Integer Parameter Synthesis for Timed Systems

First International Workshop on Synthesis of Continuous Parameters

Aleksandra Jovanović, **Didier Lime**, Olivier H. Roux

École Centrale de Nantes – IRCCyN

Grenoble, 6 April 2014

Outline

Introduction

Parametric Timed Automata

(Un)decidability

Symbolic Semi-algorithms

Computing the Integer Valuations

Time Petri Nets

Conclusion

Outline

Introduction

Parametric Timed Automata

(Un)decidability

Symbolic Semi-algorithms

Computing the Integer Valuations

Time Petri Nets

Conclusion

Parametric Verification

- Verification:
  - Does the system satisfy the property?
  - Yes / No ;
  - Witness / Counterexample.

Parametric Verification

- Verification:
  - Does the system satisfy the property?
  - Yes / No ;
  - Witness / Counterexample.
- **Parametric** Verification:
  - What are the values of the parameter such that the system satisfy the property?
  - Flexible and/or early design;
  - Symbolic proofs;
  - Robustness.

Outline

Introduction

Parametric Timed Automata

(Un)decidability

Symbolic Semi-algorithms

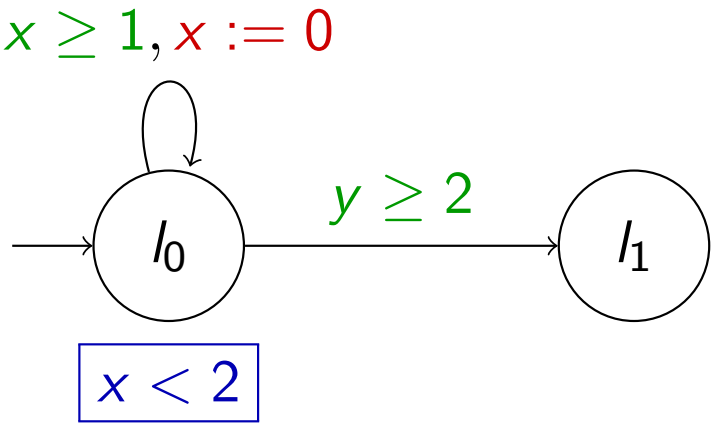
Computing the Integer Valuations

Time Petri Nets

Conclusion

Timed Automata (TA)

- Finite automata with a finite set of clocks, guards, resets and invariants:

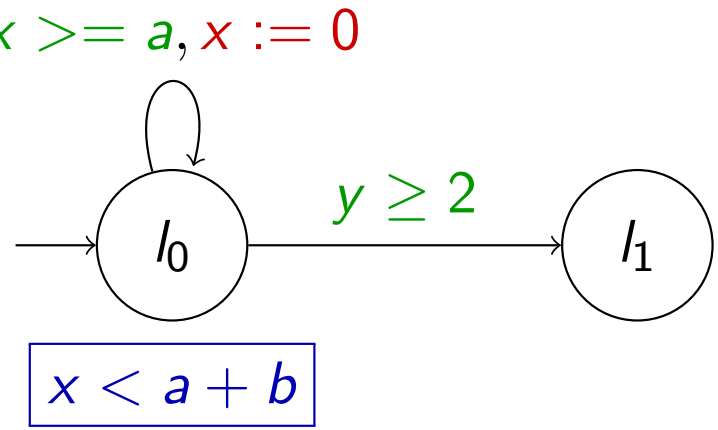


- A possible **run**:

$$(l_0,(0,0)) \xrightarrow{1} (l_0,(1,1)) \rightarrow (l_0,(0,1)) \xrightarrow{1.54} (l_0,(1.54,2.54)) \rightarrow (l_1,(1.54,2.54))$$

Parametric Timed Automata (PTA)

- Timed automata with clock constraint bounds as **linear expressions** on a finite set of parameters P:



- For any PTA $\mathcal{A}$ and any rational valuation of the parameters $v : P \to \mathbb{Q}$, "$v(\mathcal{A})$" is a **timed automaton**.

Parametric Problems

- Let Φ be some property class (reachability, unavoidability, etc.)
- Let $\mathcal{A}$ be a PTA and $\varphi \in \Phi$ be some property on $\mathcal{A}$
- Let $F_\varphi(\mathcal{A})$ be the set of valuations v s.t. $v(\mathcal{A}) \models \varphi$.

Φ-emptiness problem:

INPUTS: A PTA $\mathcal{A}$ and a property φ in Φ
PROBLEM: $F_\varphi(\mathcal{A}) = \emptyset$?

Parametric Problems

- Let Φ be some property class (reachability, unavoidability, etc.)
- Let $\mathcal{A}$ be a PTA and $\varphi \in \Phi$ be some property on $\mathcal{A}$
- Let $F_\varphi(\mathcal{A})$ be the set of valuations v s.t. $v(\mathcal{A}) \models \varphi$.

Φ-emptiness problem:

INPUTS: A PTA $\mathcal{A}$ and a property φ in Φ
PROBLEM: $F_\varphi(\mathcal{A}) = \emptyset$?

Constrained Φ-emptiness problem:

INPUTS: A PTA $\mathcal{A}$, a property φ in Φ, a set of valuations V
PROBLEM: $F_\varphi(\mathcal{A}) \cap V = \emptyset$?

Parametric Problems

- Let Φ be some property class (reachability, unavoidability, etc.)
- Let $\mathcal{A}$ be a PTA and $\varphi \in \Phi$ be some property on $\mathcal{A}$
- Let $F_\varphi(\mathcal{A})$ be the set of valuations v s.t. $v(\mathcal{A}) \models \varphi$.

Φ-emptiness problem:

INPUTS: A PTA $\mathcal{A}$ and a property φ in Φ
PROBLEM: $F_\varphi(\mathcal{A}) = \emptyset$?

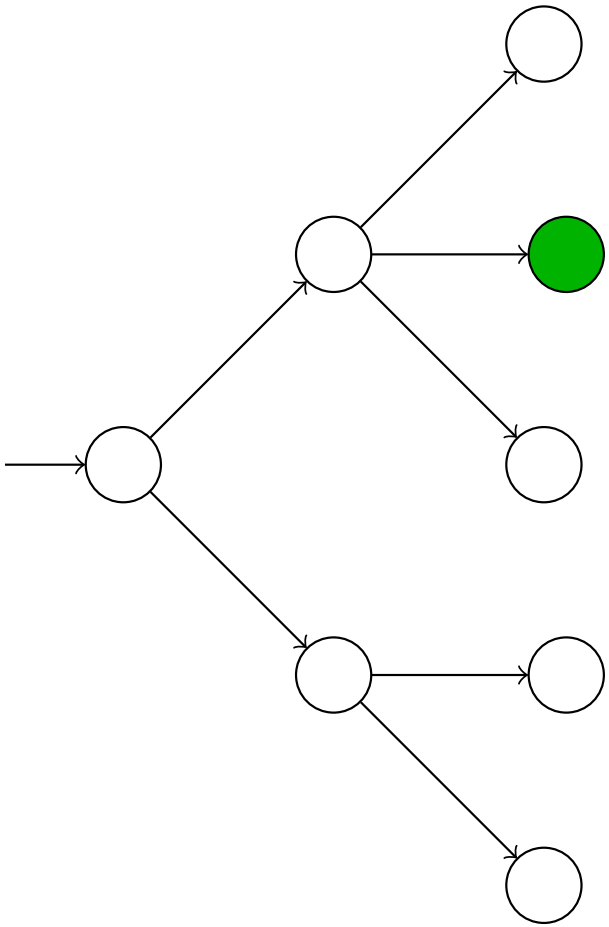
Constrained Φ-emptiness problem:

INPUTS: A PTA $\mathcal{A}$, a property φ in Φ, a set of valuations V
PROBLEM: $F_\varphi(\mathcal{A}) \cap V = \emptyset$?

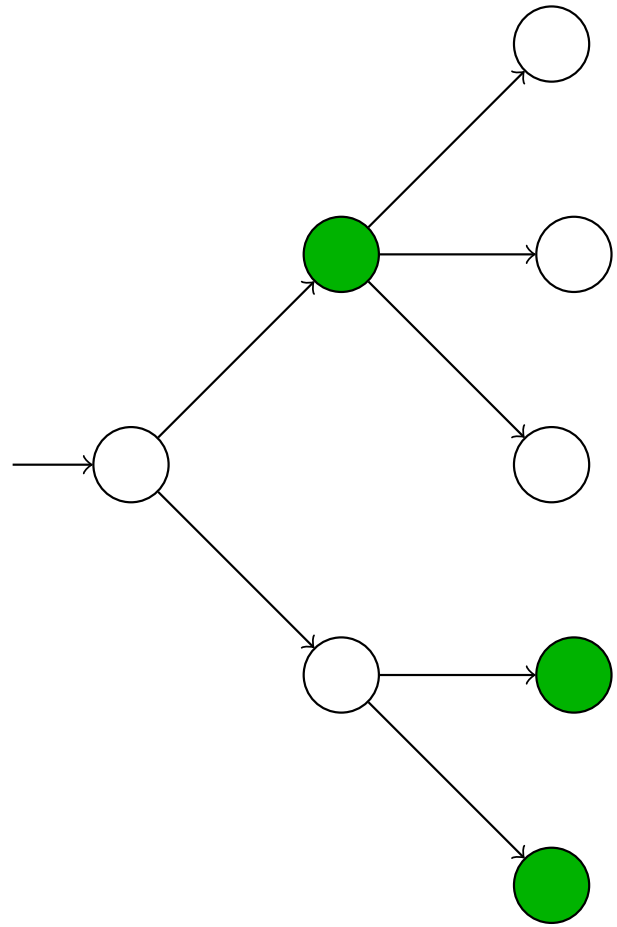
Φ-synthesis problem:

INPUTS: A PTA $\mathcal{A}$ and a property φ in Φ
PROBLEM: Compute $F_\varphi(\mathcal{A})$

Reachability and Unavoidability



Reachability (EF)



Unavoidability (AF)

Outline

Introduction

Parametric Timed Automata

(Un)decidability

Symbolic Semi-algorithms

Computing the Integer Valuations

Time Petri Nets

Conclusion

Parametric Verification is Hard!

The EF-emptiness problem for PTA is:

- in dense-time:
  - undecidable with one bounded parameter, one parametric clock, and three regular clocks [Mil00];
  - decidable with one clock [Mil00];
  - "robustly" undecidable [Doy07] (with only open constraints).
- in discrete-time [AHV93]:
  - undecidable with three parametric clocks;
  - decidable with one parametric clock.

Undecidability of the EF-emptiness problem for PTA

Theorem ([AHV93])

The EF-emptiness problem for PTA is undecidable

- Reduction of the 2-counter machine halting problem;

Undecidability of the EF-emptiness problem for PTA

Theorem ([AHV93])

The EF-emptiness problem for PTA is undecidable

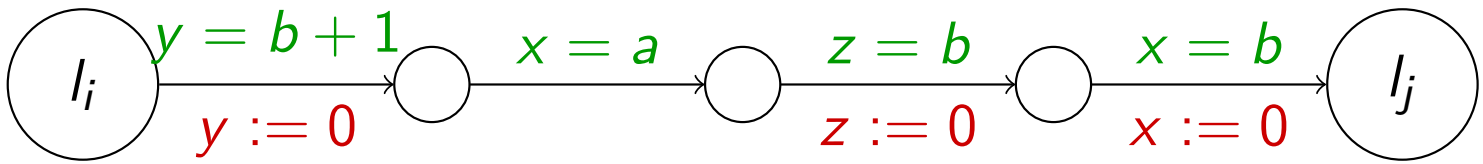
- Reduction of the 2-counter machine halting problem;
- $c_1 = b - y$ and $c_2 = b - a - z$.

Undecidability of the EF-emptiness problem for PTA

Theorem ([AHV93])

The EF-emptiness problem for PTA is undecidable

- Reduction of the 2-counter machine halting problem;
- $c_1 = b - y$ and $c_2 = b - a - z$.
- Increment of C_1



$y = b - c_1$
$z = b - a - c_2$
$x = 0$

$y = 0$
$z = b - a - c_2$
$\quad + c_1 + 1$
$x = c_1 + 1$

$y = a - c_1 - 1$
$z = b - c_2$
$x = a$

$y = a - c_1$
$\quad - 1 + c_2$
$z = 0$
$x = c_2 + a$

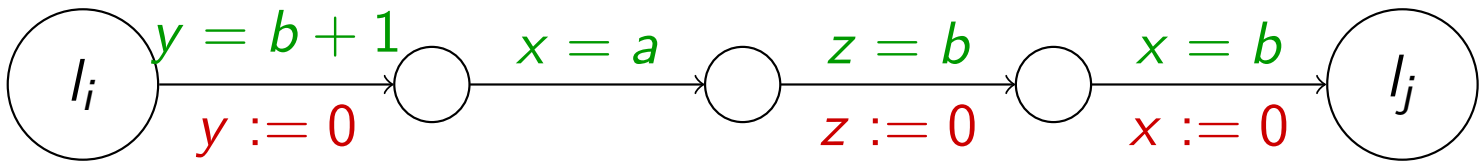
$y = b - c_1 - 1$
$z = b - a - c_2$
$x = 0$

Undecidability of the EF-emptiness problem for PTA

Theorem ([AHV93])

The EF-emptiness problem for PTA is undecidable

- Reduction of the 2-counter machine halting problem;
- $c_1 = b - y$ and $c_2 = b - a - z$.
- Increment of C_1



$y = b - c_1$
$z = b - a - c_2$
$x = 0$

$y = 0$
$z = b - a - c_2 + c_1 + 1$
$x = c_1 + 1$

$y = a - c_1 - 1$
$z = b - c_2$
$x = a$

$y = a - c_1 - 1 + c_2$
$z = 0$
$x = c_2 + a$

$y = b - c_1 - 1$
$z = b - a - c_2$
$x = 0$

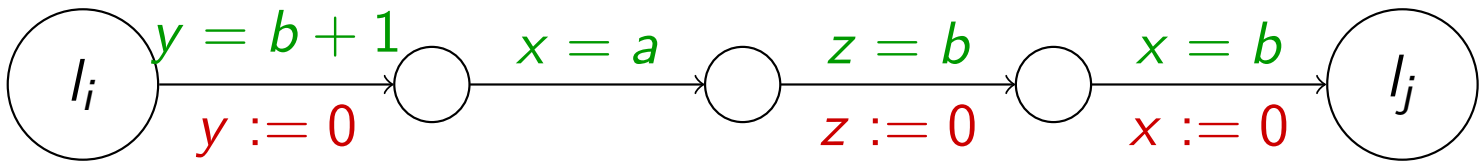
- Zero-test of C_1: a guard $y = b$ between l_i and l_j

Undecidability of the EF-emptiness problem for PTA

Theorem ([AHV93])

The EF-emptiness problem for PTA is undecidable

- Reduction of the 2-counter machine halting problem;
- $c_1 = b - y$ and $c_2 = b - a - z$.
- Increment of C_1



$y = b - c_1$
$z = b - a - c_2$
$x = 0$

$y = 0$
$z = b - a - c_2$
$+ c_1 + 1$
$x = c_1 + 1$

$y = a - c_1 - 1$
$z = b - c_2$
$x = a$

$y = a - c_1$
$- 1 + c_2$
$z = 0$
$x = c_2 + a$

$y = b - c_1 - 1$
$z = b - a - c_2$
$x = 0$

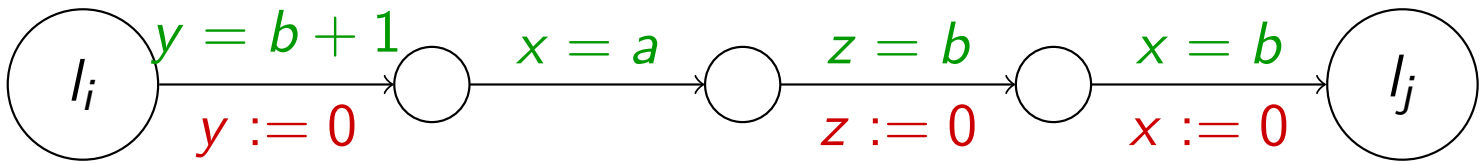
- Zero-test of C_1: a guard $y = b$ between l_i and l_j
- If the machine does not halt, the PTA cannot reach l_{halt}: $F = \emptyset$;

Undecidability of the EF-emptiness problem for PTA

Theorem ([AHV93])

The EF-emptiness problem for PTA is undecidable

- Reduction of the 2-counter machine halting problem;
- $c_1 = b - y$ and $c_2 = b - a - z$.
- Increment of C_1



$y = b - c_1$
$z = b - a - c_2$
$x = 0$

$y = 0$
$z = b - a - c_2 + c_1 + 1$
$x = c_1 + 1$

$y = a - c_1 - 1$
$z = b - c_2$
$x = a$

$y = a - c_1 - 1 + c_2$
$z = 0$
$x = c_2 + a$

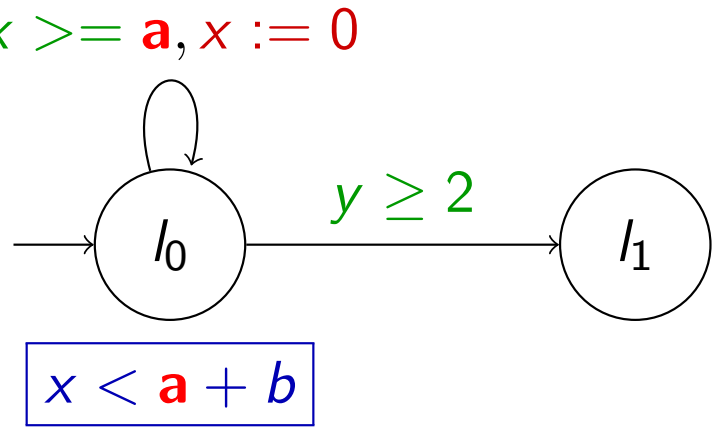
$y = b - c_1 - 1$
$z = b - a - c_2$
$x = 0$

- Zero-test of C_1: a guard $y = b$ between l_i and l_j
- If the machine does not halt, the PTA cannot reach l_{halt}: $F = \emptyset$;
- If the machine halts and c_1, c_2 are the max values of the counters then $F = \{a \geq c_1 \text{ and } b - a \geq c_2\} \neq \emptyset$.

L/U-automata [HRSV02]

- In L/U-automata, parameters are either **upper bounds** (e.g. $x \leq a$ or $x \geq 1 - a$) or **lower bounds** in the whole PTA, not both.

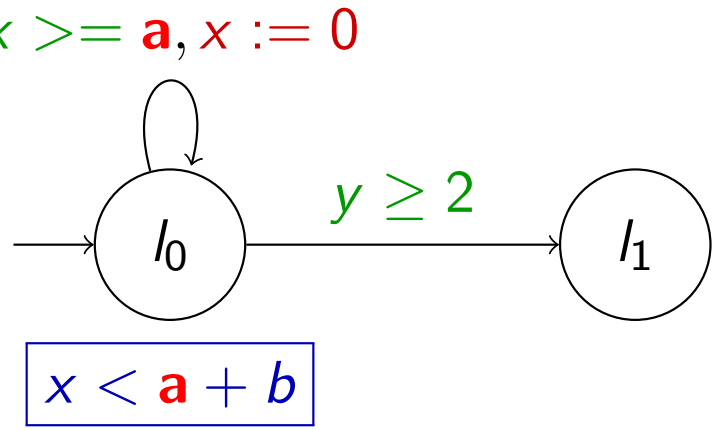
not LU



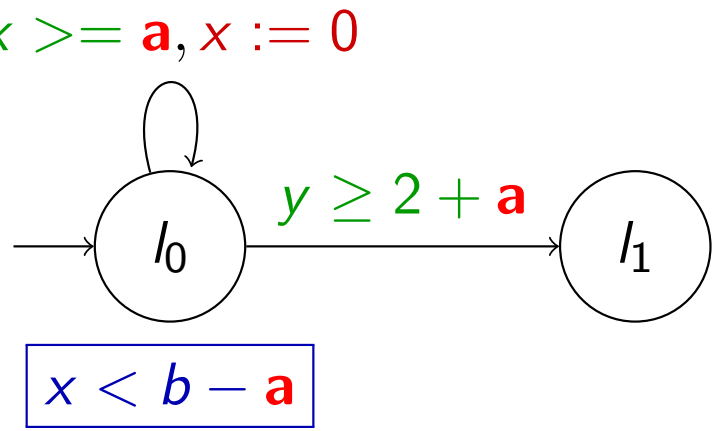
L/U-automata [HRSV02]

- In L/U-automata, parameters are either **upper bounds** (e.g. $x \leq a$ or $x \geq 1 - a$) or **lower bounds** in the whole PTA, not both.

not LU



LU



L/U-automata: Emptiness

- L/U automata have a **monotonicity** property:
  - decreasing lower bounds only **add** behaviors;
  - increasing upper bounds only **add** behaviors;
- They are "reachability-friendly": **all** the possible behaviors can be obtained by setting lower bounds to 0 and upper bounds to $+\infty$.

Theorem ([HRSV02])

The EF-emptiness problem for L/U-automata is PSPACE.

L/U-automata: Emptiness

- L/U automata have a **monotonicity** property:
  - decreasing lower bounds only **add** behaviors;
  - increasing upper bounds only **add** behaviors;
- They are "reachability-friendly": **all** the possible behaviors can be obtained by setting lower bounds to 0 and upper bounds to $+\infty$.

Theorem ([HRSV02])

The EF-emptiness problem for L/U-automata is PSPACE.

Theorem ([BT09])

The "repeated reachability"-emptiness problem for L/U-automata is PSPACE-complete (also universality and finiteness).

L/U-automata: AF-Emptiness

- L/U automata are very well suited to **reachability** properties;
- But they fail for **unavoidability**:

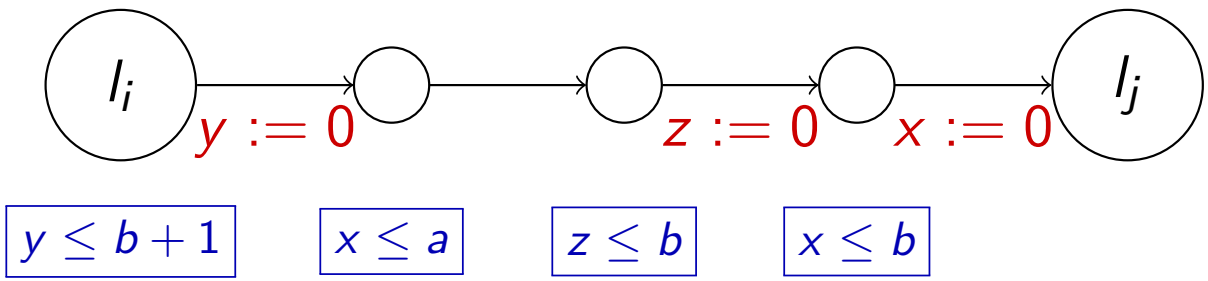
Theorem

*The AF-emptiness problem for **U**-automata is undecidable*

- We adapt the 2-counter machine reduction of [AHV93].

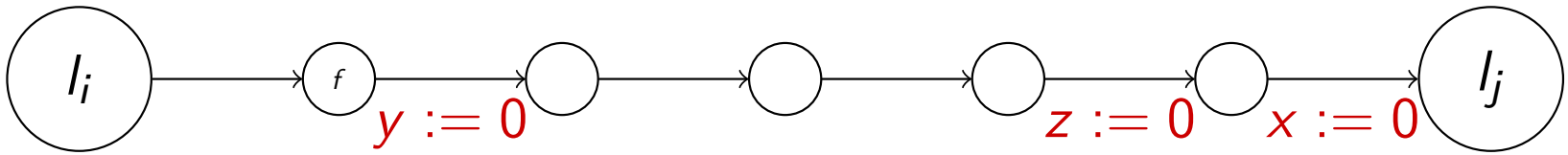
L/U-automata: AF-Emptiness

- All the runs go through the gadgets iff the **slowest run** ρ^* does:
  AF l_{halt} iff ρ^* reaches l_{halt}
- ρ^* simulates the machine;
- Increment C_1:



L/U-automata: AF-Emptiness

- Zero-testing C_1:

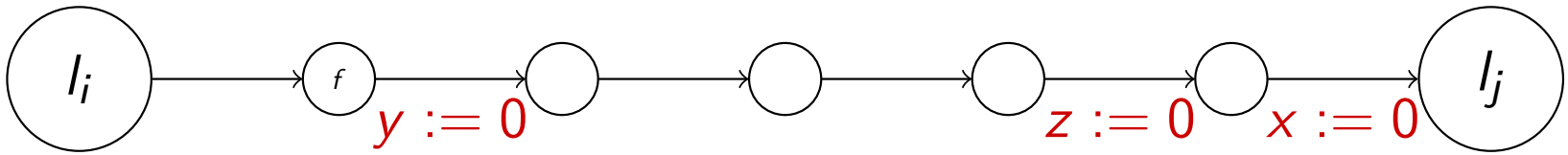


$x \leq 0$	$y \leq b$	$x \leq 0$	$x \leq a$	$z \leq b$	$x \leq b$	
$y \leq b - c_1$ $z \leq b - a - c_2$ $x = 0$	$y \leq b$ $z \leq b - a$ $\quad - c_2 + c_1$ $x \leq c_1$	$y = 0$ $z \leq b - a - c_2 + c_1$ $x \leq c_1$ $x \leq 0$				$y \leq b - c_1$ $z \leq b - a - c_2$ $x = 0$

The slowest run is **blocked** in f iff $c_1 \neq 0$

L/U-automata: AF-Emptiness

- Zero-testing C_1:



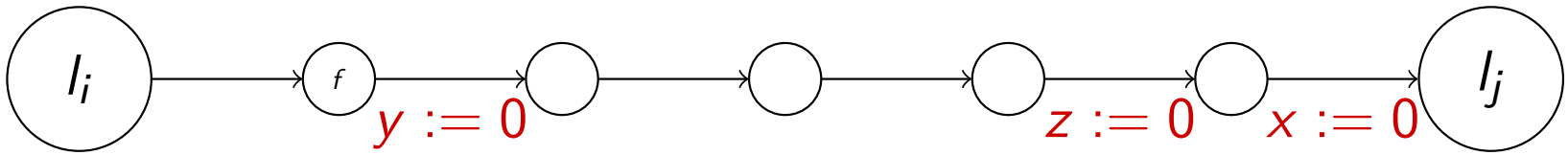
$x \leq 0$	$y \leq b$	$x \leq 0$	$x \leq a$	$z \leq b$	$x \leq b$	
$y \leq b - c_1$ $z \leq b - a - c_2$ $x = 0$	$y \leq b$ $z \leq b - a$ $\quad - c_2 + c_1$ $x \leq c_1$	$y = 0$ $z \leq b - a - c_2 + c_1$ $x \leq c_1$ $x \leq 0$				$y \leq b - c_1$ $z \leq b - a - c_2$ $x = 0$

The slowest run is **blocked** in f iff $c_1 \neq 0$

- If the machine does not halts:

L/U-automata: AF-Emptiness

- Zero-testing C_1:



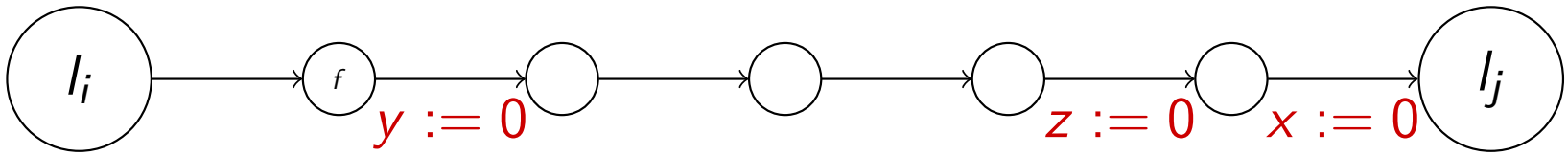
$x \leq 0$	$y \leq b$	$x \leq 0$	$x \leq a$	$z \leq b$	$x \leq b$	
$y \leq b - c_1$ $z \leq b - a - c_2$ $x = 0$	$y \leq b$ $z \leq b - a$ $\quad - c_2 + c_1$ $x \leq c_1$	$y = 0$ $z \leq b - a - c_2 + c_1$ $x \leq c_1$ $x \leq 0$				$y \leq b - c_1$ $z \leq b - a - c_2$ $x = 0$

The slowest run is **blocked** in f iff $c_1 \neq 0$

- If the machine does not halts:
  - if some runs are blocked in f in some reset ($c \neq 0$), the AF property never holds: $F = \emptyset$

L/U-automata: AF-Emptiness

- Zero-testing C_1:



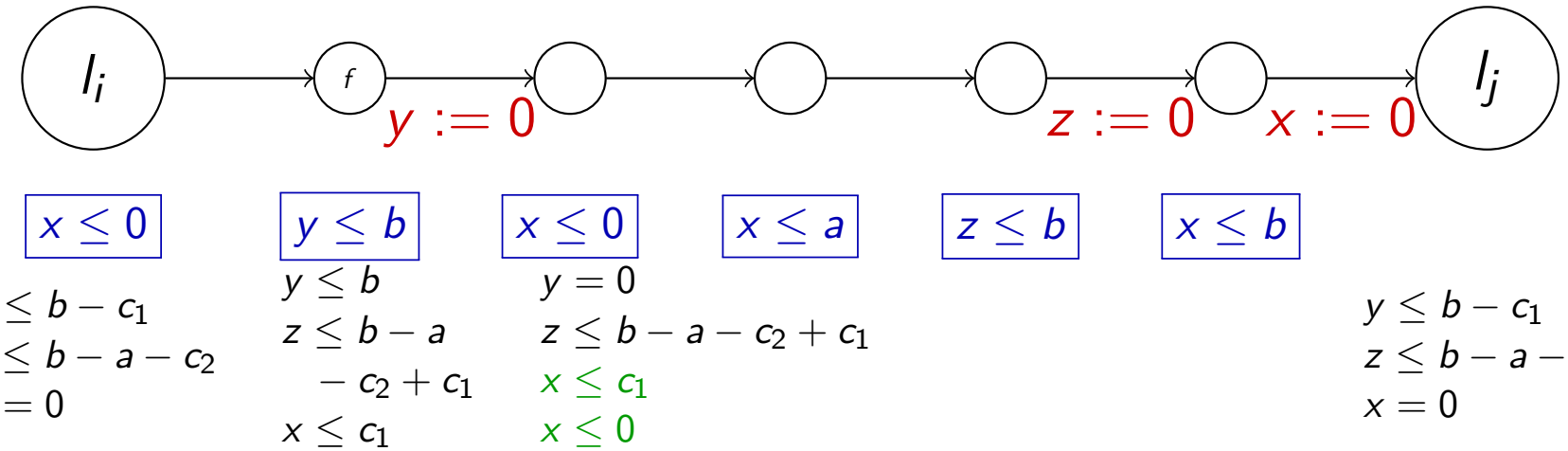
$x \leq 0$	$y \leq b$	$x \leq 0$	$x \leq a$	$z \leq b$	$x \leq b$	
$y \leq b - c_1$ $z \leq b - a - c_2$ $x = 0$	$y \leq b$ $z \leq b - a$ $\quad - c_2 + c_1$ $x \leq c_1$	$y = 0$ $z \leq b - a - c_2 + c_1$ $x \leq c_1$ $x \leq 0$				$y \leq b - c_1$ $z \leq b - a - c_2$ $x = 0$

The slowest run is **blocked** in f iff $c_1 \neq 0$

- If the machine does not halts:
  - if some runs are blocked in f in some reset ($c \neq 0$), the AF property never holds: $F = \emptyset$
  - otherwise the slowest run does not reach l_{halt} and again $F = \emptyset$.

L/U-automata: AF-Emptiness

- Zero-testing C_1:



  The slowest run is **blocked** in f iff $c_1 \neq 0$
- If the machine does not halts:
  - if some runs are blocked in f in some reset ($c \neq 0$), the AF property never holds: $F = \emptyset$
  - otherwise the slowest run does not reach l_{halt} and again $F = \emptyset$.
- if the machine halts then no runs are blocked in f, and as before $F \neq \emptyset$.

L/U-automata: Synthesis

Theorem ([BT09])

The "repeated reachability"-synthesis problem can be explicitly solved for L- and U-automata (giving finite unions of convex polyhedra).

- In a PTA, if p_i is both a lower bound and an upper bound:
  - Replace p by a fresh parameter p_i^u when p is an upper-bound;
  - Replace p by a fresh parameter p_i^l when p is a lower-bound.

L/U-automata: Synthesis

Theorem ([BT09])

The "repeated reachability"-synthesis problem can be explicitly solved for L- and U-automata (giving finite unions of convex polyhedra).

- In a PTA, if p_i is both a lower bound and an upper bound:
  - Replace p by a fresh parameter p_i^u when p is an upper-bound;
  - Replace p by a fresh parameter p_i^l when p is a lower-bound.
- $V = \bigcap_i p_i^u = p_i^l$

L/U-automata: Synthesis

Theorem ([BT09])

The "repeated reachability"-synthesis problem can be explicitly solved for L- and U-automata (giving finite unions of convex polyhedra).

- In a PTA, if p_i is both a lower bound and an upper bound:
  - Replace p by a fresh parameter p_i^u when p is an upper-bound;
  - Replace p by a fresh parameter p_i^l when p is a lower-bound.
- $V = \bigcap_i p_i^u = p_i^l$

Theorem ([BT09])

Constrained "repeated reachability"-emptiness is undecidable for L/U-automata (decidable if upper bounds are not compared to lower bounds in the initial constraints)

L/U-automata: Synthesis

- We can deduce something else from the previous reasoning:

Theorem

For L/U automata, the solution to the EF-synthesis problem cannot be represented using any formalism for which emptiness of the intersection with equality constraints is decidable.

(in particular, not finite unions of convex polyhedra)

L/U-automata: Synthesis

- We can deduce something else from the previous reasoning:

Theorem

For L/U automata, the solution to the EF-synthesis problem cannot be represented using any formalism for which emptiness of the intersection with equality constraints is decidable.

(in particular, not finite unions of convex polyhedra)

- These results can be trivially **extended** to "simple" L/U automata in which:
  - parametric constraints in the automaton are reduced to only one parameter;
  - each parameter is used only once in the whole automaton.

Bounded Integer Parametric Problems

- L/U automata:
  - Mixed decidability results regarding **emptiness**;
  - Actual **synthesis** of constraints is "complicated" if we insist on both upper and lower bounds.

Bounded Integer Parametric Problems

- L/U automata:
  - Mixed decidability results regarding **emptiness**;
  - Actual **synthesis** of constraints is "complicated" if we insist on both upper and lower bounds.
- **Syntactical** restriction of the constraints is not the only possibility;

Bounded Integer Parametric Problems

- L/U automata:
  - Mixed decidability results regarding **emptiness**;
  - Actual **synthesis** of constraints is "complicated" if we insist on both upper and lower bounds.
- **Syntactical** restriction of the constraints is not the only possibility;
- If we look for parameter values as **bounded integers** "everything" is decidable and computable!

Bounded Integer Parametric Problems

- L/U automata:
  - Mixed decidability results regarding **emptiness**;
  - Actual **synthesis** of constraints is "complicated" if we insist on both upper and lower bounds.
- **Syntactical** restriction of the constraints is not the only possibility;
- If we look for parameter values as **bounded integers** "everything" is decidable and computable!
- A finite number of possible values is somewhat trivial but:
  - EF-emptiness for **bounded rational** values is undecidable [Mil00];
  - EF-emptiness for possibly **unbounded integer** values is undecidable: in the proof of [AHV93], $b = c_1 + c_2$ and $a = c_1$ are in F when the machine halts, with c_1, c_2 max values of the counters on the run.

Outline

Introduction

Parametric Timed Automata

(Un)decidability

Symbolic Semi-algorithms

Computing the Integer Valuations

Time Petri Nets

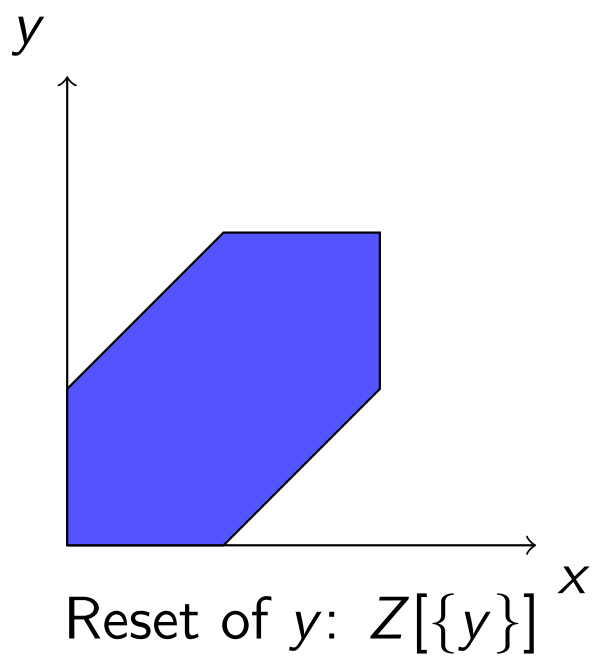
Conclusion

Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;

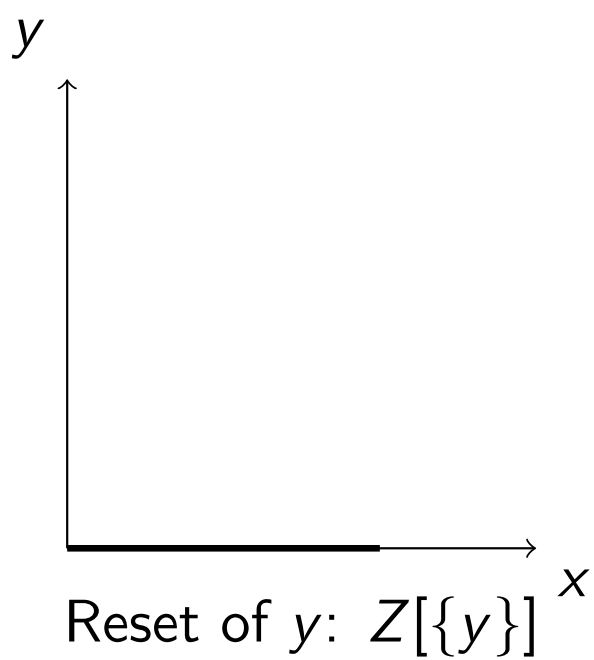
Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters;
  - Straightforward extension of the **reset** and **future** operations that act only on the **clock** variables;



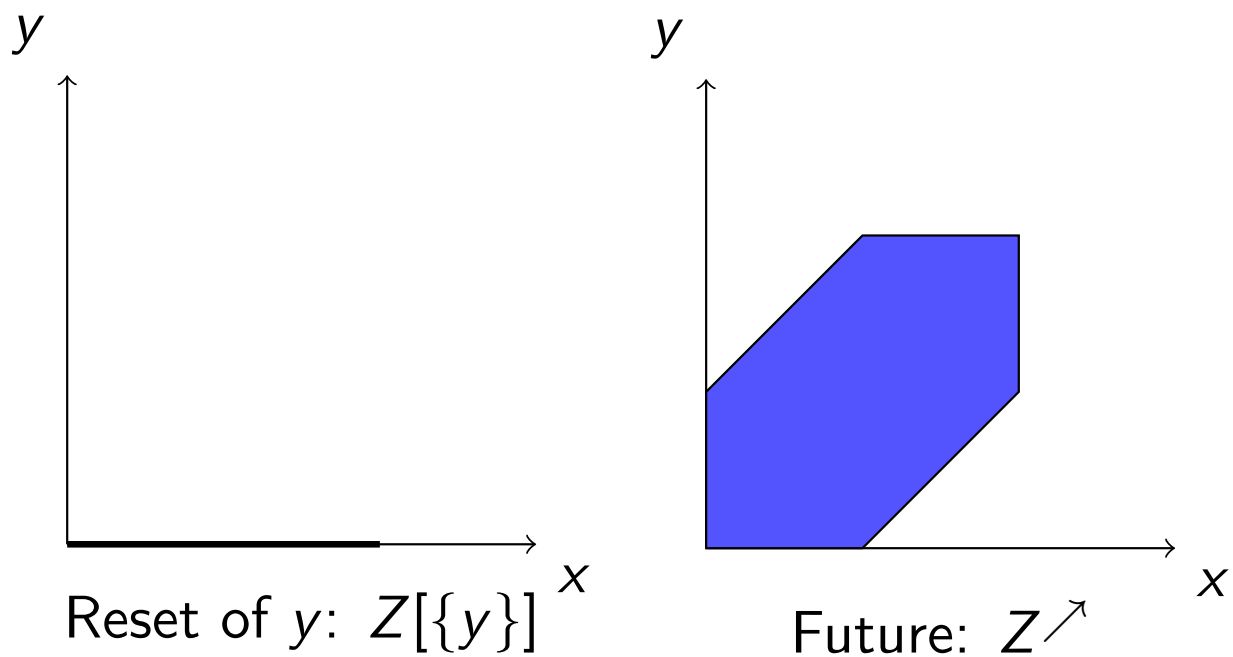
Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters;
  - Straightforward extension of the **reset** and **future** operations that act only on the **clock** variables;



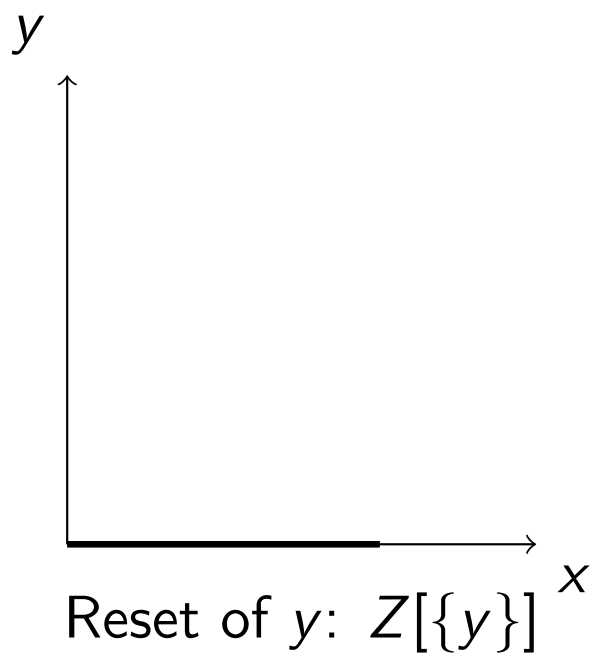
Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters;
  - Straightforward extension of the **reset** and **future** operations that act only on the **clock** variables;

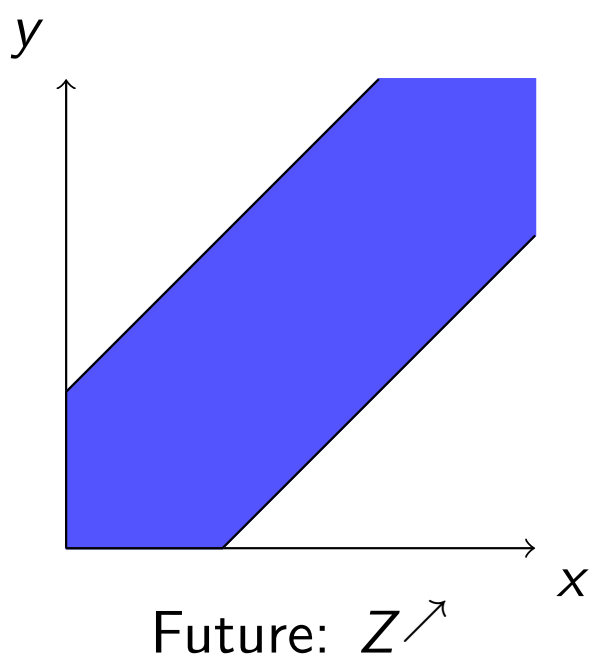


Symbolic States for PTA

- We do **not** want to **enumerate** all possible values of the parameters;
- We extend the classical **symbolic** approach to model-checking in dense-time:
  - Symbolic state (l, Z): location + polyhedron constraining **both** clocks and parameters;
  - Straightforward extension of the **reset** and **future** operations that act only on the **clock** variables;



Reset of y: $Z[\{y\}]$

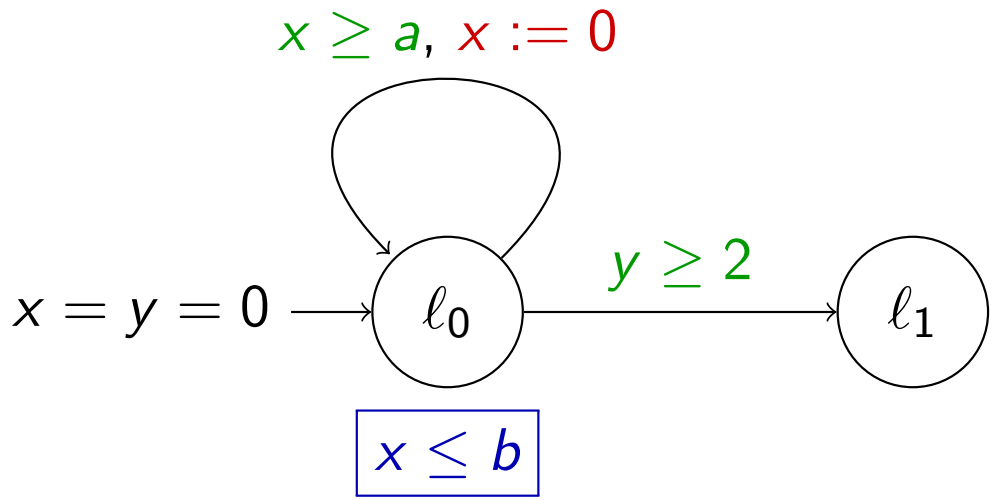


Future: $Z^{\nearrow}$

Symbolic States for PTA

- **initial** symbolic state of the PTA $\mathcal{A}$:
  $\mathsf{Init}(\mathcal{A}) = (l_0, \{v \in \mathbb{R}^{X \cup P} \mid v_{|X} \in \{\vec{0}_X\}^{\nearrow} \cap v_{|P}(\mathsf{Inv})\})$;
- **successor** by an edge $e = (l, a, \gamma, R, l')$:
  $\mathsf{Succ}((l, Z), e) = (l', (Z \cap \gamma)[R]^{\nearrow} \cap \mathsf{Inv}(l'))$

Example



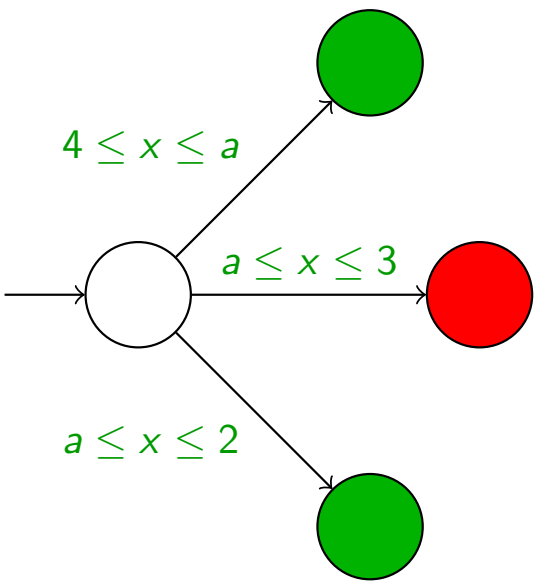
- Initial symbolic state: (ℓ_0, Z_0) with $Z_0 = \{x = y, x \leq b\}$;
- After n loops: (ℓ_0, Z_n) with

$$Z_n = \{0 \leq x \leq b, 0 \leq y, a \leq b, 0 \leq na \leq y - x \leq (n+1)b\}$$

EF semi-algorithm

- $S_{|P}$: with $S = (l, Z)$, projection of Z on parameters;
- M: passed list.

$$\mathrm{EF}_G(S, M) = \begin{cases} Z_{|P} & \text{if } S = (l, Z) \text{ and } l \in G \\ \emptyset & \text{if } S \in M \\ \bigcup_{\substack{e \in E \\ S' = \mathrm{Succ}(S,e)}} \mathrm{EF}_G\left(S', M \cup \{S\}\right) & \text{otherwise.} \end{cases}$$



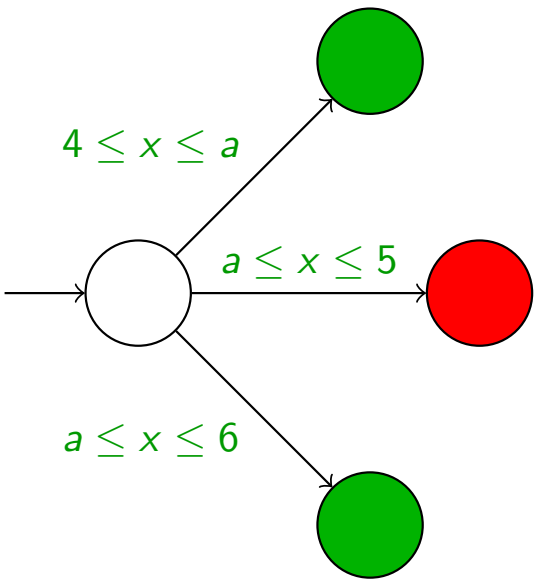
Collect the "good" parameter values:
$a \leq 2$ or $a \geq 4$

AF semi-algorithm

$$\mathsf{AF}_G(S,M) = \begin{cases} Z_{|P} & \text{if } S=(l,Z) \text{ and } l \in G \\ \emptyset & \text{if } S \in M \text{ or } S \text{ has no succ.} \\ \bigcap_{\substack{e\in E \\ S'=\mathsf{Succ}(S,e)}} \mathsf{AF}_G\big(S', M\cup\{S\}\big) \cup (\mathbb{R}^{\mathbf{P}} \setminus \mathbf{S'_{|P}}) \setminus \mathsf{dead}(S)_{|P} & \text{otherwise.} \end{cases}$$

with $\mathsf{dead}(S) = \left(\mathbb{R}^{X\cup P} \setminus \left(\bigcup_{(l,a,g,R,l')\in E} (g\cap S)^{\swarrow}\right)\right)$

States from which no outgoing transition can be taken anymore



Collect the "good" parameter values:

$a \ge 4$ and $a \le 6$

And remove the "bad" ones:

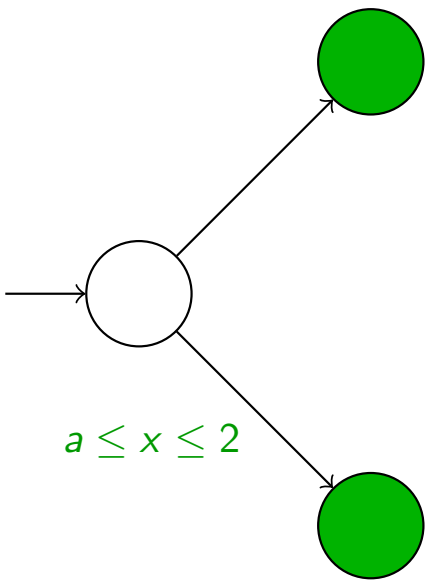
and $a > 5$

AF semi-algorithm

$$\mathrm{AF}_G(S,M) = \begin{cases} Z_{|P} & \text{if } S=(l,Z) \text{ and } l \in G \\ \emptyset & \text{if } S \in M \text{ or } S \text{ has no succ.} \\ \bigcap_{\substack{e\in E \\ S'=\mathrm{Succ}(S,e)}} \mathrm{AF}_G\big(S', M\cup\{S\}\big) \cup (\mathbb{R}^{\mathbf{P}} \setminus \mathbf{S'_{|P}}) \setminus \mathrm{dead}(S)_{|P} & \text{otherwise.} \end{cases}$$

with $\mathrm{dead}(S) = \left(\mathbb{R}^{X\cup P} \setminus \left(\bigcup_{(l,a,g,R,l')\in E} (g\cap S)^{\swarrow}\right)\right)$

States from which no outgoing transition can be taken anymore



Cutting a "good" branch may be needed:

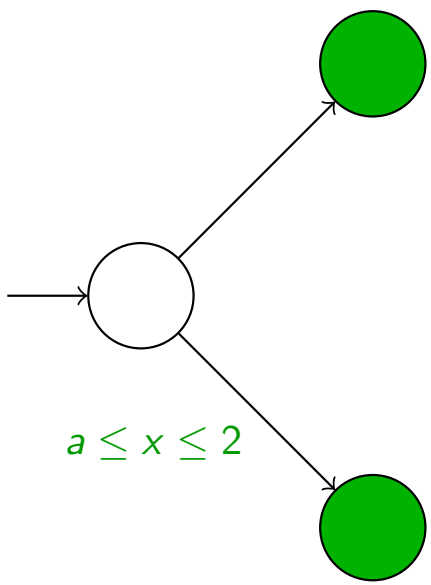
$a \geq 0$

AF semi-algorithm

$$\mathrm{AF}_G(S,M) = \begin{cases} Z_{\downarrow P} & \text{if } S=(l,Z) \text{ and } l \in G \\ \emptyset & \text{if } S \in M \text{ or } S \text{ has no succ.} \\ \bigcap_{\substack{e\in E \\ S'=\mathrm{Succ}(S,e)}} \mathrm{AF}_G\big(S', M\cup\{S\}\big) \cup (\mathbb{R}^{\mathbf{P}} \setminus \mathbf{S'_{\downarrow P}}) \setminus \mathrm{dead}(S)_{\downarrow P} & \text{otherwise.} \end{cases}$$

with $\mathrm{dead}(S) = \left(\mathbb{R}^{X\cup P} \setminus \left(\bigcup_{(l,a,g,R,l')\in E} (g\cap S)^{\swarrow}\right)\right)$

States from which no outgoing transition can be taken anymore



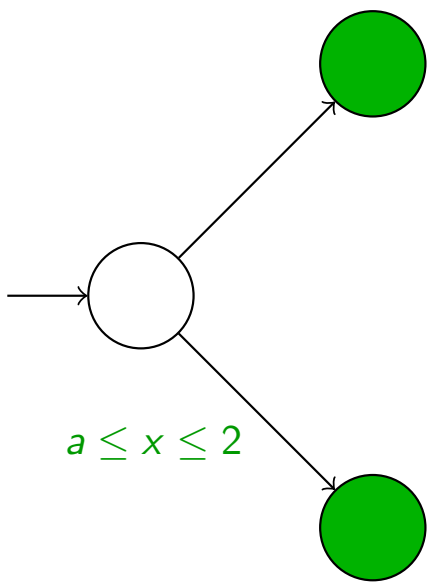
Cutting a "good" branch may be needed:
$a \geq 0$ and $a \leq 2$

AF semi-algorithm

$$\mathrm{AF}_G(S,M) = \begin{cases} Z_{|P} & \text{if } S=(l,Z) \text{ and } l \in G \\ \emptyset & \text{if } S \in M \text{ or } S \text{ has no succ.} \\ \bigcap_{\substack{e\in E \\ S'=\mathrm{Succ}(S,e)}} \mathrm{AF}_G\big(S', M\cup\{S\}\big) \cup (\mathbb{R}^{\mathbf{P}} \setminus \mathbf{S'_{|P}}) \setminus \mathrm{dead}(S)_{|P} & \text{otherwise.} \end{cases}$$

with $\mathrm{dead}(S) = \left(\mathbb{R}^{X\cup P} \setminus \left(\bigcup_{(l,a,g,R,l')\in E} (g\cap S)^{\swarrow}\right)\right)$

States from which no outgoing transition can be taken anymore



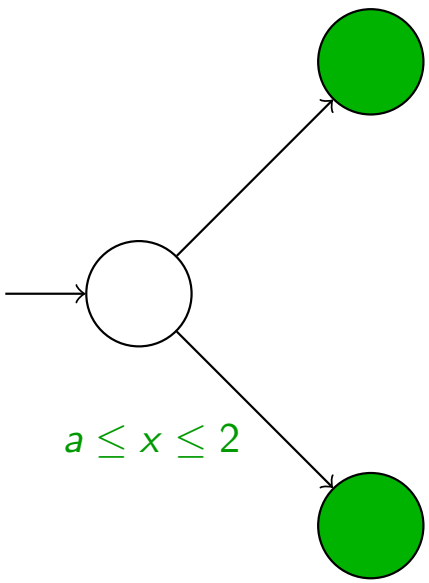
Cutting a "good" branch may be needed:
$a \geq 0$ and $(a \leq 2$ or $a > 2)$

AF semi-algorithm

$$\mathrm{AF}_G(S,M) = \begin{cases} Z_{|P} & \text{if } S=(l,Z) \text{ and } l \in G \\ \emptyset & \text{if } S \in M \text{ or } S \text{ has no succ.} \\ \bigcap_{\substack{e\in E \\ S'=\mathrm{Succ}(S,e)}} \mathrm{AF}_G\big(S', M\cup\{S\}\big) \cup (\mathbb{R}^{\mathbf{P}} \setminus \mathbf{S'_{|P}}) \setminus \mathrm{dead}(S)_{|P} & \text{otherwise.} \end{cases}$$

with $\mathrm{dead}(S) = \left(\mathbb{R}^{X\cup P} \setminus \left(\bigcup_{(l,a,g,R,l')\in E} (g\cap S)^{\swarrow}\right)\right)$

States from which no outgoing transition can be taken anymore



Cutting a "good" branch may be needed:

$a \geq 0$

EF and AF semi-algorithms

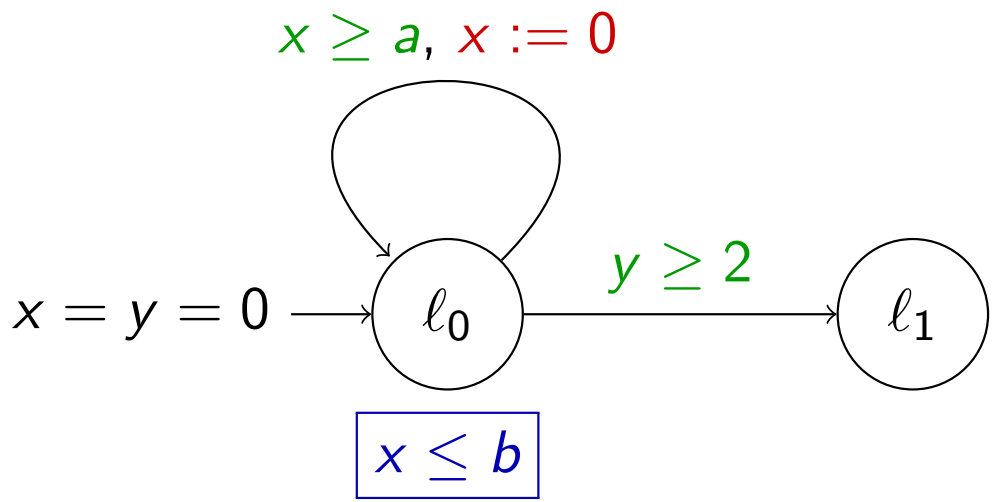
Theorem

For any PTA $\mathcal{A}$ and any subset of its locations G, upon termination, $\mathsf{EF}(\mathsf{Init}(\mathcal{A}), G, \emptyset)$ *is the solution the EF-synthesis problem for PTA $\mathcal{A}$ and set of locations to reach G.*

idem for AF

Termination is **not** guaranteed

Example



- Initial symbolic state: (ℓ_0, Z_0) with $Z_0 = \{x = y, x \leq b\}$;
- After n loops: (ℓ_0, Z_n) with:

$$Z_n = \{0 \leq x \leq b, 0 \leq y, a \leq b, 0 \leq na \leq y - x \leq (n+1)b\}$$

- $\forall n \neq m, Z_m \neq Z_n$ so the computations do not terminate.

Outline

Introduction

Parametric Timed Automata

(Un)decidability

Symbolic Semi-algorithms

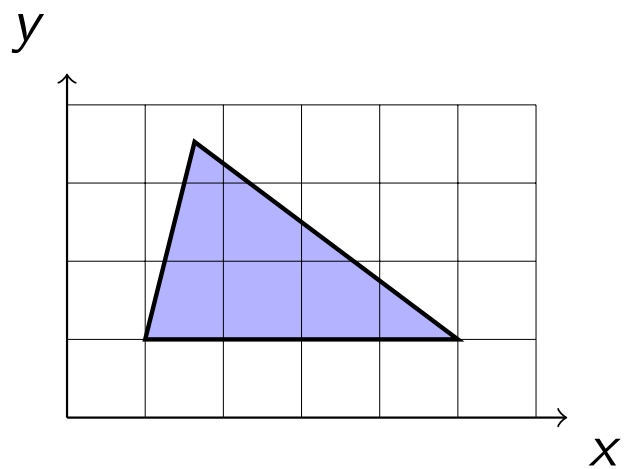
Computing the Integer Valuations

Time Petri Nets

Conclusion

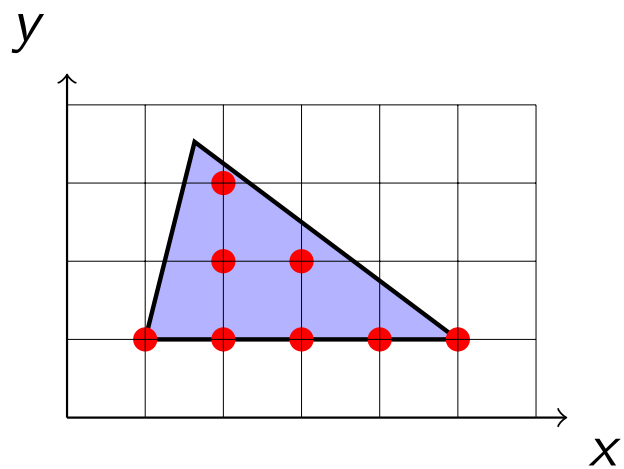
Computing the Integer Valuations: Integer Hulls

- We compute **integer hulls**: $\mathrm{IH}(Z)$



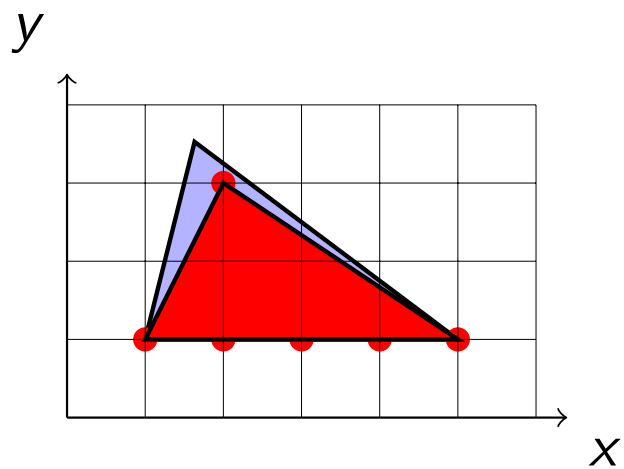
Computing the Integer Valuations: Integer Hulls

- We compute **integer hulls**: $\mathrm{IH}(Z)$



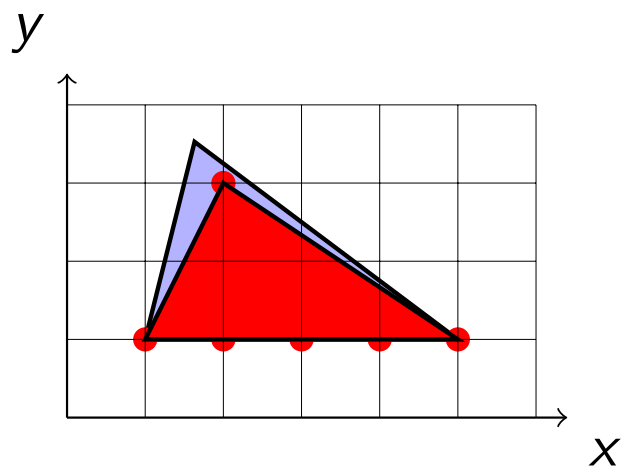
Computing the Integer Valuations: Integer Hulls

- We compute **integer hulls**: $\mathrm{IH}(Z)$



Computing the Integer Valuations: Integer Hulls

- We compute **integer hulls**: $\mathrm{IH}(Z)$



Lemma

For any symbolic state (l, Z) and any edge e:

$$\mathrm{IH}(\mathrm{Succ}(\mathrm{IH}((l, Z)), e)) = \mathrm{IH}(\mathrm{Succ}((l, Z), e))$$

Computing the Integer Valuations: Integer Hulls

- We note $\mathsf{ISucc}((l, Z), e) = \mathsf{IH}(\mathsf{Succ}((l, Z), e))$ and have:

Lemma

For a reachable symbolic states S, edge e, and **integer** *valuation v:*

$$v(\mathsf{ISucc}(S, e)) = \mathsf{Succ}(v(S), v(e))$$

Computing the Integer Valuations: Integer Hulls

- We note $\mathsf{ISucc}((l, Z), e) = \mathsf{IH}(\mathsf{Succ}((l, Z), e))$ and have:

Lemma

For a reachable symbolic states S, edge e, and **integer** *valuation v:*

$$v(\mathsf{ISucc}(S, e)) = \mathsf{Succ}(v(S), v(e))$$

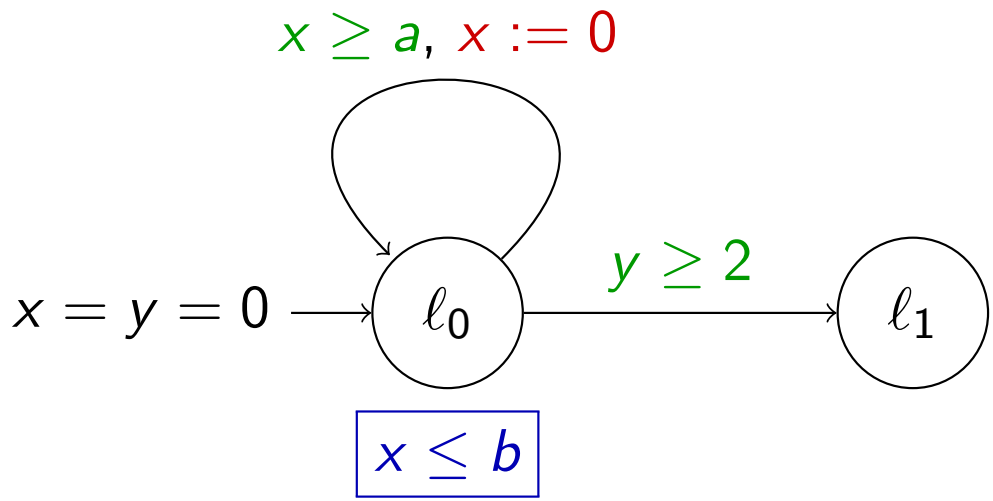
- We use ISucc instead of Succ in semi-algortihm EF to compute the **integer valuations**, giving semi-algorithm IEF.

Theorem

For any PTA $\mathcal{A}$ and any subset of its locations G, upon termination, $\mathsf{IEF}(\mathsf{Init}(\mathcal{A}), G, \emptyset)$ *is the solution the integer EF-synthesis problem for PTA $\mathcal{A}$ and set of locations to reach G.*

idem for IAF

Example



- Initial symbolic state: (ℓ_0, Z_0) with $Z_0 = \{x = y, x \leq b\}$;
- After n loops: (ℓ_0, Z_n) with **still**

$$Z_n = \{0 \leq x \leq b, 0 \leq y, a \leq b, 0 \leq na \leq y - x \leq (n+1)b\}$$

- Actually, $Z_n = \mathsf{IH}(Z_n)$

Termination in the Bounded Case

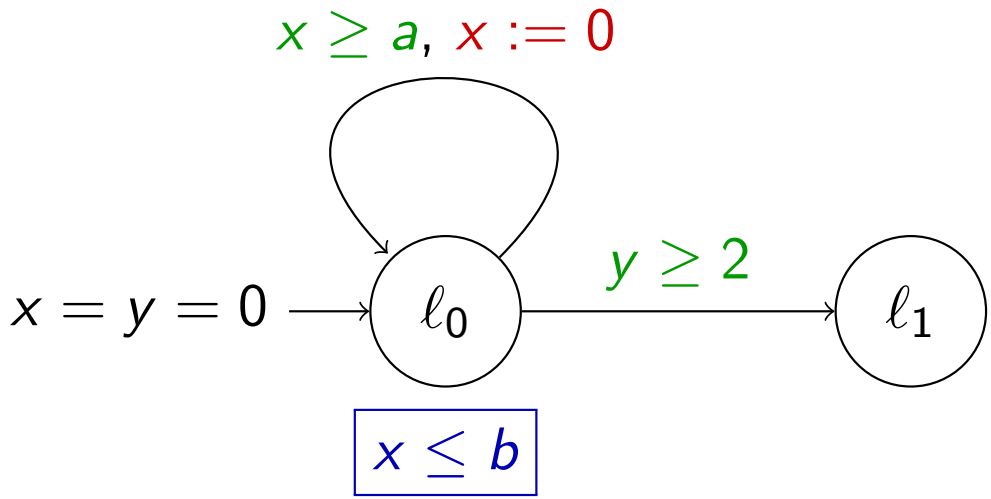
- When the possible parameter values are bounded, we can assume w.l.o.g. that all clocks are **bounded**;
- This ensures **termination** with the following result:

Lemma

For any symbolic state (l, Z) of the PTA $\mathcal{A}$:

$$\mathsf{IH}(Z) = \mathsf{Conv}\left(\bigcup_{v \in \mathsf{IntVects}(Z_{|P})} v(Z) \right)$$

Example



- Suppose all parameters are in $[0..3]$;
- Initial symbolic state: (ℓ_0, Z_0) with $Z_0 = \{x = y, x \leq b \leq 3\}$;
- After one loop: $Z_1' = Z_1 \cap \{y \leq a + 3, y \leq b + 2\}$,
- After two loops: $Z_2' = Z_2 \cap \{x \leq b - 2a + 2, y \leq a + 3, y - x \leq a + 2\}$,
- After three loops: $Z_3' = Z_3 \cap \{a \leq 1, y \leq a + 3, y \leq b + 3a\}$,
- After four loops: $Z_4' = Z_4 \cap \{y - x = 4a, x \leq 3 - 3a, x \leq b - a\}$,
- After $n > 4$ loops: $\mathbf{Z_n' = Z_{n+1}'} = \{a = 0, x = y, 0 \leq x \leq b, b \leq 3\}$.

Theoretical Complexity

- We can enumerate the parameter values in exponential time

Theorem

The ϕ-emptiness problem for PTA with bounded integer parameters is EXPTIME-complete for any class of problems ϕ that is EXPTIME-complete for TA.

- Better:

Theorem

The ϕ-emptiness problem for PTA with bounded integer parameters is PSPACE-complete for any class of problems ϕ that is PSPACE-complete for TA.

Proof by construction of a non-deterministic Turing machine that works in PSPACE

- TCTL model-checking for TA is PSPACE-complete [ACD93];

Properties Required from the Symbolic States

The proofs use (for any reachable symbolic state (l, Z)):

- Succ is non decreasing
- Z is convex
- $v(\text{Succ}((l, Z), e)) = \text{Succ}((l, v(Z)), v(e))$
- if v is an integer valuation of the parameters then $v(Z)$ has integer vertices
  $\Rightarrow$ does not work for, e.g., stopwatches

Outline

Introduction

Parametric Timed Automata

(Un)decidability

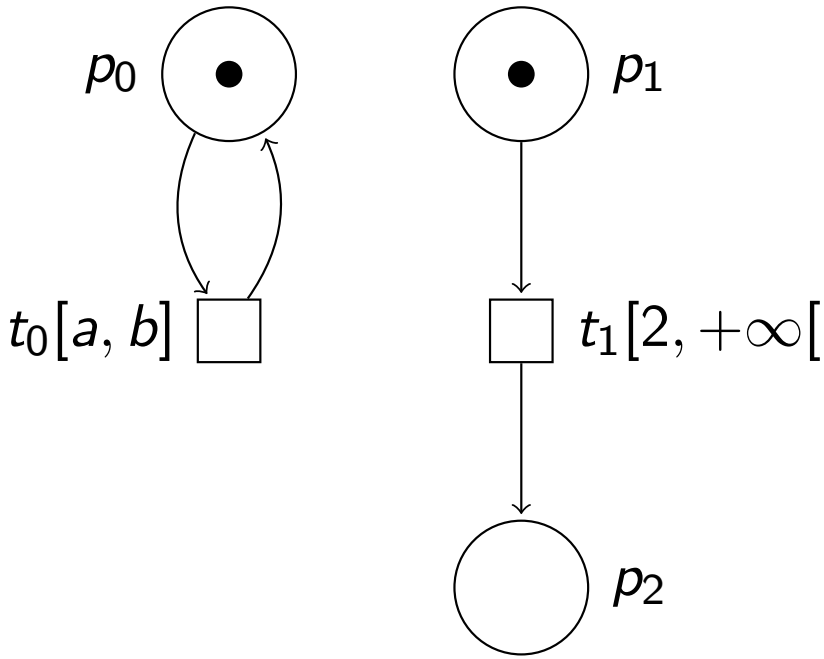
Symbolic Semi-algorithms

Computing the Integer Valuations

Time Petri Nets

Conclusion

Parametric Time Petri Nets



Results for Parametric TPNs

- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH$^+$05, CR06]);

Results for Parametric TPNs

- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH$^+$05, CR06]);
- There are two main symbolic states abstractions for TPN:

Results for Parametric TPNs

- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH$^+$05, CR06]);
- There are two main symbolic states abstractions for TPN:
  - The zone-based approach is **directly** applicable [GRR06];

Results for Parametric TPNs

- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH$^+$05, CR06]);
- There are two main symbolic states abstractions for TPN:
  - The zone-based approach is **directly** applicable [GRR06];
  - The historical approach is **state classes** [BD91]: they have **all** the good properties required before!

Results for Parametric TPNs

- There are several time-bisimulation preserving **transformations** between TA and TPNs allowing to transfer (un)decidability results (e.g. [BCH$^+$05, CR06]);
- There are two main symbolic states abstractions for TPN:
  - The zone-based approach is **directly** applicable [GRR06];
  - The historical approach is **state classes** [BD91]: they have **all** the good properties required before!
- In both cases, we can use the integer parameters approach.

Romeo: A Scheduling Problem

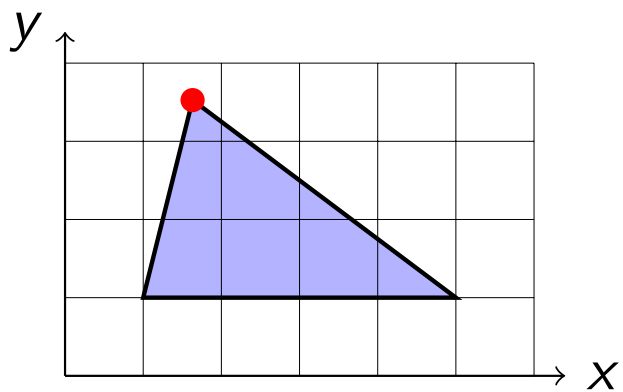
	$a \in [0, \infty)$ $b = 20$ $c = 18$ $d = 28$	$a \in [0, \infty)$ $b \in [10, \infty)$ $c = 18$ $d = 28$	$a \in [0, \infty)$ $b \in [10, \infty)$ $c \in [0, 28]$ $d = 28$	$a \in [0, \infty)$ $b \in [10, \infty)$ $c = 18$ $d \in [18, \infty)$	$a \in [0, \infty)$ $b \in [10, \infty)$ $c \geq 0$ $d \geq c$
IEF Time	1 s	2.8 s	27 s	840 s	DNF
Int. Hull	0.2 s (20%)	0.4 s (14%)	2.9 s (11%)	146 s (17%)	–
IEF Mem.	15 MB	35 MB	153 MB	1289 MB	–
EF Time	1.5 s	6.4 s	DNF	DNF	DNF
EF Mem.	19.6 MB	55 MB	–	–	–

Table : Scaling up the number of parameters in the schedulability problem

	$a = 50$	$a \in [40, 50]$	$a \in [0, 100]$	$a \in [0, 1000]$	$a \in [0, 10000]$
IEF Time	17 s	211 s	1079 s	1150 s	1178 s
Int. Hull	2 s (11.7%)	25.7 s (12.1%)	166 s (15.4%)	167 s (14.5%)	168 s (14.3%)
IEF Mem.	121 MB	425 MB	1598 MB	1667 MB	1667 MB

Table : Scaling up a's upper bound in the schedulability problem for $b \in [10, 100], c = 18$ and $d \in [18, 100]$

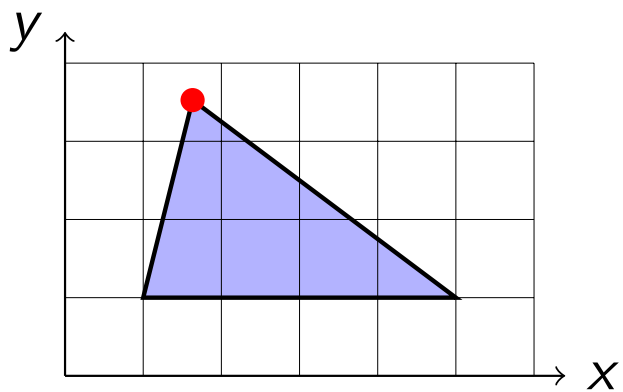
Computing the Integer Hull



We compute the **Gomory-Chvátal cuts** from integer linear programming (see, e.g. [Sch86])

- select the constraints defining a non-integer vertex:
  $y - 4x \leq -3$ and $3x + 4y \leq 19$

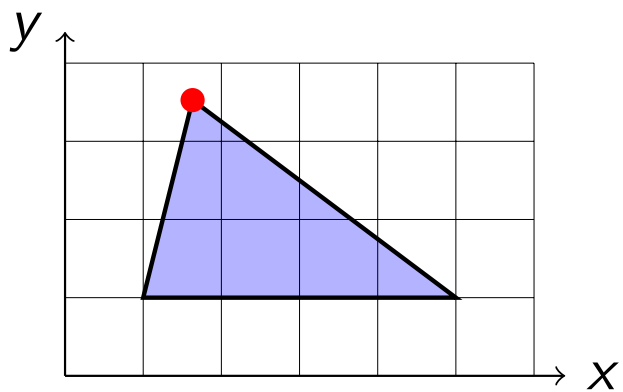
Computing the Integer Hull



We compute the **Gomory-Chvátal cuts** from integer linear programming (see, e.g. [Sch86])

- select the constraints defining a non-integer vertex:
  $y - 4x \leq -3$ and $3x + 4y \leq 19$
- add slack variables:
  $y - 4x + s = -3$ and $3x + 4y + t = 19$

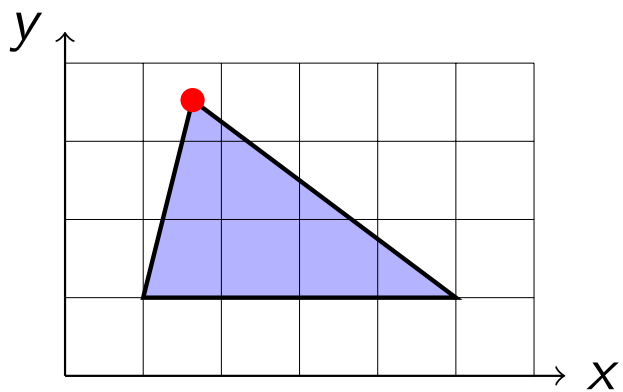
Computing the Integer Hull



We compute the **Gomory-Chvátal cuts** from integer linear programming (see, e.g. [Sch86])

- select the constraints defining a non-integer vertex:
  $y - 4x \leq -3$ and $3x + 4y \leq 19$
- add slack variables:
  $y - 4x + s = -3$ and $3x + 4y + t = 19$
- for a non-integer vertex coord. eliminate other non-slack variables:
  e.g. for y: $19y + 3s + 4t = 67$

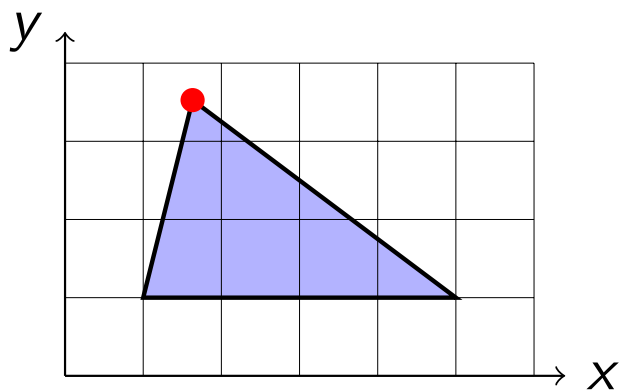
Computing the Integer Hull



We compute the **Gomory-Chvátal cuts** from integer linear programming (see, e.g. [Sch86])

- select the constraints defining a non-integer vertex:
  $y - 4x \leq -3$ and $3x + 4y \leq 19$
- add slack variables:
  $y - 4x + s = -3$ and $3x + 4y + t = 19$
- for a non-integer vertex coord. eliminate other non-slack variables:
  e.g. for y: $19y + 3s + 4t = 67$
- slacks are non negative: $y + \lfloor \frac{3}{19} \rfloor s + \lfloor \frac{4}{19} \rfloor t \leq \frac{67}{19}$

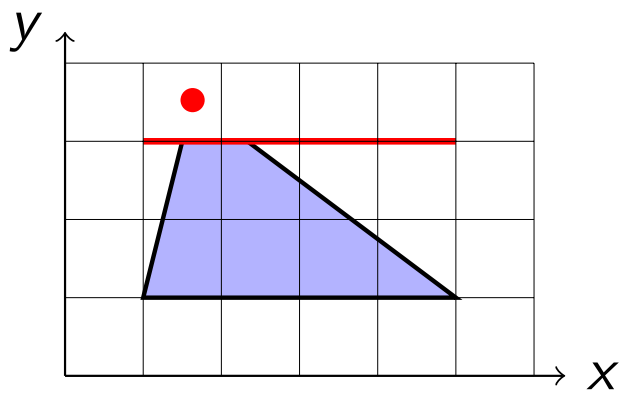
Computing the Integer Hull



We compute the **Gomory-Chvátal cuts** from integer linear programming (see, e.g. [Sch86])

- select the constraints defining a non-integer vertex:
  $y - 4x \leq -3$ and $3x + 4y \leq 19$
- add slack variables:
  $y - 4x + s = -3$ and $3x + 4y + t = 19$
- for a non-integer vertex coord. eliminate other non-slack variables:
  e.g. for y: $19y + 3s + 4t = 67$
- slacks are non negative: $y + \lfloor \frac{3}{19} \rfloor s + \lfloor \frac{4}{19} \rfloor t \leq \frac{67}{19}$
- y, s, t should be integers: $y + \lfloor \frac{3}{19} \rfloor s + \lfloor \frac{4}{19} \rfloor t \leq \lfloor \frac{67}{19} \rfloor$

Computing the Integer Hull



We compute the **Gomory-Chvátal cuts** from integer linear programming (see, e.g. [Sch86])

- select the constraints defining a non-integer vertex:
  $y - 4x \leq -3$ and $3x + 4y \leq 19$
- add slack variables:
  $y - 4x + s = -3$ and $3x + 4y + t = 19$
- for a non-integer vertex coord. eliminate other non-slack variables:
  e.g. for y: $19y + 3s + 4t = 67$
- slacks are non negative: $y + \lfloor \frac{3}{19} \rfloor s + \lfloor \frac{4}{19} \rfloor t \leq \frac{67}{19}$
- y, s, t should be integers: $y + \lfloor \frac{3}{19} \rfloor s + \lfloor \frac{4}{19} \rfloor t \leq \lfloor \frac{67}{19} \rfloor$
- Put the constraints in the above system and eliminate slacks: $y \leq 3$

Outline

Introduction

Parametric Timed Automata

(Un)decidability

Symbolic Semi-algorithms

Computing the Integer Valuations

Time Petri Nets

Conclusion

Performances Considerations

- **Termination** is guaranteed;
- We do **not** explicit **all** possible values;
- But we use **polyhedra** instead of **zones**;
- The computation of the integer hull is **not cheap** in practice
  Though most of the time is usually spent searching in the passed list

Practical Usefulness of Bounded Integer Values

- For (almost) all **practical** purposes, constants in TA or TPNs are chosen as **integers**;
- With adequate **scaling**, rationals can be transformed into integers
  This however sets the precision in advance
- The **symbolic** approach allows for **large** bounds;
- We need **no syntactical restriction** on the constraints in the PTA.

Conclusion and Perspectives

- Conclusion:
  - We have surveyed and completed a bit the theoretical results on L/U automata;
  - We offer an alternative with full PTA with **integer** parameters;
  - We argue this class has a great **practical** relevance;
  - We have shown how valuations can be computed **symbolically**;
  - The method also applies to **parametric TPNs**.

Conclusion and Perspectives

- Conclusion:
  - We have surveyed and completed a bit the theoretical results on L/U automata;
  - We offer an alternative with full PTA with **integer** parameters;
  - We argue this class has a great **practical** relevance;
  - We have shown how valuations can be computed **symbolically**;
  - The method also applies to **parametric TPNs**.
- Perspectives:
  - Improve the implementation;
  - Investigate models with **stopwatches**.
  - Prove that the symbolic results of the algorithms are actually **dense**
    - Gives "big" underapproximations;
    - Useful, e.g., for robustness issues;
    - Straightforward for EF;
    - Seemingly also true for quite generic properties.

References I


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


Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi.
Parametric real-time reasoning.
In *ACM Symposium on Theory of Computing*, pages 592–601, 1993.


Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux.
Comparison of the expressiveness of timed automata and time Petri nets.
In Paul Pettersson and Wang Yi, editors, *3rd International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS 2005)*, volume 3829 of *LNCS*, pages 211–225, Uppsala, Sweden, September 2005. Springer-Verlag.

References II


B. Berthomieu and M. Diaz.
Modeling and verification of time dependent systems using time Petri nets.
IEEE trans. on soft. eng., 17(3):259–273, 1991.


Laura Bozzelli and Salvatore La Torre.
Decision problems for lower/upper bound parametric timed automata.
Formal Methods in System Design, 35(2):121–151, 2009.


Franck Cassez and Olivier H. Roux.
Structural translation from Time Petri Nets to Timed Automata – Model-Checking Time Petri Nets via Timed Automata.
The journal of Systems and Software, 79(10):1456–1468, 2006.


L. Doyen.
Robust parametric reachability for timed automata.
Information Processing Letters, 102(5):208–213, 2007.

References III


Guillaume Gardey, Olivier H. Roux, and Olivier F. Roux.
State space computation and analysis of time Petri nets.
Theory and Practice of Logic Programming (TPLP). Special Issue on Specification Analysis and Verification of Reactive Systems, 6(3):301–320, 2006.


Thomas Hune, Judi Romijn, Marielle Stoelinga, and Frits Vaandrager.
Linear parametric model checking of timed automata.
Journal of Logic and Algebraic Programming, 52-53:183–220, 2002.


Joseph S. Miller.
Decidability and complexity results for timed automata and semi-linear hybrid automata.
In *Hybrid Systems: Computation and Control*, volume 1790 of *LNCS*, page 296–309. Springer, 2000.

References IV

Alexander Schrijver.
Theory of linear and integer programming.
John Wiley & Sons, Inc., New York, NY, USA, 1986.