

# Costs are Expensive

Patricia Bouyer<sup>1,2</sup> and Nicolas Markey<sup>1</sup>

<sup>1</sup> Lab. Spécification et Vérification – ENS Cachan & CNRS

<sup>2</sup> Computing Laboratory – Oxford University

FORMATS'07 – October 4, 2007

# Verification & Model-Checking

system:

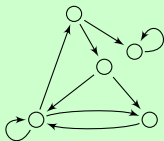


property:



# Verification & Model-Checking

system:



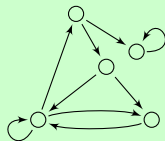
property:



$G(\text{request} \Rightarrow F \text{ grant})$

# Verification & Model-Checking

system:



model-checking  
algorithm



$G(\text{request} \Rightarrow F \text{ grant})$

property:

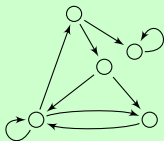


# Verification & Model-Checking

system:



property:



model-checking  
algorithm



$G(\text{request} \Rightarrow F \text{ grant})$



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;

↪ **timed 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;

↪ 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);

↪ TCTL, MTL, TPTL, timed  $\mu$ -calculus, ...

## Time is not always sufficient

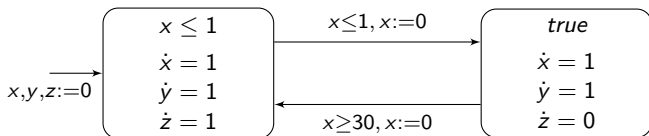
In some cases, we don't want to measure time, but rather **energy consumption**, **price to pay** for reaching some goal, ...

# 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**, ...



## Theorem (HKPV97)

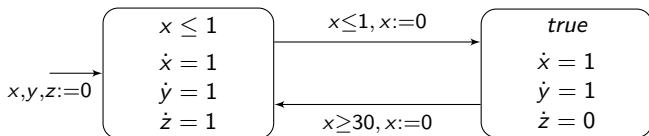
*Reachability is undecidable (even for timed automata where one single "clock" has two derivatives).*

## 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**, ...



- **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 [ATP01,BFH<sup>+</sup>01,LBB<sup>+</sup>01,BBBR06]
- Mean-cost optimality [BBL04]

- **Control games**

- Properties, and restricted decidability results [ABM04,BCFL04,BCFL05]
- Undecidability for timed game automata with more than three clocks [BBR05,BBM06]
- Decidability for timed automata with one clock [BLMR06]

- **Model-checking of WCTL**

- Undecidability for timed automata with more than three clocks [BBR04,BBM06]
- Decidability for timed automata with one clock [BLM06]

# Related work on priced timed automata

- **Basic properties**

- Optimal reachability [ATP01,BFH<sup>+</sup>01,LBB<sup>+</sup>01,BBBR06]
- Mean-cost optimality [BBL04]

- **Control games**

- Properties, and restricted decidability results [ABM04,BCFL04,BCFL05]
- Undecidability for timed game automata with more than three clocks [BBR05,BBM06]
- Decidability for timed automata with one clock [BLMR06]

- **Model-checking of WCTL**

- Undecidability for timed automata with more than three clocks [BBR04,BBM06]
- Decidability for timed automata with one clock [BLM06]

What about linear-time properties?

# Outline of the talk

- 1 Introduction
- 2 Priced timed automata and weighted temporal logics
- 3 Decidability results: 1 clock and 1 stopwatch cost
- 4 Undecidability results: all other cases...
  - 1PTAs with one (3-sloped) cost
  - 2PTAs with one stopwatch cost
  - 1PTAs with two stopwatch costs
- 5 Conclusion

# Outline of the talk

- 1 Introduction
- 2 Priced timed automata and weighted temporal logics
- 3 Decidability results: 1 clock and 1 stopwatch cost
- 4 Undecidability results: all other cases...
  - 1PTAs with one (3-sloped) cost
  - 2PTAs with one stopwatch cost
  - 1PTAs with two stopwatch costs
- 5 Conclusion

## Priced Timed automata

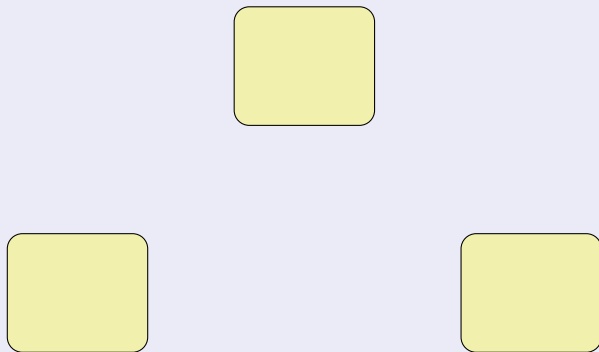
Definition (ALP01,BFH<sup>+</sup>01)

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

# Priced Timed automata

Definition (ALP01,BFH<sup>+</sup>01)

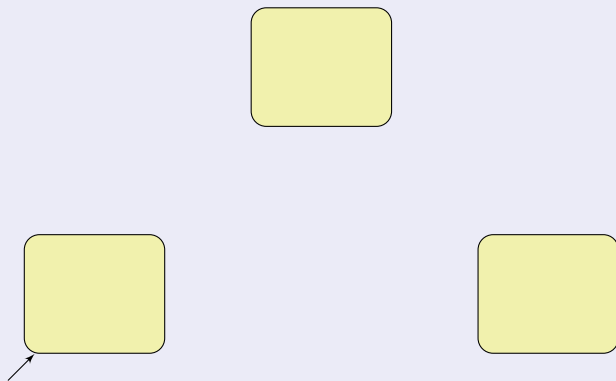
A *priced timed automaton* is a timed automaton extended with a *cost function*:  $\mathcal{P} = \langle Q, Q_0, AP, \ell, \delta, \mathcal{C}, G, R, I, P \rangle$ .



# Priced Timed automata

Definition (ALP01,BFH<sup>+</sup>01)

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



# Priced Timed automata

Definition (ALP01,BFH<sup>+</sup>01)

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

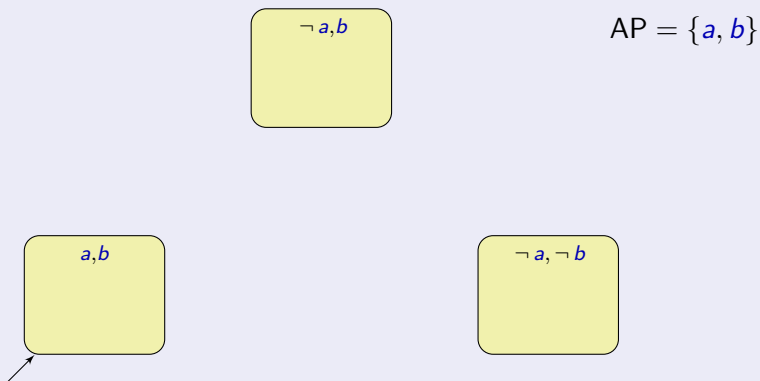
$$AP = \{a, b\}$$



# Priced Timed automata

Definition (ALP01,BFH<sup>+</sup>01)

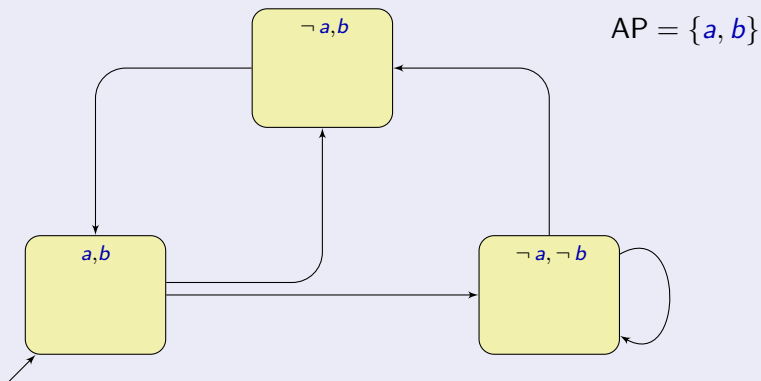
A *priced timed automaton* is a timed automaton extended with a *cost function*:  $\mathcal{P} = \langle Q, Q_0, AP, \ell, \delta, \mathcal{C}, G, R, I, P \rangle$ .



# Priced Timed automata

Definition (ALP01,BFH<sup>+</sup>01)

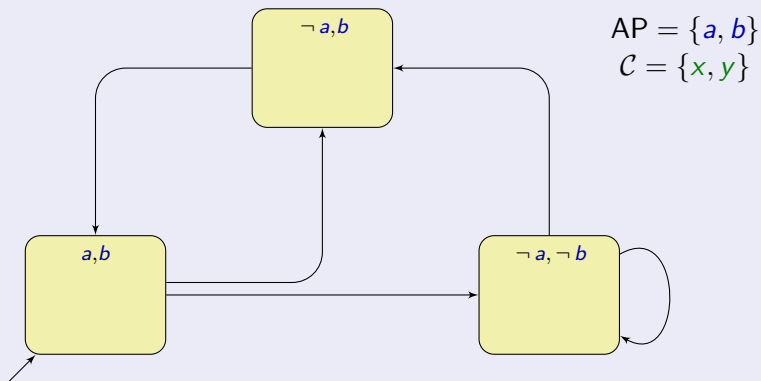
A *priced timed automaton* is a timed automaton extended with a *cost function*:  $\mathcal{P} = \langle Q, Q_0, AP, \ell, \delta, \mathcal{C}, G, R, I, P \rangle$ .



# Priced Timed automata

## Definition (ALP01,BFH<sup>+</sup>01)

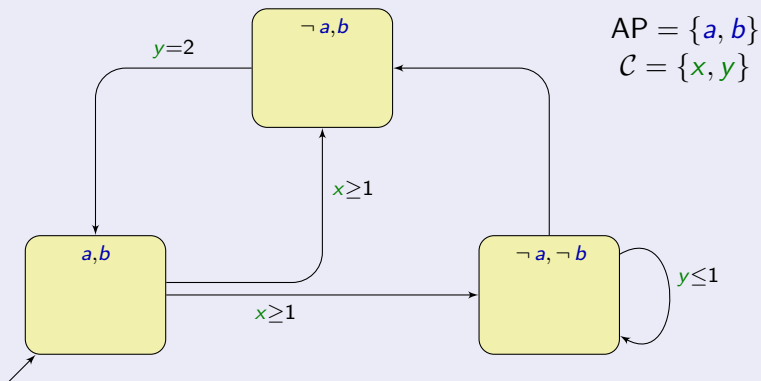
A *priced timed automaton* is a timed automaton extended with a *cost function*:  $\mathcal{P} = \langle Q, Q_0, AP, \ell, \delta, \mathcal{C}, G, R, I, P \rangle$ .



# Priced Timed automata

Definition (ALP01,BFH<sup>+</sup>01)

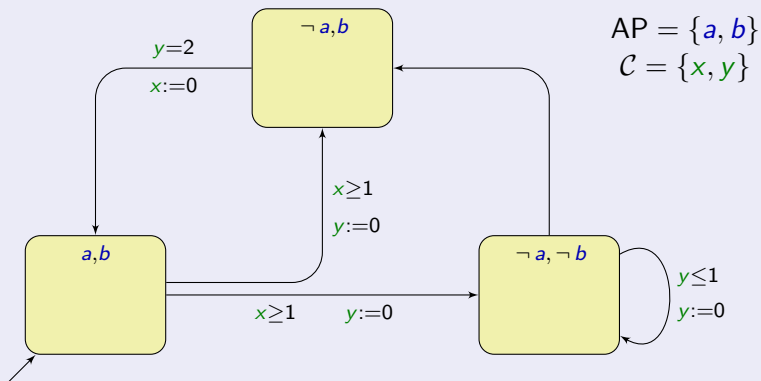
A *priced timed automaton* is a timed automaton extended with a *cost function*:  $\mathcal{P} = \langle Q, Q_0, AP, \ell, \delta, \mathcal{C}, \mathbf{G}, R, I, P \rangle$ .



# Priced Timed automata

Definition (ALP01,BFH<sup>+</sup>01)

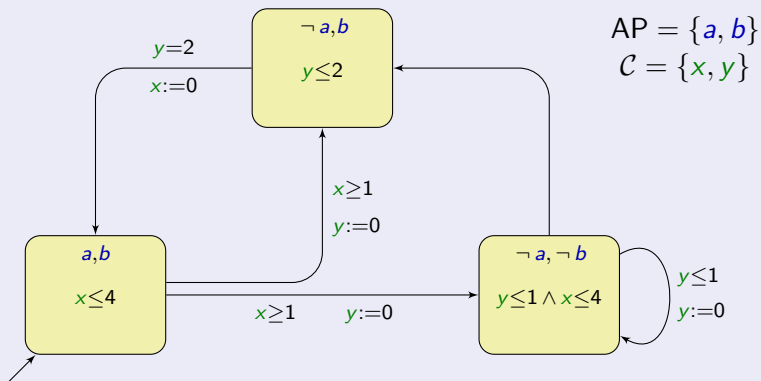
A *priced timed automaton* is a timed automaton extended with a *cost function*:  $\mathcal{P} = \langle Q, Q_0, AP, \ell, \delta, \mathcal{C}, G, R, I, P \rangle$ .



# Priced Timed automata

## Definition (ALP01,BFH<sup>+</sup>01)

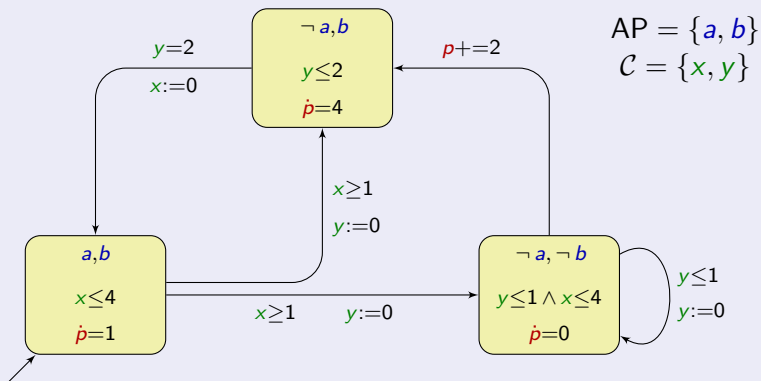
A *priced timed automaton* is a timed automaton extended with a *cost function*:  $\mathcal{P} = \langle Q, Q_0, AP, \ell, \delta, \mathcal{C}, G, R, I, P \rangle$ .



# Priced Timed automata

Definition (ALP01,BFH<sup>+</sup>01)

A *priced timed automaton* is a timed automaton extended with a *cost function*:  $\mathcal{P} = \langle Q, Q_0, AP, \ell, \delta, \mathcal{C}, G, R, I, \mathbf{P} \rangle$ .



# Priced Timed automata

## Definition (ALP01,BFH<sup>+</sup>01)

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

- the *price function* associates a **nonnegative** price with each location and each transition:
  - costs on transitions are called *discrete costs*, to be paid when firing the transition;
  - costs on locations are *continuous costs*, and they represent the price to pay per time unit elapsed in that location.

# Priced Timed automata

## Definition (ALP01,BFH<sup>+</sup>01)

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

- the *price function* associates a **nonnegative** price with each location and each transition:
  - costs on transitions are called *discrete costs*, to be paid when firing the transition;
  - costs on locations are *continuous costs*, and they represent the price to pay per time unit elapsed in that location.
- a price function is *stopwatch* if all discrete costs are zero, and all continuous costs are either 0 or 1.

# Priced Timed automata

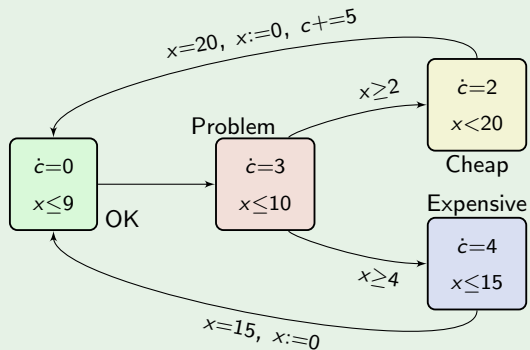
## Definition (ALP01,BFH<sup>+</sup>01)

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

- the *price function* associates a **nonnegative** price with each location and each transition:
  - costs on transitions are called *discrete costs*, to be paid when firing the transition;
  - costs on locations are *continuous costs*, and they represent the price to pay per time unit elapsed in that location.
- a price function is *stopwatch* if all discrete costs are zero, and all continuous costs are either 0 or 1.
- a *run* of a priced timed automaton is a run of the underlying timed automaton where (delay and action) transitions are labeled with their respective costs.

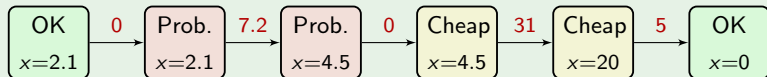
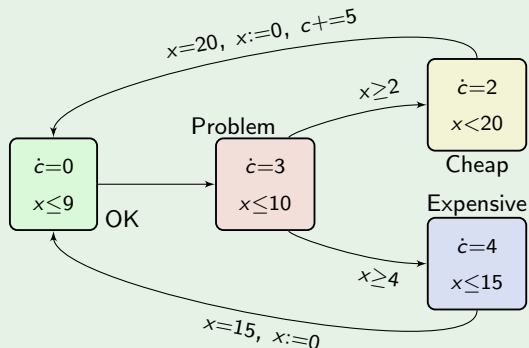
# Priced Timed automata

## Example



# Priced Timed automata

## Example



# Priced (or Weighted) Temporal Logics – Branching time

## Definition

$$\text{WCTL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \vee \varphi \mid \mathbf{E}\varphi \mathbf{U}_{\sim c} \varphi$$

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

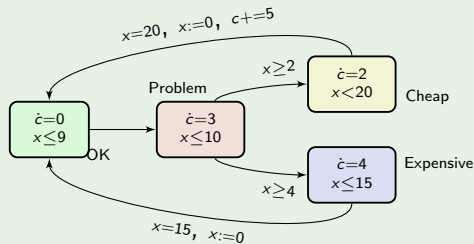
# Priced (or Weighted) Temporal Logics – Branching time

## Definition

$$\text{WCTL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \vee \varphi \mid \mathbf{E} \varphi \mathbf{U}_{\sim c} \varphi$$

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

## Example



$$\mathbf{AG}(\neg \text{OK} \Rightarrow \mathbf{EF}_{c \leq 45} \text{OK})$$

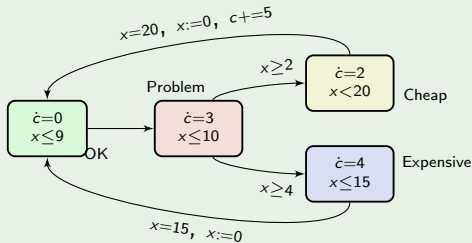
# Priced (or Weighted) Temporal Logics – Branching time

## Definition

$$\text{WCTL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \vee \varphi \mid \mathbf{E} \varphi \mathbf{U}_{\sim c} \varphi$$

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

## Example



$$\mathbf{AG}(\neg \text{OK} \Rightarrow \mathbf{EF}_{t \leq 15} \text{OK})$$

# Priced (or Weighted) Temporal Logics – Branching time

## Definition

$$\text{WCTL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \vee \varphi \mid \mathbf{E} \varphi \mathbf{U}_{\sim c} \varphi$$

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

- Model-checking TCTL is **PSPACE-complete** on TAs [ACD93].

# Priced (or Weighted) Temporal Logics – Branching time

## Definition

$$\text{WCTL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \vee \varphi \mid \mathbf{E}\varphi \mathbf{U}_{\sim c} \varphi$$

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

- Model-checking TCTL is **PSPACE-complete** on TAs [ACD93].
- Model-checking WCTL is **undecidable** on PTAs (with at least 3 clocks) [BBR05].
- Model-checking WCTL is **PSPACE-complete** on 1-clock PTAs [BLM07].

## Priced (or Weighted) Temporal Logics – Linear time

### Definition

$$\text{WMTL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \vee \varphi \mid \varphi \mathbf{U}_{\sim c} \varphi$$

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

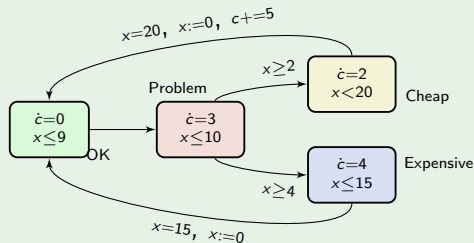
# Priced (or Weighted) Temporal Logics – Linear time

## Definition

$$\text{WMTL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \vee \varphi \mid \varphi \mathbf{U}_{\sim c} \varphi$$

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

## Example



$$\mathbf{G}(\neg \text{OK} \Rightarrow \mathbf{F}_{c \leq 60} \text{OK})$$

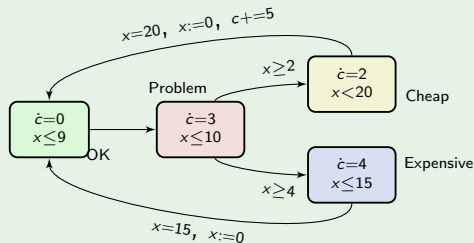
# Priced (or Weighted) Temporal Logics – Linear time

## Definition

$$\text{WMTL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \vee \varphi \mid \varphi \mathbf{U}_{\sim c} \varphi$$

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

## Example



$$\mathbf{G}(\neg \text{OK} \Rightarrow (\mathbf{F}_{t \leq 15} \text{OK} \vee \mathbf{F}_{t \leq 55} \text{OK}))$$

## Priced (or Weighted) Temporal Logics – Linear time

### Definition

$$\text{WMTL} \ni \varphi ::= a \mid \neg \varphi \mid \varphi \vee \varphi \mid \varphi \mathbf{U}_{\sim c} \varphi$$

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

- Model-checking MTL is **undecidable** on TAs [AH93].
- Model-checking MTL is **decidable** on **finite words** under the pointwise semantics [OW05].

# Outline of the talk

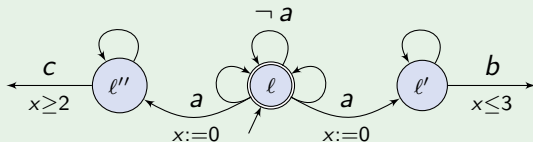
- 1 Introduction
- 2 Priced timed automata and weighted temporal logics
- 3 Decidability results: 1 clock and 1 stopwatch cost**
- 4 Undecidability results: all other cases...
  - 1PTAs with one (3-sloped) cost
  - 2PTAs with one stopwatch cost
  - 1PTAs with two stopwatch costs
- 5 Conclusion

# WMTL is decidable on 1PTAs with 1 stopwatch cost

- encode the formula as a 1-“clock” Alternating Timed Automaton:

## Example

With formula  $\mathbf{G} [a \Rightarrow (\mathbf{F}_{\leq 3} b \vee \mathbf{F}_{\geq 2} c)]$ , we associate:



# WMTL is decidable on 1PTAs with 1 stopwatch cost

- encode the formula as a 1-“clock” Alternating Timed Automaton.
- consider joint executions of the 1PTA  $\mathcal{A}$  and the 1ATA  $\mathcal{B}_\varphi$ :
  - $\rightsquigarrow$  a configuration is
    - a state of  $(q, x)$  of  $\mathcal{A}$  and
    - a set of states  $(\ell_i, x_i)$  of  $\mathcal{B}_\varphi$ .

## Example

$(\ell, 0.7), (\ell', 1.1), (\ell', 1.4), (\ell'', 1.2), (q, 2.4).$

# WMTL is decidable on 1PTAs with 1 stopwatch cost

- encode the formula as a 1-“clock” Alternating Timed Automaton.
- consider joint executions of the 1PTA  $\mathcal{A}$  and the 1ATA  $\mathcal{B}_\varphi$ :  
 $\rightsquigarrow$  a configuration is
  - a state of  $(q, x)$  of  $\mathcal{A}$  and
  - a set of states  $(\ell_i, x_i)$  of  $\mathcal{B}_\varphi$ .
- a configuration is encoded as a sequence of elements of  $(Q \cup L) \times \{0, 1, \dots, M, M + 1\}$  ordered by the fractional parts of the element they encode:

## Example

$(\ell, 0.7), (\ell', 1.1), (\ell', 1.4), (\ell'', 1.2), (q, 2.4)$

is encoded as

$\emptyset, \langle (\ell', 1) \rangle, \langle (\ell'', 1) \rangle, \langle (\ell', 1), (q, 2) \rangle, \langle (\ell, 0) \rangle.$

# WMTL is decidable on 1PTAs with 1 stopwatch cost

- encode the formula as a 1-“clock” Alternating Timed Automaton.
- consider joint executions of the 1PTA  $\mathcal{A}$  and the 1ATA  $\mathcal{B}_\varphi$ :  
 $\rightsquigarrow$  a configuration is
  - a state  $(q, x)$  of  $\mathcal{A}$  and
  - a set of states  $(\ell_i, x_i)$  of  $\mathcal{B}_\varphi$ .
- a configuration is encoded as a sequence of elements of  $(Q \cup L) \times \{0, 1, \dots, M, M + 1\}$  ordered by the fractional parts of the element they encode:
- this yields a time-abstract bisimulation (kind of “region” abstraction).

# WMTL is decidable on 1PTAs with 1 stopwatch cost

- encode the formula as a 1-“clock” Alternating Timed Automaton.
- consider joint executions of the 1PTA  $\mathcal{A}$  and the 1ATA  $\mathcal{B}_\varphi$ :  
 $\rightsquigarrow$  a configuration is
  - a state of  $(q, x)$  of  $\mathcal{A}$  and
  - a set of states  $(\ell_i, x_i)$  of  $\mathcal{B}_\varphi$ .
- a configuration is encoded as a sequence of elements of  $(Q \cup L) \times \{0, 1, \dots, M, M + 1\}$  ordered by the fractional parts of the element they encode:
- this yields a time-abstract bisimulation (kind of “region” abstraction).
- the algorithm is then similar to that of [OW05]:
  - build the resulting “region” graph (which is finite by a well-quasi-ordering argument);
  - decide the existence of an accepting execution.

# Outline of the talk

- 1 Introduction
- 2 Priced timed automata and weighted temporal logics
- 3 Decidability results: 1 clock and 1 stopwatch cost
- 4 Undecidability results: all other cases...
  - 1PTAs with one (3-sloped) cost
  - 2PTAs with one stopwatch cost
  - 1PTAs with two stopwatch costs
- 5 Conclusion

# Undecidability with a 3-sloped cost

## Theorem

*Halting of a two-counter machine can be encoded as a WMTL problem on a 1PTA with one 3-sloped cost.*

# Undecidability with a 3-sloped cost

## Theorem

*Halting of a two-counter machine can be encoded as a WMTL problem on a 1PTA with one 3-sloped cost.*

*Proof.*

- counters  $c_1$  and  $c_2$  are encoded by the clock having value  $2^{-c_1} \cdot 3^{-c_2}$  when entering some states.

# Undecidability with a 3-sloped cost

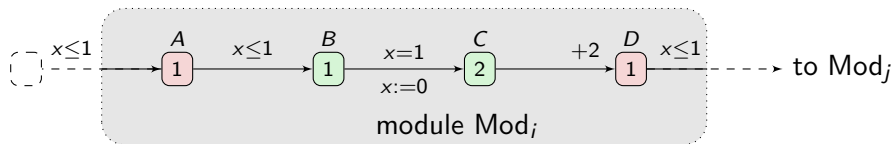
## Theorem

*Halting of a two-counter machine can be encoded as a WMTL problem on a 1PTA with one 3-sloped cost.*

*Proof.*

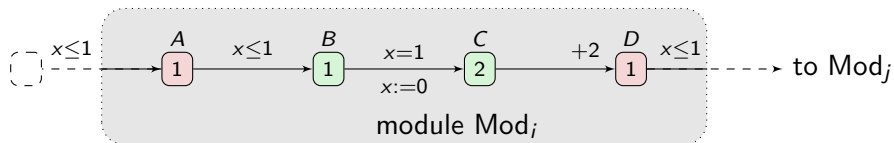
- counters  $c_1$  and  $c_2$  are encoded by the clock having value  $2^{-c_1} \cdot 3^{-c_2}$  when entering some states.
- we have to encode two kinds of instructions:
  - $p_i : c := c + 1; \text{ goto } p_j$
  - $p_i : \text{ if } (c == 0) \text{ then goto } p_j \text{ else } c := c - 1; \text{ goto } p_k$

## Incrementing counter $c_1$



$\rightsquigarrow$  red states are urgent: no time can elapse

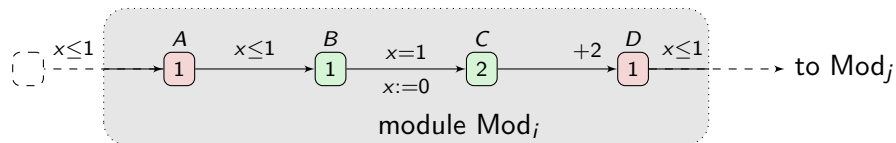
## Incrementing counter $c_1$



$\rightsquigarrow$  red states are urgent: no time can elapse

If the cost between  $A$  and  $D$  is exactly 3, then  $x_D = x_A/2$ .

## Incrementing counter $c_1$

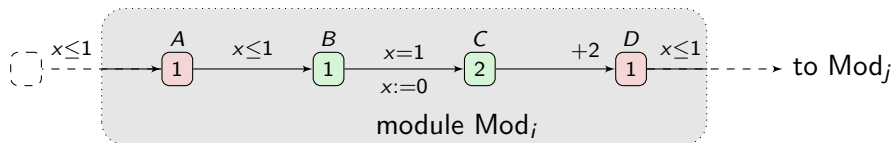


$\rightsquigarrow$  red states are urgent: no time can elapse

If the cost between  $A$  and  $D$  is exactly 3, then  $x_D = x_A/2$ .

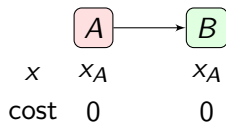
	$A$
$x$	$x_A$
cost	0

## Incrementing counter $c_1$

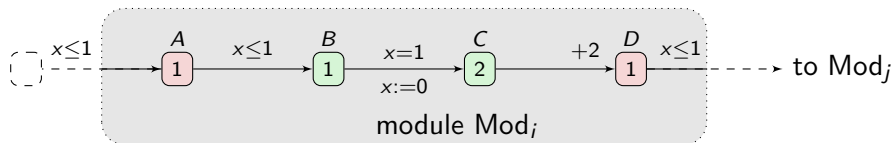


$\rightsquigarrow$  red states are urgent: no time can elapse

If the cost between  $A$  and  $D$  is exactly 3, then  $x_D = x_A/2$ .

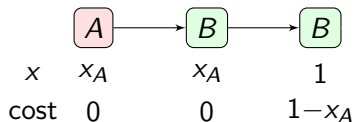


## Incrementing counter $c_1$

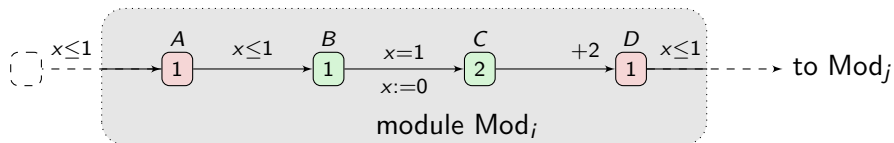


$\rightsquigarrow$  red states are urgent: no time can elapse

If the cost between  $A$  and  $D$  is exactly 3, then  $x_D = x_A/2$ .

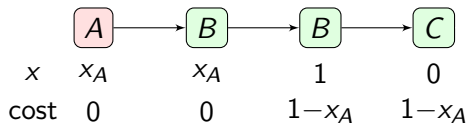


## Incrementing counter $c_1$

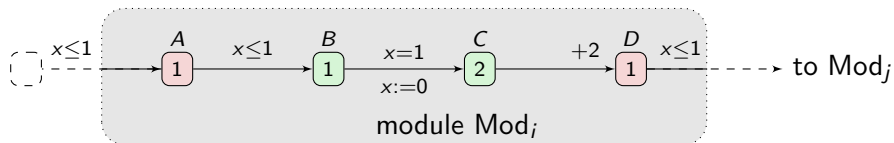


$\rightsquigarrow$  red states are urgent: no time can elapse

If the cost between  $A$  and  $D$  is exactly 3, then  $x_D = x_A/2$ .

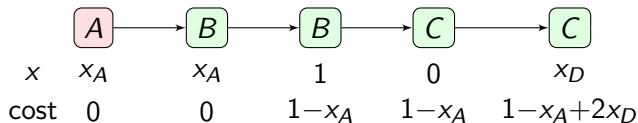


## Incrementing counter $c_1$

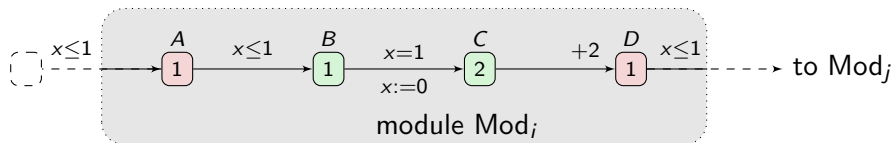


↪ red states are urgent: no time can elapse

If the cost between  $A$  and  $D$  is exactly 3, then  $x_D = x_A/2$ .

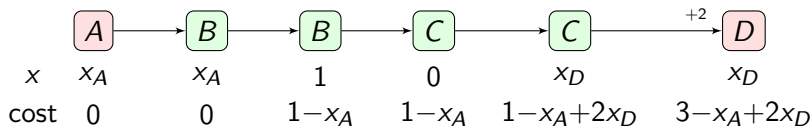


## Incrementing counter $c_1$

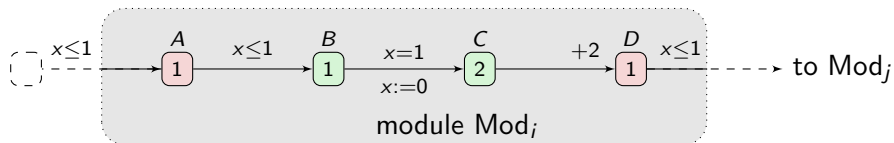


$\rightsquigarrow$  red states are urgent: no time can elapse

If the cost between  $A$  and  $D$  is exactly 3, then  $x_D = x_A/2$ .

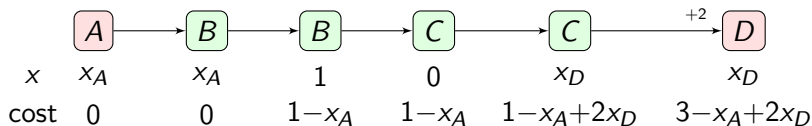


## Incrementing counter $c_1$



$\rightsquigarrow$  red states are urgent: no time can elapse

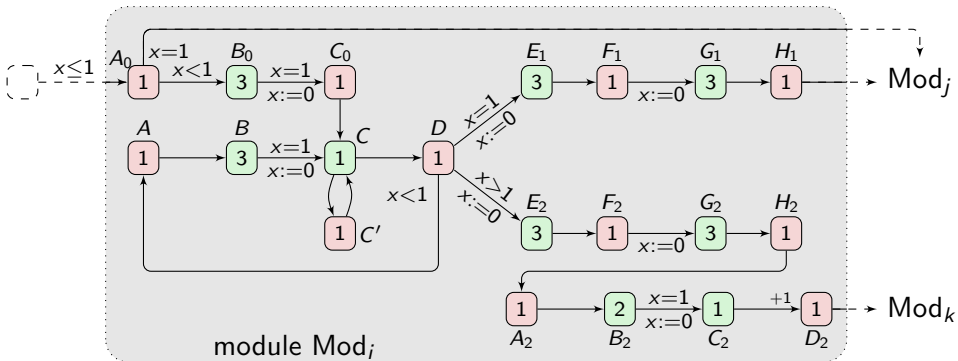
If the cost between  $A$  and  $D$  is exactly 3, then  $x_D = x_A/2$ .



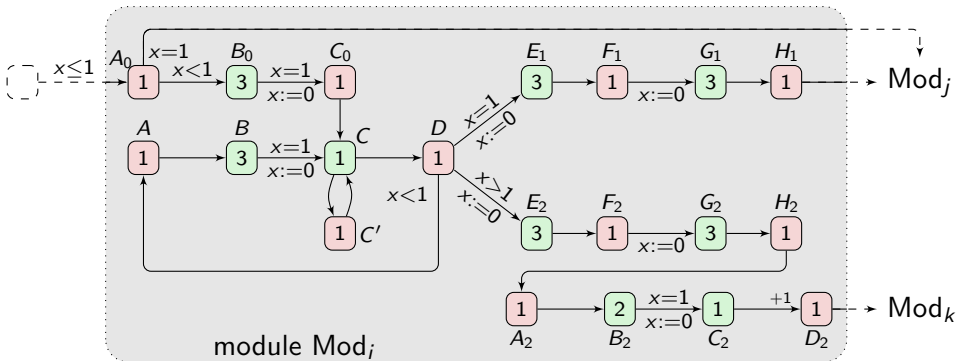
A similar module can be built for incrementing  $c_2$ .

## Testing and decrementing counter $c_1$

# Testing and decrementing counter $c_1$



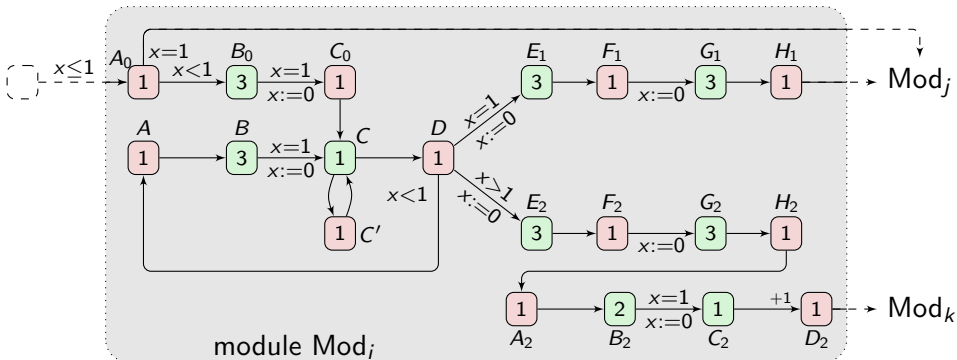
## Testing and decrementing counter $c_1$



Assume that an execution goes to  $\text{Mod}_j$  along which

- each visit to  $C_0$  or  $C'$  is followed by a visit to  $C'$  or  $F_1$ ;

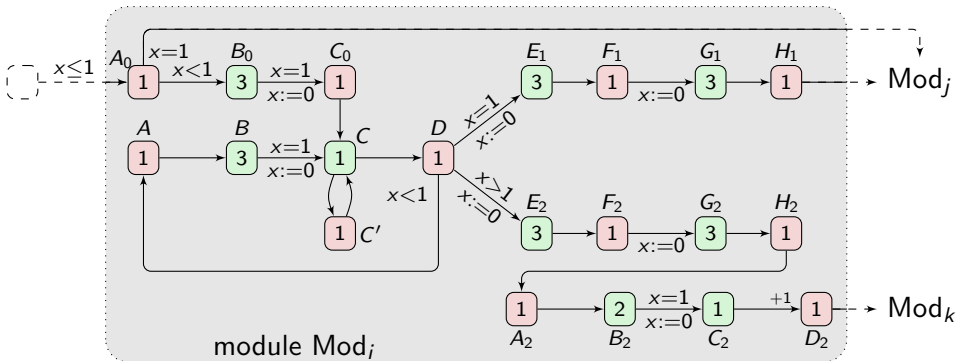
## Testing and decrementing counter $c_1$



Assume that an execution goes to  $Mod_j$  along which

- each visit to  $C_0$  or  $C'$  is followed by a visit to  $C'$  or  $F_1$ ;
- the cost is 3 between two consecutive visits to  $A$  or  $A_0$  and  $D$ ;  $C_0$  or  $C'$  and  $C'$  or  $F_1$ ;  $D$  and  $H_1$ .

## Testing and decrementing counter $c_1$

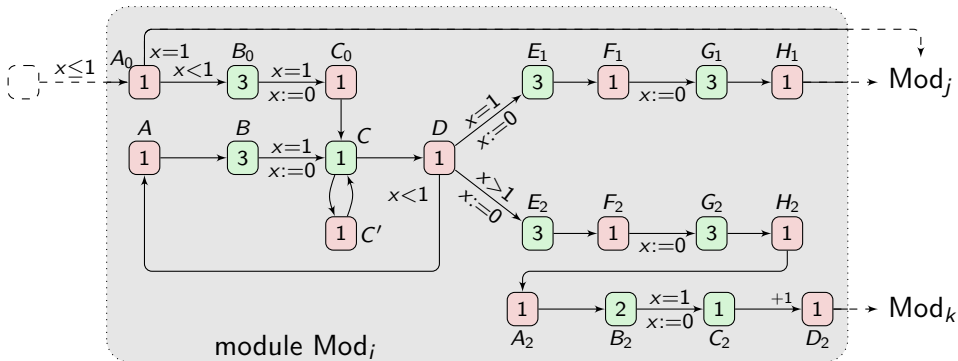


Assume that an execution goes to  $\text{Mod}_j$  along which

- each visit to  $C_0$  or  $C'$  is followed by a visit to  $C'$  or  $F_1$ ;
- the cost is 3 between two consecutive visits to  $A$  or  $A_0$  and  $D$ ;  
 $C_0$  or  $C'$  and  $C'$  or  $F_1$ ;  $D$  and  $H_1$ .

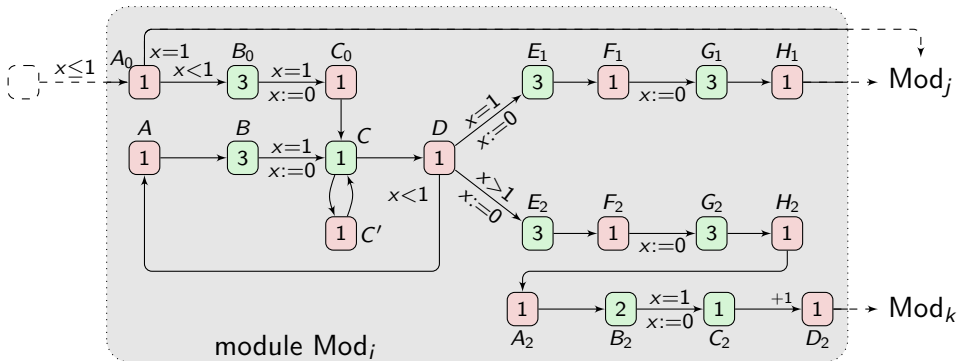
Then  $x_{H_1} = x_{A_0}$  and there is an integer  $n$  s.t.  $x_{A_0} = 3^{-n}$ .

## Testing and decrementing counter $c_1$



- at the  $k$ -th visit in  $D$ , we have  $x = 3^k \cdot x_{A_0}$ ;
- at the  $k$ -th visit in  $C_0$  or  $C'$ , we have  $x = (3^k - 3) \cdot x_{A_0}$ ;

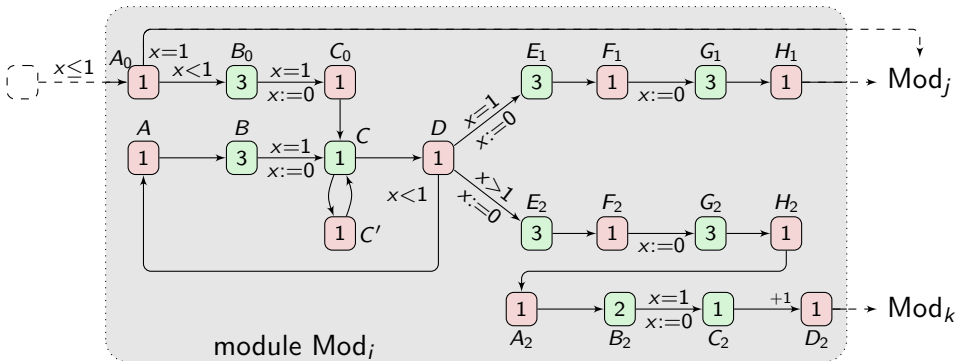
## Testing and decrementing counter $c_1$



- at the  $k$ -th visit in  $D$ , we have  $x = 3^k \cdot x_{A_0}$ ;
- at the  $k$ -th visit in  $C_0$  or  $C'$ , we have  $x = (3^k - 3) \cdot x_{A_0}$ ;

It easily follows that  $x = x_{A_0}$  in  $H_1$ .

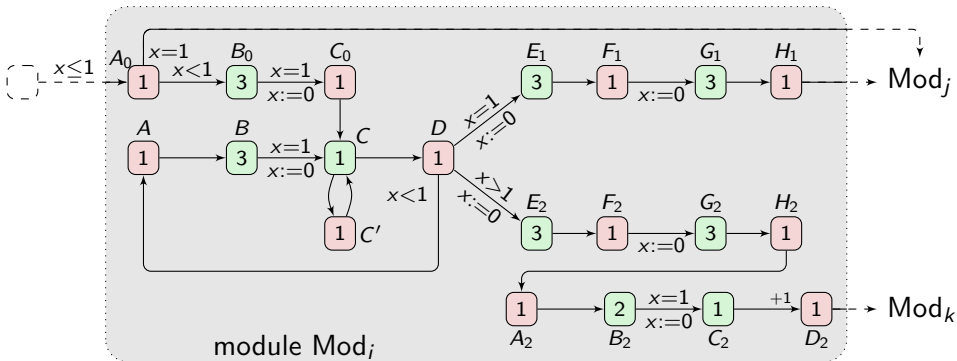
## Testing and decrementing counter $c_1$



Assume that an execution goes to  $\text{Mod}_k$  along which

- each visit to  $C_0$  or  $C'$  is followed by a visit to  $C'$  or  $F_2$ ;

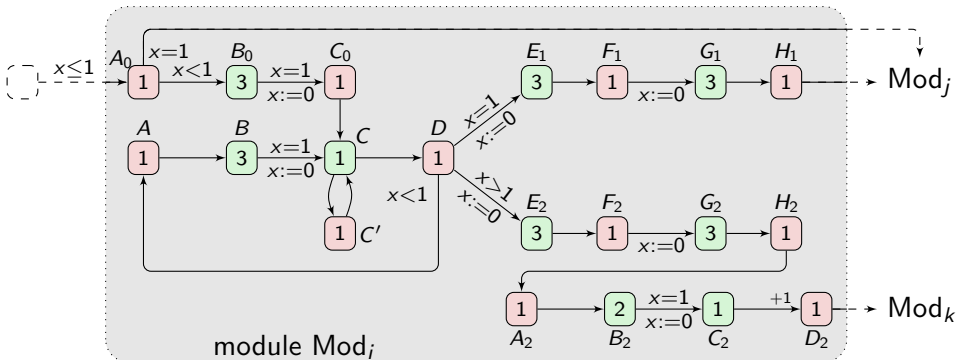
## Testing and decrementing counter $c_1$



Assume that an execution goes to  $\text{Mod}_k$  along which

- each visit to  $C_0$  or  $C'$  is followed by a visit to  $C'$  or  $F_2$ ;
- the cost is 3 between two consecutive visits to  $A$  or  $A_0$  and  $D$ ;  $C_0$  or  $C'$  and  $C'$  or  $F_2$ ;  $D$  and  $H_2$ ;  $H_2$  and  $D_2$ .

## Testing and decrementing counter $c_1$



Assume that an execution goes to  $\text{Mod}_k$  along which

- each visit to  $C_0$  or  $C'$  is followed by a visit to  $C'$  or  $F_2$ ;
- the cost is 3 between two consecutive visits to  $A$  or  $A_0$  and  $D$ ;  
 $C_0$  or  $C'$  and  $C'$  or  $F_2$ ;  $D$  and  $H_2$ ;  $H_2$  and  $D_2$ .

Then  $x_{D_2} = 2 \cdot x_{A_0}$  and for all integer  $n$ ,  $x_{A_0} \neq 3^{-n}$ .

## Global reduction

- Plug the modules according to the 2-counter machine;

## Global reduction

- Plug the modules according to the 2-counter machine;
- urgency in any state  $S$  is enforced by a formula

$$\mathbf{G}(S \Rightarrow S \mathbf{U}_{=0} \neg S);$$

## Global reduction

- Plug the modules according to the 2-counter machine;
- urgency in any state  $S$  is enforced by a formula

$$\mathbf{G}(S \Rightarrow S \mathbf{U}_{=0} \neg S);$$

- cost-constraints between the states of the modules are enforced with formulas of the form

$$\mathbf{G}[(A \vee A_0) \Rightarrow (\neg D \mathbf{U}_{=3} D)].$$

## Global reduction

- Plug the modules according to the 2-counter machine;
- urgency in any state  $S$  is enforced by a formula

$$\mathbf{G}(S \Rightarrow S \mathbf{U}_{=0} \neg S);$$

- cost-constraints between the states of the modules are enforced with formulas of the form

$$\mathbf{G}[(A \vee A_0) \Rightarrow (\neg D \mathbf{U}_{=3} D)].$$

- Finally, the aim is to reach the halting state:

**F** Halt.

# Outline of the talk

- 1 Introduction
- 2 Priced timed automata and weighted temporal logics
- 3 Decidability results: 1 clock and 1 stopwatch cost
- 4 Undecidability results: all other cases...**
  - 1PTAs with one (3-sloped) cost
  - 2PTAs with one stopwatch cost**
  - 1PTAs with two stopwatch costs
- 5 Conclusion

# Undecidability with two clocks and one stopwatch cost

## Theorem

*Halting of a two-counter machine can be encoded as a WMTL problem on a 2PTA with a stopwatch cost.*

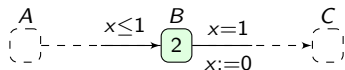
# Undecidability with two clocks and one stopwatch cost

## Theorem

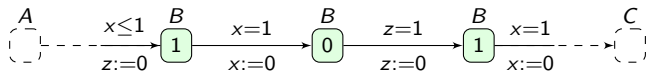
*Halting of a two-counter machine can be encoded as a WMTL problem on a 2PTA with a stopwatch cost.*

*Proof.*

In the previous construction, replace



with



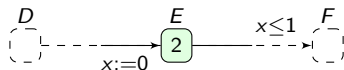
# Undecidability with two clocks and one stopwatch cost

## Theorem

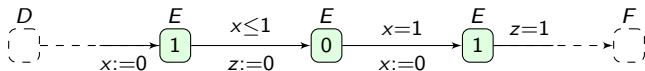
*Halting of a two-counter machine can be encoded as a WMTL problem on a 2PTA with a stopwatch cost.*

*Proof.*

and



with



# Outline of the talk

- 1 Introduction
- 2 Priced timed automata and weighted temporal logics
- 3 Decidability results: 1 clock and 1 stopwatch cost
- 4 Undecidability results: all other cases...**
  - 1PTAs with one (3-sloped) cost
  - 2PTAs with one stopwatch cost
  - **1PTAs with two stopwatch costs**
- 5 Conclusion

# Undecidability with two stopwatch costs

## Theorem

*Halting of a two-counter machine can be encoded as a WMTL problem on a 2PTA with a stopwatch cost.*

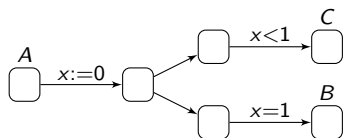
# Undecidability with two stopwatch costs

## Theorem

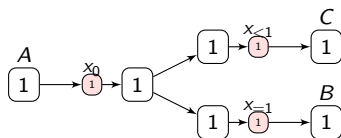
*Halting of a two-counter machine can be encoded as a WMTL problem on a 2PTA with a stopwatch cost.*

*Proof.*

In the previous construction, replace



with



and enforce that

$$\mathbf{G} \left[ (x_0 \wedge \neg x_0 \mathbf{U} x_{=1}) \Rightarrow (\neg x_0 \mathbf{U}_{c=1} x_{=1}) \right]$$

# Outline of the talk

- 1 Introduction
- 2 Priced timed automata and weighted temporal logics
- 3 Decidability results: 1 clock and 1 stopwatch cost
- 4 Undecidability results: all other cases...
  - 1PTAs with one (3-sloped) cost
  - 2PTAs with one stopwatch cost
  - 1PTAs with two stopwatch costs
- 5 Conclusion

# Conclusion

- Priced timed automata are a nice extension of TAs
- Unfortunately, this extension comes with a price (!):
  - model checking for this extension is undecidable;
  - it can be made decidable under very restrictive conditions.

# Conclusion

- Priced timed automata are a nice extension of TAs
- Unfortunately, this extension comes with a price (!):
  - model checking for this extension is undecidable;
  - it can be made decidable under very restrictive conditions.
- Corollary of our results: undecidability of reachability for linear hybrid automata with 2 clocks and one 3-sloped variable.