Verifying (One-clock) Priced Timed Automata

Nicolas Markey

Joint works with Patricia Bouyer, Thomas Brihaye, Kim G. Larsen and Jacob Illum Rasmussen

Lab. Specification et Verification
ENS Cachan & CNRS, France

December 1, 2006
Verification & Model-Checking

system:

⇒

property:
Verification & Model-Checking

system:

property:

G(request ⇒ F grant)
Verification & Model-Checking

system:

⇒

property:

G(request⇒Fgrant)

model-checking algorithm

G(request⇒Fgrant)
Verification & Model-Checking

system:

⇒

property:

G(request ⇒ F grant)

model-checking algorithm

yes/no
Verification & Control

system:

property:

G(request ⇒ F grant)
Verification & Control

system:

⇒

property:

⇒

G(request⇒F grant)

controller synthesis
Verification & Control

system:

⇒

property:

G(request⇒F grant)

controller synthesis

yes/no
Verification & Control

system:
⇒
property:

G(request ⇒ F grant)

controller synthesis

yes/no
Adding timing requirements

- Need for timed models:
  - the behaviour of most systems depends on time;
  - (faithful) modelling has to take time into account;

\sim twed automata, timed Petri nets, timed process algebras, ...
Adding timing requirements

- Need for **timed models**:
  - the behaviour of most systems depends on time;
  - (faithful) modelling has to take time into account;

  \[\rightarrow\text{ timed automata, timed Petri nets, timed process algebras, ...}\]

- Need for **time in specification**:
  - again, the behaviour of most systems depends on time;
  - untimed specifications are not enough (e.g., *bounded response property*);

  \[\rightarrow\text{ TCTL, MTL, TPTL, timed }\mu\text{-calculus, ...}\]
Time is not always sufficient

In some cases, we don’t want to measure time, but rather energy consumption, price to pay for reaching some goal, ...
Time is not always sufficient

In some cases, we don’t want to measure time, but rather energy consumption, price to pay for reaching some goal, ...

- hybrid automata: timed automata augmented with variables whose derivative is not constant.

Examples: leaking gas burner, water-level monitor, ...

\[
\begin{align*}
  x &\leq 1 \\
  \dot{x} &= 1 \\
  \dot{y} &= 1 \\
  \dot{z} &= 1 \\
  x &\leq 1, x:=0 \\
  x &\geq 30, x:=0 \\
  \dot{x} &= 1 \\
  \dot{y} &= 1 \\
  \dot{z} &= 0 \\
  \text{true}
\end{align*}
\]

**Theorem (HKPV97)**

Reachability is undecidable (even for timed automata where one single “clock” has two derivatives).
Time is not always sufficient

In some cases, we don’t want to measure time, but rather energy consumption, price to pay for reaching some goal, ...

- hybrid automata: timed automata augmented with variables whose derivative is not constant.

≈ examples: leaking gas burner, water-level monitor, ...

\[
\begin{align*}
    x &\leq 1 \\
    \dot{x} &= 1 \\
    \dot{y} &= 1 \\
    \dot{z} &= 1
\end{align*}
\]

\[
\begin{align*}
    x &\leq 1, x := 0 \\
    x &\geq 30, x := 0
\end{align*}
\]

- priced timed automata: similar to hybrid automata, but the behavior only depends on clock variables.
Related work on priced timed automata

- **Basic properties**
  - Optimal reachability
  - Mean-cost optimality

- **Control games**
  - Properties, and restricted decidability results
  - Undecidability for timed game automata with more than three clocks
  - Decidability for timed automata with one clock

- **Model-checking of WCTL**
  - Undecidability for timed automata with more than three clocks
  - Decidability for timed automata with one clock
Outline of the talk

1 Introduction

2 Priced Timed Game Automata
   - Definitions and examples
   - Existence of optimal strategies in PTGAs is undecidable
   - Existence of optimal strategies in 1PTGAs is decidable
   - (Pseudo-)algorithm for computing the optimal cost

3 Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
   - Model-checking Priced CTL on PTAs is undecidable
   - Model-checking Priced CTL on 1PTAs is decidable

4 Conclusion
Outline of the talk

1 Introduction

2 Priced Timed Game Automata
   - Definitions and examples
     - Existence of optimal strategies in PTGAs is undecidable
     - Existence of optimal strategies in 1PTGAs is decidable
     - (Pseudo-)algorithm for computing the optimal cost

3 Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
   - Model-checking Priced CTL on PTAs is undecidable
   - Model-checking Priced CTL on 1PTAs is decidable

4 Conclusion
A *priced timed game automaton* is a timed automaton with costs where states are partitionned into *controllable* and *uncontrollable* ones:

\[ \mathcal{G} = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]
Priced timed game automata

Definition (ALP01, BFH+01)

A *priced timed game automaton* is a timed automaton with costs where states are partitioned into *controllable* and *uncontrollable* ones:

\[ \mathcal{G} = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]
Priced timed game automata

Definition (ALP01,BFH^+01)

A *priced timed game automaton* is a timed automaton with costs where states are partitionned into *controllable* and *uncontrollable* ones:

\[ G = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]
A **priced timed game automaton** is a timed automaton with costs where states are partitioned into *controllable* and *uncontrollable* ones:

\[ G = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{\text{urg}}, P \rangle \]
Priced timed game automata

Definition (ALP01, BFH^+01)

A *priced timed game automaton* is a timed automaton with costs where states are partitionned into *controllable* and *uncontrollable* ones:

\[ \mathcal{G} = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]

\[ AP = \{ a, b \} \]
Priced timed game automata

Definition (ALP01,BFH$^+$01)

A *priced timed game automaton* is a timed automaton with costs where states are partitioned into *controllable* and *uncontrollable* ones:

\[ G = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]

\[ \neg a, b \]

\[ a, b \]

\[ \neg a, \neg b \]

\[ AP = \{a, b\} \]
Priced timed game automata

Definition (ALP01, BFH⁺01)

A *priced timed game automaton* is a timed automaton with costs where states are partitioned into *controllable* and *uncontrollable* ones:

\[ \mathcal{G} = \langle Q_c \cup Q_u, Q_0, \text{AP, } \ell, \delta, C, G, R, I, Q_{\text{urg}}, P \rangle \]

\( \text{AP} = \{a, b\} \)
Priced timed game automata

Definition (ALP01,BFH⁺01)

A *priced timed game automaton* is a timed automaton with costs where states are partitionned into *controllable* and *uncontrollable* ones:

\[ G = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]

\[ AP = \{a, b\} \]

\[ C = \{x, y\} \]
Priced timed game automata

Definition (ALP01, BFH\(^+\)01)

A *priced timed game automaton* is a timed automaton with costs where states are partitioned into *controllable* and *uncontrollable* ones:

\[
G = \langle Q_c \cup Q_u, Q_0, \mathcal{AP}, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle
\]

\[
\begin{align*}
AP &= \{a, b\} \\
C &= \{x, y\}
\end{align*}
\]
Priced timed game automata

Definition (ALP01,BFH+01)

A \textit{priced timed game automaton} is a timed automaton with costs where states are partitioned into \textit{controllable} and \textit{uncontrollable} ones:

\[ G = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]

\[ AP = \{a, b\} \]
\[ C = \{x, y\} \]
Priced timed game automata

**Definition (ALP01,BFH⁺01)**

A priced timed game automaton is a timed automaton with costs where states are partitionned into *controllable* and *uncontrollable* ones:

\[ \mathcal{G} = \langle Q_c \cup Q_u, Q_0, \text{AP}, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]

**AP = \{a, b\}**

\[ C = \{x, y\} \]
Priced timed game automata

Definition (ALP01,BFH\(^{+}01\))

A *priced timed game automaton* is a timed automaton with costs where states are partitionned into *controllable* and *uncontrollable* ones:

\[ G = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]

\[ AP = \{a, b\} \]
\[ C = \{x, y\} \]
Priced timed game automata

Definition (ALP01, BFH+01)

A **priced timed game automaton** is a timed automaton with costs where states are partitioned into *controllable* and *uncontrollable* ones:

\[ \mathcal{G} = \langle Q_c \cup Q_u, Q_0, AP, \ell, \delta, C, G, R, I, Q_{urg}, P \rangle \]

\[ AP = \{a, b\} \]

\[ C = \{x, y\} \]
Example

\[
\begin{align*}
\dot{p} &= 5 \\
x &\leq 2 \\
y &:= 0 \\
\dot{y} &= 0 \\
\dot{p} &= 6 \\
x &\geq 3 \\
p &:= 1 \\
\dot{p} &= 1 \\
x &\geq 3 \\
p &:= 7
\end{align*}
\]

Minimal cost for reaching smiley:

\[
\inf_{0 \leq t \leq 2} \max \left(5t + 6(3-t) + 1, 5t + (3-t) + 7 \right) = 17.2
\]

(when \( t = 1.8 \))
Example

Minimal cost for reaching \( y = 0 \):
\[
\inf_{0 \leq t \leq 2} \max \left( 5t + 6(3-t) + 1, 5t + (3-t) + 7 \right) = 17.2 \quad \text{(when} \quad t = 1.8) \]
Example

Minimal cost for reaching a goal state:
\[
\inf_{0 \leq t \leq 2} \max (5t + 6(3 - t) + 1, 5t + (3 - t) + 7) = 17.2
\]
when \( t = 1.8 \).
Minimal cost for reaching 😃:

\[
\begin{align*}
\min_{0 \leq t \leq 2} & \max \left( 5t + 6(3-t) + 1, 5t + (3-t) + 7 \right) = 17.2 \\
& \text{when } t = 1.8
\end{align*}
\]
Minimal cost for reaching ☺:

\[ 5t + 6(3 - t) + 1 \]
Minimal cost for reaching 😊:

\[ 5t + 6(3 - t) + 1, \quad 5t + (3 - t) + 7 \]
Minimal cost for reaching 😊:

\[
\max (5t + 6(3 - t) + 1, 5t + (3 - t) + 7)
\]
Minimal cost for reaching 😊:

$$\inf_{0 \leq t \leq 2} \max (5t + 6(3 - t) + 1, 5t + (3 - t) + 7)$$
Minimal cost for reaching 😊:

\[
\inf_{0 \leq t \leq 2} \max (5t + 6(3 - t) + 1, 5t + (3 - t) + 7) = 17.2
\]

(when \( t = 1.8 \))
Strategies

Definitions

- \( \text{Run}(A, B) \) is the set of trajectories from some state in \( A \) to some state in \( B \);
- given \( \rho = ((q_i, v_i))_{i \in \mathbb{Z}^+} \), we write \( \rho_{\leq n} \) for the \( n \)-th prefix of \( \rho \);
- a strategy is a function
  \[
  \sigma : \text{Run}(Q \times \mathbb{R}^+ \cup C, Q_c \times \mathbb{R}^+ \cup C) \to \delta \cup \mathbb{R}^+ \]
Strategies

Definitions

- A run $\rho = ((q_i, v_i))_{i \in \mathbb{Z}^+}$ is **compatible** with a strategy $\sigma$ from step $i_0$ if, for each $i \geq i_0$ s.t. $q_i \in Q_c$,
  - if $\sigma(\rho_{\leq i}) = e \in \delta$ and $v_i \models I(q_i)$ and $v_i \models G(e)$, then $e = (q_i, q_{i+1})$ and $v_{i+1} = v_i[R(e) \leftarrow 0]$.
  - if $\sigma(\rho_{\leq i}) = r \in \mathbb{R}^+_\geq 0$ and, for all $t \in [0, r]$, $v_i + t \models I(q_i)$, then $q_{i+1} = q_i$ and $v_{i+1} = v_i + r$.

- A strategy $\sigma$ is **winning** (for some reachability objective $W \subseteq Q$) after some finite prefix $\rho_0$ if any “prolongation” of $\rho_0$ that is compatible with $\sigma$ after $\rho_0$, reaches a location in $W$. 
Strategies

Definitions

- the cost of a winning strategy $\sigma$ from $\rho_0$ is
  \[
  \text{Cost}(\sigma, \rho_0) = \sup \{ \text{cost}(\rho) \mid \rho \text{ compatible execution after } \rho_0 \}
  \]

  (assuming that the trajectory stops as soon as it enters any location in $W$).
Strategies

Definitions

- the **cost** of a winning strategy $\sigma$ from $\rho_0$ is

  \[
  \text{Cost}(\sigma, \rho_0) = \sup \{ \text{cost}(\rho) \mid \rho \text{ compatible execution after } \rho_0 \}
  \]

(assuming that the trajectory stops as soon as it enters any location in $W$).

Example

Consider strategy $\sigma$:

- in $\circ$, wait until $x = 2$;
- in $\tint$, wait until $x = 3$;
- in $\bullet$, wait until $x = 4$;
the cost of a winning strategy $\sigma$ from $\rho_0$ is

$$\text{Cost}(\sigma, \rho_0) = \sup \{ \text{cost}(\rho) \mid \rho \text{ compatible execution after } \rho_0 \}$$

(assuming that the trajectory stops as soon as it enters any location in $W$).

Consider strategy $\sigma$:
- in $\bigcirc$, wait until $x = 2$;
- in $\bigcirc$, wait until $x = 3$;
- in $\bullet$, wait until $x = 4$;

$\text{Cost}(\sigma, (\bigcirc, x = 0)) = \sup(17, 19) = 19$. 

\[\dot{p} = 5 \quad x \leq 2 \quad y := 0 \]
\[\dot{x} = 6 \quad p := 1 \quad x \geq 3 \quad p := 7 \]
\[\dot{y} = 0 \]
\[\bigcirc \bigcirc \bigcirc \bullet \bigcirc \]
Outline of the talk

1 Introduction

2 Priced Timed Game Automata
   - Definitions and examples
   - Existence of optimal strategies in PTGAs is undecidable
   - Existence of optimal strategies in 1PTGAs is decidable
   - (Pseudo-)algorithm for computing the optimal cost

3 Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
   - Model-checking Priced CTL on PTAs is undecidable
   - Model-checking Priced CTL on 1PTAs is decidable

4 Conclusion
Timed games are undecidable

**Theorem (BBR05,BBM06)**

The existence of a strategy with cost less than or equal to a given value is **undecidable** on PTGAs with (at least) three clocks.
Timed games are undecidable

**Theorem (BBR05,BBM06)**

The existence of a strategy with cost less than or equal to a given value is *undecidable* on PTGAs with (at least) three clocks.

*Proof.* Encoding of a two-counter machine.
Timed games are undecidable

Theorem (BBR05, BBM06)

The existence of a strategy with cost less than or equal to a given value is undecidable on PTGAs with (at least) three clocks.

Proof. Encoding of a two-counter machine.

We first define some auxiliary modules:

\[
\begin{align*}
x &= x_0 \in [0, 1] \\
y &= y_0 \in (0, 1) \\
z &= 0 \\
p &= p_0
\end{align*}
\]

\[
\begin{align*}
\dot{p} &= 0 \\
x &= 1 \\
y &= 0 \\
z &= 0
\end{align*}\quad
\begin{align*}
\dot{p} &= 1 \\
x &= 0 \\
y &= 0 \\
z &= 0
\end{align*}
\]

**Add**\(+^+ (x, \{z\})\)

\[
\begin{align*}
x &= x_0 \\
y &= y_0 \\
z &= 0 \\
p &= p_0 + x_0
\end{align*}
\]
Timed games are undecidable

**Theorem (BBR05,BBM06)**

The existence of a strategy with cost less than or equal to a given value is **undecidable** on PTGAs with (at least) three clocks.

**Proof.** Encoding of a two-counter machine.

We first define some auxiliary modules:

\[
\begin{align*}
x &= x_0 \in [0,1] \\
y &= y_0 \in (0,1) \\
z &= 0 \\
p &= p_0
\end{align*}
\]

\[
\begin{align*}
\text{Add}^{-}(x, \{z\}) \\
p = p_0 + 1 - x_0
\end{align*}
\]
Timed games are undecidable

*Proof.* Encoding of a two-counter machine. We first define some auxiliary modules:

\[
\text{Test}(\ y = 2x, \ \{z\} \ )
\]

\[
\dot{p} = 0
\]

\[
t_0
\]

\[
\begin{align*}
\text{Add}^+(x, \{z\}) & \quad \text{Add}^+(x, \{z\}) & \quad \text{Add}^-(y, \{z\}) \\
\text{Add}^-(x, \{z\}) & \quad \text{Add}^-(x, \{z\}) & \quad \text{Add}^+(y, \{z\})
\end{align*}
\]

\[
\dot{p} = 0 \quad p = 0 \quad p = 0
\]

\[
t_1
\]

\[
z = 0 \\
z = 0
\]

\[
\begin{align*}
z &= 0 \\
z &= 0
\end{align*}
\]

\[
y_0 = 2x_0 \quad \text{iff} \quad \text{we reach } t_1 \text{ with cost less than or equal to } 3.
\]
Timed games are undecidable

Proof. Encoding of a two-counter machine.

We first define some auxiliary modules:

\[
\begin{align*}
\text{Power}_2(x, \{y,z\}) &= \\
&= x=1, \ x:=0 \\
&\quad \rightarrow z=1 \land x \leq 1 \\
&\quad \rightarrow \hat{p}=0 \\
&\quad \rightarrow \text{Test}(y = 2x, \{z\}) \\
&\quad \rightarrow z=0 \\
&\quad \rightarrow x=1 \land z=0 \\
&\rightarrow p_3 \\
\end{align*}
\]

\[
x_0 = 2^{-k} \quad \text{iff} \quad \text{we reach } t_1 \text{ or } p_3 \text{ with cost less than or equal to 3.}
\]
Timed games are undecidable

**Proof.** Encoding of a two-counter machine.

Counters are encoded by $x_1 = \frac{1}{2c_1+1}, \quad x_2 = \frac{1}{3c_2+1}$

$p_j : c_1 := c_1 + 1; \text{ goto } p_k$

The execution halts iff we can reach $\text{Halt}$ or $t_1$ or $p_3$ with cost $\leq 3$. 
Timed games are undecidable

**Proof.** Encoding of a two-counter machine.

Counters are encoded by $x_1 = \frac{1}{2c_1+1}$, $x_2 = \frac{1}{3c_2+1}$

$p_j \colon c_1 := c_1 + 1; \text{ goto } p_k$

The execution halts iff we can reach \texttt{Halt} or $t_1$ or $p_3$ with cost $\leq 3$. 
Outline of the talk

1. Introduction

2. Priced Timed Game Automata
   - Definitions and examples
   - Existence of optimal strategies in PTGAs is undecidable
   - Existence of optimal strategies in 1PTGAs is decidable
   - (Pseudo-)algorithm for computing the optimal cost

3. Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
   - Model-checking Priced CTL on PTAs is undecidable
   - Model-checking Priced CTL on 1PTAs is decidable

4. Conclusion
Strategies

Definitions

- Run\((A, B)\) is the set of trajectories from some state in \(A\) to some state in \(B\);
- given \(\rho = ((q_i, x_i))_{i \in \mathbb{Z}^+}\), we write \(\rho \leq n\) for the \(n\)-th prefix of \(\rho\);
- a strategy is a function
  \[\sigma : \text{Run}(Q \times \mathbb{R}^+, Q_c \times \mathbb{R}^+) \rightarrow \delta \cup \mathbb{R}^+_>\]
- a strategy is memoryless if it only depends on the present state:
  \[\sigma : Q_c \times \mathbb{R}^+ \rightarrow \delta \cup \mathbb{R}^+_>\]
Strategies

Definitions

- **the cost** of a winning strategy $\sigma$ from $\rho_0$ is
  \[
  \text{Cost}(\sigma, \rho_0) = \sup\{\text{cost}(\rho) \mid \rho \text{ compatible execution after } \rho_0\}
  \]
  (assuming that the trajectory stops as soon as it enters any location in $W$).

- **the optimal cost** of winning from some state $s$ is
  \[
  \text{OptCost}(s) = \inf\{\text{Cost}(\sigma, \rho_0) \mid \rho_0 \text{ ending in } s, \sigma \text{ winning strategy}\}
  \]

- A strategy $\sigma$ is $\varepsilon$-optimal from a trajectory $\rho_0$ ending in $s$ if
  \[
  \text{OptCost}(s) \leq \text{Cost}(\sigma, \rho_0) \leq \text{OptCost}(s) + \varepsilon
  \]

- A strategy is **optimal** if it is 0-optimal.
Memorylessness and optimality

**Fact**

*In our PTGAs, optimal strategies do not always exist.*

**Example**

In this example, only $\varepsilon$-optimal strategies exist, for any $\varepsilon > 0$. 
Memorylessness and optimality

**Fact**

*In our PTGAs, optimal strategies do not always exist.*

**Fact**

*When optimal strategies exist, they might require some memory.*

**Example**

The **optimal strategy** depends on the date at which the blue state is entered.
Memorylessness and optimality

Fact

In our PTGAs, optimal strategies do not always exist.

Fact

When optimal strategies exist, they might require some memory.

Example

The optimal strategy depends on the date at which the blue state is entered. But there is a memoryless $\varepsilon$-optimal strategy.
Decidability of 1PTGAs

Definition

Given $\varepsilon > 0$ and $N \in \mathbb{Z}^+$, a strategy $\sigma$ is $(\varepsilon, N)$ acceptable if

- $\sigma$ is $\varepsilon$-optimal and memoryless,
- there is a partition $(I_n)_{n \leq N}$ of $[0, M]$ (where $M$ is the maximal constant of the guards and invariants of the game) s.t., for any $q \in Q_c$, $x \mapsto \sigma(q, x)$ is constant on each $I_n$. 
Decidability of 1PTGAs

Definition

Given $\varepsilon > 0$ and $N \in \mathbb{Z}^+$, a strategy $\sigma$ is $(\varepsilon, N)$ acceptable if

- $\sigma$ is $\varepsilon$-optimal and memoryless,
- there is a partition $(I_n)_{n \leq N}$ of $[0, M]$ (where $M$ is the maximal constant of the guards and invariants of the game) s.t., for any $q \in Q_c$, $x \mapsto \sigma(q, x)$ is constant on each $I_n$.

Main Theorem

- For every location, the optimal cost is computable and is piecewise affine.
- There exists $N \in \mathbb{Z}^+$ s.t., for any $\varepsilon > 0$, we can effectively compute an $(\varepsilon, N)$-acceptable (thus, almost-optimal and memoryless) strategy.
Simplifying the problem

We restrict to TGAs with maximal constant 1 (in clock constraints)
Simplifying the problem

We restrict to TGAs with maximal constant 1 (in clock constraints)

Example

\[ \dot{p} = 2, \quad x \leq 4 \]
\[ x < 3 \]
\[ x \geq 2 \]

\[ \dot{p} = 2, \quad x \leq 1 \]
\[ x = 1, \quad x := 0 \]

\[ \dot{p} = 2, \quad x \leq 1 \]
\[ x = 1, \quad x := 0 \]

\[ \dot{p} = 2, \quad x \leq 1 \]
\[ x = 1, \quad x := 0 \]
Simplifying the problem

We restrict to strongly-connected TGAs without resets.
Simplifying the problem

We restrict to strongly-connected TGAs without resets.

Example
Simplifying the problem

We restrict to strongly-connected TGAs without resets.

Example

\[
\begin{align*}
\dot{p} &= 1 \\
\dot{p} &= 4 \\
\dot{p} &= 3 \\
\dot{p} &= 1 \\
x &\leq 1 \\
x &= 0 \\
x &= 0
\end{align*}
\]
Simplifying the problem

We restrict to strongly-connected TGAs without resets.

Example

\[
\begin{aligned}
\dot{p} &= 1 \\
\dot{p} &= 4 \\
\dot{p} &= 3 \\
\dot{p} &= 1
\end{aligned}
\]

\[
\begin{aligned}
x &\leq 1 \\
x &\leq 0 \\
x &\leq 1 \\
x &\leq 0
\end{aligned}
\]
Simplifying the problem

We restrict to strongly-connected TGAs without resets.

Example

\[ \dot{p} = 1 \quad \dot{p} = 4 \]
\[ \dot{p} = 3 \quad \dot{p} = 1 \]
\[ x \leq 1 \quad x \leq 1 \]
\[ x = 0 \quad x = 0 \]
\[ x = 0 \quad +\infty \]

\[ x = 0 \]
Simplifying the problem

We restrict to strongly-connected TGAs without resets.

\[ G \]

\[ G' \]

\[ x := 0 \]

\[ m \]

\[ m_1 \]

\[ n \]

\[ n_2 \]

\[ +\infty \]
Simplifying the problem

We restrict to strongly-connected TGAs without resets.

Let $\sigma$ be a winning strategy in $G$.

$$\sigma'(\rho, (q_1, x)) = \begin{cases} \sigma(\rho, (q, x)) & \text{if not } m \rightarrow n \\ m_1 \rightarrow n_2 & \text{otherwise} \end{cases}$$

$$\sigma'(\rho, (q_2, x)) = \sigma(\rho', (q, x))$$

where $\rho = \rho_1 \rho_2$ and $\rho' = \rho_1 \rho^* \rho_2$ s.t.

the outcomes of $\sigma$ from $\rho'$ never fire

transition $m \rightarrow n$ any more.

$$\text{OptCost}_{G'}(q_1, x) \leq \text{OptCost}_G(q, x).$$
Simplifying the problem

We restrict to strongly-connected TGAs without resets.

\[
G_m^n x := 0 \\
G'_m^n x := 0 \\
\infty
\]

Any (possibly memoryfull) winning strategy \( \sigma' \) in \( G' \) can be mimicked by a memoryfull strategy \( \sigma \) in \( G \) having at most the same cost:

\[
\text{OptCost}_{G}(q, x) \leq \text{OptCost}_{G'}(q, x).
\]
Simplifying the problem

We restrict to strongly-connected TGAs without resets.

\[ G_m^n(x := 0) \]

Any (possibly memoryfull) winning strategy \( \sigma' \) in \( G' \) can be mimicked by a memoryfull strategy \( \sigma \) in \( G \) having at most the same cost:

\[ \text{OptCost}_{G}(q, x) \leq \text{OptCost}_{G'}(q, x). \]

Theorem

\[ \text{OptCost}_{G}(q, x) = \text{OptCost}_{G'}(q_1, x). \]
Simplifying the problem

We restrict to strongly-connected TGAs without resets.

Theorem
If $\sigma'$ is $(\varepsilon', N')$-acceptable in $G'$, then

$$\sigma(q, x) = \begin{cases} 
\sigma'(q_2, x) & \text{if } \text{Cost}(q_2, x) \leq \text{Cost}(q_1, x) \\
\sigma'(q_1, x) & \text{otherwise}
\end{cases}$$

is $(2\varepsilon', N')$-acceptable in $G$. 

\[ +\infty \]

\[ m \]
\[ n \]
\[ \downarrow \]
\[ x:=0 \]

\[ m_1 \]
\[ n_2 \]
\[ \downarrow \]
\[ x:=0 \]
Simplifying the problem

Reduced to
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Simplifying the problem

Reduced to
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Simplifying the problem

Reduced to
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Simplifying the problem

Reduced to:
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Simplifying the problem

Reduced to
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Simplifying the problem

Reduced to
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Simplifying the problem

Reduced to
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Simplifying the problem

Reduced to
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Simplifying the problem

Reduced to
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Simplifying the problem

Reduced to
- strongly-connected PTGAs
- clock is bounded by 1
- no resetting transitions.
Outside cost-functions

**Theorem**

Let $G$ be a strongly-connected non-resetting 1PTGA with outside cost-functions.

- OptCost$_G$ is computable;
- in each location, function $x \mapsto \text{OptCost}_G(q, x)$ is decreasing, piecewise affine and continuous. Its finitely many segments either have slope $-c$ where $c$ is the price of some locations, or are fragments of the outside cost-functions;
- There exists $N \in \mathbb{Z}^+$ s.t., for any $\varepsilon > 0$, we can compute an $(\varepsilon, N)$-acceptable strategy $\sigma$. 
Operations on cost functions: controllable locations

\[ \dot{p} = 3 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]

\[ \dot{p} = 5 \]
Operations on cost functions: controllable locations

\[ \dot{p} = 3 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]
Operations on cost functions: controllable locations

\[ \dot{p} = 3 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]

Legend:
- Yellow line: \( \dot{p} = 5 \)
- Red line: \( \dot{p} = 3 \)
- Green line: \( \dot{p} = 2 \)
- Blue line: \( \dot{p} = 1 \)
Operations on cost functions: controllable locations

\[ \dot{p} = 3 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]
Operations on cost functions: controllable locations

\[ \dot{p} = 3 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]
Operations on cost functions: controllable locations

\[ \dot{p} = 3 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]
Operations on cost functions: controllable locations

\[ \dot{p} = 3 \]

\[ \min_r (f_1, f_2)(x) = \min_{x \leq x' \leq 1} \left( r \cdot (x - x') + \min(f_1, f_2)(x') \right) \]
Operations on cost functions: uncontrollable locations

\[ \dot{p} = 2 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]
Operations on cost functions: uncontrollable locations

\[ \dot{p} = 2 \]

\[ \dot{p} = 5 \]
\[ \dot{p} = 3 \]
\[ \dot{p} = 2 \]
\[ \dot{p} = 1 \]
Operations on cost functions: uncontrollable locations

\[ \dot{p} = 2 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]
Operations on cost functions: uncontrollable locations

\[ \dot{p} = 2 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]
Operations on cost functions: uncontrollable locations

\[ \dot{p} = 2 \]

\[
\begin{align*}
\dot{p} &= 5 \\
\dot{p} &= 3 \\
\dot{p} &= 2 \\
\dot{p} &= 1
\end{align*}
\]
Operations on cost functions: uncontrollable locations

\[ \dot{p} = 2 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 2 \]

\[ \dot{p} = 1 \]
\[
\dot{p} = 2
\]

\[
\max_r (f_1, f_2)(x) = \max_{x \leq x' \leq 1} \left( r \cdot (x - x') + \max(f_1, f_2)(x') \right)
\]
**Inductive proof**

**Ideas of the proof**

*Induction on the number of non-urgent locations in the SCC*

- **base cases:**
  - all locations are urgent (thus uncontrollable);
  - there is only one location, which is controllable (thus non-urgent).

- **induction step:**
  we consider one of the non-urgent locations having minimal cost rate:
  - if it is controllable, we create two SCCs having one less non-urgent location;
  - if it is uncontrollable, we make it urgent and add an extra outside cost function to which it can go.
Inductive proof – base cases

\[ p = 1 \]

\[ p = 3 \]

\[ p = 5 \]

\[ \dot{p} = 1 \]

\[ x \leq 1 \]
Inductive proof – base cases

\[ p = 1 \]

\[ x \leq 1 \]
Inductive proof – base cases

\[ \dot{p} = 1 \quad \dot{p} = 3 \quad \dot{p} = 5 \quad \dot{p} = 1 \]

\[ x \leq 1 \]
Inductive proof – base cases
Inductive proof – base cases

\[ \dot{p} = 3 \]
Inductive proof – base cases
Inductive proof – base cases

\[ \dot{p}=3 \]
Inductive proof – base cases

\[ \dot{p} = 3 \]
Inductive proof – inductive cases

When $q_{\text{min}}$ is controllable:

Let $\sigma$ be a winning strategy. Assume there exists an outcome of $\sigma$ s.t.:

$(q_{\text{min}}, u) \rightarrow^* (q_{\text{min}}, v) \rightarrow^* \text{win}$ with $0 \leq u < v \leq 1$.

Then $\sigma$ is not optimal: waiting in $q_{\text{min}}$ would have been cheaper.
Inductive proof – inductive cases

- When $q_{\text{min}}$ is controllable:

Let $\sigma$ be a winning strategy.

\[
\begin{align*}
\dot{p} &= 3 \\
\dot{p} &= 5 \\
\dot{p} &= 1 \\
\dot{p} &= 2 \\
x &\leq 1
\end{align*}
\]
Inductive proof – inductive cases

- When $q_{\text{min}}$ is controllable:

Let $\sigma$ be a winning strategy.

Assume there exists an outcome of $\sigma$ s.t.:

$$(q_{\text{min}}, u) \xrightarrow{*} (q_{\text{min}}, v) \xrightarrow{*} \text{win}$$

with $0 \leq u < v \leq 1$. 
Inductive proof – inductive cases

- When $q_{\text{min}}$ is controllable:

Let $\sigma$ be a winning strategy.

Assume there exists an outcome of $\sigma$ s.t.:

$$(q_{\text{min}}, u) \rightarrow^* (q_{\text{min}}, v) \rightarrow^* \text{win}$$

with $0 \leq u < v \leq 1$.

Then $\sigma$ is not optimal: waiting in $q_{\text{min}}$ would have been cheaper.
Inductive proof – inductive cases

- When $q_{\min}$ is controllable:
Inductive proof – inductive cases

- When \( q_{\text{min}} \) is controllable:
Inductive proof – inductive cases

- When $q_{\text{min}}$ is controllable:
Inductive proof – inductive cases

- When $q_{\text{min}}$ is controllable:

\[ G \]

\[ G' \]

Any (possibly memoryfull) winning strategy $\sigma'$ in $G'$ can be mimicked by a memoryfull strategy $\sigma$ in $G$ having at most the same cost:

\[ \text{OptCost}_{G}(q, x) \leq \text{OptCost}_{G'}(q_1, x) \]

Let $\sigma$ winning in $G$. Then there exists $\sigma_1$ winning in $G$ that visits $q_{\text{min}}$ at most once. That strategy can be played in $G'$.
Inductive proof – inductive cases

When $q_{\text{min}}$ is controllable:

Any (possibly memoryfull) winning strategy $\sigma'$ in $G'$ can be mimicked by a memoryfull strategy $\sigma$ in $G$ having at most the same cost:

$$\text{OptCost}_{G}(q, x) \leq \text{OptCost}_{G'}(q_1, x).$$
Inductive proof – inductive cases

- When $q_{\text{min}}$ is controllable:

Any (possibly memoryfull) winning strategy $\sigma'$ in $G'$ can be mimicked by a memoryfull strategy $\sigma$ in $G$ having at most the same cost:

$$\text{OptCost}_{G}(q, x) \leq \text{OptCost}_{G'}(q_{1}, x).$$

Let $\sigma$ winning in $G$. Then there exists $\sigma_{1}$ winning in $G$ that visits $q_{\text{min}}$ at most once. That strategy can be played in $G'$.

$$\text{OptCost}_{G'}(q_{1}, x) \leq \text{OptCost}_{G}(q, x).$$
Inductive proof – inductive cases

- When $q_{\text{min}}$ is controllable:

Let $\sigma'$ be an ($\varepsilon', N'$)-acceptable strategy for $G'$. Let

$$
\sigma(q, x) = \begin{cases} 
\sigma'(q_2, x) & \text{if } \text{Cost}_{G'}(q_2, x) \leq \text{OptCost}_{G'}(q_{\text{min}}, x) \\
\sigma'(q_1, x) & \text{otherwise}
\end{cases}
$$

Then $\sigma$ is ($3\varepsilon', N$)-acceptable in $G$, for some $N$ independant of $\varepsilon'$. 
Inductive proof – inductive cases

When $q_{\text{min}}$ is uncontrollable:

\[ \dot{p} = 1 \]
\[ \dot{p} = 3 \]
\[ \dot{p} = 5 \]
\[ \dot{p} = 1 \]
\[ x \leq 1 \]
Inductive proof – inductive cases

- When $q_{\text{min}}$ is uncontrollable:

\[
\dot{p} = 1 \\
\dot{p} = 1 \\
\dot{p} = 5 \\
\dot{p} = 1 \\
x \leq 1
\]

Make $q_{\text{min}}$ urgent and apply I.H.
Inductive proof – inductive cases

- When $q_{\text{min}}$ is uncontrollable:

  \[ \dot{p} = 1 \]
  \[ \dot{p} = 5 \]

  Make $q_{\text{min}}$ urgent and apply I.H.:
Inductive proof – inductive cases

- When $q_{\text{min}}$ is uncontrollable:

Make $q_{\text{min}}$ urgent and apply I.H.:

First instance where slope less than $c_{\text{min}}$
Inductive proof – inductive cases

- When $q_{\text{min}}$ is uncontrollable:

  $\dot{p} = 1$

  $\dot{p} = 3$

  $\dot{p} = 5$

  $\dot{p} = 1$

  Make $q_{\text{min}}$ urgent and apply I.H.:

  It’s better to wait in $q_{\text{min}}$...
Inductive proof – inductive cases

- When \( q_{\text{min}} \) is uncontrollable:

\[
\dot{p} = 1 \\
\dot{p} = 3 \\
\dot{p} = 5 \\
\dot{p} = 1
\]

\( x \leq 1 \)

Make \( q_{\text{min}} \) urgent and apply I.H.:
Inductive proof – inductive cases

- When $q_{\text{min}}$ is uncontrollable:

\[ \dot{p} = 1 \quad \dot{\rho} = 1 \quad \dot{p} = 5 \quad \dot{p} = 3 \quad x \leq 1 \]

Apply I.H. again:
Inductive proof – inductive cases

- When $q_{min}$ is uncontrollable:

\[ \dot{p} = 1 \]

\[ \dot{p} = 3 \]

\[ x \leq 1 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 1 \]

Apply I.H. again:
Inductive proof – inductive cases

- When $q_{\text{min}}$ is uncontrollable:

  \[
  \begin{align*}
  \dot{p} &= 1 \\
  \dot{p} &= 5 \\
  \dot{p} &= 3 \\
  \dot{p} &= 1 \\
  x &\leq 1
  \end{align*}
  \]

  Apply I.H. again:

  First instance where slope less than $c_{\text{min}}$
Inductive proof – inductive cases

- When $q_{\text{min}}$ is uncontrollable:

Apply I.H. again:

It’s better to wait in $q_{\text{min}}$...
Inductive proof – inductive cases

When $q_{\min}$ is uncontrollable:

This procedure terminates because fragments having slope less than $c_{\min}$ are fragments of outside functions.
Outline of the talk

1. Introduction

2. Priced Timed Game Automata
   - Definitions and examples
   - Existence of optimal strategies in PTGAs is undecidable
   - Existence of optimal strategies in 1PTGAs is decidable
   - (Pseudo-)algorithm for computing the optimal cost

3. Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
   - Model-checking Priced CTL on PTAs is undecidable
   - Model-checking Priced CTL on 1PTAs is decidable

4. Conclusion
Example (iterative pseudo-algorithm of [BCFL04])

\[ \dot{p} = 1 \]

\[ \dot{p} = 5 \]

\[ x \leq 1 \]
Example (iterative pseudo-algorithm of [BCFL04])

\[ \dot{p} = \begin{cases} 1 & \text{if } x \leq 1 \\ 3 & \text{otherwise} \end{cases} \]
Example (iterative pseudo-algorithm of [BCFL04])

\[ \dot{p} = 1 \]

\[ \dot{p} = 3 \]

\[ x \leq 1 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 1 \]
Example (iterative pseudo-algorithm of [BCFL04])

\[
\begin{align*}
\dot{p} &= 1 \\
\dot{p} &= 3 \\
\dot{p} &= 5 \\
\dot{p} &= 1
\end{align*}
\]

\[x \leq 1\]
Example (iterative pseudo-algorithm of [BCFL04])

\[ \dot{p} = 1 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 1 \]

\[ x \leq 1 \]
Example (iterative pseudo-algorithm of [BCFL04])

\[ \dot{p} = 1 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 5 \]

\[ x \leq 1 \]
Example (iterative pseudo-algorithm of [BCFL04])

\[
\dot{p} = 1
\]

\[
\dot{p} = 3
\]

\[
\dot{p} = 5
\]

\[
x \leq 1
\]
Example (iterative pseudo-algorithm of [BCFL04])

\[ \dot{p} = 1 \]

\[ \dot{p} = 5 \]

\[ x \leq 1 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 1 \]
Example (iterative pseudo-algorithm of [BCFL04])

\[
\begin{align*}
\dot{p} &= 1 \\
\dot{p} &= 3 \\
\dot{p} &= 5 \\
\dot{p} &= 1 \quad (x \leq 1)
\end{align*}
\]
Example (iterative pseudo-algorithm of [BCFL04])
Example (iterative pseudo-algorithm of [BCFL04])

\[ p = 1 \]

\[ p = 5 \]

\[ p = 3 \]

\[ x \leq 1 \]
Example (iterative pseudo-algorithm of [BCFL04])

\[ \dot{p} = 1 \]
\[ \dot{p} = 5 \]
\[ x \leq 1 \]
\[ \dot{p} = 3 \]
\[ \dot{p} = 1 \]
Example (iterative pseudo-algorithm of [BCFL04])
Example (iterative pseudo-algorithm of [BCFL04])

\[
\begin{align*}
\dot{p} = 1 & \quad \dot{p} = 3 \\
\dot{p} = 5 & \quad \dot{p} = 1 \\
\end{align*}
\]

\(x \leq 1\)
Example (iterative pseudo-algorithm of [BCFL04])

\[ \dot{p} = 1 \]

\[ \dot{p} = 3 \]

\[ \dot{p} = 5 \]

\[ \dot{p} = 1 \]

\[ x \leq 1 \]
Example (iterative pseudo-algorithm of [BCFL04])

**Theorem**

*This algorithm terminates on 1PTGAs.*
Example (iterative pseudo-algorithm of [BCFL04])

**Theorem**

This algorithm terminates on 1PTGAs.

**Proof.**

- The cost functions computed at round $i$ represent the cost of winning in at most $i$ steps.
- Since there exists $N \in \mathbb{Z}^+$ s.t., for any $\varepsilon > 0$, there exists an $(\varepsilon, N)$-acceptable strategy, we know that there exists $\varepsilon$-optimal strategies that are guaranteed to win in at most $N \times |Q|$ steps.
Outline of the talk

1. Introduction

2. Priced Timed Game Automata
   - Definitions and examples
   - Existence of optimal strategies in PTGAs is undecidable
   - Existence of optimal strategies in 1PTGAs is decidable
   - (Pseudo-)algorithm for computing the optimal cost

3. Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
   - Model-checking Priced CTL on PTAs is undecidable
   - Model-checking Priced CTL on 1PTAs is decidable

4. Conclusion
A priced timed automaton is a timed automaton extended with a cost function: $\mathcal{P} = \langle Q, Q_0, AP, \ell, \delta, C, G, R, I, P \rangle$
Outline of the talk

1. Introduction

2. Priced Timed Game Automata
   - Definitions and examples
   - Existence of optimal strategies in PTGAs is undecidable
   - Existence of optimal strategies in 1PTGAs is decidable
   - (Pseudo-)algorithm for computing the optimal cost

3. Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
     - Model-checking Priced CTL on PTAs is undecidable
     - Model-checking Priced CTL on 1PTAs is decidable

4. Conclusion
Priced (or Weighted) CTL

Definition

\[
WCTL \ni \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid E \varphi U_{\sim c} \varphi
\]

where \( a \in \text{AP} \), \( \sim \in \{<, \leq, =, \geq, >\} \), and \( c \in \mathbb{Z}^+ \).
Priced (or Weighted) CTL

**Definition**

\[ \text{WCTL} \models \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid E\varphi U_{\sim c} \varphi \]

where \( a \in \text{AP} \), \( \sim \in \{<, \leq, =, \geq, >\} \), and \( c \in \mathbb{Z}^+ \).

**Example**

\[
\begin{align*}
\dot{p} &= 5 \\
& \xleftarrow{y := 0} y = 0 \\
& \xrightarrow{x \leq 2} \dot{p} = 6 \\
& \xrightarrow{x \geq 3, p++=1} \dot{p} = 6 \\
& \xrightarrow{x \geq 3, p++=7} \dot{p} = 1 \\
& \xrightarrow{} \dot{p} = 1 \\
& \xrightarrow{} \text{E F } p \leq 10
\end{align*}
\]
Priced (or Weighted) CTL

**Definition**

\[ WCTL \ni \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid E\varphi U_{\sim c} \varphi \]

where \( a \in AP, \sim \in \{<, \leq, =, \geq, >\} \), and \( c \in Z^+ \).

**Example**

![Example Diagram]
Priced (or Weighted) CTL

**Definition**

\[ WCTL \ni \varphi ::= a \mid \neg \varphi \mid \varphi \lor \varphi \mid E\varphi U\sim c \varphi \]

where \( a \in AP \), \( \sim \in \{<, \leq, =, \geq, >\} \), and \( c \in \mathbb{Z}^+ \).

**Example**

\[ A \mathcal{G}_{t=2} (E F p \leq 6 \circ) \]
Outline of the talk

1 Introduction

2 Priced Timed Game Automata
   - Definitions and examples
   - Existence of optimal strategies in PTGAs is undecidable
   - Existence of optimal strategies in 1PTGAs is decidable
   - (Pseudo-)algorithm for computing the optimal cost

3 Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
     - Model-checking Priced CTL on PTAs is undecidable
     - Model-checking Priced CTL on 1PTAs is decidable

4 Conclusion
The power of costs

- Adding the value of a clock to the cost:

\[
\dot{p} = 0 \\
\ell_0 \\
z := 0 \\
x := 0 \\
y = 1, \ y := 0
\]

\[
\dot{p} = 1 \\
\ell_1 \\
z := 0 \\
x := 0 \\
y = 1, \ y := 0
\]

Add\(^+\)\((x, \{z\})\)
The power of costs

- Adding the value of a clock to the cost:

\[
\dot{p} = 0 \\
\dot{\ell}_0 = 1 \\
\ell_1 = 0 \\
\dot{z} = 0 \\
\dot{x} = 1 \\
\dot{y} = 1 \\
\dot{p} = 0 + x_0
\]

\[x, y, z, p \in [0, 1], (0, 1), 0, p_0\]
The power of costs

- Adding the value of a clock to the cost:

  \[ \dot{p} = 0 \quad \ell_0 \quad z = 0 \]

  \[ \dot{p} = 1 \quad \ell_1 \quad x = 1 \]

  \[ y = 1, \ y := 0 \quad y = 1, \ y := 0 \]

  \[ \text{Add}^+(x, \{ z \}) \]

- Subtracting the value of a clock to the cost:

  \[ \dot{p} = 0 \quad \ell_0 \quad x = 0 \]

  \[ \dot{p} = 1 \quad \ell_1 \quad z = 0 \]

  \[ y = 1, \ y := 0 \quad y = 1, \ y := 0 \]

  \[ \text{Add}^-(x, \{ z \}) \]
The power of costs

- Adding the value of a clock to the cost:

  \[
  \dot{p} = 0 \\
  \ell_0 \\
  \dot{p} = 1 \\
  \ell_1 \\
  \]

  \[
  y=1, \ y:=0 \\
  x=1 \\
  z=1 \\
  \]

  \[
  x=x_0 \in [0,1] \\
  y=y_0 \in (0,1) \\
  z=0 \\
  p=p_0 \\
  \]

  \[
  \text{Add}^+(x, \{z\}) \\
  \]

- Substracting the value of a clock to the cost:

  \[
  \dot{p} = 1 \\
  \ell_0 \\
  \dot{p} = 0 \\
  \ell_1 \\
  \]

  \[
  y=1, \ y:=0 \\
  x=1 \\
  z=1 \\
  \]

  \[
  x=x_0 \\
  y=y_0 \\
  z=0 \\
  p=p_0 + 1 - x_0 \\
  \]

  \[
  \text{Add}^-(x, \{z\}) \\
  \]
The power of costs

- Testing if \( y = 2x \):

\[
\begin{align*}
\dot{p} &= 0 \\
\dot{z} &= 0 \\
z &= 0 \\
x &= 0 \\
\text{Add}^+(x, \{ z \}) \\
\text{Add}^+(x, \{ z \}) \\
\text{Add}^-(y, \{ z \}) \\
\dot{p} &= 0
\end{align*}
\]

Test(\( y=2x, \{ z \} \))
The power of costs

- Testing if $y = 2x$:

$$\dot{p}=0 \quad z=0 \quad \text{Add}^+(x, \{z\}) \quad \text{Add}^+(x, \{z\}) \quad \text{Add}^-(y, \{z\}) \quad \dot{p}=0$$

$z:=0 \quad t_0 \quad \text{Test}(y=2x, \{z\})$

$$t_0 \models EF_{p \leq 1} \quad \text{and} \quad EF_{p \geq 1} \quad t_1 \quad \Leftrightarrow \quad y_0 = 2x_0.$$
Testing that $x$ is a (negative) power of 2:

$\text{Power}_2(x, \{y,z\})$

- $z := 0$
- $y := 0$
- $z = 0, x := y$
- $z = 1 \land x \leq 1$
- $x = 1 \land z = 0$
- $\text{Test}(y = 2x, \{z\})$
- $x = 1, x := 0$
- $z = 0$

$p_0$

$p_1$

$p_2$

$p_3$
The power of costs

Testing that $x$ is a (negative) power of 2:

$$\text{Power}_2(x,\{y,z\})$$

$$p_0 \models E(p_2 \rightarrow (E p_2 \cup \varphi_{\text{test}})) \cup p_3 \iff x_0 = 2^{-d} \text{ for some } d \geq 0$$
Undecidability of WCTL model-checking

**Theorem (BBR05,BBM06)**

*Model-checking WCTL on PTAs is undecidable.*
Undecidability of WCTL model-checking

Theorem (BBR05, BBM06)

Model-checking WCTL on PTAs is undecidable.

Proof. Encoding of a two-counter machine:

- two counters $c_1$ and $c_2$ encoded by two clocks:

\[
\begin{align*}
x_1 &= \frac{1}{2c_1+1} \\
x_2 &= \frac{1}{2c_2+1}
\end{align*}
\]
Undecidability of WCTL model-checking

Theorem (BBR05,BBM06)

Model-checking WCTL on PTAs is undecidable.

Proof. Encoding of a two-counter machine:
- two counters $c_1$ and $c_2$ encoded by two clocks.
- Encoding instruction “$p_j : c_1 := c_1 + 1; \text{goto } p_k$”:

$\dot{x} = 0$ \hspace{1cm} $\dot{y} = 0$ \hspace{1cm} $\dot{z} = 0$

$E(a_3 \Rightarrow \Phi) \ U_{p \leq 0} \text{Halt.}$
Undecidability of WCTL model-checking

**Theorem (BBR05,BBM06)**

Model-checking WCTL on PTAs is undecidable.

**Proof.** Encoding of a two-counter machine:
- two counters $c_1$ and $c_2$ encoded by two clocks.
- Encoding instruction “$p_j : c_1 := c_1 + 1; \text{goto } p_k$”:

\[
\begin{aligned}
    \dot{p} &= 0 \\
    x &= 1 \\
    x &= 0 \\
    a_0 &
\end{aligned} \quad \begin{aligned}
    \dot{p} &= 0 \\
    z &= 0 \\
    a_1 &
\end{aligned} \quad \begin{aligned}
    \dot{p} &= 0 \\
    a_2 &
\end{aligned} \quad \begin{aligned}
    \dot{p} &= 1 \\
    a_3 &
\end{aligned}
\]

\[
\text{Aut}^j(x, y, z) \quad \text{Power}_3(y, \{x, z\}) \quad \text{Power}_2(x, \{y, z\}) \quad \text{Test}(x=2z, \{y\})
\]

\[
E(a_3 \Rightarrow \Phi) \cup_{p \leq 0} \text{Halt}.
\]
Outline of the talk

1. Introduction

2. Priced Timed Game Automata
   - Definitions and examples
   - Existence of optimal strategies in PTGAs is undecidable
   - Existence of optimal strategies in 1PTGAs is decidable
   - (Pseudo-)algorithm for computing the optimal cost

3. Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
   - Model-checking Priced CTL on PTAs is undecidable
   - Model-checking Priced CTL on 1PTAs is decidable

4. Conclusion
Region equivalence and TCTL

Theorem ([ACD90])

TCTL model-checking on timed automata is PSPACE-complete.
Region equivalence and TCTL

Theorem ([ACD90])

*TCTL* model-checking on timed automata is PSPACE-complete.

The algorithm relies on the following equivalence on clock valuations:

**Definition**

Two clock valuations *v* and *v'*** are equivalent (w.r.t. *M*) iff

- for all *c* ∈ *C*, *v*(c) > *M* iff *v'*(c) > *M;
- for all *c* ∈ *C* s.t. *v*(c) ≤ *M*, ⌈*v*(c)⌉ = ⌈*v'*(c)⌉;
- for all *c* ∈ *C* s.t. *v*(c) ≤ *M*, {*v*(c)} = 0 iff {*v'*(c)} = 0;
- for all *c, c'* ∈ *C* s.t. *v*(c) ≤ *M* and *v*(c') ≤ *M*, {*v*(c)} ≤ {*v*(c')} iff {*v'*(c)} ≤ {*v'*(c')}.
Region equivalence and TCTL

**Theorem ([ACD90])**

*TCTL* model-checking on timed automata is PSPACE-complete.

The algorithm relies on the following equivalence on clock valuations:
Region equivalence and TCTL

Theorem ([ACD90])

*TCTL* model-checking on timed automata is PSPACE-complete.

The algorithm relies on the following equivalence on clock valuations:
Region equivalence and TCTL

**Theorem ([ACD90])**

\textit{TCTL model-checking on timed automata is PSPACE-complete.}

The algorithm relies on the following equivalence on clock valuations:
Theorem ([ACD90])

**TCTL model-checking on timed automata is PSPACE-complete.**

The algorithm relies on the following equivalence on clock valuations:

Lemma

- *Region equivalence is a time-abstract bisimulation.*
- *Two equivalent valuations satisfy the same TCTL formulas.*
Fact

*Regions are not a correct abstraction for WCTL (even with one single clock).*
Fact

Regions are not a correct abstraction for WCTL (even with one single clock).

Example

\[ \dot{p} = 2 \quad x = 1 \quad q_0 \rightarrow q_1 \]

\[ E F_{\leq 1} q_1 \]
Region equivalence and WCTL

Fact

*Regions are not a correct abstraction for WCTL (even with one single clock).*

Example

\[
\begin{aligned}
\dot{p} &= 2 \\
\dot{q} &= 1 \\
\dot{x} &= 1 \\
q_0 \rightarrow q_1 \\
q_0 \rightarrow q_3 \\
q_3 \rightarrow q_4 \\
q_1 \rightarrow q_4 \\
E(\neg EF_{\leq 1} q_1) \cup_{\geq 1} q_4
\end{aligned}
\]
Refining regions

**Theorem**

Model-checking WCTL on one-clock PTAs is decidable.
Refining regions

Theorem

Model-checking WCTL on one-clock PTAs is decidable.

Proof. We refine regions as suggested by the previous example:

Main lemma

Let $\varphi \in \text{WCTL}$, and $A$ be a PTA. Let $C$ be the l.c.m. of the positive costs of $A$, and $M$ the maximal constant of its guards and invariants. There exists finitely many constants

$$0 = a_0 < a_1 < \cdots < a_n < a_{n+1} = +\infty$$

s.t.

- the truth value of $\varphi$ is uniform on each region $(q, (a_1, a_{i+1}))$;
- each $a_i$ is in $\mathbb{Z}^+ / C \hat{h}(\varphi)$, and $a_n = M$.

where $\hat{h}(\varphi)$ is the maximal number of constrained modalities in $\varphi$. 
Refining regions

Main lemma

Let $\varphi \in \text{WCTL}$, and $\mathcal{A}$ be a PTA. Let $C$ be the l.c.m. of the positive costs of $\mathcal{A}$, and $M$ the maximal constant of its guards and invariants. There exists finitely many constants

$$0 = a_0 < a_1 < \cdots < a_n < a_{n+1} = +\infty$$

s.t.

- the truth value of $\varphi$ is uniform on each region $(q, (a_1, a_{i+1}))$;
- each $a_i$ is in $\mathbb{Z}^+ / C^{\mathcal{h}(\varphi)}$, and $a_n = M$.

where $\mathcal{h}(\varphi)$ is the maximal number of constrained modalities in $\varphi$. 
Refining regions

Main lemma

Let $\varphi \in \text{WCTL}$, and $\mathcal{A}$ be a PTA. Let $C$ be the l.c.m. of the positive costs of $\mathcal{A}$, and $M$ the maximal constant of its guards and invariants. There exists finitely many constants

$$0 = a_0 < a_1 < \cdots < a_n < a_{n+1} = +\infty$$

s.t.

- the truth value of $\varphi$ is uniform on each region $(q, (a_1, a_{i+1}))$;
- each $a_i$ is in $\mathbb{Z}^+ / C^{h(\varphi)}$, and $a_n = M$.

where $h(\varphi)$ is the maximal number of constrained modalities in $\varphi$.

*Proof.* Inductive proof:

- for atomic propositions, the $a_i$’s are the constants that appear in the constraints of the automaton.
Refining regions

Main lemma

Let $\varphi \in \text{WCTL}$, and $\mathcal{A}$ be a PTA. Let $C$ be the l.c.m. of the positive costs of $\mathcal{A}$, and $M$ the maximal constant of its guards and invariants. There exists finitely many constants

$$0 = a_0 < a_1 < \cdots < a_n < a_{n+1} = +\infty$$

s.t.

- the truth value of $\varphi$ is uniform on each region $(q, (a_1, a_{i+1}))$;
- each $a_i$ is in $\mathbb{Z}^+/C\bar{h}(\varphi)$, and $a_n = M$.

where $\bar{h}(\varphi)$ is the maximal number of constrained modalities in $\varphi$.

Proof. Inductive proof:

- If $\varphi = E(\varphi_1 \bigcup_{c} \varphi_2)$, assume we have computed the $a_i$’s for $\varphi_1$ and $\varphi_2$. 


Refining regions

Proof. Inductive proof:

- If \( \varphi = E(\varphi_1 \cup_c \varphi_2) \), assume we have computed the \( a_i \)'s for \( \varphi_1 \) and \( \varphi_2 \).
Refining regions

*Proof.* Inductive proof:

- If $\varphi = E(\varphi_1 \cup_c \varphi_2)$, assume we have computed the $a_i$’s for $\varphi_1$ and $\varphi_2$.

**Lemma**

*The set of costs of non-resetting trajectories between $(q, a_i)$ and $(q, a_{i+1})$ is an interval.*
Refining regions

Proof. Inductive proof:

- If $\varphi = \mathbf{E}(\varphi_1 \mathbf{U}_{c} \varphi_2)$, assume we have computed the $a_i$'s for $\varphi_1$ and $\varphi_2$.

Lemma

The set of costs of non-resetting trajectories between $(q, a_i)$ and $(q, a_{i+1})$ is an interval.

Proof.
Refining regions

Proof. Inductive proof:

- If $\varphi = E(\varphi_1 U_{\sim c} \varphi_2)$, assume we have computed the $a_i$’s for $\varphi_1$ and $\varphi_2$.

Lemma

The set of costs of non-resetting trajectories between $(q, a_i)$ and $(q, a_{i+1})$ is an interval.

Proof.

$$\text{costs} \subseteq [c_{\text{min}}, c_{\text{max}}] \cdot (a_{i+1} - a_i)$$
Refining regions

Proof. Inductive proof:

- If $\varphi = E(\varphi_1 \cup_c \varphi_2)$, assume we have computed the $a_i$'s for $\varphi_1$ and $\varphi_2$.

Lemma

The set of costs of non-resetting trajectories between $(q, a_i)$ and $(q, a_{i+1})$ is an interval.

Proof.

Costs $\subseteq [c_{\text{min}}, c_{\text{max}}] \cdot (a_{i+1} - a_i)$
Refining regions

Proof. Inductive proof:

- If $\varphi = E(\varphi_1 U_c \varphi_2)$, assume we have computed the $a_i$’s for $\varphi_1$ and $\varphi_2$.

Lemma

The set of costs of non-resetting trajectories between $(q, a_i)$ and $(q, a_{i+1})$ is an interval.

Proof.

$\left( c_{\min}, \frac{c + c'}{2} \right) \cdot (a_{i+1} - a_i) \subseteq \text{costs} \subseteq \left[ \frac{c + c'}{2}, c_{\max} \right) \cdot (a_{i+1} - a_i)$
Refining regions

**Proof.** Inductive proof:

- If \( \varphi = E(\varphi_1 U_{\sim c} \varphi_2) \), assume we have computed the \( a_i \)'s for \( \varphi_1 \) and \( \varphi_2 \).

\[
\begin{align*}
(q, \{a_i\}) & \xrightarrow{\langle c_{\text{min}}, c_{\text{max}} \rangle \cdot (a_{i+1} - a_i)} (q, \{a_{i+1}\}) \\
(q, \{a_i\}) & \xrightarrow{\langle 0, c_{\text{max}} \rangle \cdot (a_{i+1} - a_i)} (q, (a_i, a_{i+1})) \\
(q, \{a_n\}) & \xrightarrow{[0,0] \text{ or } \langle 0, +\infty \rangle} (q, (a_n, +\infty)) \\
(q, \text{any}) & \xrightarrow{(x := 0)} [0,0] \xrightarrow{(x := 0)} (q, \{0\}) \\
(q, x, (a_i, a_{i+1})) & \xrightarrow{\langle c_{\text{min}}, c_{\text{max}} \rangle \cdot (a_{i+1} - x)} (q, \{a_{i+1}\}) \\
\vdots
\end{align*}
\]
Refining regions

Proof. Inductive proof:

- If $\varphi = E(\varphi_1 U_{\sim c} \varphi_2)$, assume we have computed the $a_i$'s for $\varphi_1$ and $\varphi_2$. 
Refining regions

Proof. Inductive proof:

- If $\varphi = E(\varphi_1 \U_c \varphi_2)$, assume we have computed the $a_i$’s for $\varphi_1$ and $\varphi_2$.

From this graph, we get that:

- the set of costs between any two regions is a union of intervals of the form $\langle \alpha - \beta x, \alpha' - \beta' x \rangle$ where
  - $\alpha$ and $\alpha'$ are integral multiples of $1/C^{\max\{h(\varphi_1), h(\varphi_2)\}}$,
  - $\beta$ and $\beta'$ are costs of the automaton.

- the set of values for $x$ s.t. $(q, x) \models E\varphi_1 \U_c \varphi_2$ is a finite union of intervals whose bounds are multiples of $1/C^{h(\varphi)}$ and bounded by $M$.

- if the formula holds, it has an exponential-sized witness.
The exponential blowup cannot be avoided

\[ \dot{p} = 1 \quad \dot{p} = 4 \quad \dot{p} = 2 \quad \dot{p} = 2 \quad \dot{p} = 1 \]

\[ x < 1 \quad x < 1 \quad x < 1 \quad x < 2 \quad x < 2 \]

\[ x = 2 \quad x = 2 \quad x = 2 \quad x = 0 \quad x = 0 \]

\[ x = 0 \quad x = 0 \quad x = 0 \quad x = 0 \quad x = 0 \]

\[ a \quad b \quad c \]

\[ \phi(X) = E(F = 0(c)) \iff x \in \{0, 1\} \]

\[ \phi(X) = E(F = 0(c)) \iff x \in \{0, 1/2, 1, 3/2\} \]

\[ \phi(X) = E(F = 0(c)) \iff x \in \{p/2^n - 1 \mid 0 \leq p < 2^n\} \]
The exponential blowup cannot be avoided

- value of $x$ when leaving state $a$: $x_0, x_1 x_2 x_3 \ldots x_n$
The exponential blowup cannot be avoided

- value of $x$ when leaving state $a$: $x_0, x_1 x_2 x_3 \ldots x_n$
- + cost 4 between $a$ and $b$
The exponential blowup cannot be avoided

\[ \dot{p} = 1 \quad \dot{p} = 2 \quad \dot{p} = 2 \quad \dot{p} = 1 \quad \dot{p} = 1 \]

- Value of \( x \) when leaving state \( a \): \( x_0, x_1 x_2 x_3 \ldots x_n \)
- \( + \) cost 4 between \( a \) and \( b \)
- \( \rightarrow \) Value of \( x \) when entering state \( b \): \( x_1, x_2 x_3 \ldots x_n \)
The exponential blowup cannot be avoided

\[ \dot{p} = \begin{cases} 1, & x < 1 \\ 2, & x = 2 \\ 2, & x < 2 \\ 1, & x = 0 \end{cases} \]

\[ \begin{align*}
\text{value of } x \text{ when leaving state } a: & \quad x_0, x_1 x_2 x_3 \ldots x_n \\
\text{+ cost 4 between } a \text{ and } b \\
\rightarrow \text{ value of } x \text{ when entering state } b: & \quad x_1, x_2 x_3 \ldots x_n \\
\end{align*} \]

\[ \varphi(X) = E((a \lor b) U_{=0} (\neg a \land E(\neg b U_{=4} (b \land X)))) \]
The exponential blowup cannot be avoided

\[ \dot{p} = 1 \quad \dot{p} = 2 \quad \dot{p} = 4 \quad \dot{p} = 2 \]

- Value of \( x \) when leaving state \( a \): \( x_0, x_1 x_2 x_3 \ldots x_n \)
- Cost 4 between \( a \) and \( b \)
- Value of \( x \) when entering state \( b \): \( x_1, x_2 x_3 \ldots x_n \)

\[ \varphi(X) = E((a \lor b) U_{=0} (\neg a \land E(\neg b U_{=4} (b \land X)))) \]
- \( a, x \models \varphi(E F_{=0} c) \) iff \( x \in \{0, 1\} \)
The exponential blowup cannot be avoided

value of $x$ when leaving state $a$: $x_0, x_1 x_2 x_3 \ldots x_n$
+ cost 4 between $a$ and $b$
→ value of $x$ when entering state $b$: $x_1, x_2 x_3 \ldots x_n$

$\varphi(X) = E\left( (a \lor b) \ U_{=0} (\neg a \land E(\neg b \ U_{=4} (b \land X))) \right)$

- $a, x \models \varphi(E F_{=0} c)$ iff $x \in \{0, 1\}$
- $a, x \models \varphi\left(\varphi(E F_{=0} c)\right)$ iff $x \in \{0, 1/2, 1, 3/2\}$
The exponential blowup cannot be avoided

\[
\dot{p} = \begin{cases} 1 & \text{if } x > 1 \\ 4 & \text{if } x = 2 \\ 2 & \text{if } x < 2 \\ 0 & \text{otherwise} \end{cases}
\]

- value of \( x \) when leaving state \( a \): \( x_0, x_1 x_2 x_3 \ldots x_n \)
- cost 4 between \( a \) and \( b \)
- value of \( x \) when entering state \( b \): \( x_1, x_2 x_3 \ldots x_n \)

\[\varphi(X) = E((a \lor b) \ U_{=0} (\neg a \land E(\neg b \ U_{=4} (b \land X))))\]

- \( a, x \models \varphi(E F_{=0} c) \) iff \( x \in \{0, 1\} \)
- \( a, x \models \varphi(\varphi(E F_{=0} c)) \) iff \( x \in \{0, 1/2, 1, 3/2\} \)
- \( a, x \models \varphi^n(E F_{=0} c) \) iff \( x \in \{p/2^{n-1} \mid 0 \leq p < 2^n\} \)
Assuming we have computed the $a_i$’s for $\varphi_1$ and $\varphi_2$:

**EXPTIME Algorithm**

- for each $x = k/2C^\mathcal{h}(\varphi)$:
  - non-deterministically guess a witnessing trajectory in the graph we just built,
  - check that this graph satisfies $\varphi_1 \mathcal{U} \varphi_2$,
  - check that the total cost satisfies condition $\sim c$. 
Assuming we have computed the $a_i$’s for $\varphi_1$ and $\varphi_2$:

**EXPTIME Algorithm**

- for each $x = k/2C^{\overline{h}(\varphi)}$:
  - non-deterministically guess a witnessing trajectory in the graph we just built,
  - check that this graph satisfies $\varphi_1 U \varphi_2$,
  - check that the total cost satisfies condition $\sim c$.

$\leadsto$ each step is in (N)PSPACE;

$\leadsto$ the whole algorithm is in EXPTIME (there may be an exponential number of constants to store).
Algorithms

Assuming we have computed the $a_i$’s for $\varphi_1$ and $\varphi_2$:

**EXPTIME Algorithm**

- for each $x = k/2C^h(\varphi)$:
  - non-deterministically guess a witnessing trajectory in the graph we just built,
  - check that this graph satisfies $\varphi_1 \cup \varphi_2$,
  - check that the total cost satisfies condition $\sim c$.

$\leadsto$ each step is in (N)PSPACE;

$\leadsto$ the whole algorithm is in EXPTIME (there may be an exponential number of constants to store).

$\leadsto$ can be achieved in PSPACE if we don’t store all the constants, but re-compute them when we need them.
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \quad q_0 \quad \dot{p} = 1 \quad q_2 \]

\[ x = 1 \quad q_1 \quad x = 1 \quad q_3 \]

\[ E \cup_{\geq 1} \]

\[ E F_{\leq 1} \]

\[ q_1 \]

\[ q_3 \]
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \]
\[ \dot{p} = 1 \]
\[ x = 1 \]

\[ q_0, x, \{0\} \]
\[ \text{step} = 0 \]
\[ \text{cost} = [0,0] \]
Execution of the PSPACE algorithm
Execution of the PSPACE algorithm
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \quad p_0 \quad q_1 \]

\[ \dot{p} = 1 \quad x = 1 \quad q_2 \quad q_3 \]

\[ q_0, x, \{0\} \]
step = 0
\[ \text{cost} = [0,0] \]

\[ \text{E U}_{\geq 1} \]

\[ \text{q}_3 \]

\[ q_0, x, \{0\} \]
step = 0
\[ \text{cost} = [0,0] \]

\[ \text{E F}_{\leq 1} \]

\[ q_1 \]

\[ q_0, (0, 1/4) \]
step = 1
\[ \text{cost} = (0, 1/2) \]
Execution of the PSPACE algorithm

$q_0, x, \{0\}$
step = 0
$\text{cost} = [0,0]$

$E \cup \geq 1$

$q_0, x, \{0\}$
step = 0
$\text{cost} = [0,0]$

$F \leq 1$

$q_0, \{1/4\}$
step = 2
$\text{cost} = [1/2,1/2]$
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \]
\[ \dot{p} = 1 \]
\[ x = 1 \]
\[ q_0 \]
\[ \dot{q}_0 \]
\[ q_1 \]
\[ q_2 \]
\[ x = 1 \]
\[ q_3 \]

\[ q_0, x, \{0\} \]
\[ \text{step} = 0 \]
\[ \text{cost} = [0,0] \]

\[ q_0, (1/4, 1/2) \]
\[ \text{step} = 3 \]
\[ \text{cost} = (1/2, 1) \]

\[ \mathbf{E} \mathbf{U}_{\geq 1} \]
\[ \mathbf{E} \mathbf{F}_{\leq 1} \]

\[ q_3 \]

\[ q_1 \]
Execution of the PSPACE algorithm

- $p = 2$
- $x = 1$
- $q_0 \rightarrow q_1$

- $p = 1$
- $x = 1$
- $q_2 \rightarrow q_3$

- $q_0, x, \{0\}$
- step = 0
- cost = [0,0]

- $E \cup \geq 1$

- $q_0, x, \{0\}$
- step = 0
- cost = [0,0]

- $E F \leq 1$

- $q_0, \{1/2\}$
- step = 4
- cost = [1,1]

- $q_1$

- $q_3$
Execution of the PSPACE algorithm

\[
\dot{p} = 2, q_0 \\
\dot{p} = 1, q_2
\]

\[
x = 1 \\
x = 1
\]

\[
E U_{\geq 1}
\]

\[
q_0, x, \{0\} \\
\text{step} = 0 \\
\text{cost} = [0,0]
\]

\[
q_0, x, \{0\} \\
\text{step} = 0 \\
\text{cost} = [0,0]
\]

\[
q_3
\]

\[
E F_{\leq 1}
\]

\[
q_0, (1/2, 3/4) \\
\text{step} = 5 \\
\text{cost} = (1,3/2)
\]

\[
q_1
\]
Execution of the PSPACE algorithm

\[
\begin{align*}
\dot{p} &= 2 \\
q_0, x, \{0\} &\quad \text{step} = 0 \\
&\quad \text{cost} = [0,0] \\
\dot{p} &= 1 \\
q_0, x, \{0\} &\quad \text{step} = 0 \\
&\quad \text{cost} = [0,0] \\
q_0 \rightarrow q_1 \\
\dot{q}_1 &= 1 \\
q_0, (1/2, 3/4) &\quad \text{step} = 5 \\
&\quad \text{cost} = (1,3/2) \\
q_0 \rightarrow q_3 \\
q_0 \rightarrow \neg \quad \text{E U}_{\geq 1} \\
q_0 \rightarrow q_1 \\
q_0 \rightarrow \neg \quad \text{E F}_{\leq 1} \\
q_0 \rightarrow q_1
\end{align*}
\]
Execution of the PSPACE algorithm

$q_0, x, \{0\}$
step = 0
cost = [0,0]

$q_0, x, \{0\}$
step = 0
cost = [0,0]

$q_0, x, \{0\}$
step = 0
cost = [0,0]
Execution of the PSPACE algorithm

$q_0, (0, 1/4)$
step = 1
(cost = (0, 1/2))
Execution of the PSPACE algorithm

$q_0, (0, 1/4)$

step = 1

cost = (0, 1/2)

$q_0, (0, 1/4)$

step = 0

cost = [0, 0]

$\dot{p} = 2$

$x = 1$

$q_1$

$q_0$

$\dot{p} = 1$

$x = 1$

$q_3$

$q_2$
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \]
\[ \dot{q} = 1 \]

\[ q_0, (0, 1/4) \]
\[ \text{step} = 1 \]
\[ \text{cost} = (0, 1/2) \]

\[ E U_{\geq 1} \]

\[ q_1 \]

\[ \neg \]

\[ q_3 \]

\[ q_0, (0, 1/4) \]
\[ \text{step} = 0 \]
\[ \text{cost} = [0,0] \]

\[ E F_{\leq 1} \]

\[ q_1 \]

\[ q_0, (0, 1/4) \]
\[ \text{step} = 0 \]
\[ \text{cost} = [0,0] \]

\[ x = 1 \]

\[ \text{E U}_{\geq 1} \]

\[ q_1 \]

\[ q_3 \]
Execution of the PSPACE algorithm

\[
\begin{align*}
\dot{p} &= 2 \\
\dot{p} &= 1
\end{align*}
\]

\[
\begin{align*}
q_0, (0, 1/4) \\
\text{step} &= 1 \\
\text{cost} &= (0, 1/2)
\end{align*}
\]

\[
\begin{align*}
q_0, (0, 1/2) \\
\text{step} &= 0 \\
\text{cost} &= [0, 0]
\end{align*}
\]

\[
\begin{align*}
q_0, \{1/4\} \\
\text{step} &= 1 \\
\text{cost} &= (0, 1/2)
\end{align*}
\]
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \]
\[ \dot{q} = 1 \]

\[ x = 1 \]

\[ q_0, (0, 1/4) \]
\[ \text{step} = 1 \]
\[ \text{cost} = (0, 1/2) \]

\[ q_0, (0, 1/4) \]
\[ \text{step} = 0 \]
\[ \text{cost} = [0, 0] \]

\[ q_0, (1/4, 1/2) \]
\[ \text{step} = 2 \]
\[ \text{cost} = (0, 1) \]
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \]
\[ \dot{p} = 1 \]

\[ q_0, (0, 1/4) \]
- step = 1
- cost = (0, 1/2)

\[ q_0, (0, 1/4) \]
- step = 0
- cost = [0, 0]

\[ q_0, \{1/2\} \]
- step = 3
- cost = (1/2, 1)

\[ q_0, \{0, 1/4\} \]
Execution of the PSPACE algorithm

$q_0, (0, 1/4)$
step = 1
cost = (0,1/2)

$q_0, (0, 1/4)$
step = 0
cost = [0,0]

$q_0, \{3/4\}$
step = 5
cost = (1,3/2)
Execution of the PSPACE algorithm

$\dot{p} = 2$

$q_0, (0, 1/4)$
step = 1
cost = (0, 1/2)

$q_0, (0, 1/4)$
step = 0
cost = [0, 0]

$q_0, \{3/4\}$
step = 5
cost = (1, 3/2)

$\dot{p} = 1$

$x = 1$

$q_1$

$q_3$

$E \cup_{\geq 1}$

$E F_{\leq 1}$

$q_1$
Execution of the PSPACE algorithm

$q_0, (0, 1/4)$
step = 1
cost = $[0, 1/2]$

$q_0, (0, 1/4)$
step = 0
cost = $[0, 0]$
Execution of the PSPACE algorithm

$q_2, (0, 1/4)$
step = 2
\[\text{cost} = (0, 1/2)\]
Execution of the PSPACE algorithm

\[ \dot{p} = 2, q_0 \xrightarrow{x=1} q_1 \]

\[ \dot{p} = 1, q_2 \xrightarrow{x=1} q_3 \]

\[ q_2, (0, 1/4) \]
\[ \text{step} = 2 \]
\[ \text{cost} = (0, 1/2) \]

\[ E U_{\geq 1} \]

\[ q_1 \]

\[ E F_{\leq 1} \]

\[ q_3 \]

\[ E \]
Execution of the PSPACE algorithm

\[ p = 2 \]
\[ q_0 \]
\[ \dot{p} = 1 \]
\[ q_2 \]
\[ x = 1 \]
\[ q_1 \]
\[ q_2, (0, 1/4) \]
\[ \text{step} = 0 \]
\[ \text{cost} = [0, 0] \]
\[ \dot{p} = 1 \]
\[ q_1 \]
\[ x = 1 \]
\[ q_3 \]
\[ q_2, (0, 1/4) \]
\[ \text{step} = 0 \]
\[ \text{cost} = [0, 0] \]

\[ E \cup_{\geq 1} \]
\[ q_3 \]
\[ q_2, (0, 1/4) \]
\[ \text{step} = 2 \]
\[ \text{cost} = (0, 1/2) \]

\[ E \preceq_{1} \]
\[ q_1 \]
\[ q_2, (0, 1/4) \]
\[ \text{step} = 0 \]
\[ \text{cost} = [0, 0] \]
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \]

\[ \dot{q} = 1 \]

\[ x = 1 \]

\[ q_0 \rightarrow q_1 \]

\[ q_2 \rightarrow q_3 \]

\[ q_2, (0, 1/4) \]

\[ \text{step} = 2 \]

\[ \text{cost} = (0, 1/2) \]

\[ \text{E} \cup \geq_1 \]

\[ q_2, (0, 1/4) \]

\[ \text{step} = 0 \]

\[ \text{cost} = [0, 0] \]

\[ \text{E} \preceq_1 \]

\[ q_2, (1, 5/4) \]

\[ \text{step} = 5 \]

\[ \text{cost} = (3/4, 5/4) \]

\[ \neg \]

\[ q_3 \]

\[ \text{cost} = (0, 1/2) \]
Execution of the PSPACE algorithm

\[ p = 2 \quad q_0 \quad q_1 \quad \dot{p} = 1 \quad q_2 \quad q_3 \]

- \( q_2, (0, 1/4) \)
  - \( \text{step} = 2 \)
  - \( \text{cost} = (0, 1/2) \)

- \( \text{step} = 0 \)
  - \( \text{cost} = [0,0] \)

- \( E U_{\geq 1} \)
  - \( q_3 \)

- \( E F_{\leq 1} \)
  - \( q_1 \)
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \]
\[ \dot{q} = 1 \]

\[ x = 1 \]
\[ \dot{p} = 1 \]
\[ \dot{q} = 2 \]

\[ q_0 \]
\[ q_1 \]
\[ q_2 \]
\[ q_3 \]

\[ E \ U_{\geq 1} \]
\[ E F_{\leq 1} \]

\[ q_2, \{1/4\} \]
\[ \text{step} = 3 \]
\[ \text{cost} = (1/4, 1/2) \]
Execution of the PSPACE algorithm

\[ q_2, \{1\} \]
\[ \text{step} = 9 \]
\[ \text{cost} = [1, 5/4) \]

\[ E \cup_{\geq 1} \]

\[ E \mathrel{\text{F}}_{\leq 1} \]

\[ q_1 \]

\[ q_3 \]
Execution of the PSPACE algorithm

\[ \dot{p} = 2 \quad q_0 \rightarrow q_1 \]
\[ \dot{p} = 1 \quad q_2 \rightarrow q_3 \]

\[ x = 1 \]

\[ q_3, \{1\} \]
\[ \text{step} = 10 \]
\[ \text{cost} = [1, 5/4) \]

\[ E U_{\geq 1} \]

\[ E F_{\leq 1} \]

\[ q_1 \]

\[ q_3 \]
Execution of the PSPACE algorithm

\[
\begin{align*}
\dot{p} &= 2 \\
\dot{q} &= 1 \\
\dot{x} &= 1
\end{align*}
\]

\[
\begin{align*}
q_0 &\rightarrow q_1 \\
q_2 &\rightarrow q_3
\end{align*}
\]

\[
\begin{align*}
E U_{\geq 1} &\rightarrow q_3, \{1\} \\
&\text{step} = 10 \\
&\text{cost} = [1, 5/4)
\end{align*}
\]

\[
\begin{align*}
E F_{\leq 1} &\rightarrow q_1 \\
&\text{step} = 0 \\
&\text{cost} = [0, 0]
\end{align*}
\]
Execution of the PSPACE algorithm

\[ \dot{p} = 2, \quad q_0 \xrightarrow{x=1} q_1 \]
\[ \dot{p} = 1, \quad q_2 \xrightarrow{x=1} q_3 \]

\[ \text{E U}_{\geq 1} \]
\[ q_3, \{1\} \]
\[ \text{step} = 10 \]
\[ \text{cost} = [1, 5/4) \]

\[ \text{E F}_{\leq 1} \]
\[ q_3, \{1\} \]
\[ \text{step} = 0 \]
\[ \text{cost} = [0, 0] \]
Execution of the PSPACE algorithm

\[
\begin{align*}
\dot{p} &= 2 \\
\dot{q} &= 1 \\
\end{align*}
\]

\[
\begin{align*}
x &= 1 \\
\end{align*}
\]

\[
\begin{align*}
q &= 3, \{1\} \\
\text{step} &= 10 \\
\text{cost} &= [1, 5/4) \\
\end{align*}
\]

\[
\begin{align*}
E \cup_{\geq 1} & \\
\end{align*}
\]

\[
\begin{align*}
E \preceq_{\leq 1} & \\
\end{align*}
\]

\[
\begin{align*}
q &= 3 \\
\end{align*}
\]

\[
\begin{align*}
q &= 1 \\
\end{align*}
\]
Outline of the talk

1. Introduction

2. Priced Timed Game Automata
   - Definitions and examples
   - Existence of optimal strategies in PTGAs is undecidable
   - Existence of optimal strategies in 1PTGAs is decidable
   - (Pseudo-)algorithm for computing the optimal cost

3. Priced Timed Automata and WCTL
   - Priced timed automata
   - Priced CTL
   - Model-checking Priced CTL on PTAs is undecidable
   - Model-checking Priced CTL on 1PTAs is decidable

4. Conclusion
Conclusion and Perspectives

Summary of our works:

- Adding costs to timed automata provides a natural way for modeling resource consumption.
- Unfortunately, costs are expensive!
  - Undecidable for three-clock automata;
  - Complex algorithms for one-clock automata;
  - Convergence of the pseudo-algorithm of [BCFL04].

Perspectives:

- Complexity gap for Priced Timed Games;
- Two-clock Priced Timed Automata;
- Priced ATL model-checking: mixing games and WCTL;
- Multi-constrained objectives/modalities.