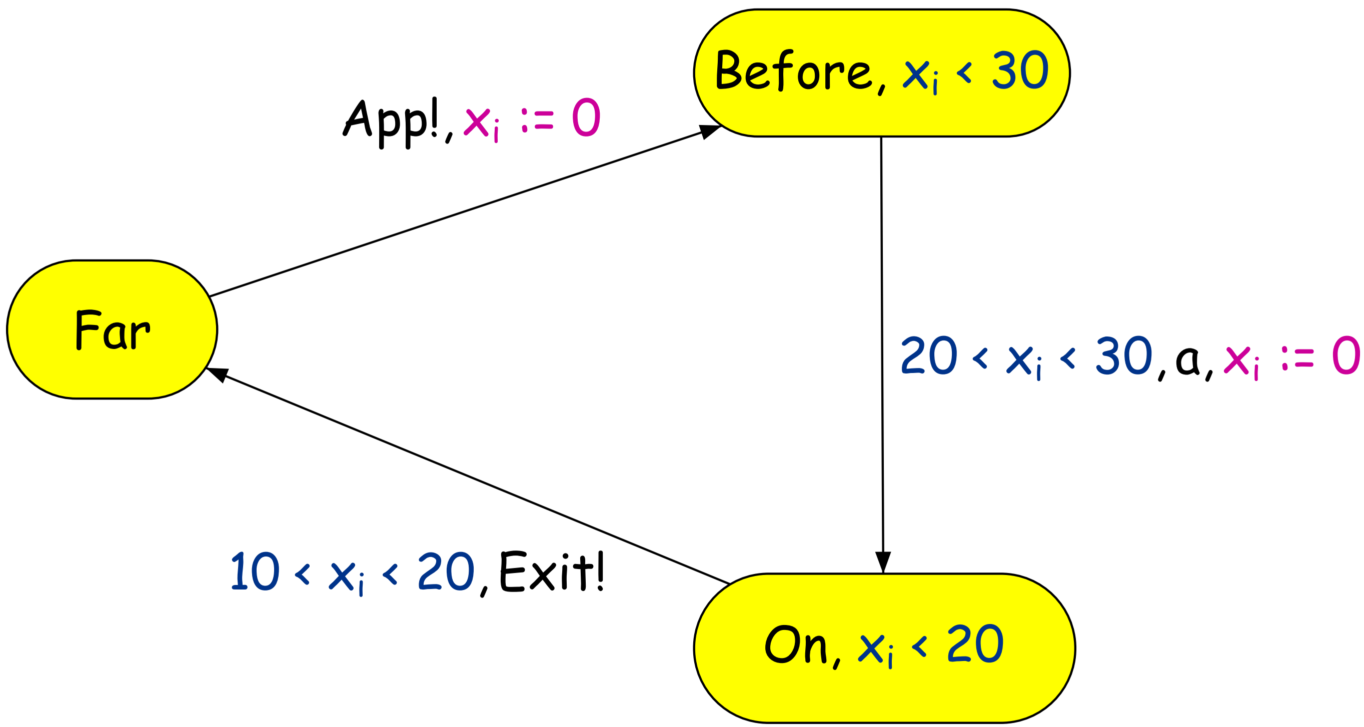
Note: e.g. in **Uppaal**: binary synchronization, in **HyTech**: binary synchronization, in **Kronos**: binary synchronization, **(H)CMC**: n-ary synchronization

Remark: concurrent timed automata **[Lanotte, Maggiolo-Schettini & Tini 2003]**

- notion of private/shared clocks
- relative conciseness

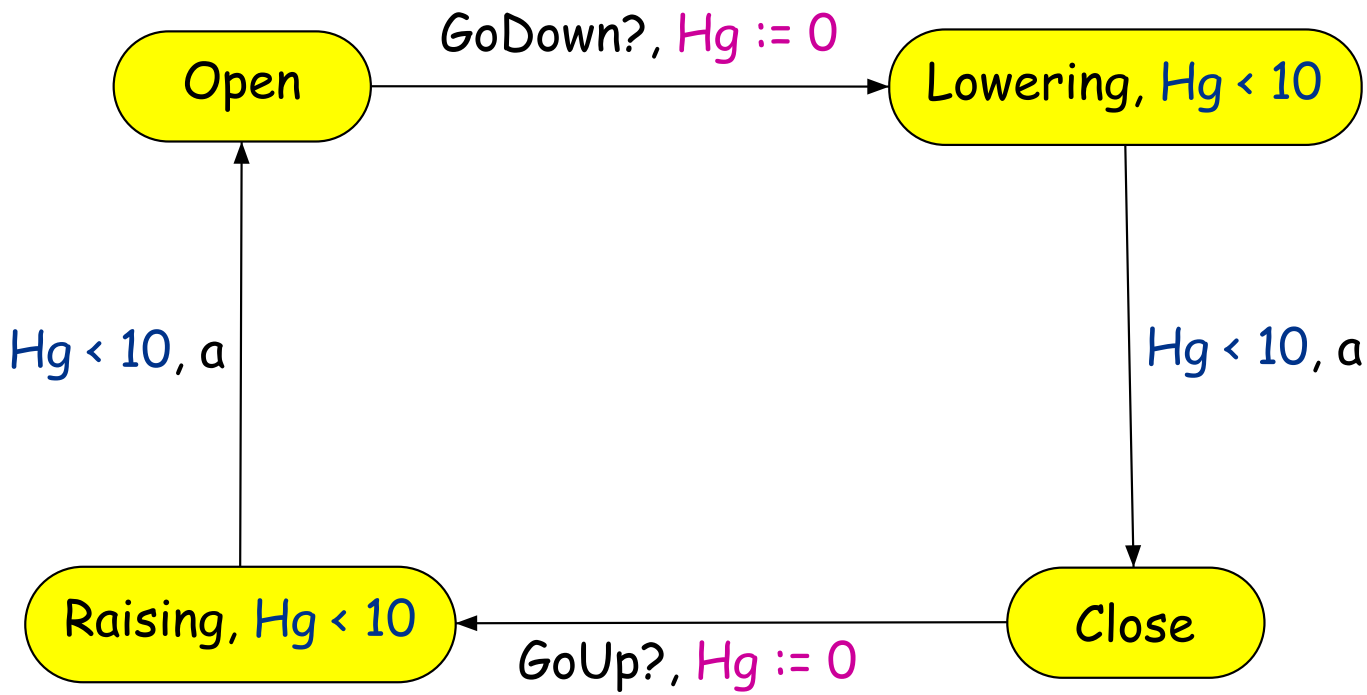
The Train Crossing Example (1)

Train$_i$ with $i = 1, 2, \ldots$



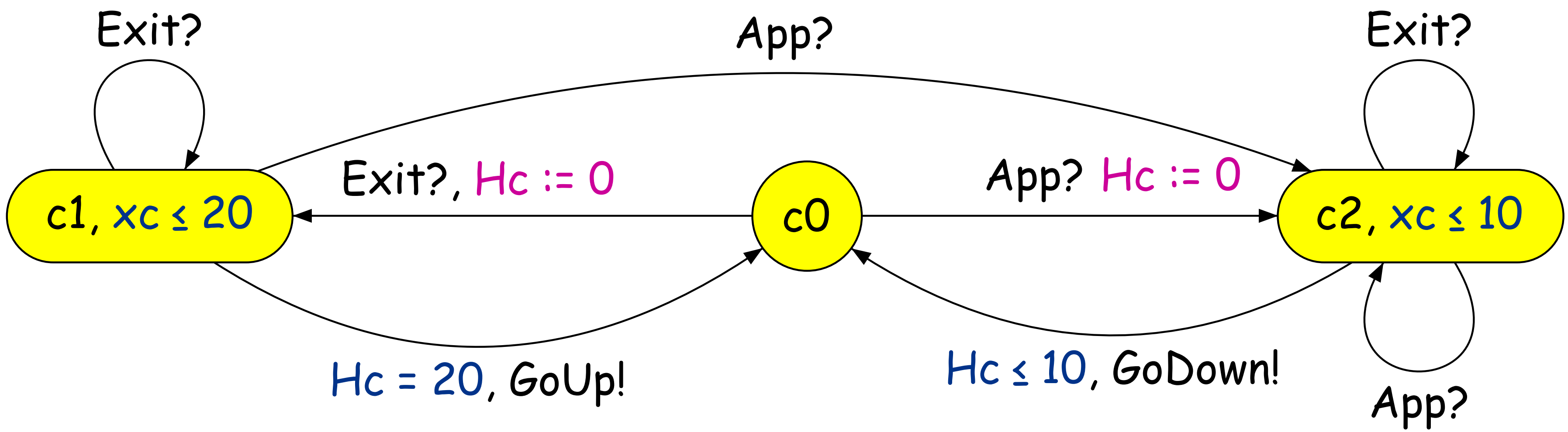
The Train Crossing Example (2)

The gate:



The Train Crossing Example (3)

The controller:



The Train Crossing Example (4)

We use the synchronization function f:

$Train_1$	$Train_2$	Gate	Controller	
App!	.	.	App?	App
.	App!	.	App?	App
Exit!	.	.	Exit?	Exit
.	Exit!	.	Exit?	Exit
a	.	.	.	a
.	a	.	.	a
.	.	a	.	a
.	.	GoUp?	GoUp!	GoUp
.	.	GoDown?	GoDown!	GoDown

to define the parallel composition $(Train_1 \parallel Train_2 \parallel Gate \parallel Controller)$

NB: the parallel composition does not add expressive power !

The Train Crossing Example (5)

Some properties one could check:

- ✔ Is the gate closed when a train crosses the road?

The Train Crossing Example (5)

Some properties one could check:

- ✔ Is the gate closed when a train crosses the road?

$$AG(\text{train.On} \Rightarrow \text{gate.Close})$$

The Train Crossing Example (5)

Some properties one could check:

- ✔ Is the gate closed when a train crosses the road?

$$AG(\text{train.On} \Rightarrow \text{gate.Close})$$

- ✔ Is the gate always closed for less than 5 minutes?

The Train Crossing Example (5)

Some properties one could check:

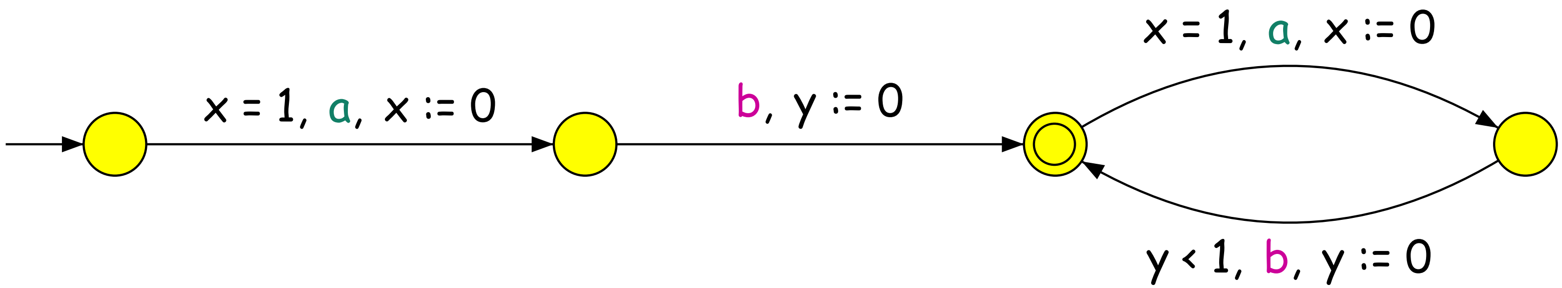
- ✔ Is the gate closed when a train crosses the road?

$$AG(\text{train.On} \Rightarrow \text{gate.Close})$$

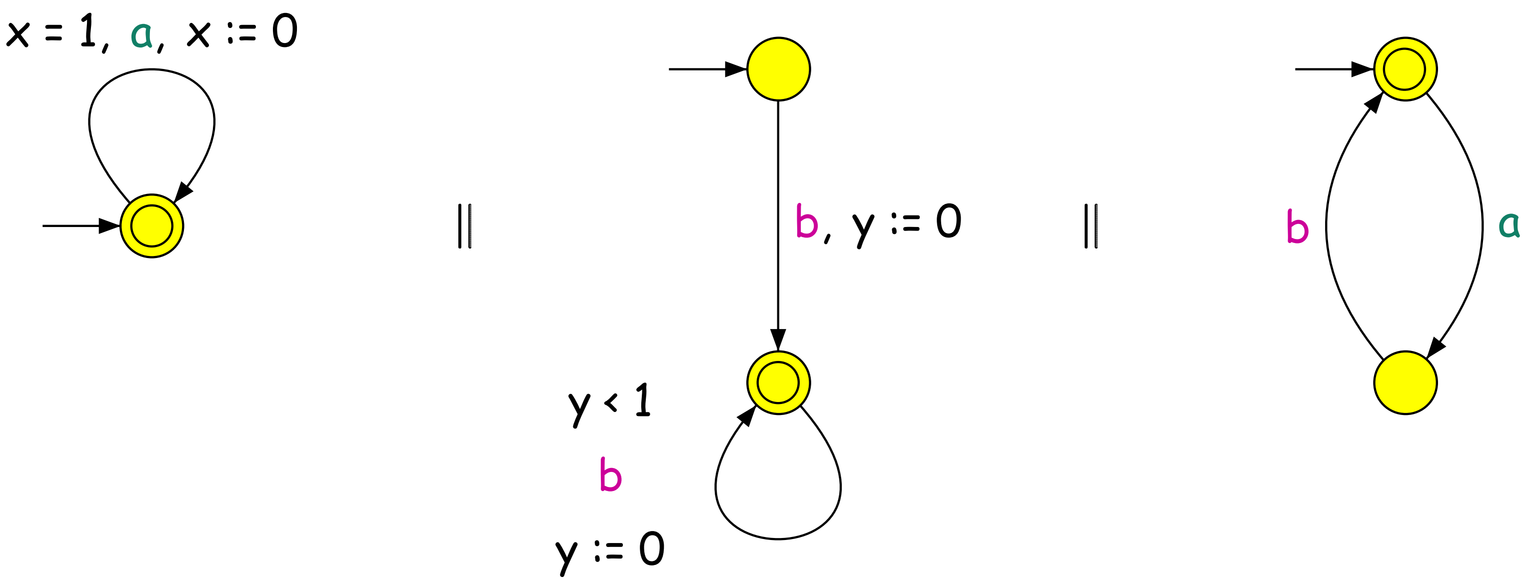
- ✔ Is the gate always closed for less than 5 minutes?

$$\neg EF(\text{gate.Close} \wedge (\text{gate.Close}\ U_{>5\ \text{min}}\ \neg\text{gate.Close}))$$

Discrete vs Dense-Time Semantics



- ✔ Dense-time: $L_{dense} = \{((ab)^\omega, \tau) \mid \forall i,\ \tau_{2i-1} = i \text{ and } \tau_{2i} - \tau_{2i-1} > \tau_{2i+2} - \tau_{2i+1}\}$
- ✔ Discrete-time: $L_{discrete} = \emptyset$



Verification of TA

Problem: the set of configurations is infinite

➜ classical methods can not be applied

Verification of TA

Problem: the set of configurations is infinite

➜ classical methods can not be applied

Positive key point: variables (clocks) have the same speed

Verification of TA

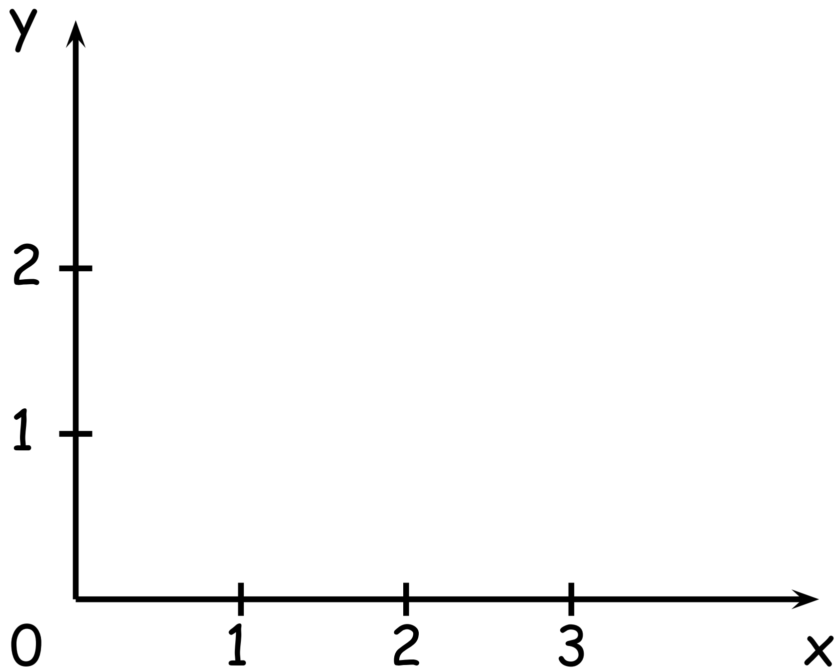
Problem: the set of configurations is infinite

➜ classical methods can not be applied

Positive key point: variables (clocks) have the same speed

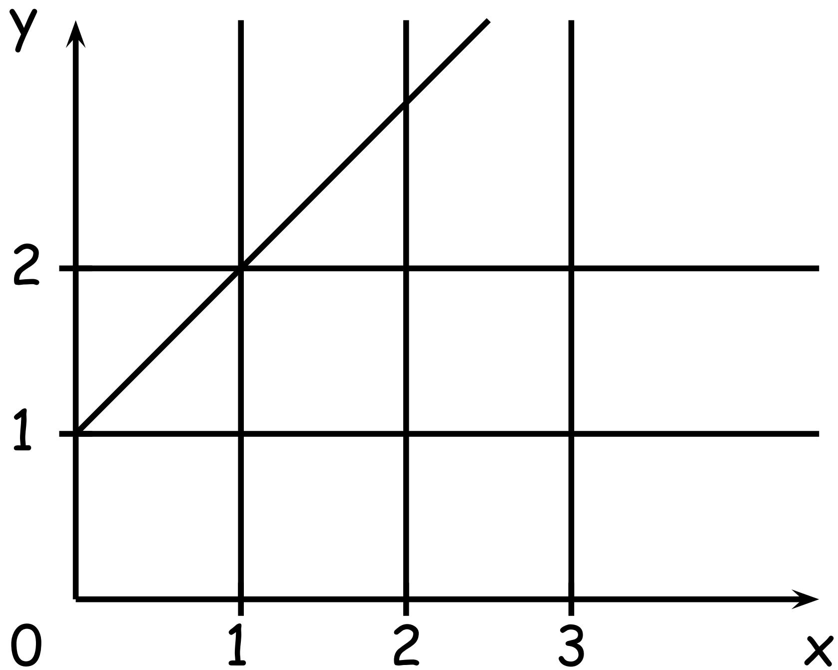
Aim: construct a finite abstraction

The Region Abstraction



Equivalence of finite index

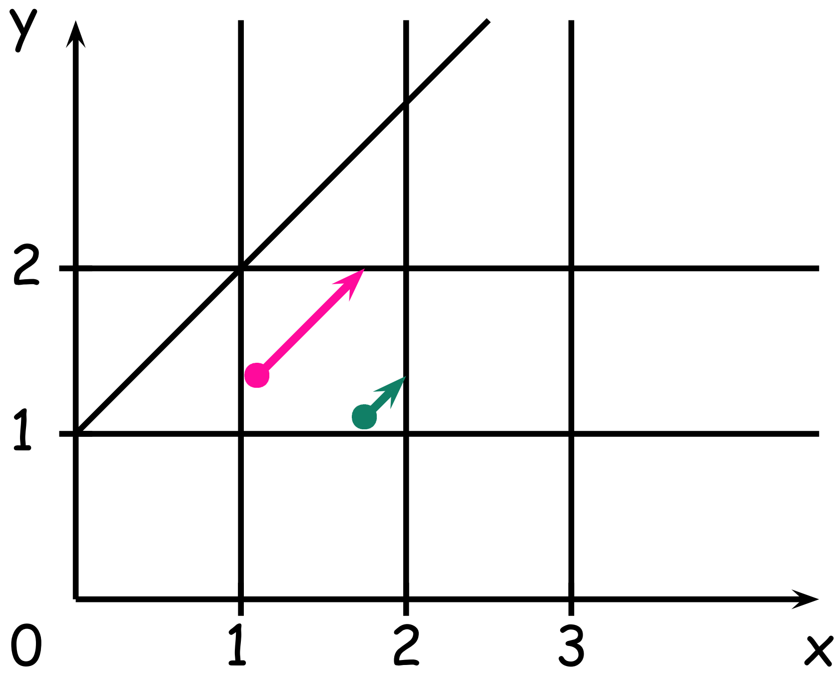
The Region Abstraction



Equivalence of finite index

- ✔ “compatibility” between regions and constraints

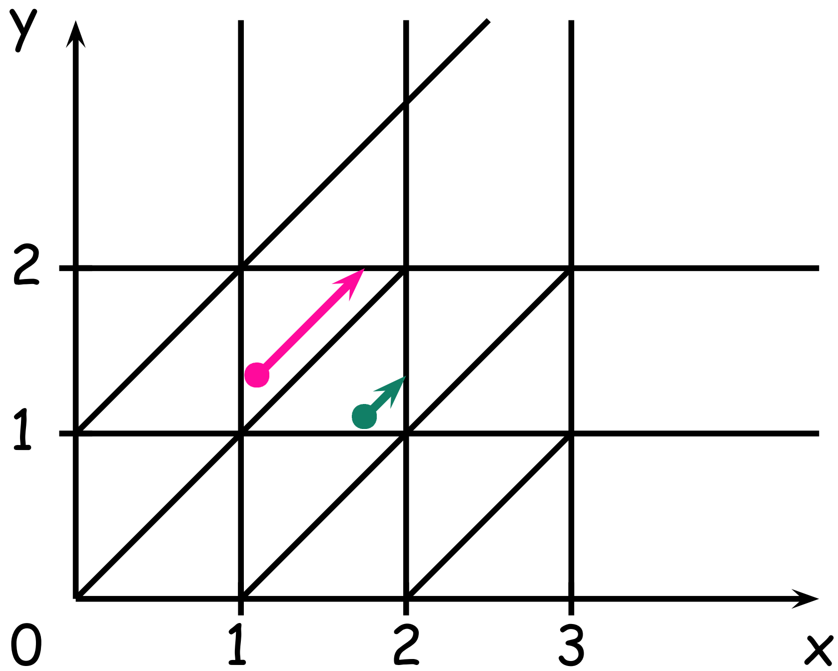
The Region Abstraction



Equivalence of finite index

- ✔ "compatibility" between regions and constraints
- ✔ "compatibility" between regions and time elapsing

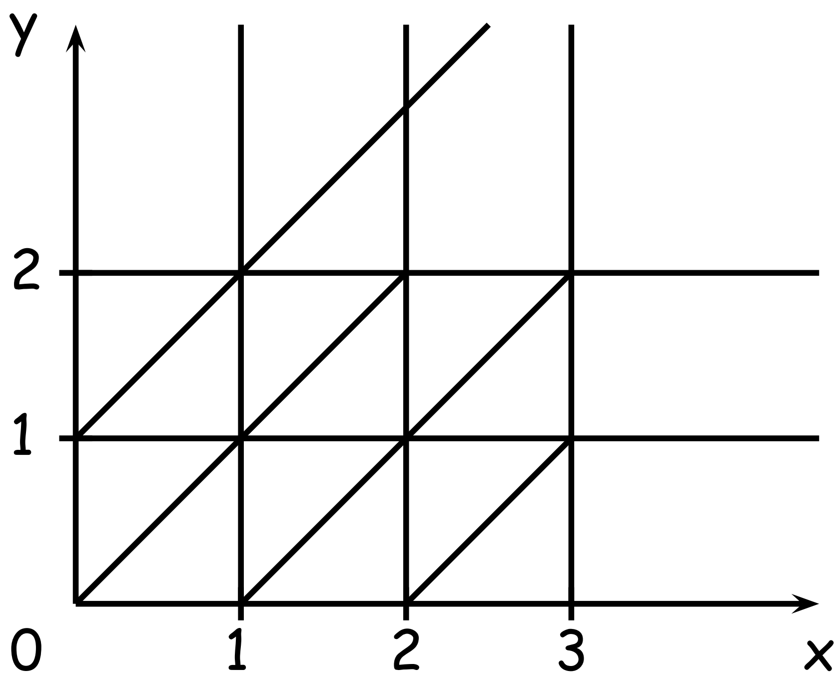
The Region Abstraction



Equivalence of finite index

- ✔ "compatibility" between regions and constraints
- ✔ "compatibility" between regions and time elapsing

The Region Abstraction

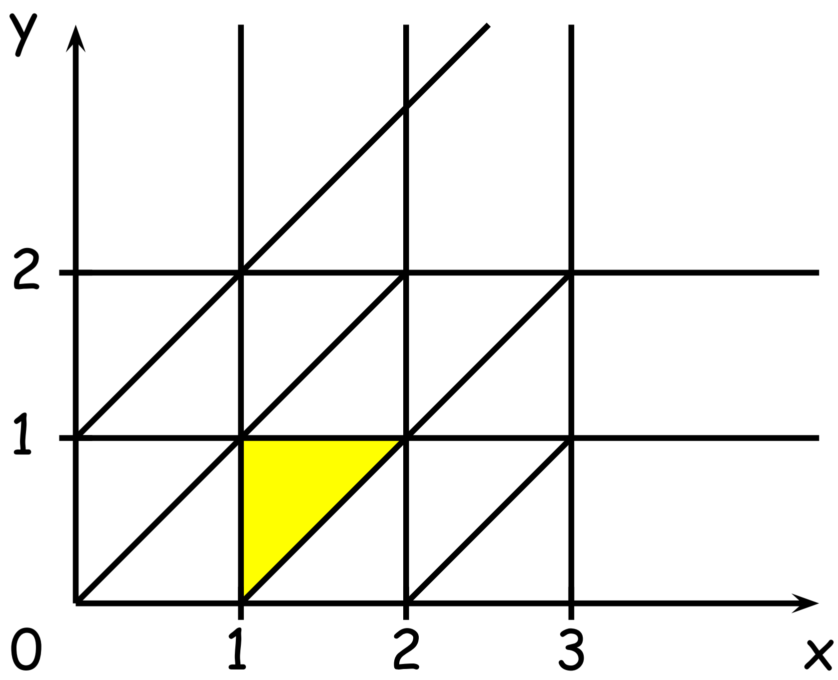


Equivalence of finite index

- ✔ "compatibility" between regions and constraints
- ✔ "compatibility" between regions and time elapsing

→ a bisimulation property

The Region Abstraction



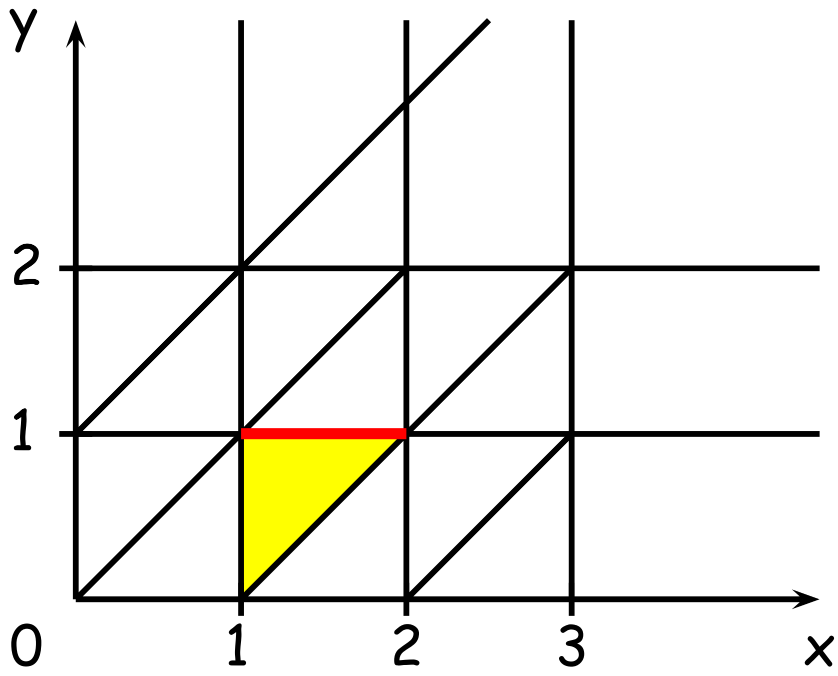
Equivalence of finite index

region defined by
$I_x =]1;2[$, $I_y =]0;1[$
$\{x\} < \{y\}$

- ✔ "compatibility" between regions and constraints
- ✔ "compatibility" between regions and time elapsing

➜ a **bisimulation** property

The Region Abstraction



Equivalence of finite index

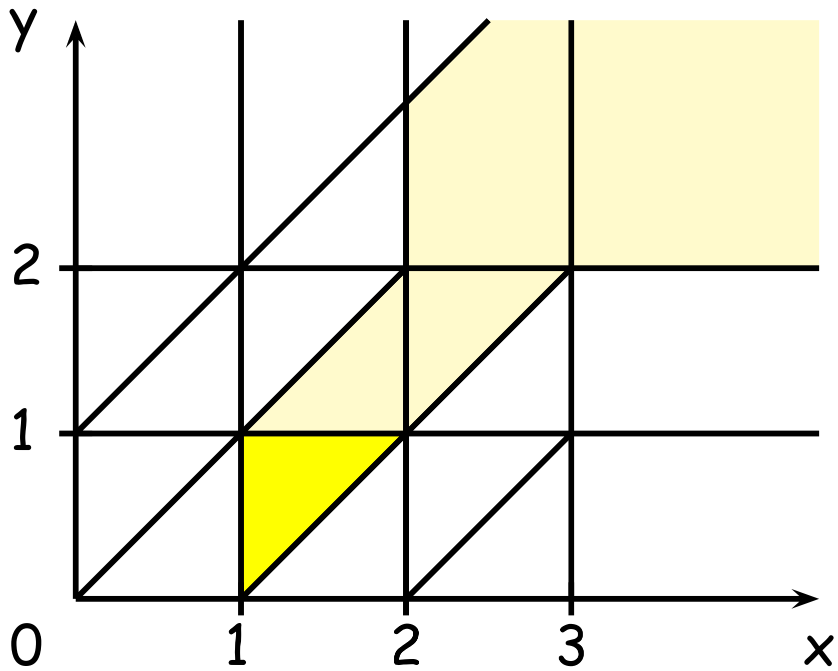
region defined by
$I_x =]1;2[,\ I_y =]0;1[$
$\{x\} < \{y\}$

successor region
$I_x =]1;2[,\ I_y = \{1\}$

- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

→ a bisimulation property

The Region Abstraction



Equivalence of finite index

region defined by
$I_x =]1;2[$, $I_y =]0;1[$
$\{x\} < \{y\}$

successor regions

- ✔ "compatibility" between regions and constraints
- ✔ "compatibility" between regions and time elapsing

➜ a bisimulation property

The Region Automaton

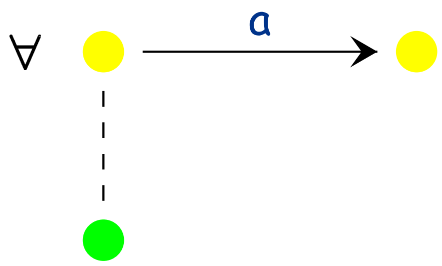
timed automaton $\otimes$ region abstraction

$\ell \xrightarrow{g,a,C:=0} \ell'$ is transformed into:

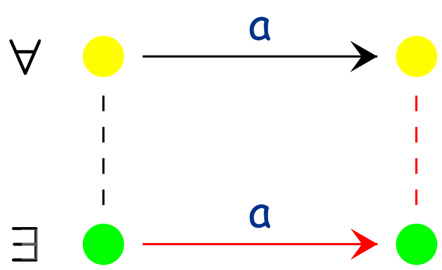
$(\ell, R) \xrightarrow{a} (\ell', R')$ if there exists $R'' \in \mathrm{Succ}_t^*(R)$ s.t.

- ✔ $R'' \subseteq g$
- ✔ $[C \leftarrow 0]R'' \subseteq R'$

Time-Abstract Bisimulation

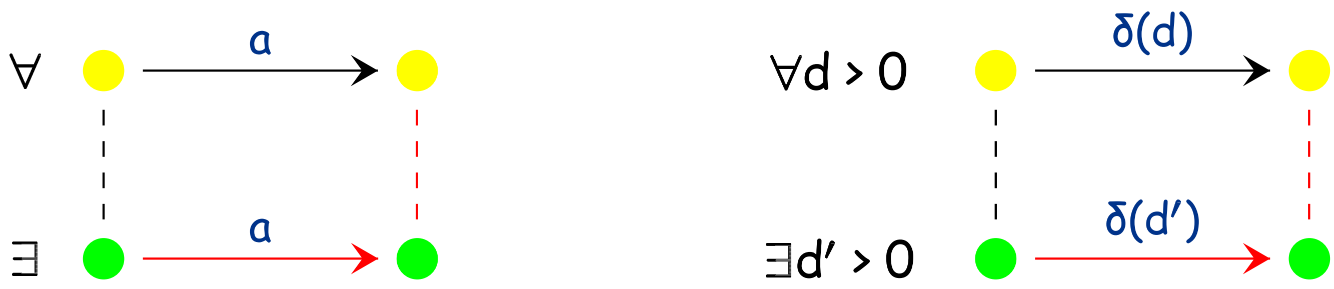


Time-Abstract Bisimulation

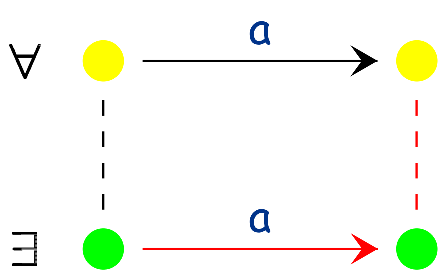


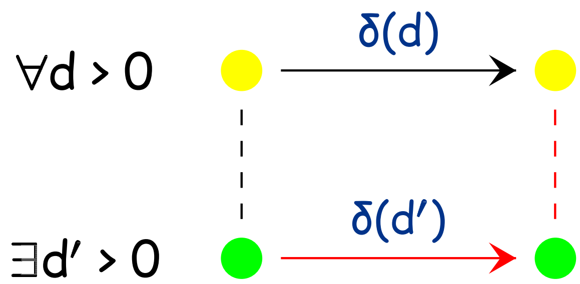
Time-Abstract Bisimulation

Time-Abstract Bisimulation



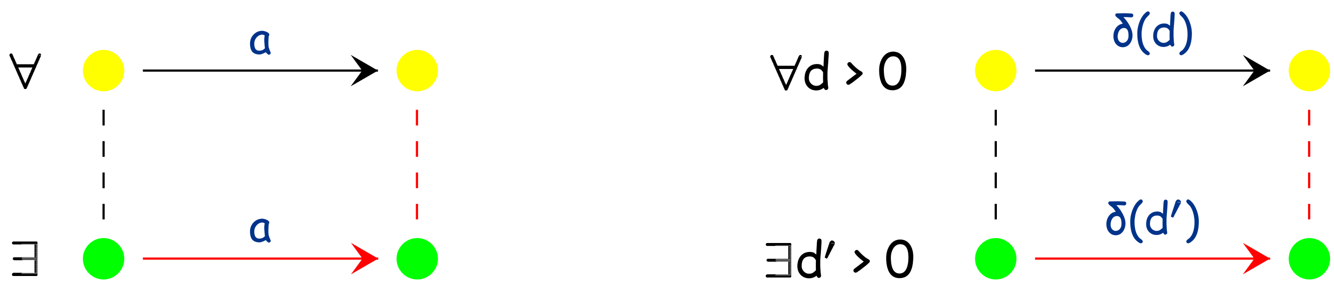
Time-Abstract Bisimulation





$$(\ell_0, v_0) \xrightarrow{a_1, t_1} (\ell_1, v_1) \xrightarrow{a_2, t_2} (\ell_2, v_2) \xrightarrow{a_3, t_3} \ldots$$

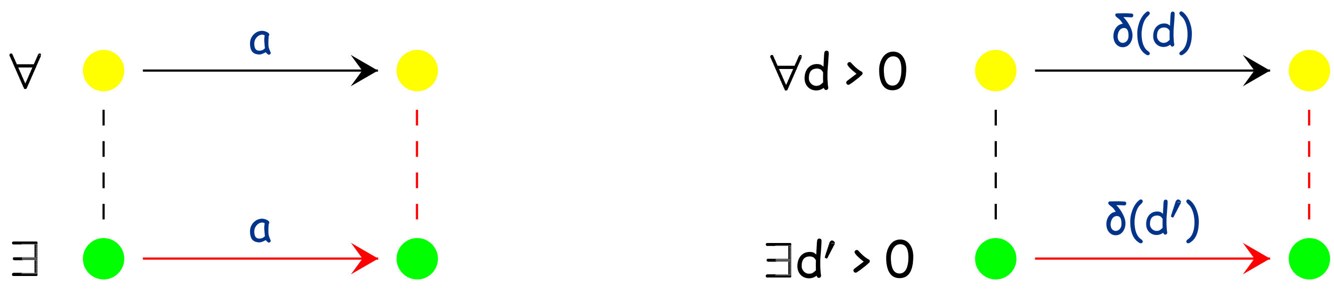
Time-Abstract Bisimulation



$$
\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \ldots \\
\downarrow & & \downarrow & & \downarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3} & \ldots
\end{array}
$$

with $v_i \in R_i$ for all i.

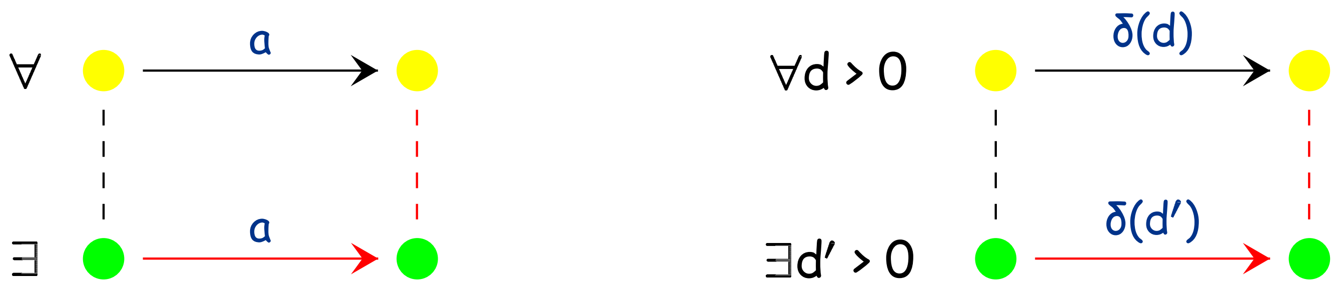
Time-Abstract Bisimulation



$$
\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \ldots \\
\updownarrow & & \updownarrow & & \updownarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3} & \ldots
\end{array}
$$

with $v_i \in R_i$ for all i.

Time-Abstract Bisimulation



$$\begin{array}{ccccccc}
(\ell_0, v_0) & \xrightarrow{a_1, t_1} & (\ell_1, v_1) & \xrightarrow{a_2, t_2} & (\ell_2, v_2) & \xrightarrow{a_3, t_3} & \ldots \\
\updownarrow & & \updownarrow & & \updownarrow & & \\
(\ell_0, R_0) & \xrightarrow{a_1} & (\ell_1, R_1) & \xrightarrow{a_2} & (\ell_2, R_2) & \xrightarrow{a_3} & \ldots
\end{array}$$

with $v_i \in R_i$ for all i.

Remark: We can not check real-time properties with a time-abstract bisimulation.
We need to add clocks for the formula we want to check.

The Region Automaton

timed automaton $\otimes$ region abstraction

$\ell \xrightarrow{g,a,C:=0} \ell'$ is transformed into:

$(\ell, R) \xrightarrow{a} (\ell', R')$ if there exists $R'' \in \mathrm{Succ}_t^*(R)$ s.t.

- ✔ $R'' \subseteq g$
- ✔ $[C \leftarrow 0]R'' \subseteq R'$

The Region Automaton

timed automaton $\otimes$ region abstraction

$\ell \xrightarrow{g,a,C:=0} \ell'$ is transformed into:

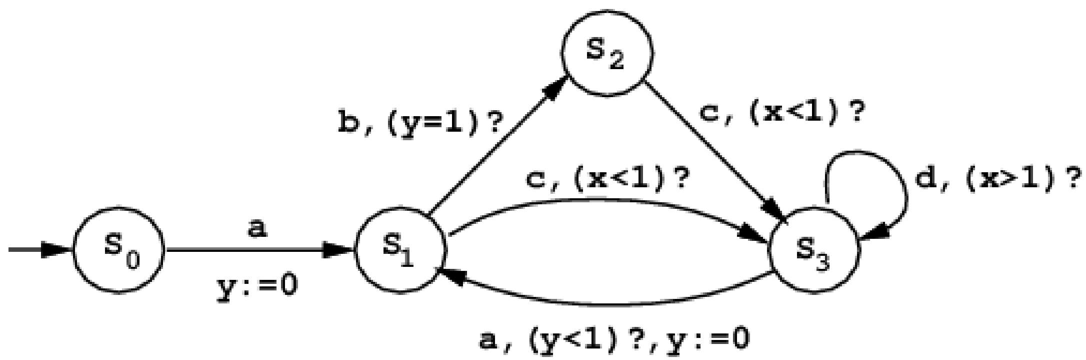
$(\ell, R) \xrightarrow{a} (\ell', R')$ if there exists $R'' \in \mathrm{Succ}_t^*(R)$ s.t.

- ✔ $R'' \subseteq g$
- ✔ $[C \leftarrow 0]R'' \subseteq R'$

$\mathcal{L}$(reg. aut.) = UNTIME($\mathcal{L}$(timed aut.))

where $\mathrm{UNTIME}((a_1, t_1)(a_2, t_2)\ldots) = a_1 a_2 \ldots$

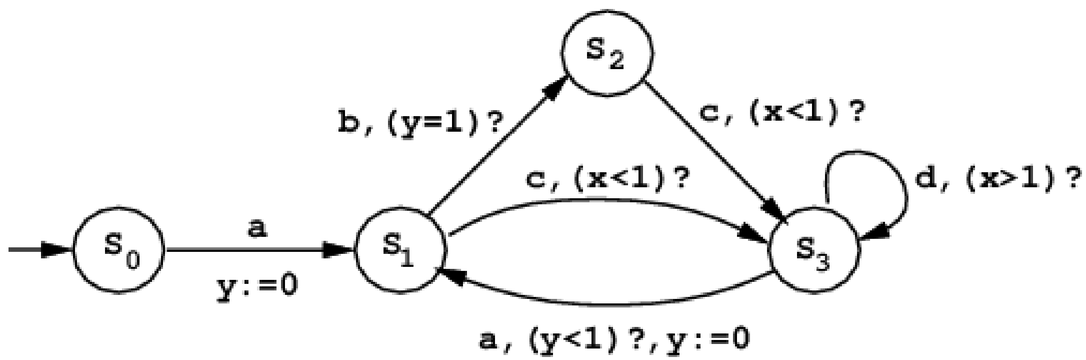
An Example [Alur & Dill 90's]



Questions:

- ✔ Is s_3 reachable?

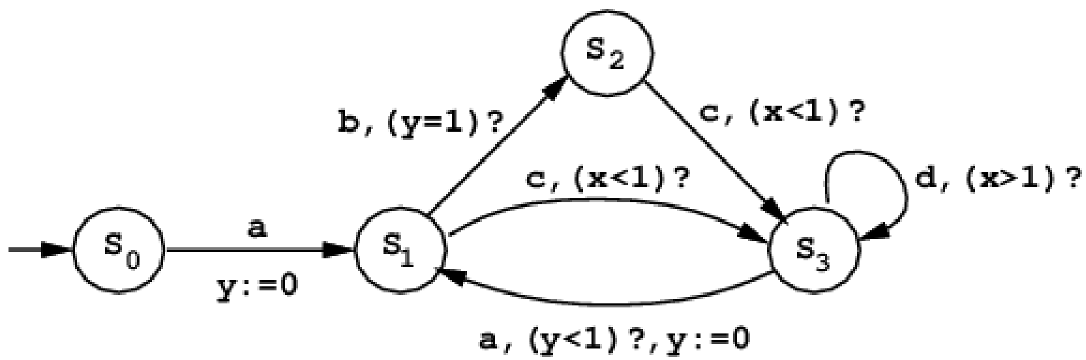
An Example [Alur & Dill 90's]



Questions:

- ✔ Is s_3 reachable?
- ✔ If s_2 is a repeated state (for a Büchi condition), what is the language recognized by this automaton?

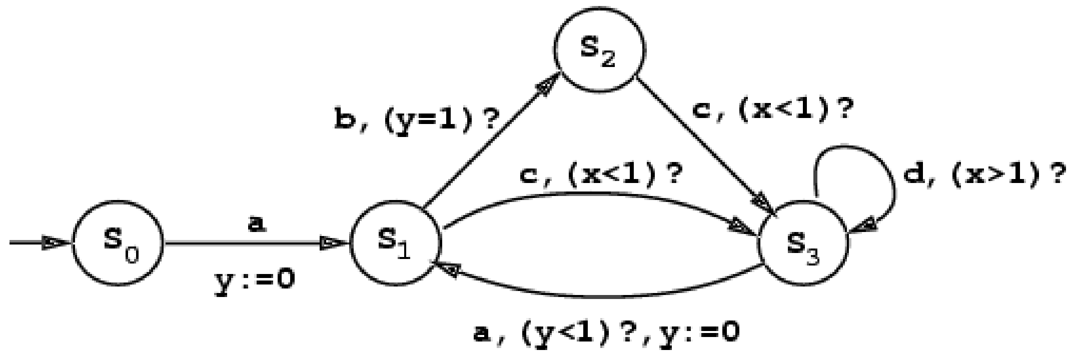
An Example [Alur & Dill 90's]

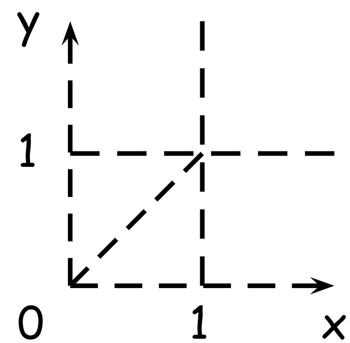


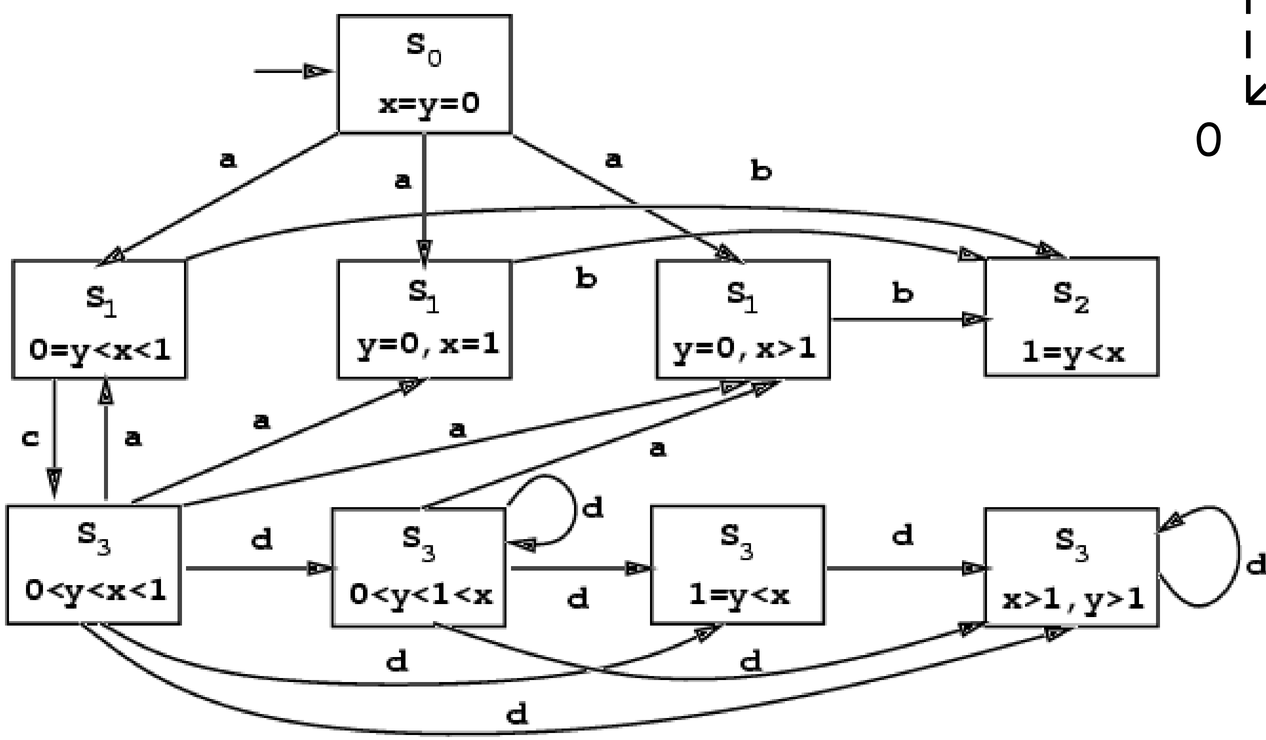
Questions:

- ✔ Is s_3 reachable?
- ✔ If s_2 is a repeated state (for a Büchi condition), what is the language recognized by this automaton?
- ✔ Is there an infinite timed word accepted by this automaton with no d?

An Example [Alur & Dill 90's]







Main Basis Result

Theorem **[Alur & Dill 90's]** Reachability is decidable for TA.

Main Basis Result

¡ The size of the region graph is in $\mathcal{O}(|X|!.2^{|X|})$!

Theorem **[Alur & Dill 90's]** Reachability is decidable for TA.
It is even PSPACE-complete.

PSPACE-Easyness

✔ **One configuration:** a discrete location + a region

PSPACE-Easyness

- ✔ **One configuration:** a discrete location + a region
  - a discrete location: log-space

PSPACE-Easyness

✔ **One configuration:** a discrete location + a region

- a discrete location: log-space
- a region:
  - an interval for each clock
  - an interval for each pair of clocks

PSPACE-Easyness

- **One configuration:** a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks

➜ needs polynomial space

PSPACE-Easyness

- ✔ **One configuration:** a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks

➜ needs polynomial space

- ✔ By guessing a path: needs only to store two configurations

PSPACE-Easyness

- ✔ **One configuration:** a discrete location + a region
  - a discrete location: log-space
  - a region:
    - an interval for each clock
    - an interval for each pair of clocks

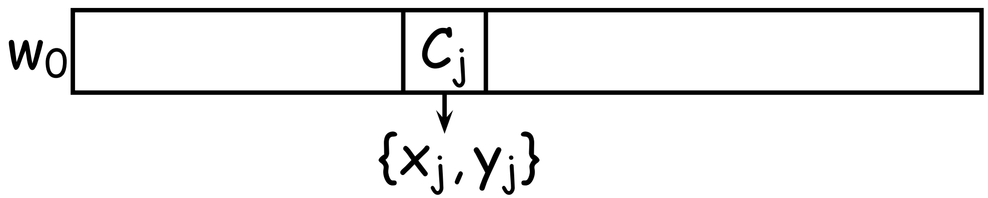
➜ needs polynomial space

- ✔ By guessing a path: needs only to store two configurations

➜ in NPSPACE, thus in PSPACE

PSPACE-Hardness

$\left.\begin{array}{c}\mathcal{M} \text{ LBTM} \\ w_0 \in \{a,b\}^*\end{array}\right\} \rightsquigarrow A_{\mathcal{M},w_0}$ s.t. $\mathcal{M}$ accepts w_0 iff the final state of $A_{\mathcal{M},w_0}$ is reachable



C_j contains a "a" iff $x_j = y_j$

C_j contains a "b" iff $x_j < y_j$

(these conditions are invariant by time elapsing)

➜ proof taken in **[Aceto & Laroussinie 2002]**

PSPACE-Hardness (cont.)

If $q \xrightarrow{\alpha,\alpha',\delta} q'$ is a transition of $\mathcal{M}$, then for each position i of the tape, we have a transition

$$(q,i) \xrightarrow{g,r:=0} (q',i')$$

where:

- g is $x_i = y_i$ (resp. $x_i < y_i$) if $\alpha = a$ (resp. $\alpha = b$)
- $r = \{x_i, y_i\}$ (resp. $r = \{x_i\}$) if $\alpha = a$ (resp. $\alpha = b$)
- $i' = i + 1$ (resp. $i' = i - 1$) if δ is right and $i < n$ (resp. left)

Enforcing time elapsing: on each transition, add the condition $t = 1$ and clock t is reset.

Initialization: $\text{init} \xrightarrow{t=1, r_0:=0} (q_0, 1)$ where $r_0 = \{x_i \mid w_0[i] = b\} \cup \{t\}$

Termination: $(q_f, i) \longrightarrow \text{end}$

Tighter Results

- ✔ Reachability in TA is PSPACE-complete even if the time is discrete!
  [Alur & Dill 90's]

- ✔ Reachability in TA with integer constants in $\{1, 2\}$ is PSPACE-complete.
  [Courcoubetis & Yannakakis 1992]

- ✔ Reachability in TA with 3 clocks is PSPACE-complete.
  [Courcoubetis & Yannakakis 1992]

- ✔ Reachability in TA with 1 clock is NLOGSPACE-complete.
  [Laroussinie, Markey & Schnoebelen 2004]

- ✔ Reachability in TA with 2 clocks is NP-hard.
  [Laroussinie, Markey & Schnoebelen 2004]

Verification of TCTL

How to check that $\mathcal{A} \models \varphi$?

Verification of TCTL

How to check that $\mathcal{A} \models \varphi$?

✔ Add the clocks of φ, and consider the new bigger region automaton

"Two equivalent states satisfy the same subformulas of φ"

Verification of TCTL

How to check that $\mathcal{A} \models \varphi$?

- Add the clocks of φ, and consider the new bigger region automaton

  "Two equivalent states satisfy the same subformulas of φ"

- Label all the states of the region automaton with the subformulas of φ

Verification of TCTL

How to check that $\mathcal{A} \models \varphi$?

- ✔ Add the clocks of φ, and consider the new bigger region automaton

  "Two equivalent states satisfy the same subformulas of φ"

- ✔ Label all the states of the region automaton with the subformulas of φ

Theorem [Alur, Courcoubetis & Dill 1990]
Model-checking of TCTL is PSPACE-complete for TA.

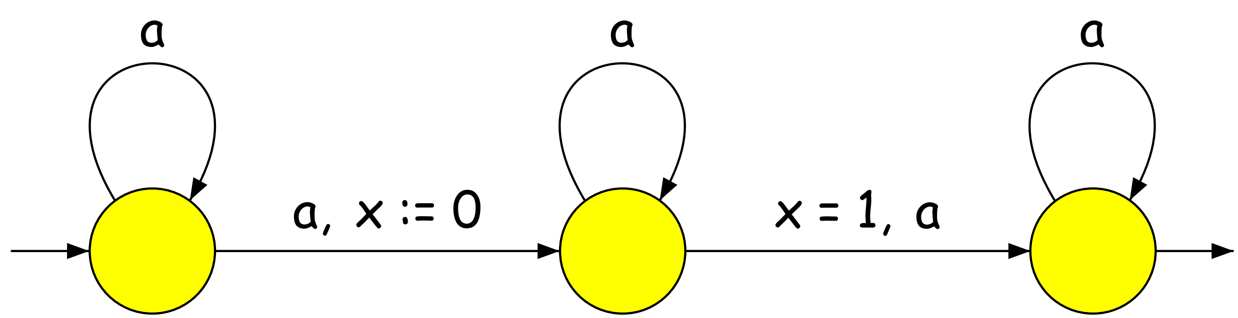
A Model Not Far From Undecidability

- ✔ Universality is undecidable **[Alur & Dill 90's]**
- ✔ Inclusion is undecidable **[Alur & Dill 90's]**
- ✔ Satisfiability of TCTL is undecidable **[Alur & Dill 90's]**
- ✔ Determinizability is undecidable **[Tripakis 2003]**
- ✔ ...

A Model Not Far From Undecidability

- ✔ Universality is undecidable [Alur & Dill 90's]
- ✔ Inclusion is undecidable [Alur & Dill 90's]
- ✔ Satisfiability of TCTL is undecidable [Alur & Dill 90's]
- ✔ Determinizability is undecidable [Tripakis 2003]
- ✔ ...

An example of non-deterministic TA:



A Model Not Far From Undecidability

- ✔ Universality is undecidable **[Alur & Dill 90's]**
- ✔ Inclusion is undecidable **[Alur & Dill 90's]**
- ✔ Satisfiability of TCTL is undecidable **[Alur & Dill 90's]**
- ✔ Determinizability is undecidable **[Tripakis 2003]**
- ✔ ...

Extending timed automata:

- ✔ add silent actions **[Bérard, Diekert, Gastin, Petit 1998]** Decidable!

A Model Not Far From Undecidability

- ✔ Universality is undecidable **[Alur & Dill 90's]**
- ✔ Inclusion is undecidable **[Alur & Dill 90's]**
- ✔ Satisfiability of TCTL is undecidable **[Alur & Dill 90's]**
- ✔ Determinizability is undecidable **[Tripakis 2003]**
- ✔ ...

Extending timed automata:

- ✔ add silent actions **[Bérard, Diekert, Gastin, Petit 1998]** Decidable!
- ✔ add guards, e.g. $x + y \bowtie c$ **[Bérard, Dufourd 2000]** Undecidable!

A Model Not Far From Undecidability

- ✔ Universality is undecidable **[Alur & Dill 90's]**
- ✔ Inclusion is undecidable **[Alur & Dill 90's]**
- ✔ Satisfiability of TCTL is undecidable **[Alur & Dill 90's]**
- ✔ Determinizability is undecidable **[Tripakis 2003]**
- ✔ ...

Extending timed automata:

- ✔ add silent actions **[Bérard, Diekert, Gastin, Petit 1998]** Decidable!
- ✔ add guards, e.g. $x + y \bowtie c$ **[Bérard, Dufourd 2000]** Undecidable!
- ✔ add operations on clocks, e.g. $x := y + 1$ Decidable/Undecidable
  [Bouyer, Dufourd, Fleury, Petit 2000]

A Model Not Far From Undecidability

- ✔ Universality is undecidable [Alur & Dill 90's]
- ✔ Inclusion is undecidable [Alur & Dill 90's]
- ✔ Satisfiability of TCTL is undecidable [Alur & Dill 90's]
- ✔ Determinizability is undecidable [Tripakis 2003]
- ✔ ...

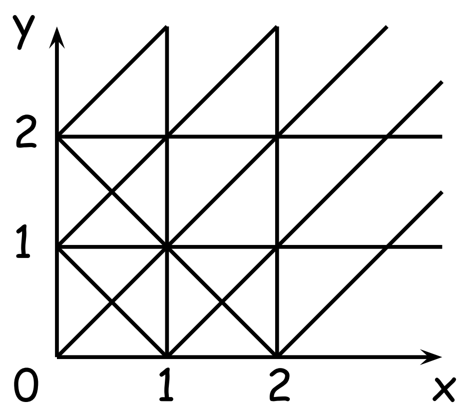
Extending timed automata:

- ✔ add silent actions [Bérard, Diekert, Gastin, Petit 1998] Decidable!
- ✔ add guards, e.g. $x + y \bowtie c$ [Bérard, Dufourd 2000] Undecidable!
- ✔ add operations on clocks, e.g. $x := y + 1$ Decidable/Undecidable
  [Bouyer, Dufourd, Fleury, Petit 2000]
- ✔ more general variables, e.g. hybrid systems Undecidable!
  [Alur, Courcoubetis, Henzinger, Ho 1993]
  [Henzinger 1996] [Henzinger, Kopke, Puri, Varaiya 1998]
  ...

Adding Constraints of the Form $x + y \sim c$

$x + y \sim c$ and $x \sim c$ [Bérard,Dufourd 2000]

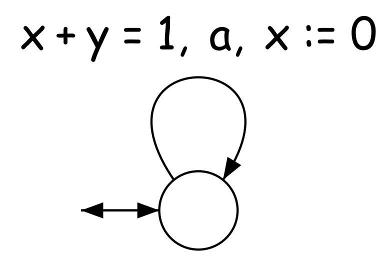
✔ **Decidability:** - for two clocks, decidable using the abstraction



- for four clocks (or more), undecidable!

✔ **Expressiveness:** more expressive! (even using two clocks)

$$\{(a^n, t_1 \ldots t_n) \mid n \geq 1 \text{ and } t_i = 1 - \tfrac{1}{2^i}\}$$



The Two-Counter Machine

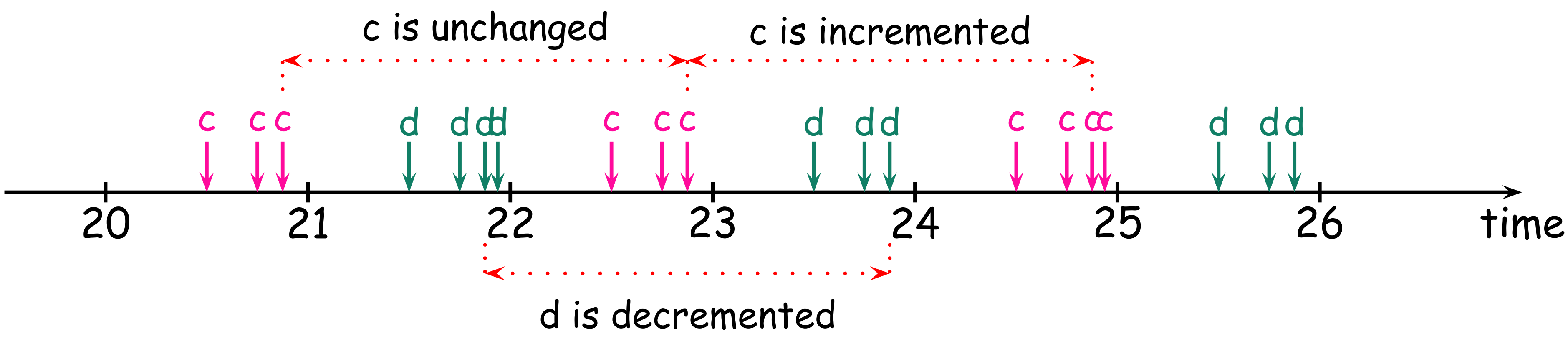
Definition. A two-counter machine is a finite set of instructions over two counters (x and y):

- Incrementation:
  (p): $x := x + 1$; **goto (q)**

- Decrementation:
  (p): if $x > 0$ **then** $x := x - 1$; **goto (q) else goto (r)**

Theorem. [Minsky 67] The emptiness problem for two counter machines is undecidable.

Undecidability Proof



➜ simulation of
- decrement of d
- increment of c

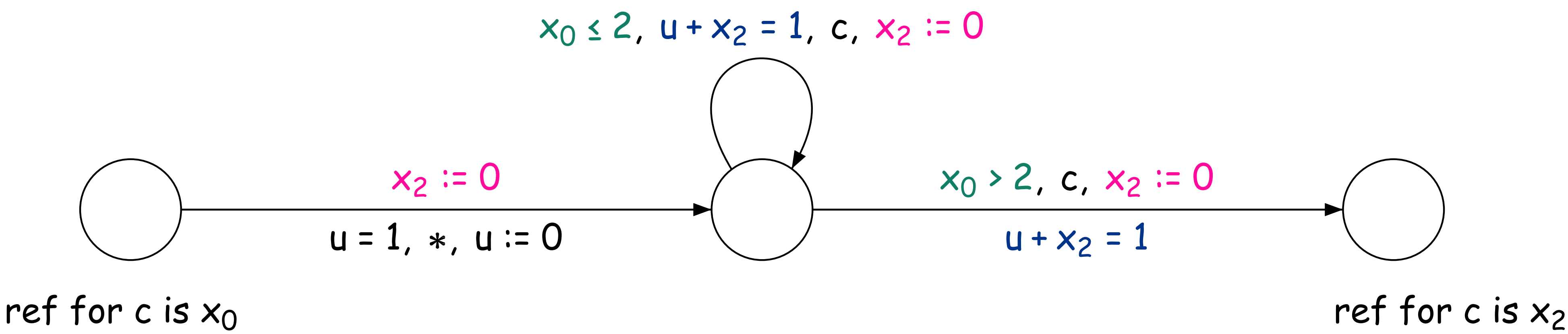
We will use 4 clocks:
- u, "tic" clock (each time unit)
- x_0, x_1, x_2: reference clocks for the two counters

"x_i reference for c" $\equiv$ "the last time x_i has been reset is the last time action c has been performed"

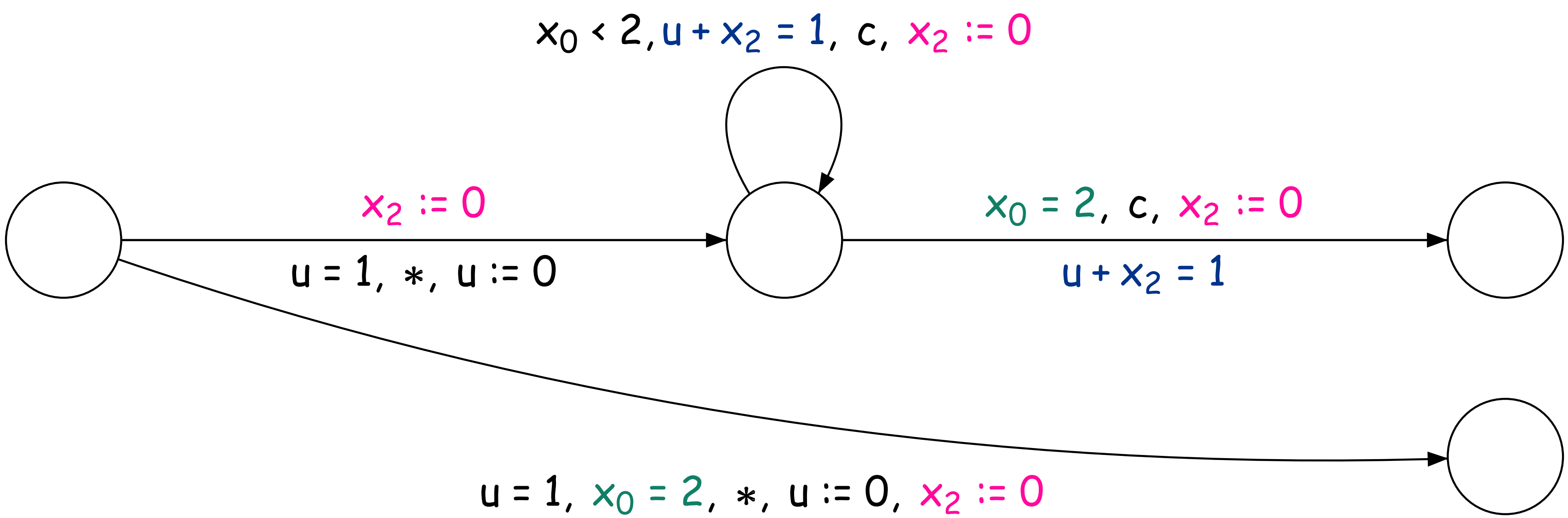
[Bérard, Dufourd 2000]

Undecidability Proof (cont.)

- ✔ **Increment of counter** c:

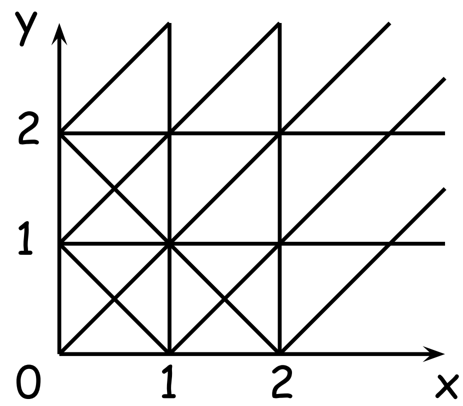


- ✔ **Decrement of counter** c:



Adding Constraints of the Form $x + y \sim c$

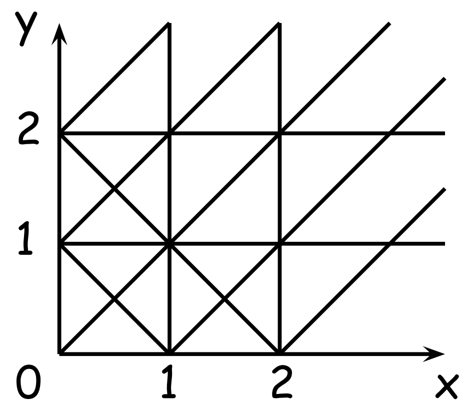
- Two clocks: decidable! using the abstraction



- Four clocks (or more): undecidable!

Adding Constraints of the Form $x + y \sim c$

- ✔ Two clocks: decidable! using the abstraction



- ✔ Three clocks: **open question**

- ✔ Four clocks (or more): undecidable!

Networks of TA, discussion

Complexity of Model-Checking

	Kripke structures S	Timed automata A
Reachability	NLOGSPACE-complete	
CTL/TCTL	P-complete	
AF-μ-calc./$L_{\mu,\nu}$	P-complete	
full μ-calc./$L^{+}_{\mu,\nu}$	UP $\cap$ co-UP	

Complexity of Model-Checking

	Kripke structures S	Timed automata A
Reachability	NLOGSPACE-complete	PSPACE-complete
CTL/TCTL	P-complete	PSPACE-complete
AF-μ-calc./$L_{\mu,\nu}$	P-complete	EXPTIME-complete
full μ-calc./$L^{+}_{\mu,\nu}$	UP ∩ co-UP	EXPTIME-complete

Timing constraints induce a complexity blowup !

[Alur 1991, Alur Henzinger 1994, Alur Courcoubetis Dill 1993, Aceto Laroussinie 1999]

Complexity of Model-Checking

	Kripke structures S	Timed automata A or $(S_1 \parallel \ldots \parallel S_n)$
Reachability	NLOGSPACE-complete	PSPACE-complete
CTL/TCTL	P-complete	PSPACE-complete
AF-μ-calc./$L_{\mu,\nu}$	P-complete	EXPTIME-complete
full μ-calc./$L^{+}_{\mu,\nu}$	UP $\cap$ co-UP	EXPTIME-complete

Timing constraints induce a complexity blowup !

From a complexity point of view, adding clocks = adding components !

[Alur 1991, Alur Henzinger 1994, Alur Courcoubetis Dill 1993, Aceto Laroussinie 1999]

Complexity of Model-Checking

	Kripke structures S	Timed automata A or $(S_1 \parallel \ldots \parallel S_n)$ or $(A_1 \parallel \ldots \parallel A_n)$
Reachability	NLOGSPACE-complete	PSPACE-complete
CTL/TCTL	P-complete	PSPACE-complete
AF-μ-calc./$L_{\mu,\nu}$	P-complete	EXPTIME-complete
full μ-calc./$L^{+}_{\mu,\nu}$	UP $\cap$ co-UP	EXPTIME-complete

Timing constraints induce a complexity blowup !

From a complexity point of view, adding clocks = adding components !

[Alur 1991, Alur Henzinger 1994, Alur Courcoubetis Dill 1993, Aceto Laroussinie 1999]

State Explosion Problem

- ✔ due to parallel composition
- ✔ due to timing constraints

State Explosion Problem

- ✔ due to parallel composition
- ✔ due to timing constraints

From a complexity point of view:

no double complexity gap!

State Explosion Problem

- ✔ due to parallel composition
- ✔ due to timing constraints

From a complexity point of view: ☺

no double complexity gap!

State Explosion Problem

- ✔ due to parallel composition
- ✔ due to timing constraints

From a complexity point of view: ☺

no double complexity gap!

In practice:

- ✔ BDD-like techniques try to avoid discrete state explosion problem in untimed systems ➜ SMV verifies very large systems

State Explosion Problem

- ✔ due to parallel composition
- ✔ due to timing constraints

From a complexity point of view: ☺

no double complexity gap!

In practice:

- ✔ BDD-like techniques try to avoid discrete state explosion problem in untimed systems → SMV verifies very large systems
- ✔ **Timed systems:** problems to deal with both explosions. Much smaller systems can be analyzed in practice.

State Explosion Problem

- ✔ due to parallel composition
- ✔ due to timing constraints

From a complexity point of view: ☺

no double complexity gap!

In practice: ☹

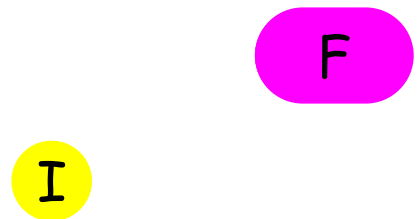
- ✔ BDD-like techniques try to avoid discrete state explosion problem in untimed systems ➜ SMV verifies very large systems
- ✔ **Timed systems:** problems to deal with both explosions. Much smaller systems can be analyzed in practice.

Verification Methods

- ✔ on-the-fly backward algorithms
- ✔ on-the-fly forward algorithms
- ✔ compositional algorithms

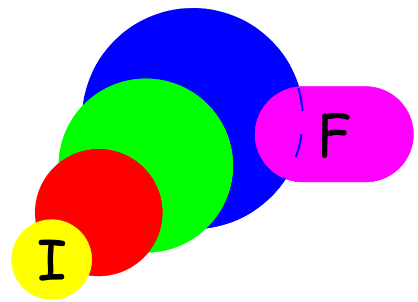
Reachability Analysis

✔ **forward analysis algorithm:**
compute the successors of initial configurations



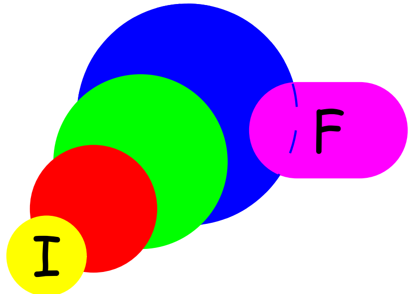
Reachability Analysis

- **forward analysis algorithm:**
  compute the successors of initial configurations

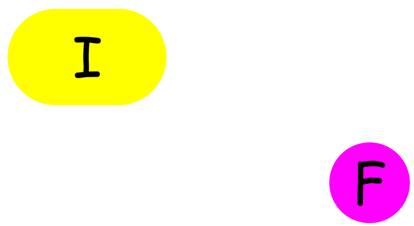


Reachability Analysis

- ✔ **forward analysis algorithm:**
  compute the successors of initial configurations

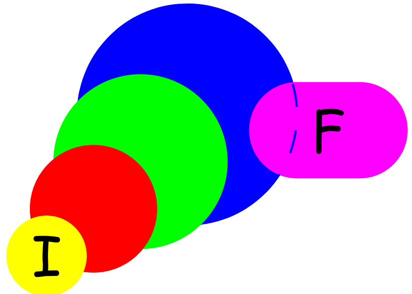


- ✔ **backward analysis algorithm:**
  compute the predecessors of final configurations

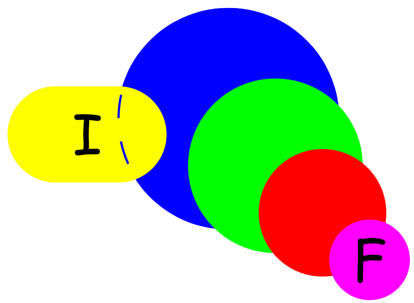


Reachability Analysis

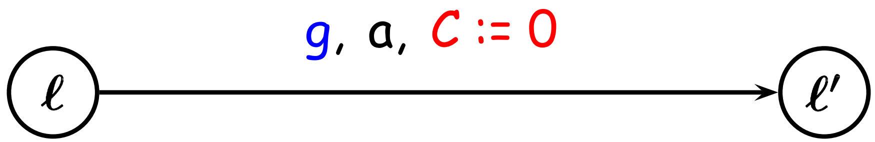
- ✔ **forward analysis algorithm:**
  compute the successors of initial configurations



- ✔ **backward analysis algorithm:**
  compute the predecessors of final configurations



Note on the Backward Analysis

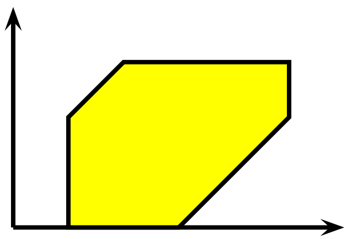


$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\qquad$ Z

Note on the Backward Analysis

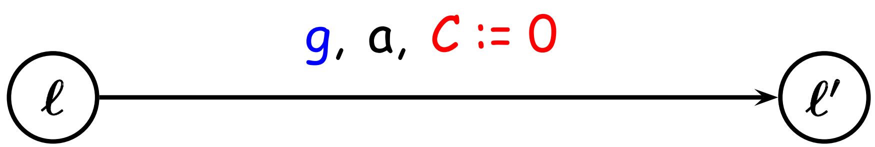
$$\ell \xrightarrow{g,\ a,\ C := 0} \ell'$$

$$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g} \qquad\qquad Z$$

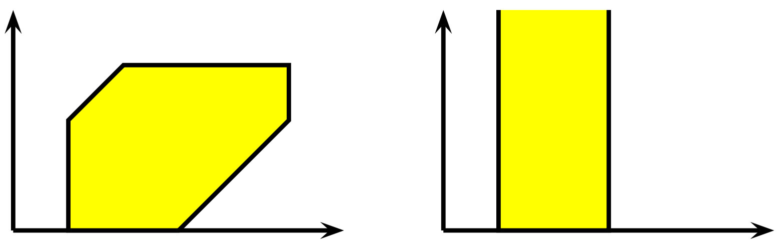


Z

Note on the Backward Analysis



$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ Z

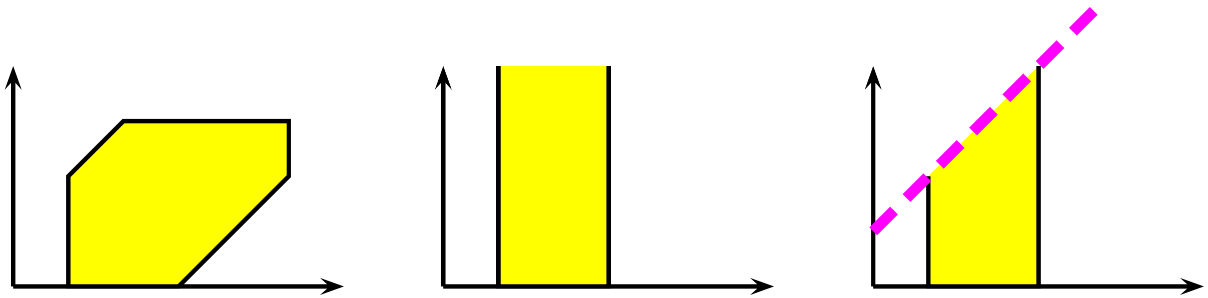


Z $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$

Note on the Backward Analysis

$$\ell \xrightarrow{g,\ a,\ C := 0} \ell'$$

$$[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g \longleftarrow Z$$

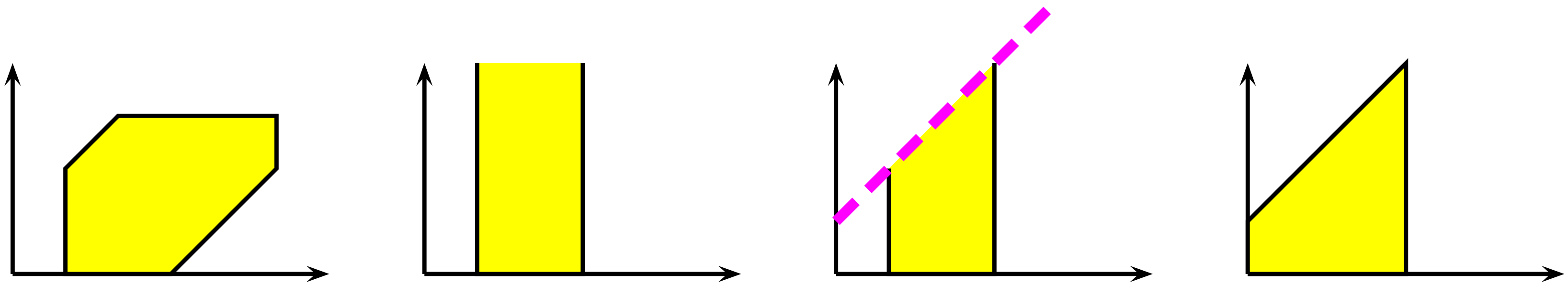


Z $\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$

Note on the Backward Analysis

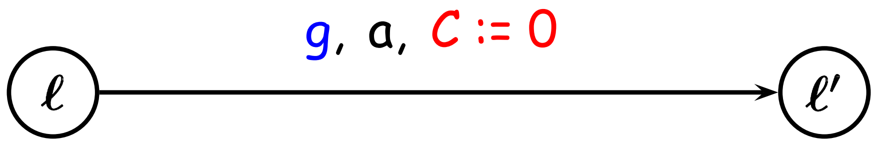
$$\ell \xrightarrow{\;g,\ a,\ C := 0\;} \ell'$$

$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\qquad$ Z

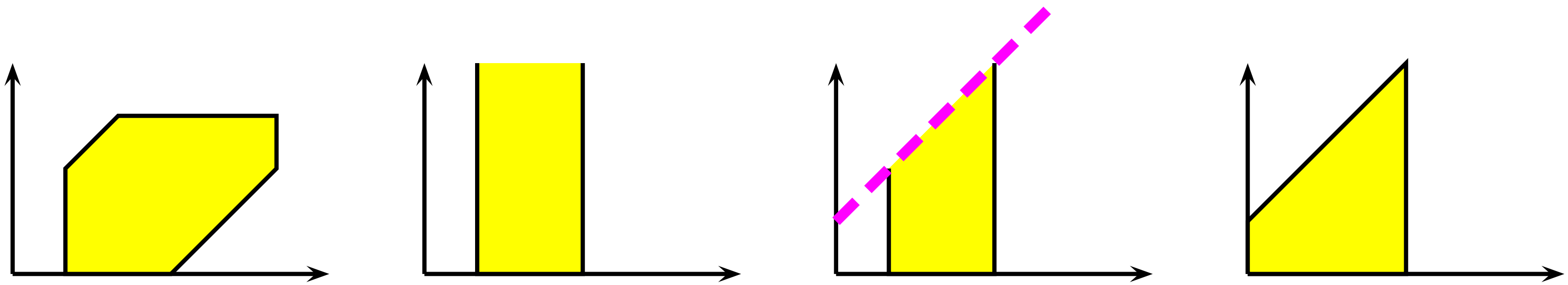


Z $\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$ $\qquad$ $\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$

Note on the Backward Analysis



$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\qquad\qquad$ Z



Z $\qquad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$ $\qquad$ $\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$

The exact backward computation terminates and is correct!

Note on the Backward Analysis (cont.)

If $\mathcal{A}$ is a timed automaton, we construct its corresponding set of regions.

Because of the bisimulation property, we get that:

"Every set of valuations which is computed along the backward computation is a finite union of regions"

Note on the Backward Analysis (cont.)

If $\mathcal{A}$ is a timed automaton, we construct its corresponding set of regions.

Because of the bisimulation property, we get that:

"Every set of valuations which is computed along the backward computation is a finite union of regions"

Let R be a region. Assume:

- ✔ $v \in \overleftarrow{R}$ (for ex. $v + t \in R$)
- ✔ $v' \equiv_{reg.} v$

There exists t' s.t. $v' + t' \equiv_{reg.} v + t$, which implies that $v' + t' \in R$ and thus $v' \in \overleftarrow{R}$.

Note on the Backward Analysis (cont.)

If $\mathcal{A}$ is a timed automaton, we construct its corresponding set of regions.

Because of the bisimulation property, we get that:

"Every set of valuations which is computed along the backward computation is a finite union of regions"

But, the backward computation is not so nice, when also dealing with integer variables...

$$i := j.k + \ell.m$$

Remark: Verification of TCTL

For checking $\mathcal{S} \models \varphi$:

- for all subformulas ψ of φ, compute the states $[\psi]$ satisfying ψ
- can be done using backward computations, f.ex.

$$\mathrm{Pre}[\psi](\varphi) = \{v \mid \exists \delta \text{ s.t. } v + \delta \in [\varphi] \wedge \forall 0 \leq \delta' \leq \delta,\ v + \delta' \in [\psi]\}$$

- as previously, everything computed is a finite union of regions...

[Henzinger, Nicollin, Sifakis & Yovine 1994] [Yovine 1998]

Forward Analysis of TA

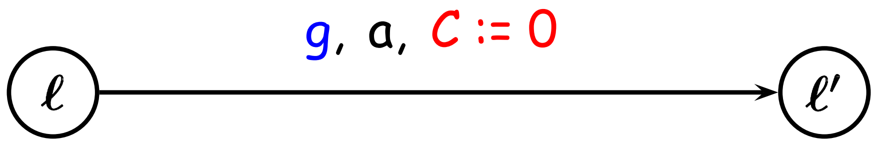
$$\ell \xrightarrow{\quad g,\ a,\ C := 0 \quad} \ell'$$

zones	Z	$[C \leftarrow 0](\overrightarrow{Z} \cap g)$

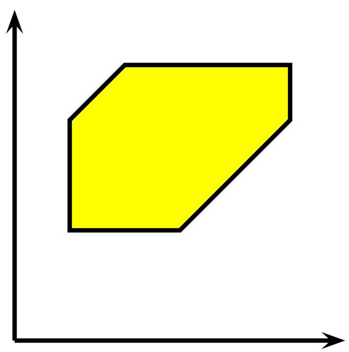
A zone is a set of valuations defined by a clock constraint

$$\varphi \ ::= \ x \sim c \ \mid \ x - y \sim c \ \mid \ \varphi \wedge \varphi$$

Forward Analysis of TA



zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

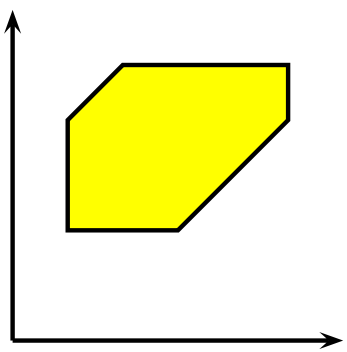


Z

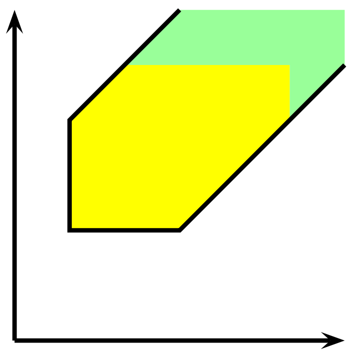
Forward Analysis of TA

$$\ell \xrightarrow{\;g,\ a,\ C := 0\;} \ell'$$

zones $\quad Z \quad\quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

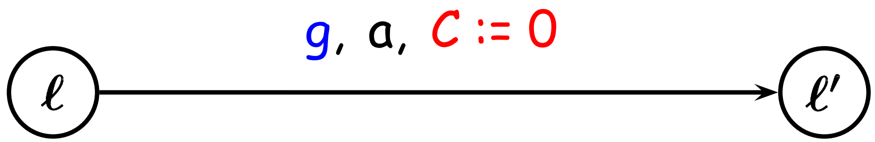


Z

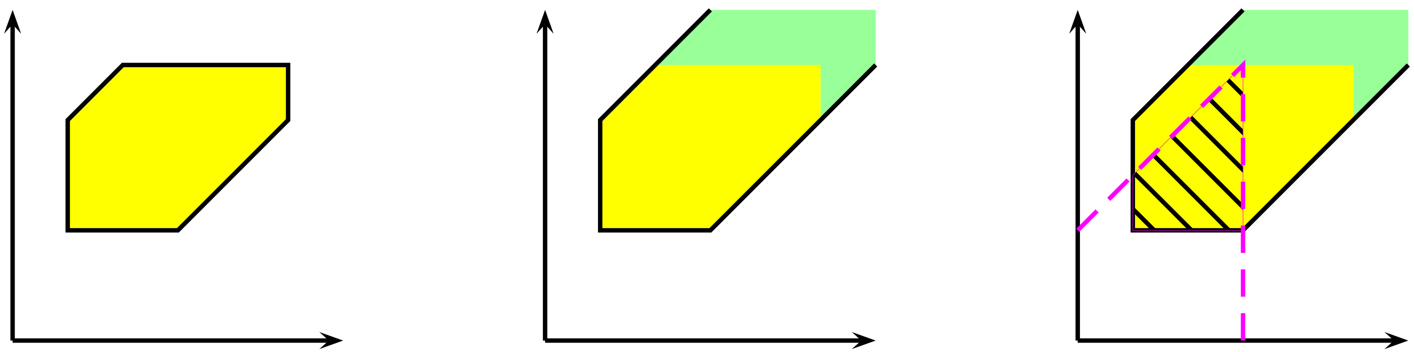


$\overrightarrow{Z}$

Forward Analysis of TA



zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

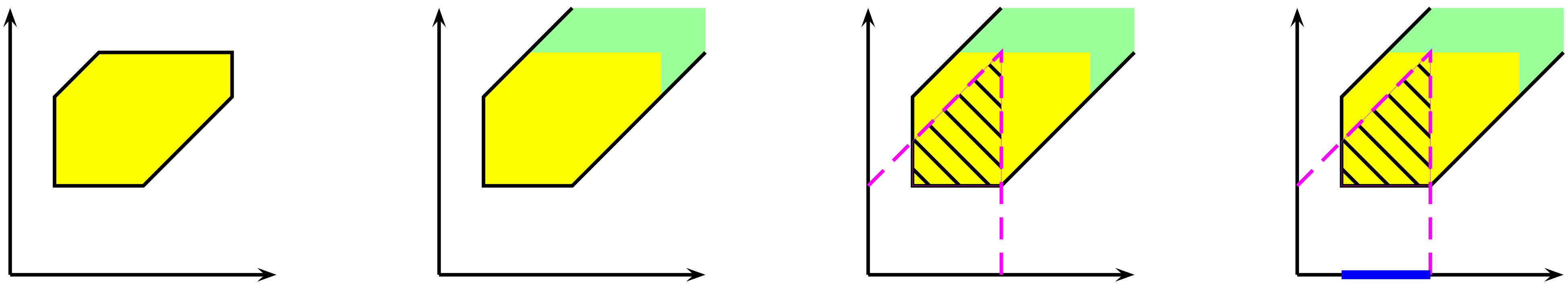


$Z \qquad \overrightarrow{Z} \qquad \overrightarrow{Z} \cap g$

Forward Analysis of TA

$$\ell \xrightarrow{g,\ a,\ C := 0} \ell'$$

zones $\quad Z \quad [C \leftarrow 0](\overrightarrow{Z} \cap g)$

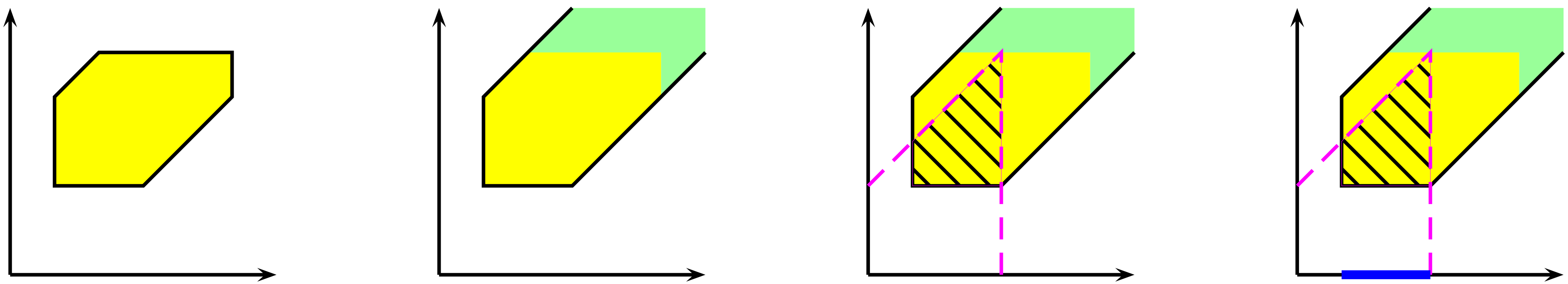


$Z \qquad \overrightarrow{Z} \qquad \overrightarrow{Z} \cap g \qquad [y \leftarrow 0](\overrightarrow{Z} \cap g)$

Forward Analysis of TA

$$\ell \xrightarrow{g,\ a,\ C := 0} \ell'$$

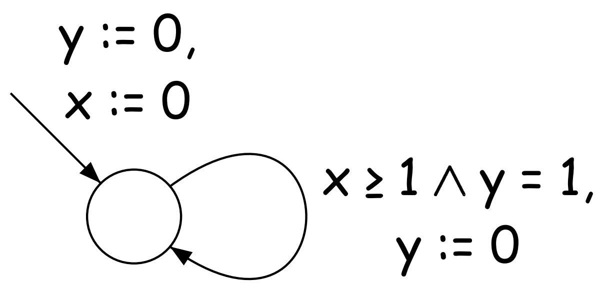
zones	Z	$[C \leftarrow 0](\overrightarrow{Z} \cap g)$

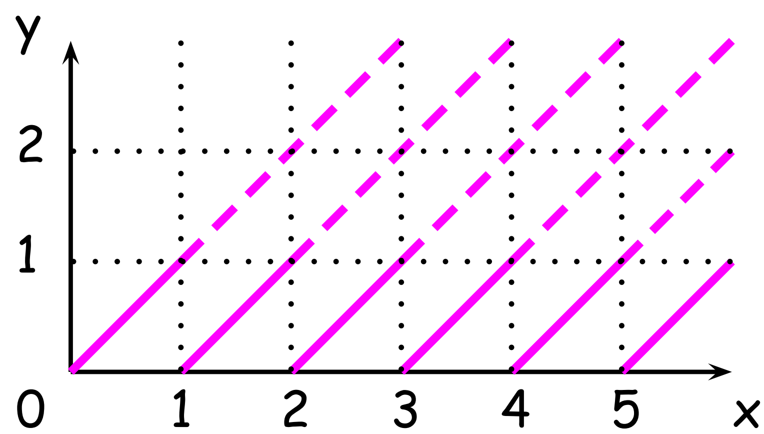


Z $\overrightarrow{Z}$ $\overrightarrow{Z} \cap g$ $[y \leftarrow 0](\overrightarrow{Z} \cap g)$

➜ a termination problem

Non Termination of the Forward Analysis





➜ an infinite number of steps...

"Solutions" to this Problem

(f.ex. in [Larsen,Pettersson,Yi 1997] or in [Daws,Tripakis 1998])

✔ **inclusion checking**: if $Z \subseteq Z'$ and Z' still handled, then we don't need to handle Z

→ correct w.r.t. reachability

...

"Solutions" to this Problem

(f.ex. in [Larsen,Pettersson,Yi 1997] or in [Daws,Tripakis 1998])

- ✔ **inclusion checking**: if $Z \subseteq Z'$ and Z' still handled, then we don't need to handle Z

  ➜ correct w.r.t. reachability

- ✔ **activity**: eliminate redundant clocks [Daws,Yovine 1996]

  ➜ correct w.r.t. reachability

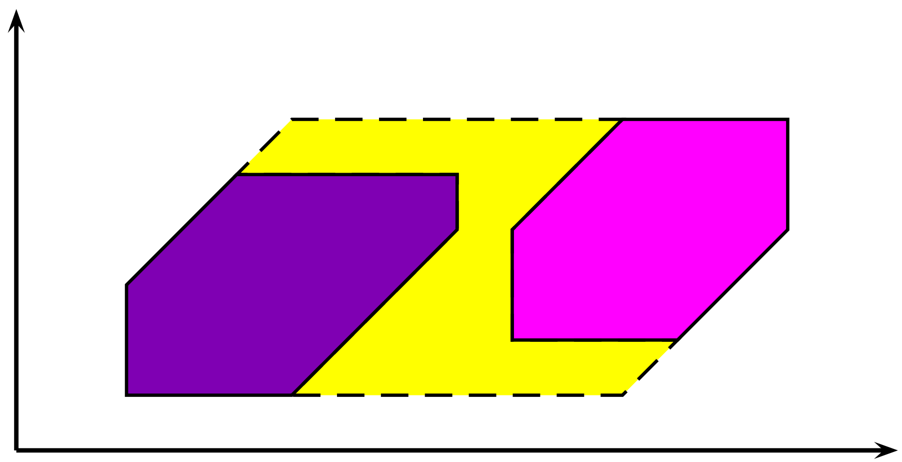
$$q \xrightarrow{g,a,C:=0} q' \quad\Longrightarrow\quad Act(q) = clocks(g) \cup (Act(q') \setminus C)$$

...

"Solutions" to this Problem (cont.)

✔ **convex-hull approximation**: if Z and Z' are computed then we overapproximate using "$Z \sqcup Z'$".

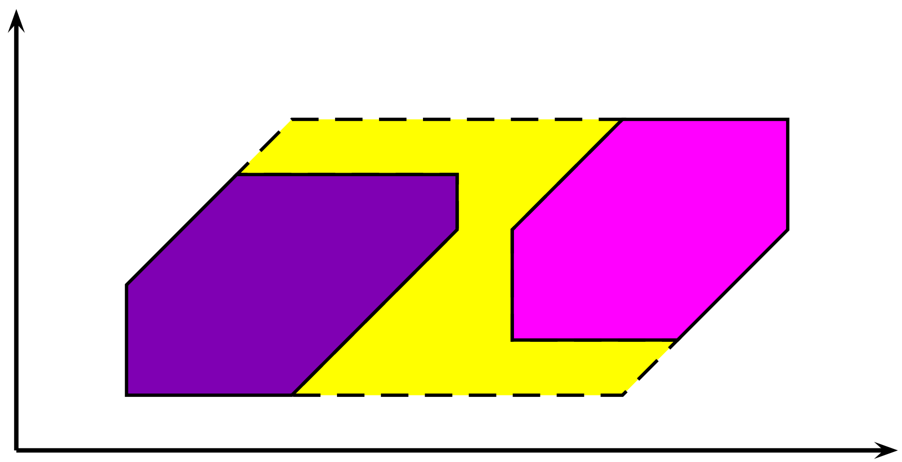
➜ "semi-correct" w.r.t. reachability



“Solutions” to this Problem (cont.)

- ✔ **convex-hull approximation**: if Z and Z' are computed then we overapproximate using “$Z \sqcup Z'$”.

  ➜ “semi-correct” w.r.t. reachability



- ✔ **extrapolation**, a widening operator on zones

The DBM Data Structure

DBM (Difference Bounded Matrice) data structure [Dill 1989]

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

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

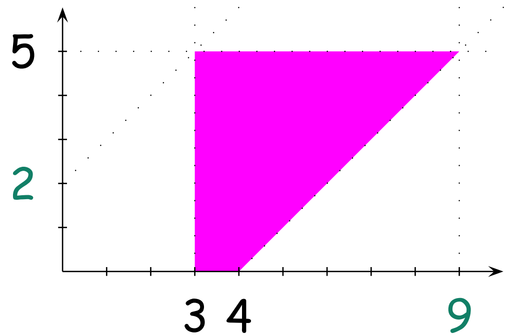
The DBM Data Structure

DBM (Difference Bounded Matrice) data structure [Dill 1989]

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

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

- ✔ Existence of a normal form



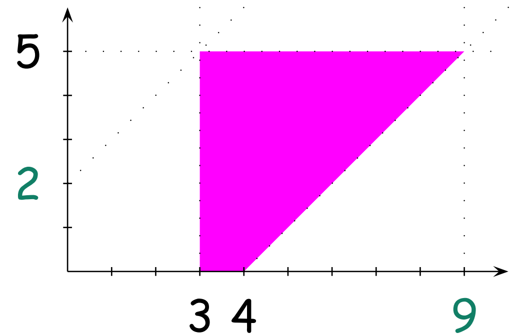
$$\begin{bmatrix} 0 & \mathbf{-3} & 0 \\ 9 & 0 & \mathbf{4} \\ \mathbf{5} & 2 & 0 \end{bmatrix}$$

The DBM Data Structure

DBM (Difference Bounded Matrice) data structure [Dill 1989]

$$(x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4) \qquad \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{bmatrix} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{bmatrix} \end{array}$$

- ✔ Existence of a normal form



$$\begin{bmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{bmatrix}$$

- ✔ All previous operations on zones can be computed using DBMs

The Extrapolation Operator

Fix an integer k (“$*$” represents an integer between $-k$ and $+k$)

$$\begin{bmatrix} * & \boxed{>k} & * \\ * & * & * \\ \boxed{<-k} & * & * \end{bmatrix} \rightsquigarrow \begin{bmatrix} * & \boxed{+\infty} & * \\ * & * & * \\ \boxed{-k} & * & * \end{bmatrix}$$

- ✔ “intuitively”, erase non-relevant constraints

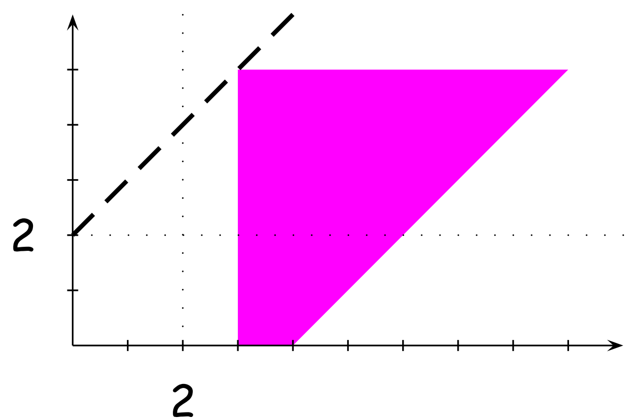
➜ *ensures termination*

The Extrapolation Operator

Fix an integer k ("$*$" represents an integer between $-k$ and $+k$)

$$\begin{bmatrix} * & \boxed{>k} & * \\ * & * & * \\ \boxed{<-k} & * & * \end{bmatrix} \rightsquigarrow \begin{bmatrix} * & \boxed{+\infty} & * \\ * & * & * \\ \boxed{-k} & * & * \end{bmatrix}$$

✔ "intuitively", erase non-relevant constraints



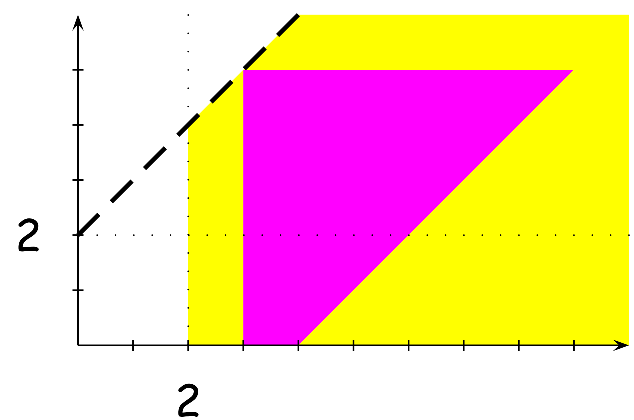
➜ ensures termination

The Extrapolation Operator

Fix an integer k ("$*$" represents an integer between −k and +k)

$$\begin{bmatrix} * & \boxed{>k} & * \\ * & * & * \\ \boxed{<-k} & * & * \end{bmatrix} \rightsquigarrow \begin{bmatrix} * & \boxed{+\infty} & * \\ * & * & * \\ \boxed{-k} & * & * \end{bmatrix}$$

✔ "intuitively", erase non-relevant constraints



➜ ensures termination

Challenge

Propose a **good** constant for the extrapolation:

- ✔ keep the correctness of the forward computation

Solution by the past: maximal constant appearing in the automaton

- ✔ Several correctness proofs can be found
- ✔ Implemented in tools like UPPAAL, KRONOS, RT-SPIN...
- ✔ Successfully used on real-life examples

Challenge

Propose a **good** constant for the extrapolation:

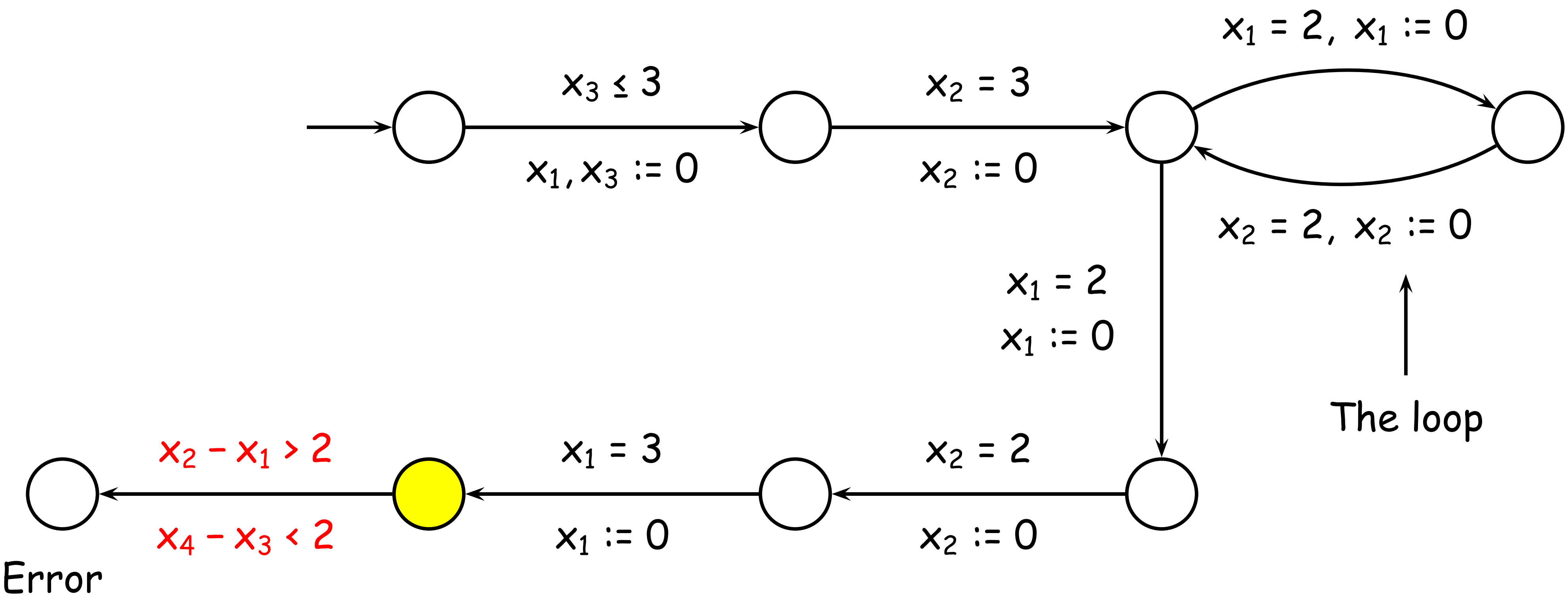
- ✔ keep the correctness of the forward computation

Solution by the past: maximal constant appearing in the automaton

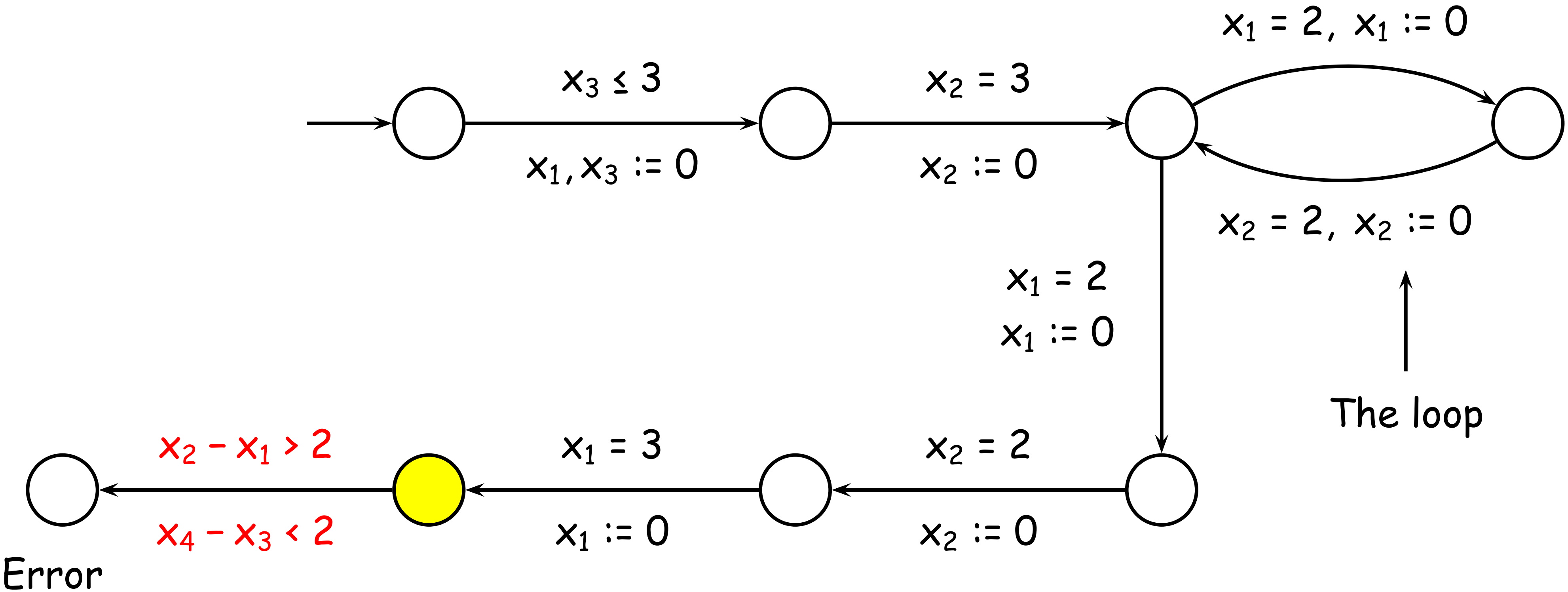
- ✔ Several correctness proofs can be found
- ✔ Implemented in tools like UPPAAL, KRONOS, RT-SPIN...
- ✔ Successfully used on real-life examples

However...

A Problematic Automaton

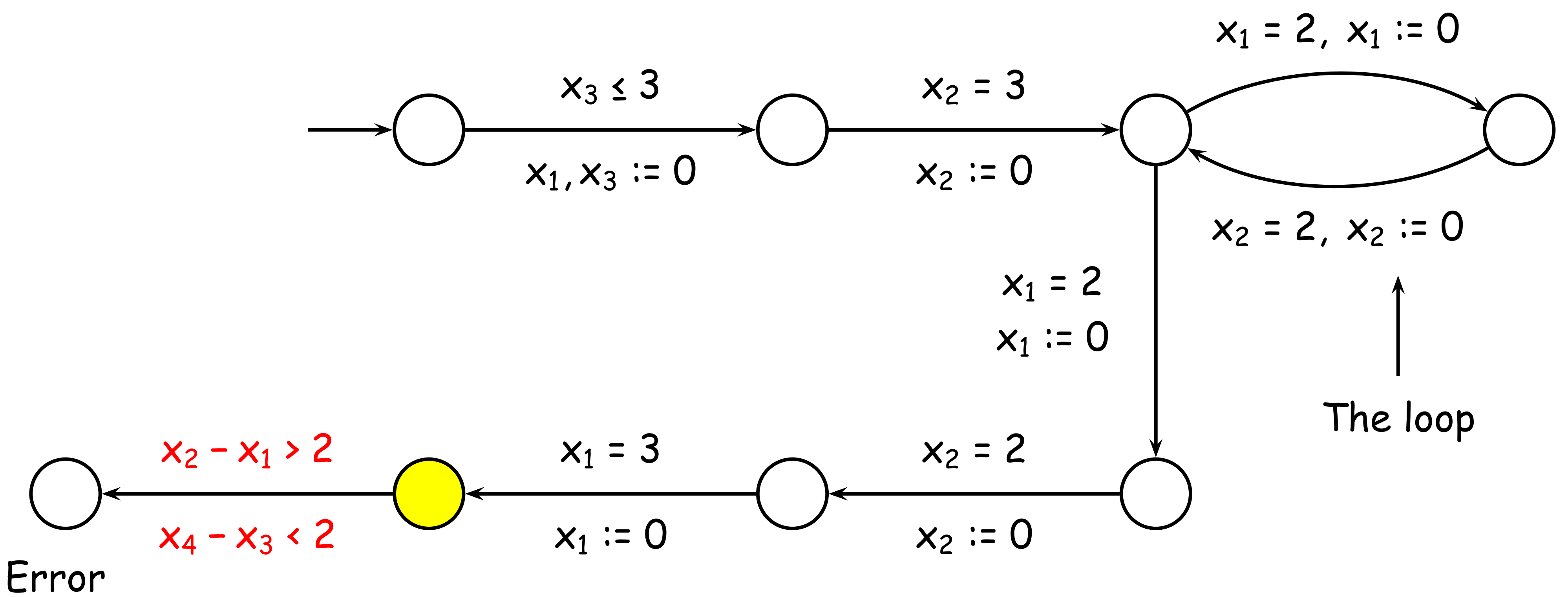


A Problematic Automaton

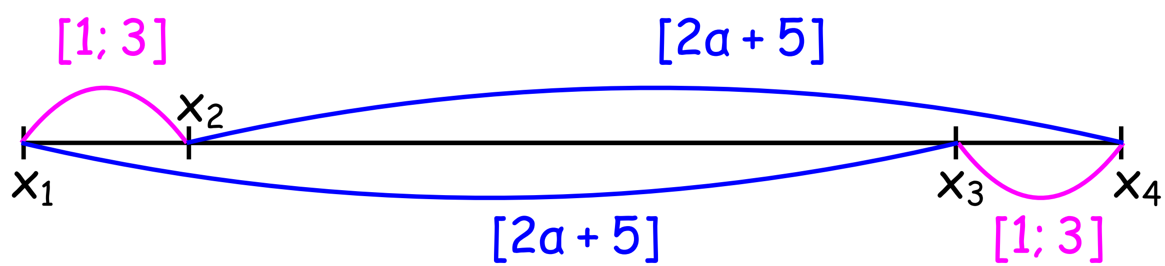


$v(x_1) = 0$
$v(x_2) = d$
$v(x_3) = 2a + 5$
$v(x_4) = 2a + 5 + d$

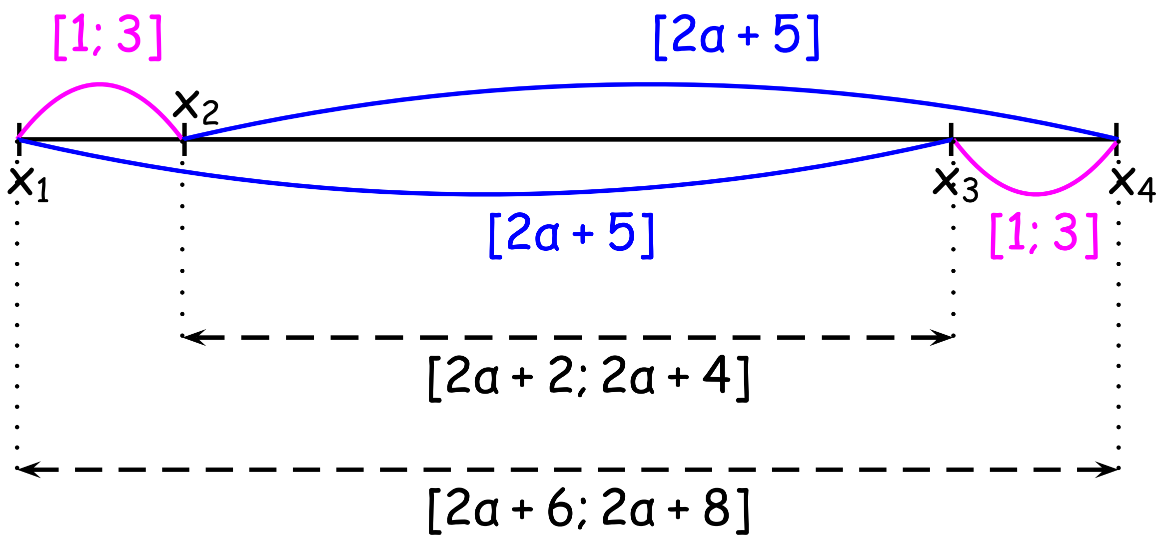
A Problematic Automaton



$v(x_1) = 0$
$v(x_2) = d$
$v(x_3) = 2a + 5$
$v(x_4) = 2a + 5 + d$

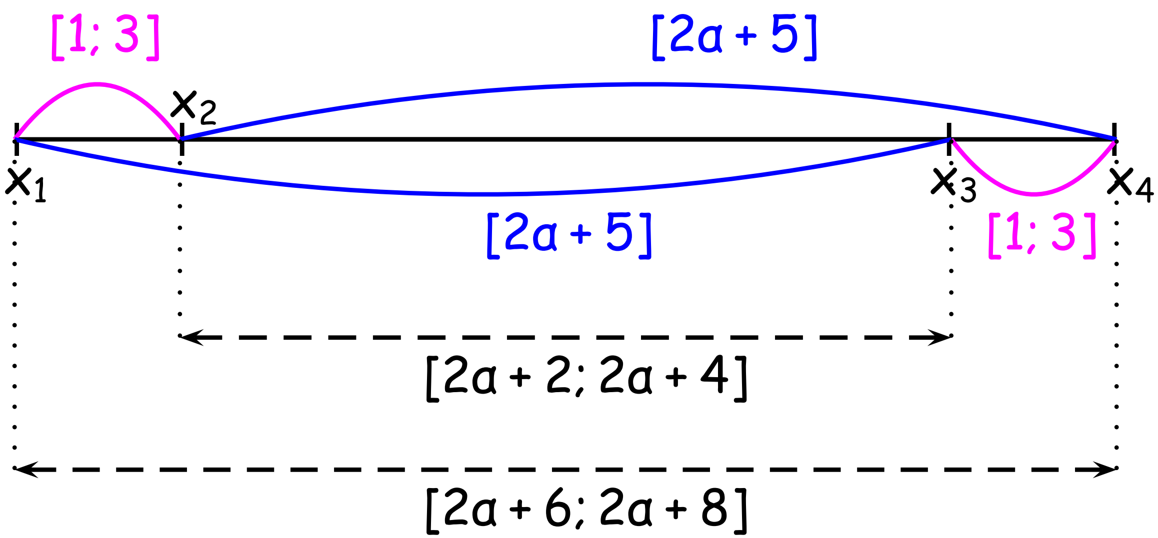


The Problematic Zone



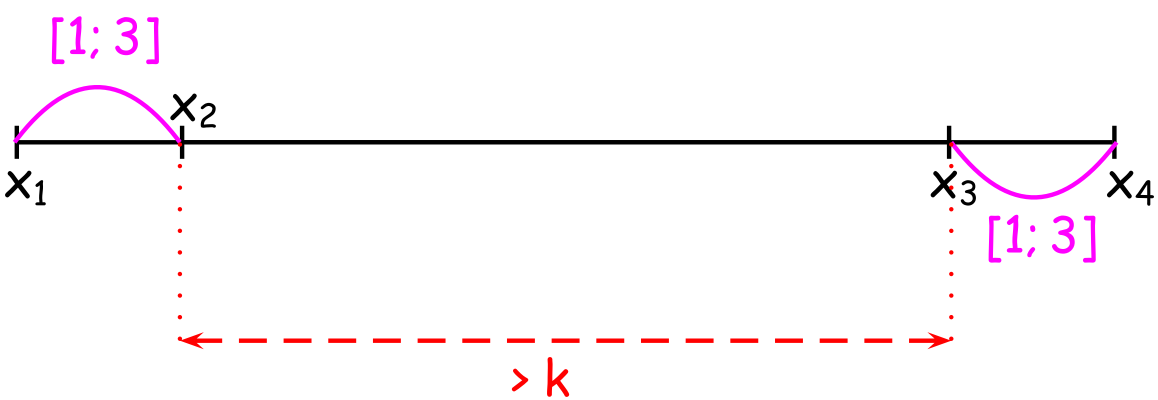
implies $x_1 - x_2 = x_3 - x_4$.

The Problematic Zone



implies $x_1 - x_2 = x_3 - x_4$.

If a is sufficiently large, after extrapolation:



does not imply $x_1 - x_2 = x_3 - x_4$.

General Abstractions

Criteria for a good abstraction operator Abs:

General Abstractions

Criteria for a good abstraction operator Abs:

- ✔ easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone

General Abstractions

Criteria for a good abstraction operator Abs:

- ✔ easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone
- ✔ finiteness of the abstraction [Termination]
  {Abs(Z) | Z zone} is finite

General Abstractions

Criteria for a good abstraction operator Abs:

- ✔ easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone
- ✔ finiteness of the abstraction [Termination]
  {Abs(Z) | Z zone} is finite
- ✔ completeness of the abstraction [Completeness]
  $Z \subseteq \text{Abs}(Z)$

General Abstractions

Criteria for a good abstraction operator Abs:

- ✔ easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone
- ✔ finiteness of the abstraction [Termination]
  {Abs(Z) | Z zone} is finite
- ✔ completeness of the abstraction [Completeness]
  $Z \subseteq \mathrm{Abs}(Z)$
- ✔ soundness of the abstraction [Soundness]
  the computation of $(\mathrm{Abs} \circ \mathrm{Post})^*$ is correct w.r.t. reachability

General Abstractions

Criteria for a good abstraction operator Abs:

- ✔ easy computation [Effectiveness]
  Abs(Z) is a zone if Z is a zone
- ✔ finiteness of the abstraction [Termination]
  {Abs(Z) | Z zone} is finite
- ✔ completeness of the abstraction [Completeness]
  $Z \subseteq \mathrm{Abs}(Z)$
- ✔ soundness of the abstraction [Soundness]
  the computation of $(\mathrm{Abs} \circ \mathrm{Post})^*$ is correct w.r.t. reachability

For the previous automaton,

no abstraction operator can satisfy all these criteria!

Why That?

Assume there is a "nice" operator Abs.

The set {M DBM representing a zone Abs(Z)} is finite.

➜ k the max. constant defining one of the previous DBMs

We get that, for every zone Z,

$$Z \subseteq \mathrm{Extra}_k(Z) \subseteq \mathrm{Abs}(Z)$$

Problem!

Open questions:

- which conditions can be made weaker?
- find a clever termination criterium?
- use an other data structure than zones/DBMs?
- ?

What Can We Cling To?

Diagonal-free: only guards $x \sim c$
(no guard $x - y \sim c$)

Theorem: the classical algorithm is correct for diagonal-free timed automata.

[Bouyer 2003]

What Can We Cling To?

Diagonal-free: only guards $x \sim c$
(no guard $x - y \sim c$)

Theorem: the classical algorithm is correct for diagonal-free timed automata.

General: both guards $x \sim c$ and $x - y \sim c$

Proposition: the classical algorithm is correct for timed automata that use **less than 3 clocks.**

(the constant used is bigger than the maximal constant...)

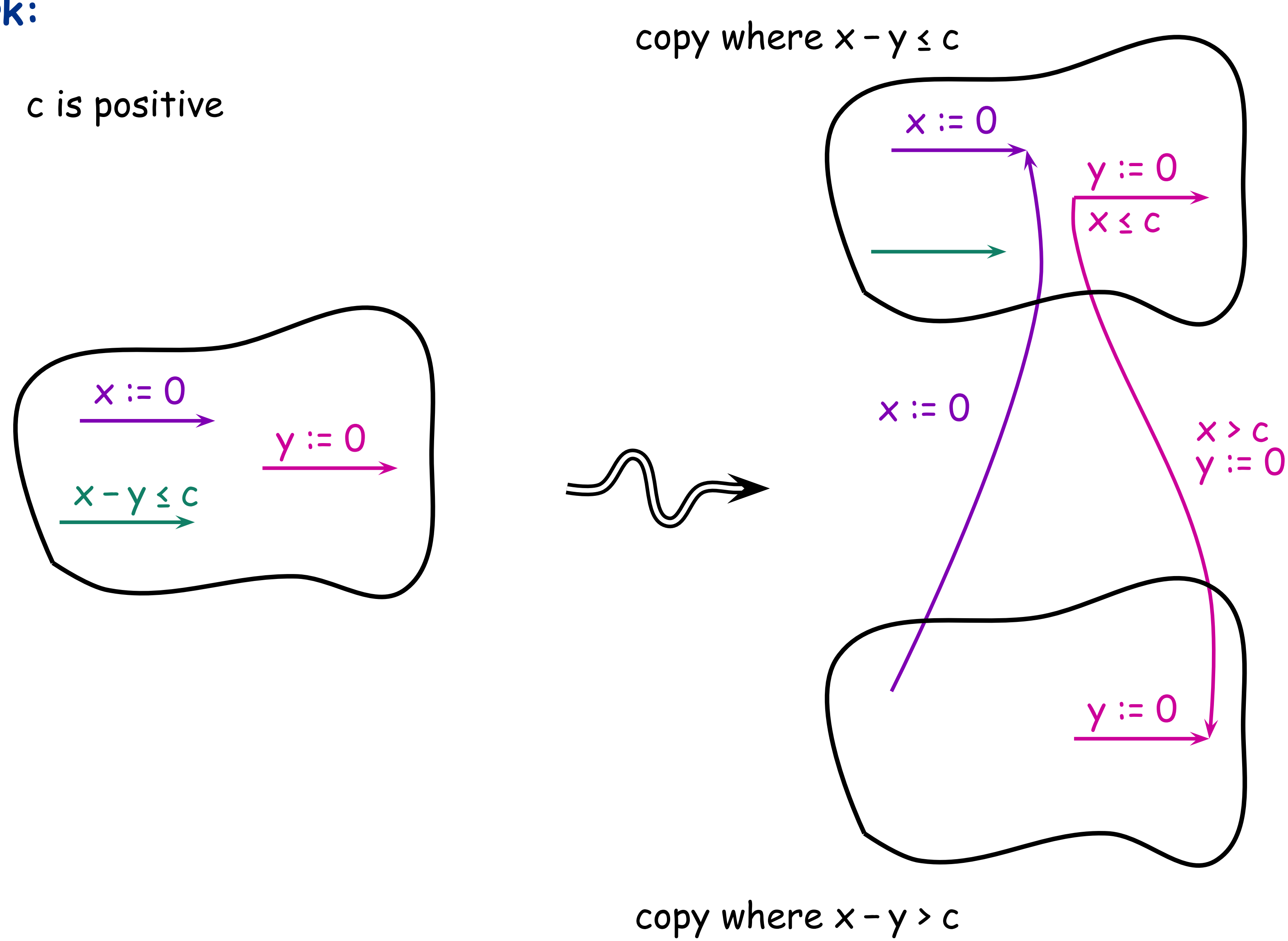
[Bouyer 2003]

How to Deal with Diagonals?

→ proof in **[Bérard, Diekert, Gastin & Petit 1998]**

Remark:

c is positive



How to Deal with Diagonals?

Remark: diagonal can be eliminated (but blowup of the number of discrete states)

How to Deal with Diagonals?

Remark: diagonal can be eliminated (but blowup of the number of discrete states)

Solution: eliminate on-the-fly diagonals

How to Deal with Diagonals?

Remark: diagonal can be eliminated (but blowup of the number of discrete states) ☹

Solution: eliminate on-the-fly diagonals

How to Deal with Diagonals?

Remark: diagonal can be eliminated (but blowup of the number of discrete states)

Solution: eliminate on-the-fly diagonals ☹

Actual work: counter-example refinement

How to Deal with Diagonals?

Remark: diagonal can be eliminated (but blowup of the number of discrete states)

Solution: eliminate on-the-fly diagonals ☹

Actual work: counter-example refinement hope ☺

A Note on Compositional Methods

Basic idea:

untimed: **[Andersen 1995]**

timed: **[Laroussinie, Larsen 1995]**

$$(A_1 \parallel \cdots \parallel A_n) \models \varphi \quad \Leftrightarrow \quad (A_1 \parallel \cdots \parallel A_{n-1}) \models \varphi / A_n$$

$$\vdots$$

$$\Leftrightarrow \quad \mathrm{nil} \models \varphi / A_n / \ldots / A_2 / A_1$$

A Note on Compositional Methods

Basic idea:

untimed: **[Andersen 1995]**

timed: **[Laroussinie, Larsen 1995]**

$$(A_1 \parallel \cdots \parallel A_n) \models \varphi \quad \Leftrightarrow \quad (A_1 \parallel \cdots \parallel A_{n-1}) \models \varphi / A_n$$

$$\vdots$$

$$\Leftrightarrow \quad \mathrm{nil} \models \varphi / A_n / \ldots / A_2 / A_1$$

Need of:

- ✔ a compositional logic, e.g. L_μ, $L^+_{\mu,\nu}$...

$$([a]\varphi)/q = \bigwedge_{\substack{q \xrightarrow{g,c,r} q' \\ f(b,c)=a}} (g \Rightarrow [b](\varphi/q'))$$

A Note on Compositional Methods

Basic idea:

untimed: **[Andersen 1995]**

timed: **[Laroussinie, Larsen 1995]**

$$
\begin{aligned}
(A_1 \parallel \cdots \parallel A_n) \models \varphi \quad & \Leftrightarrow \quad (A_1 \parallel \cdots \parallel A_{n-1}) \models \varphi / A_n \\
& \vdots \\
& \Leftrightarrow \quad \mathrm{nil} \models \varphi / A_n / \ldots / A_2 / A_1
\end{aligned}
$$

Need of:

- ✔ a compositional logic, e.g. L_μ, $L^{+}_{\mu,\nu}$...

$$([a]\varphi)/q = \bigwedge_{\substack{q \xrightarrow{g,c,r} q' \\ f(b,c)=a}} (g \Rightarrow [b](\varphi/q'))$$

- ✔ simplification rules

A Note on Compositional Methods

Basic idea:

untimed: [Andersen 1995]

timed: [Laroussinie, Larsen 1995]

$$(A_1 \parallel \cdots \parallel A_n) \models \varphi \quad \Leftrightarrow \quad (A_1 \parallel \cdots \parallel A_{n-1}) \models \varphi / A_n$$

$$\vdots$$

$$\Leftrightarrow \quad \text{nil} \models \varphi / A_n / \ldots / A_2 / A_1$$

Need of:

- ✔ a compositional logic, e.g. L_μ, $L^+_{\mu,\nu}$...

$$([a]\varphi)/q = \bigwedge_{\substack{q \xrightarrow{g,c,r} q' \\ f(b,c)=a}} (g \Rightarrow [b](\varphi/q'))$$

- ✔ simplification rules

Bad news: for those logics, nil model-checking is as difficult as simple m.-c.

[Aceto, Laroussinie 2002]

Existing Tools

- ✔ **Uppaal**: made in Uppsala (Sweden) & Aalborg (Denmark)
  - reachability, deadlock, a simple fragment of TCTL
  - forward analysis

  `http://www.uppaal.com`

Existing Tools

- ✔ **Uppaal**: made in Uppsala (Sweden) & Aalborg (Denmark)
  - reachability, deadlock, a simple fragment of TCTL
  - forward analysis

  `http://www.uppaal.com`
- ✔ **HyTech**: made in Berkeley (USA)
  - no specification logic, a rich computation language, hybrid models
  - forward and backward computations

  `http://www-cad.eecs.berkeley.edu/~tah/HyTech/`

Existing Tools

- **Uppaal**: made in Uppsala (Sweden) & Aalborg (Denmark)
  - reachability, deadlock, a simple fragment of TCTL
  - forward analysis

  `http://www.uppaal.com`
- **HyTech**: made in Berkeley (USA)
  - no specification logic, a rich computation language, hybrid models
  - forward and backward computations

  `http://www-cad.eecs.berkeley.edu/~tah/HyTech/`
- **CMC**: made in Cachan (France)
  - modal logic L_ν
  - compositional method

  `http://www.lsv.ens-cachan.fr/~fl/cmcweb.html`

Existing Tools

- ✔ **Uppaal**: made in Uppsala (Sweden) & Aalborg (Denmark)
  - reachability, deadlock, a simple fragment of TCTL
  - forward analysis

  `http://www.uppaal.com`
- ✔ **HyTech**: made in Berkeley (USA)
  - no specification logic, a rich computation language, hybrid models
  - forward and backward computations

  `http://www-cad.eecs.berkeley.edu/~tah/HyTech/`
- ✔ **CMC**: made in Cachan (France)
  - modal logic L_ν
  - compositional method

  `http://www.lsv.ens-cachan.fr/~fl/cmcweb.html`
- ✔ **Kronos**: made in Grenoble (France)
  - full TCTL
  - forward and backward analysis

  `http://www-verimag.imag.fr/TEMPORISE/kronos/`

Conclusion Remarks

Actual Challenges

Deal with both discrete and time explosions!

untimed systems	time information
BBD-like techniques	more and more optimizations
	static analysis of TA
	[BBFL03,BBLP04...]

Some attempts for the data-structures:

- ✔ the CDD data-structure [Larsen, Pearson, Weise & Yi 1999]
- ✔ the data-structure of RED [Wang since 2000]

Some attempts for the techniques:

- ✔ partial-order reduction [Bengtsson, Jonsson, Lilius & Yi 1998]
- ✔ partial-order semantics approach [Lugiez, Niebert & Zennou 2004]

Actual Challenges (cont.)

Intermediate challenges

- better understand **geometry** of reachable state spaces
  (in particular, find a satisfactory solution for dealing with diagonals)
- data-structures for both **discrete and dense** parts
  (up to now: time is not really integrated, it is only added as a feature)
- propose **true concurrent models**?
- and then use techniques from concurrency theory?

Other challenges

- controller synthesis,
- implementability issues (program synthesis)

Thanks to F. Laroussinie, F. Cassez, O.-H. Roux and J.-F. Raskin

Bibliography

[ACD90] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking for real-time systems. In *Proc. 5th IEEE Symposium on Logic in Computer Science (LICS'90)*, pages 414–425. IEEE Computer Society Press, 1990.

[ACD93] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: an algorithmic approach to specification and verification of hybrid systems. In *Proc. Workshop on Hybrid Systems (1991 & 1992)*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1993.

[AD90] Rajeev Alur and David Dill. Automata for modeling real-time systems. In *Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[AD94] Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science (TCS)*, 126(2):183–235, 1994.

[AH94] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *Journal of the Association for Computing Machinery (JACM)*, 41(1):181–204, 1994.

[AL99] Luca Aceto and François Laroussinie. Is your model-checker on time? In *Proc. 24th International Symposium on Mathematical Foundations of Computer Science (MFCS'99)*, volume 1672 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 1999.

Bibliography

[AL02] Luca Aceto and François Laroussinie. Is your model-checker on time? on the complexity of model-checking for timed modal logics. *Journal of Logic and Algebraic Programming (JLAP)*, 52–53:7–51, 2002.

[Alu91] Rajeev Alur. *Techniques for Automatic Verification of Real-Time Systems.* PhD thesis, Stanford University, Stanford, CA, USA, 1991.

[And95] Henrik R. Andersen. Partial model-checking (extended abstract). In *Proc. 10th IEEE Symposium on Logic in Computer Science (LICS'95)*, pages 398–407. IEEE Computer Society Press, 1995.

[BBFL03] Gerd Behrmann, Patricia Bouyer, Emmanuel Fleury, and Kim G. Larsen. Static guard analysis in timed automata verification. In *Proc. 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2003)*, volume 2619 of *Lecture Notes in Computer Science*, pages 254–277. Springer, 2003.

[BBLP04] Gerd Behrmann, Patricia Bouyer, Kim G. Larsen, and Radek Pelánek. Lower and upper bounds in zone based abstractions of timed automata. In *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 312–326. Springer, 2004.

Bibliography

[BD91] Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Transactions in Software Engineering*, 17(3):259–273, 1991.

[BD00] Béatrice Bérard and Catherine Dufourd. Timed automata and additive clock constraints. *Information Processing Letters (IPL)*, 75(1–2):1–7, 2000.

[BDFP00a] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Are timed automata updatable? In *Proc. 12th International Conference on Computer Aided Verification (CAV'2000)*, volume 1855 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2000.

[BDFP00b] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Expressiveness of updatable timed automata. In *Proc. 25th International Symposium on Mathematical Foundations of Computer Science (MFCS'2000)*, volume 1893 of *Lecture Notes in Computer Science*, pages 232–242. Springer, 2000.

[BDGP98] Béatrice Bérard, Volker Diekert, Paul Gastin, and Antoine Petit. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2–3):145–182, 1998.

[BFHR92] Ahmed Bouajjani, J.C. Fernandez, Nicolas Halbwachs, and Pascal Raymond. Minimal state graph generation. *Science of Computer Programming*, 18(3):247–269, 1992.

Bibliography

[BJLY98] Johan Bengtsson, Bengt Jonsson, Johan Lilius, and Wang Yi. Partial order reductions for timed systems. In *Proc. 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *Lecture Notes in Computer Science*, pages 485–500. Springer, 1998.

[BM83] Bernard Berthomieu and Miguel Menasche. An enumerative approach for analyzing time Petri nets. In *Proc. IFIP 9th World Computer Congress*, volume 83 of *Information Processing*, pages 41–46. North-Holland/ IFIP, 1983.

[Bou03] Patricia Bouyer. Untameable timed automata! In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'03)*, volume 2607 of *Lecture Notes in Computer Science*, pages 620–631. Springer, 2003.

[Bou04] Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 2004. To appear.

[BS91] Janusz A. Brzozowski and Carl-Johan H. Seger. Advances in asynchronous circuit theory. Bulletin of the European Association of Theoretical Computer Science (EATCS), 1991.

[BV03] Bernard Berthomieu and François Vernadat. State class constructions for branching analysis of time Petri nets. In *Proc. 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *Lecture Notes in Computer Science*, pages 442–457. Springer, 2003.

Bibliography

[CR03] Franck Cassez and Olivier-H. Roux. Traduction structurelle des réseaux de petri temporels vers les automates temporisés. In *Actes 4ième Colloque sur la Modélisation des Systèmes Réactifs (MSR'2003)*, pages 311–326. Hermès, 2003.

[CY92] Costas Courcoubetis and Mihalis Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415, 1992.

[Dil89] David Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proc. of the Workshop on Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.

[DT98] Conrado Daws and Stavros Tripakis. Model-checking of real-time reachability properties using abstractions. In *Proc. 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98)*, volume 1384 of *Lecture Notes in Computer Science*, pages 313–329. Springer, 1998.

[DY96] Conrado Daws and Sergio Yovine. Reducing the number of clock variables of timed automata. In *Proc. 17th IEEE Real-Time Systems Symposium (RTSS'96)*, pages 73–81. IEEE Computer Society Press, 1996.

[EC82] E. Allen Emerson and Edmund M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.

Bibliography

[GRR03] Guillaume Gardey, Olivier H. Roux, and Olivier F. Roux. A zone-based method for computing the state space of a time Petri net. In *Proc. 1st International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, Lecture Notes in Computer Science. Springer, 2003. To appear.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *Proc. 11th IEEE Annual Symposim on Logic in Computer Science (LICS'96)*, pages 278–292. IEEE Computer Society Press, 1996.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, 1998.

[HNSY94] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model-checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[LL95] François Laroussinie and Kim G. Larsen. Compositional model-checking of real-time systems. In *Proc. 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 1995.

Bibliography

[LLW95] François Laroussinie, Kim G. Larsen, and Carsten Weise. From timed automata to logic – and back. In *Proc. 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, pages 529–539. Springer, 1995.

[LMS04] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Model-checking "real-time automata" and other simple timed automata. In submission, 2004.

[LMST03] Ruggero Lanotte, Andrea Maggiolo-Schettini, and Simone Tini. Concurrency in timed automata. *Theoretical Computer Science*, 309(1–3):503–527, 2003.

[LNZ04] Denis Lugiez, Peter Niebert, and Sarah Zennou. A partial order semantics approach to the clock explosion problem of timed automata. In *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 296–311. Springer, 2004.

[LPWY99] Kim G. Larsen, Justin Pearson, Carsten Weise, and Wang Yi. Clock difference diagrams. *Nordic Journal of Computing*, 6(3):271–298, 1999.

[LPY97] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *Journal of Software Tools for Technology Transfer (STTT)*, 1(1–2):134–152, 1997.

Bibliography

[Mer74] Philip M. Merlin. *A Study of the Recoverability of Computing Systems.* PhD thesis, University of California, Irvine, CA, USA, 1974.

[Min67] Marvin Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall International, 1967.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium on Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE Computer Society Press, 1977.

[Tri03] Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. In *Proc. 1st International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, Lecture Notes in Computer Science. Springer, 2003. To appear.

[Wang03] Farn Wang. Efficient verification of timed automata with BDD-like data-structures. In *Proc. 4th Internationsl Conference on Verification, Model-Checking and Abstract Interpretation (VMCAI'03)*, volume 2575 of *Lecture Notes in Computer Science*, pages 189–205. Springer, 2003.

[Yov98] Sergio Yovine. Model-checking timed automata. In *School on Embedded Systems*, volume 1494 of *Lecture Notes in Computer Science*, pages 114–152. Springer, 1998.

[YR98] Tomohiro Yoneda and Hikaru Ryuba. CTL model-checking of time petri nets using geometric regions. *IEICE Transactions Information and Systems*, E81-D(3), 1998.

Advertisement!

Don't forget to have a look at the posters!!!!