Model-Checking One-Clock Priced Timed Automata

Patricia Bouyer[1,2] Kim G. Larsen[3] Nicolas Markey[1]

[1]LSV, CNRS & ENS Cachan, France
[2]Oxford University, England
[3]Aalborg University, Denmark

Model checking

system:

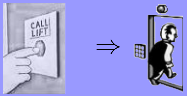
property:

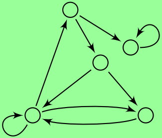
$\Rightarrow$

Model checking

system:

property:





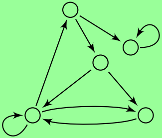
$\mathbf{G}(\texttt{request} \Rightarrow \mathbf{F}\texttt{grant})$

Model checking

system:

property:

$\Rightarrow$



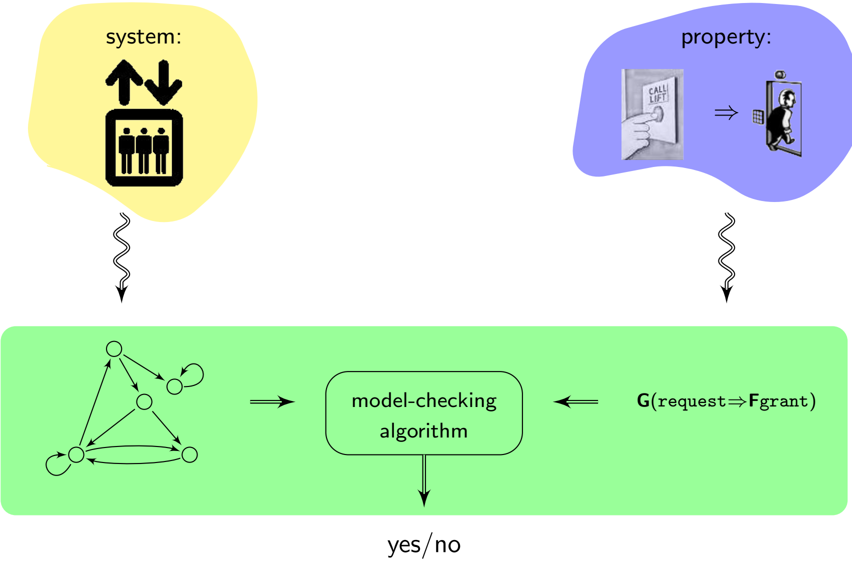
$\Longrightarrow$

model-checking algorithm

$\Longleftarrow$

$\mathbf{G}(\mathtt{request} \Rightarrow \mathbf{F}\mathtt{grant})$

Model checking



Motivation

Adding timing requirements

Motivation

Adding timing requirements

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

Motivation

Adding timing requirements

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

$\rightsquigarrow$ timed automata, time(d) Petri nets, timed process algebras. . .

Motivation

Adding timing requirements

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

  $\rightsquigarrow$ timed automata, time(d) Petri nets, timed process algebras...
- Need for time in specifications:
  - again, the behaviour of most systems depends on time
  - untimed specifications are not enough (e.g., *bounded* response prop.)

Motivation

Adding timing requirements

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

  $\rightsquigarrow$ timed automata, time(d) Petri nets, timed process algebras...
- Need for time in specifications:
  - again, the behaviour of most systems depends on time
  - untimed specifications are not enough (e.g., *bounded* response prop.)

  $\rightsquigarrow$ TCTL, MTL, TPTL, timed μ-calculus...

Motivation

Adding timing requirements

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

  $\rightsquigarrow$ timed automata, time(d) Petri nets, timed process algebras...
- Need for time in specifications:
  - again, the behaviour of most systems depends on time
  - untimed specifications are not enough (e.g., *bounded* response prop.)

  $\rightsquigarrow$ TCTL, MTL, TPTL, timed μ-calculus...

Time is not always sufficient! We may want to measure not only time, but also energy consumption, price to pay...

Motivation

Adding timing requirements

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

  $\rightsquigarrow$ timed automata, time(d) Petri nets, timed process algebras...
- Need for time in specifications:
  - again, the behaviour of most systems depends on time
  - untimed specifications are not enough (e.g., *bounded* response prop.)

  $\rightsquigarrow$ TCTL, MTL, TPTL, timed μ-calculus...

Time is not always sufficient! We may want to measure not only time, but also energy consumption, price to pay...

- hybrid automata: timed automata augmented with variables whose derivatives are not constant

Motivation

Adding timing requirements

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

  $\rightsquigarrow$ timed automata, time(d) Petri nets, timed process algebras...
- Need for time in specifications:
  - again, the behaviour of most systems depends on time
  - untimed specifications are not enough (e.g., *bounded* response prop.)

  $\rightsquigarrow$ TCTL, MTL, TPTL, timed μ-calculus...

Time is not always sufficient! We may want to measure not only time, but also energy consumption, price to pay...

- hybrid automata: timed automata augmented with variables whose derivatives are not constant

  "Hybrid automata are mostly undecidable." [HKPV97]

Motivation

Adding timing requirements

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

  $\rightsquigarrow$ timed automata, time(d) Petri nets, timed process algebras...
- Need for time in specifications:
  - again, the behaviour of most systems depends on time
  - untimed specifications are not enough (e.g., *bounded* response prop.)

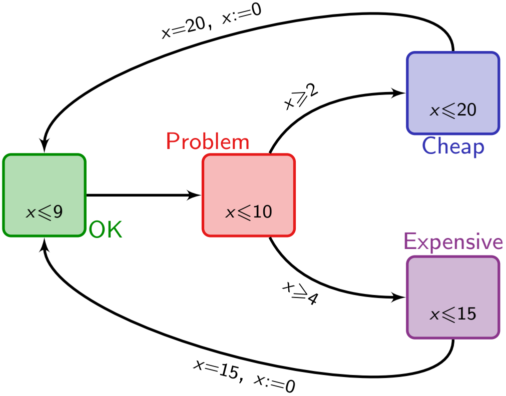
  $\rightsquigarrow$ TCTL, MTL, TPTL, timed μ-calculus...

Time is not always sufficient! We may want to measure not only time, but also energy consumption, price to pay...

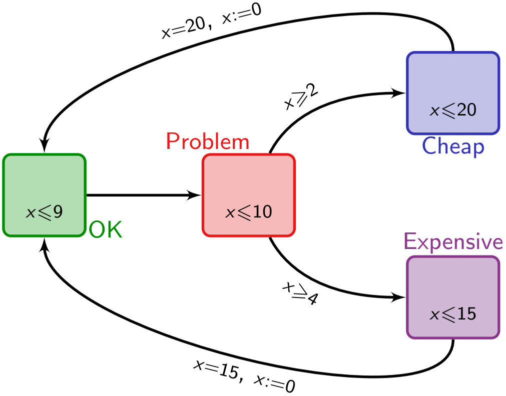
- hybrid automata: timed automata augmented with variables whose derivatives are not constant

  "Hybrid automata are mostly undecidable." [HKPV97]
- priced timed automata: similar to hybrid automata, but the behaviour only depends on clock variables

Timed automata

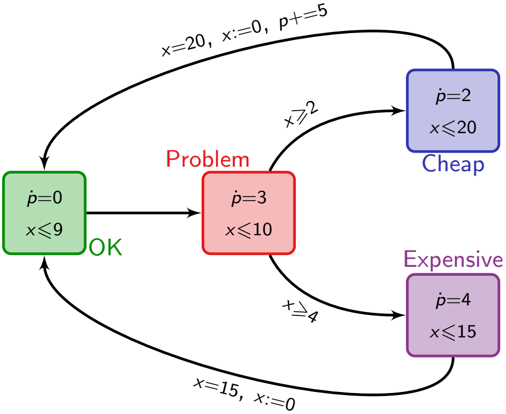


Timed automata



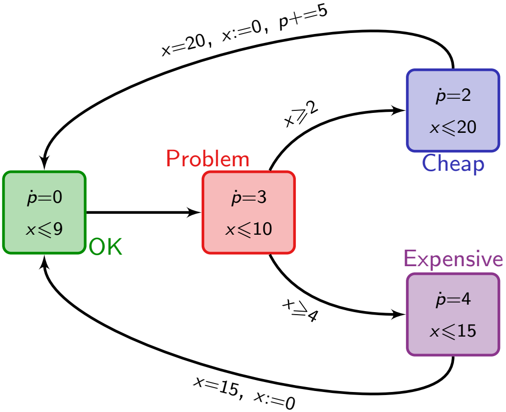
$$\begin{matrix}\text{OK}\\0\end{matrix} \dashrightarrow \begin{matrix}\text{OK}\\5.2\end{matrix} \longrightarrow \begin{matrix}\text{Problem}\\5.2\end{matrix} \dashrightarrow \begin{matrix}\text{Problem}\\6.8\end{matrix} \longrightarrow \begin{matrix}\text{Cheap}\\6.8\end{matrix} \dashrightarrow \begin{matrix}\text{Cheap}\\20\end{matrix} \longrightarrow \begin{matrix}\text{OK}\\0\end{matrix}$$

Priced (weighted) timed automata



$$\begin{array}{c}\text{OK}\\0\end{array} \dashrightarrow \begin{array}{c}\text{OK}\\5.2\end{array} \longrightarrow \begin{array}{c}\text{Problem}\\5.2\end{array} \dashrightarrow \begin{array}{c}\text{Problem}\\6.8\end{array} \longrightarrow \begin{array}{c}\text{Cheap}\\6.8\end{array} \dashrightarrow \begin{array}{c}\text{Cheap}\\20\end{array} \longrightarrow \begin{array}{c}\text{OK}\\0\end{array}$$

Priced (weighted) timed automata



$$\begin{matrix}\text{OK}\\0\end{matrix} \overset{0}{\dashrightarrow} \begin{matrix}\text{OK}\\5.2\end{matrix} \xrightarrow{0} \begin{matrix}\text{Problem}\\5.2\end{matrix} \overset{4.8}{\dashrightarrow} \begin{matrix}\text{Problem}\\6.8\end{matrix} \xrightarrow{0} \begin{matrix}\text{Cheap}\\6.8\end{matrix} \overset{26.4}{\dashrightarrow} \begin{matrix}\text{Cheap}\\20\end{matrix} \xrightarrow{5} \begin{matrix}\text{OK}\\0\end{matrix}$$

Related works

- **Basic properties**
  - Optimal reachability [ATP01,BFH+01,BFH+01b,LBB+01,BBBR07]
  - Mean-cost optimality [BBL04]

Related works

- **Basic properties**
  - Optimal reachability [ATP01,BFH+01,BFH+01b,LBB+01,BBBR07]
  - Mean-cost optimality [BBL04]

- **Model-checking of WCTL**
  - Undecidability for timed automata with more than three clocks [BBR04,BBM06]
  - Decidability for timed automata with one clock [this paper]

Related works

- **Basic properties**
  - Optimal reachability [ATP01,BFH+01,BFH+01b,LBB+01,BBBR07]
  - Mean-cost optimality [BBL04]
- **Model-checking of WCTL**
  - Undecidability for timed automata with more than three clocks [BBR04,BBM06]
  - Decidability for timed automata with one clock [this paper]
- **Control games**
  - Properties, and restricted decidability results [ABM04,BCFL04,BCFL05]
  - Undecidability for timed automata with more than three clocks [BBR05,BBM06]
  - Decidability for timed automata with one clock [BLMR06]

The logic WCTL

$$\mathsf{CTL} \ni \varphi \ ::= \ \texttt{true} \ | \ \alpha \ | \ \neg\varphi \ | \ \varphi \vee \varphi \ | \ \mathbf{E}\,\varphi\mathbf{U}\varphi \ | \ \mathbf{A}\,\varphi\mathbf{U}\varphi$$

The logic WCTL

$$\mathsf{WCTL} \ni \varphi \ ::= \ \mathtt{true} \ \mid \ \alpha \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \mathbf{E}\,\varphi \mathbf{U}_{P\sim c}\varphi \ \mid \ \mathbf{A}\,\varphi \mathbf{U}_{P\sim c}\varphi$$

where P is a cost variable, $c \in \mathbb{N}$, and $\sim \in \{<, \leqslant, =, \geqslant, >\}$.

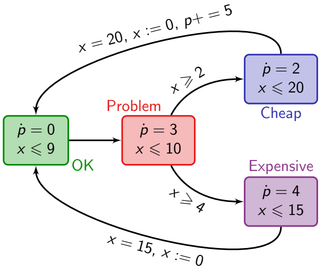
The logic WCTL

$$\text{WCTL} \ni \varphi \ ::= \ \texttt{true} \ \mid \ \alpha \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \mathbf{E}\,\varphi \mathbf{U}_{P\sim c}\varphi \ \mid \ \mathbf{A}\,\varphi \mathbf{U}_{P\sim c}\varphi$$

where P is a cost variable, $c \in \mathbb{N}$, and $\sim \in \{<, \leqslant, =, \geqslant, >\}$.

Particular case: P is the time elapsed $\rightarrow$ TCTL

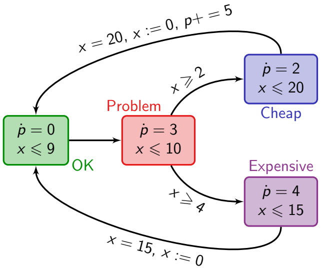
An example



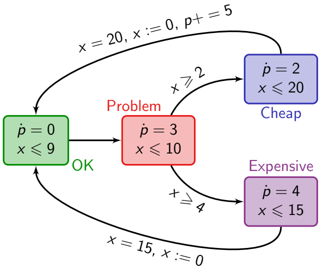
x = 20, x := 0, p+ = 5
$\dot{p} = 2$
$x \leqslant 20$
Cheap
$x \geqslant 2$
Problem
$\dot{p} = 0$
$x \leqslant 9$
OK
$\dot{p} = 3$
$x \leqslant 10$
$x \geqslant 4$
Expensive
$\dot{p} = 4$
$x \leqslant 15$
$x = 15, x := 0$

An example

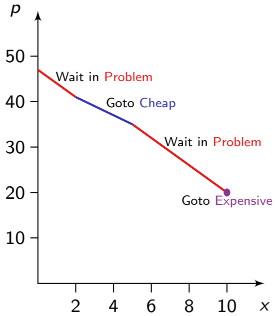
- $\mathbf{A}\,\mathbf{G}(\text{Problem} \implies \mathbf{E}\,\mathbf{F}_{p \leqslant 47}\text{OK})$



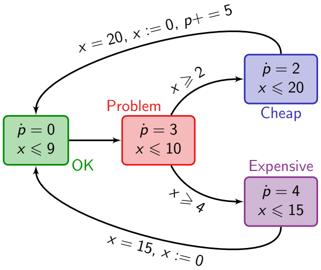
An example



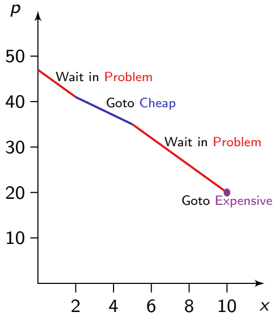
- $\mathbf{A}\,\mathbf{G}(\text{Problem} \implies \mathbf{E}\,\mathbf{F}_{p \leqslant 47}\text{OK})$



An example

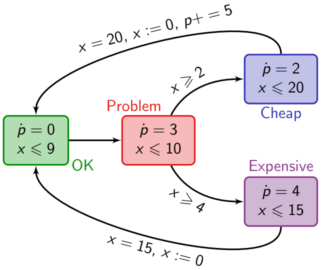


- **A G**(Problem $\implies$ **E F**$_{p \leqslant 47}$OK)

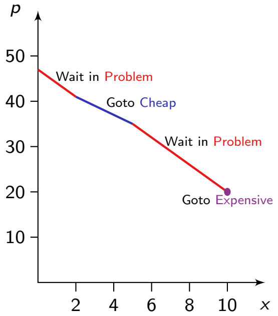


- **A G**(Problem $\implies$ **A F**$_{p \leqslant 56}$OK)

An example



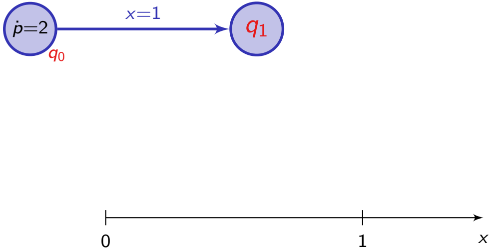
- $\mathbf{A\,G}(\text{Problem} \implies \mathbf{E\,F}_{p \leqslant 47}\text{OK})$



- $\mathbf{A\,G}(\text{Problem} \implies \mathbf{A\,F}_{p \leqslant 56}\text{OK})$
- $\mathbf{A\,G}(\neg\mathbf{E}\,(\text{OK}\,\mathbf{U}_{t \geqslant 8}(\text{Problem} \wedge \neg\mathbf{E\,F}_{p < 30}\text{OK})))$

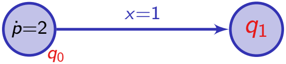
Model-checking WCTL

First remark: Regions cannot be used for model-checking WCTL...



Model-checking WCTL

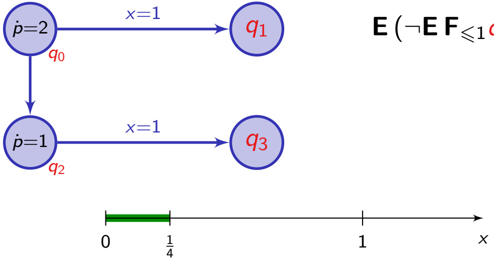
First remark: Regions cannot be used for model-checking WCTL...



$\mathbf{E}\,\mathbf{F}_{\leqslant 1}\,q_1$

Model-checking WCTL

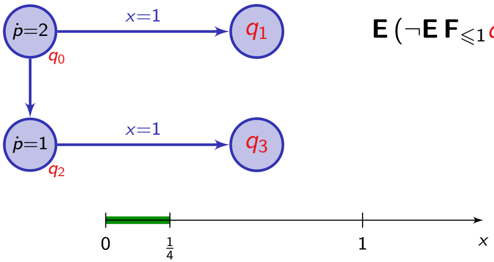
First remark: Regions cannot be used for model-checking WCTL...



$$\mathbf{E}\,(\neg \mathbf{E}\,\mathbf{F}_{\leqslant 1}\, q_1)\,\mathbf{U}_{\geqslant 1}\, q_3$$

Model-checking WCTL

First remark: Regions cannot be used for model-checking WCTL...



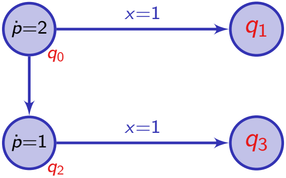
$$\mathbf{E}\,(\neg \mathbf{E}\,\mathbf{F}_{\leqslant 1}\, q_1)\,\mathbf{U}_{\geqslant 1}\, q_3$$

Theorem [BBR05,BBM06]

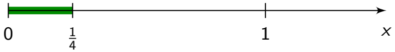
Model-checking three-clock priced timed automata is undecidable.

Model-checking WCTL

First remark: Regions cannot be used for model-checking WCTL...



$$\mathbf{E}\,(\neg \mathbf{E}\,\mathbf{F}_{\leqslant 1}\, q_1)\,\mathbf{U}_{\geqslant 1}\, q_3$$



Theorem [BBR05,BBM06]

Model-checking three-clock priced timed automata is undecidable.

Our result

Model-checking one-clock priced timed automata is PSPACE-complete.

Refining regions

We refine regions as suggested by the previous example:

A sufficient granularity

Let $\Phi \in \mathsf{WCTL}$, 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 exist finitely many constants

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

such that

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

where $h(\Phi)$ is the maximal number of *constrained* modalities in Φ.

Refining regions

We refine regions as suggested by the previous example:

A sufficient granularity

Let $\Phi \in$ 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 exist finitely many constants

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

such that

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

where $h(\Phi)$ is the maximal number of *constrained* modalities in Φ.

Inductive proof:

- for atomic propositions, the a_i's are the constants that appear in the constraints of the automaton.

Refining regions

We refine regions as suggested by the previous example:

A sufficient granularity

Let $\Phi \in$ 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 exist finitely many constants

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

such that

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

where $h(\Phi)$ is the maximal number of *constrained* modalities in Φ.

- If $\Phi = \mathbf{E}\,(\varphi_1 \mathbf{U}_{\sim c} \varphi_2)$, assume we have computed the a_i's for φ_1 and φ_2.

Refining regions

- If $\Phi = \mathbf{E}\left(\varphi_1 \mathbf{U}_{\sim c} \varphi_2\right)$, assume we have computed the a_i's for φ_1 and φ_2.

Refining regions

- If $\Phi = \mathbf{E}\,(\varphi_1 \mathbf{U}_{\sim c} \varphi_2)$, assume we have computed the a_i's for φ_1 and φ_2.

Lemma

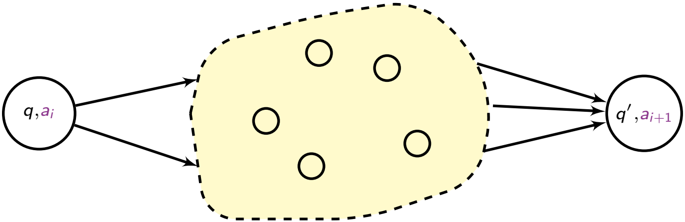
The set of costs of non-resetting runs between (q, a_i) and (q', a_{i+1}) is an interval.

Refining regions

- If $\Phi = \mathbf{E}\left(\varphi_1 \mathbf{U}_{\sim c} \varphi_2\right)$, assume we have computed the a_i's for φ_1 and φ_2.

Lemma

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

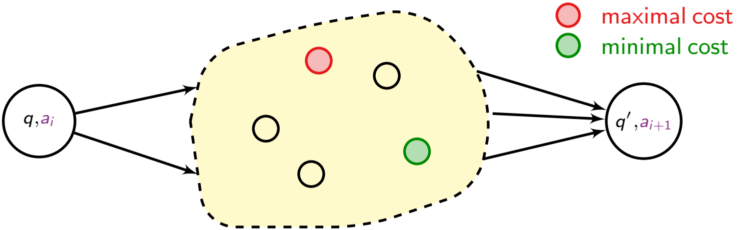


Refining regions

- If $\Phi = \mathbf{E}\,(\varphi_1 \mathbf{U}_{\sim c} \varphi_2)$, assume we have computed the a_i's for φ_1 and φ_2.

Lemma

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



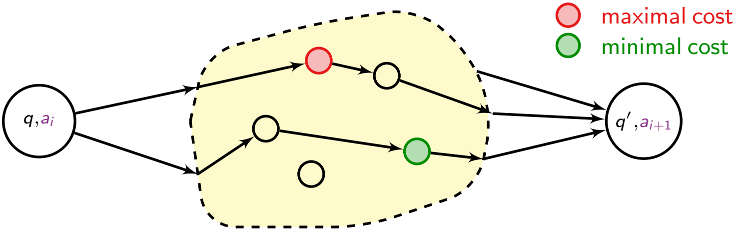
$$\text{costs} \quad \subseteq \quad [c_{\min},\ c_{\max}] \cdot (a_{i+1} - a_i)$$

Refining regions

- If $\Phi = \mathbf{E}(\varphi_1 \mathbf{U}_{\sim c} \varphi_2)$, assume we have computed the a_i's for φ_1 and φ_2.

Lemma

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



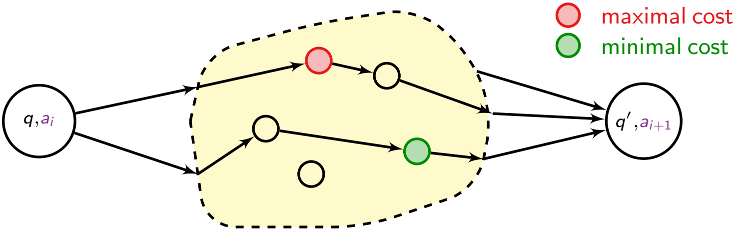
$$\text{costs} \quad \subseteq \quad [c_{\min},\ c_{\max}] \cdot (a_{i+1} - a_i)$$

Refining regions

- If $\Phi = \mathbf{E}\,(\varphi_1 \mathbf{U}_{\sim c} \varphi_2)$, assume we have computed the a_i's for φ_1 and φ_2.

Lemma

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



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

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

Refining regions

- If $\Phi = \mathbf{E}\,(\varphi_1 \mathbf{U}_{\sim c} \varphi_2)$, assume we have computed the a_i's for φ_1 and φ_2.

$$\boxed{q, a_i} \xrightarrow{\langle c_{\min}, c_{\max}\rangle \cdot (a_{i+1} - a_i)} \boxed{q, a_{i+1}}$$

$$\boxed{q, a_i} \xrightarrow{\langle 0, c_{\max}\rangle \cdot (a_{i+1} - a_i)} \boxed{q, (a_i, a_{i+1})}$$

$$\boxed{q, a_n} \xrightarrow{[0,0] \text{ or } \langle 0, +\infty\rangle} \boxed{q, (a_n, +\infty)}$$

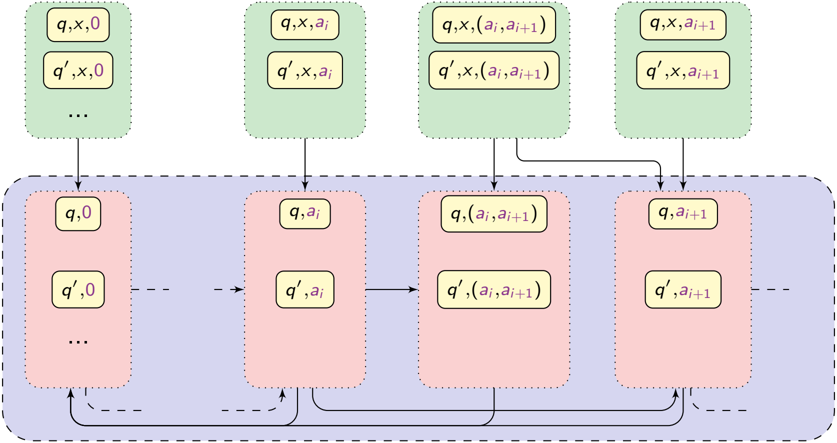
$$\boxed{q, \text{any}} \xrightarrow{(x{:=}0)\quad [0,0]} \boxed{q, 0}$$

$$\boxed{q, x, (a_i, a_{i+1})} \xrightarrow{\langle c_{\min}, c_{\max}\rangle \cdot (a_{i+1} - x)} \boxed{q, a_{i+1}}$$

$$\vdots$$

Refining regions

- If $\Phi = \mathbf{E}\,(\varphi_1 \mathbf{U}_{\sim c} \varphi_2)$, assume we have computed the a_i's for φ_1 and φ_2.



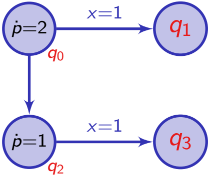
Refining regions

- If $\Phi = \mathbf{E}\,(\varphi_1 \mathbf{U}_{\sim c} \varphi_2)$, assume we have computed the a_i's for φ_1 and φ_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
  - α and α' are in $\mathbb{N}/C^{\max\{h(\varphi_1),h(\varphi_2)\}}$,
  - β and β' are costs of the automaton (in $\mathbb{N}/C$).
- the set of values for x s.t. $(q, x) \models \mathbf{E}\,\varphi_1 \mathbf{U}_{\sim c} \varphi_2$ is a finite union of intervals whose bounds are multiples of $1/C^{h(\Phi)}$ and bounded by M.
- if the formula holds, it has an exponential-sized witness.

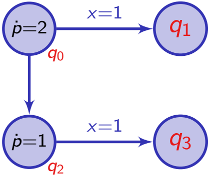
Back to the example



$$\mathbf{E}\,(\neg \mathbf{E}\,\mathbf{F}_{\leqslant 1} q_1)\,\mathbf{U}_{\geqslant 1} q_3$$

Constants for $\mathbf{E}\,\mathbf{F}_{\leqslant 1} q_1$ are 0, 1/2 and 1.

Back to the example

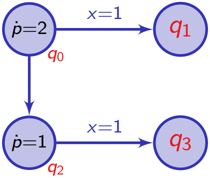


$$\mathbf{E}\,(\neg \mathbf{E}\,\mathbf{F}_{\leqslant 1}\, q_1)\,\mathbf{U}_{\geqslant 1}\, q_3$$

Constants for $\mathbf{E}\,\mathbf{F}_{\leqslant 1} q_1$ are 0, 1/2 and 1.

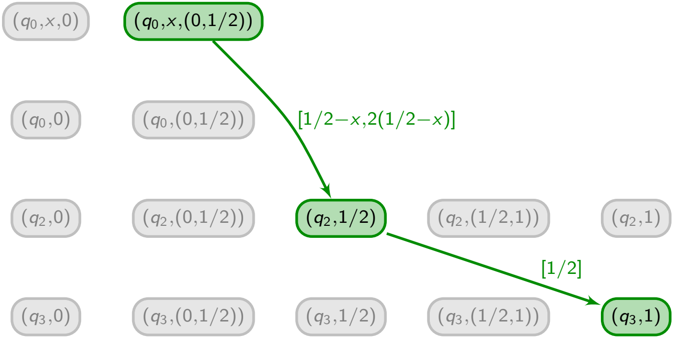
$(q_0,x,0)$	$(q_0,x,(0,1/2))$	$(q_0,x,1/2)$	$(q_0,x,(1/2,1))$	$(q_0,x,1)$
$(q_0,0)$	$(q_0,(0,1/2))$	$(q_0,1/2)$	$(q_0,(1/2,1))$	$(q_0,1)$
$(q_2,0)$	$(q_2,(0,1/2))$	$(q_2,1/2)$	$(q_2,(1/2,1))$	$(q_2,1)$
$(q_3,0)$	$(q_3,(0,1/2))$	$(q_3,1/2)$	$(q_3,(1/2,1))$	$(q_3,1)$

Back to the example

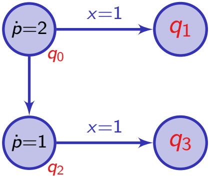


$$\mathbf{E}\,(\neg\mathbf{E}\,\mathbf{F}_{\leqslant 1}\,q_1)\,\mathbf{U}_{\geqslant 1}\,q_3$$

Constants for $\mathbf{E}\,\mathbf{F}_{\leqslant 1}\,q_1$ are 0, 1/2 and 1.



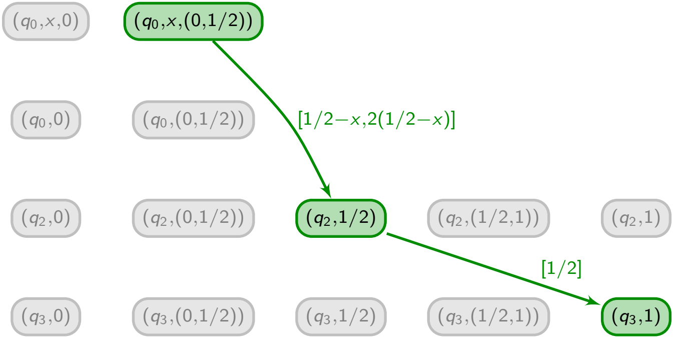
Back to the example



$$\mathbf{E}\,(\neg\mathbf{E}\,\mathbf{F}_{\leqslant 1}q_1)\,\mathbf{U}_{\geqslant 1}\,q_3$$

Constants for $\mathbf{E}\,\mathbf{F}_{\leqslant 1}q_1$ are 0, 1/2 and 1.

$$[1-x, 3/2-2x] \cap [1, +\infty) \neq \emptyset \iff x \leqslant 1/4$$



Algorithms

If $\Phi = \varphi_1 \mathbf{U}_{\sim c} \varphi_2$, assume we have computed the a_i's for φ_1 and φ_2:

An "obvious" EXPTIME algorithm

for every $x = k/2C^{h(\Phi)}$,

Algorithms

If $\Phi = \varphi_1 \mathbf{U}_{\sim c} \varphi_2$, assume we have computed the a_i's for φ_1 and φ_2:

An "obvious" EXPTIME algorithm

for every $x = k/2C^{h(\Phi)}$,

- non-deterministically guess a witnessing path in the graph,

Algorithms

If $\Phi = \varphi_1 \mathbf{U}_{\sim c} \varphi_2$, assume we have computed the a_i's for φ_1 and φ_2:

An "obvious" EXPTIME algorithm

for every $x = k/2C^{h(\Phi)}$,

- non-deterministically guess a witnessing path in the graph,
- check that this path satisfies $\varphi_1 \mathbf{U} \varphi_2$,

Algorithms

If $\Phi = \varphi_1 \mathbf{U}_{\sim c} \varphi_2$, assume we have computed the a_i's for φ_1 and φ_2:

An "obvious" EXPTIME algorithm

for every $x = k/2C^{h(\Phi)}$,

- non-deterministically guess a witnessing path in the graph,
- check that this path satisfies $\varphi_1 \mathbf{U} \varphi_2$,
- check that the cost of this path satisfies "$\sim c$".

Algorithms

If $\Phi = \varphi_1 \mathbf{U}_{\sim c} \varphi_2$, assume we have computed the a_i's for φ_1 and φ_2:

An "obvious" EXPTIME algorithm

for every $x = k/2C^{h(\Phi)}$,

- non-deterministically guess a witnessing path in the graph,
- check that this path satisfies $\varphi_1 \mathbf{U} \varphi_2$,
- check that the cost of this path satisfies "$\sim c$".

$\rightsquigarrow$ each step is in (N)PSPACE

Algorithms

If $\Phi = \varphi_1 \mathbf{U}_{\sim c} \varphi_2$, assume we have computed the a_i's for φ_1 and φ_2:

An "obvious" EXPTIME algorithm

for every $x = k/2C^{h(\Phi)}$,

- non-deterministically guess a witnessing path in the graph,
- check that this path satisfies $\varphi_1 \mathbf{U} \varphi_2$,
- check that the cost of this path satisfies "$\sim c$".

$\rightsquigarrow$ each step is in (N)PSPACE

$\rightsquigarrow$ the whole algorithm is in EXPTIME (there may be an exponential number of constants)

Algorithms

If $\Phi = \varphi_1 \mathbf{U}_{\sim c} \varphi_2$, assume we have computed the a_i's for φ_1 and φ_2:

An "obvious" EXPTIME algorithm

for every $x = k/2C^{h(\Phi)}$,

- non-deterministically guess a witnessing path in the graph,
- check that this path satisfies $\varphi_1 \mathbf{U} \varphi_2$,
- check that the cost of this path satisfies "$\sim c$".

$\rightsquigarrow$ each step is in (N)PSPACE

$\rightsquigarrow$ the whole algorithm is in EXPTIME (there may be an exponential number of constants)

$\rightsquigarrow$ however, we can avoid storing all constants and re-compute them when needed...

Algorithms

If $\Phi = \varphi_1 \mathbf{U}_{\sim c} \varphi_2$, assume we have computed the a_i's for φ_1 and φ_2:

An "obvious" EXPTIME algorithm

for every $x = k/2C^{h(\Phi)}$,

- non-deterministically guess a witnessing path in the graph,
- check that this path satisfies $\varphi_1 \mathbf{U} \varphi_2$,
- check that the cost of this path satisfies "$\sim c$".

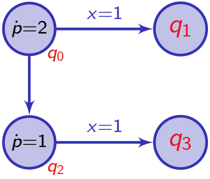
$\rightsquigarrow$ each step is in (N)PSPACE

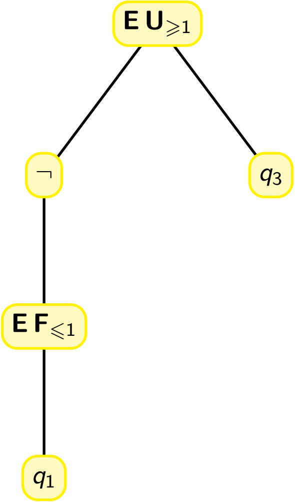
$\rightsquigarrow$ the whole algorithm is in EXPTIME (there may be an exponential number of constants)

$\rightsquigarrow$ however, we can avoid storing all constants and re-compute them when needed... and get a PSPACE algorithm.

Execution of the PSPACE algorithm

Same idea as [HKV96]

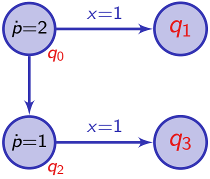




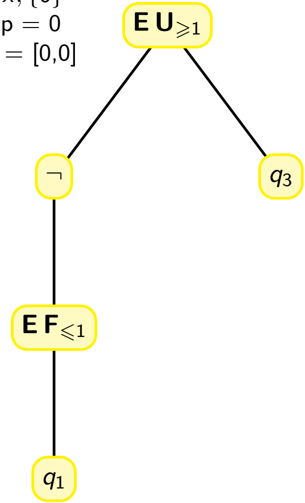
Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]



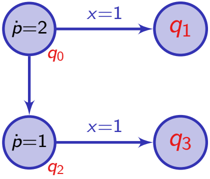
$q_0, x, \{0\}$
step $= 0$
cost $=$ [0,0]

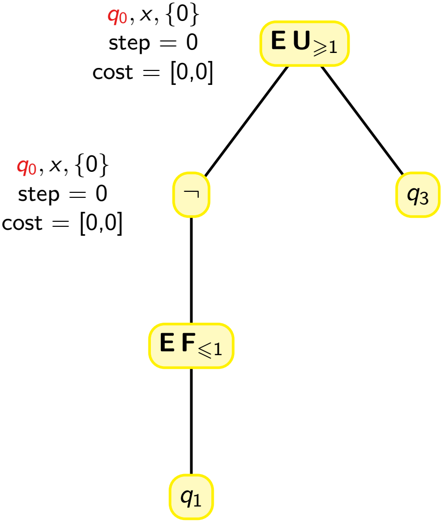


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

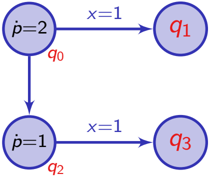


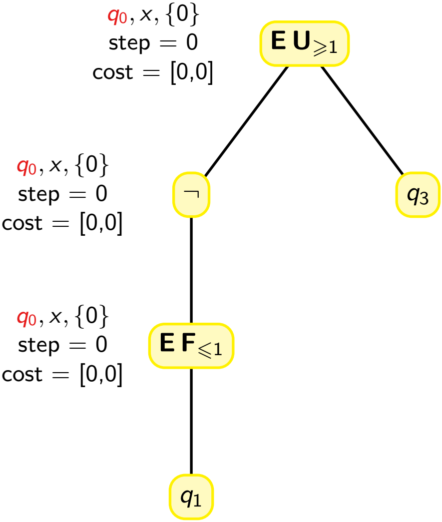


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

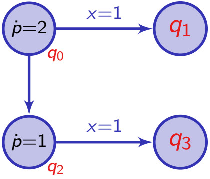




Skip

Execution of the PSPACE algorithm

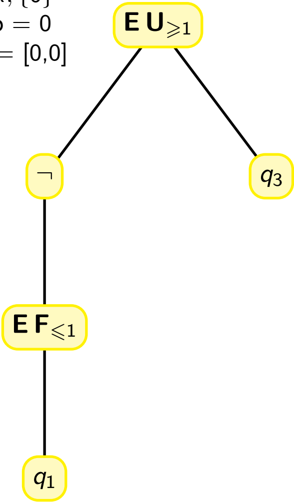
Same idea as [HKV96]



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

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

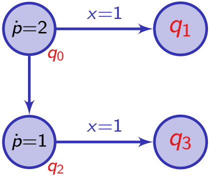
$q_0, (0, 1/4)$
step $= 1$
cost $= (0,1/2)$

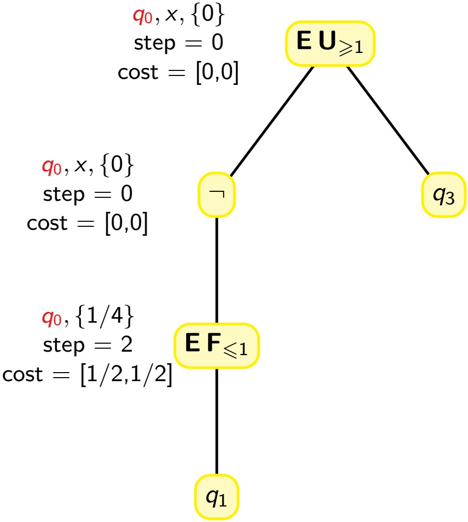


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

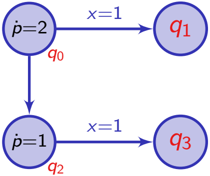


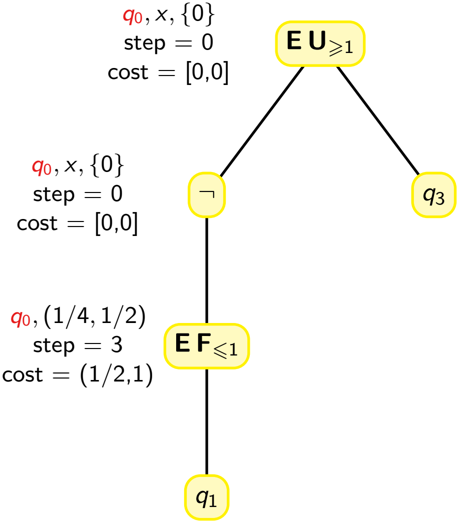


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

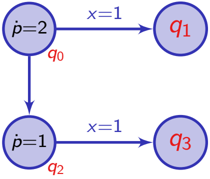


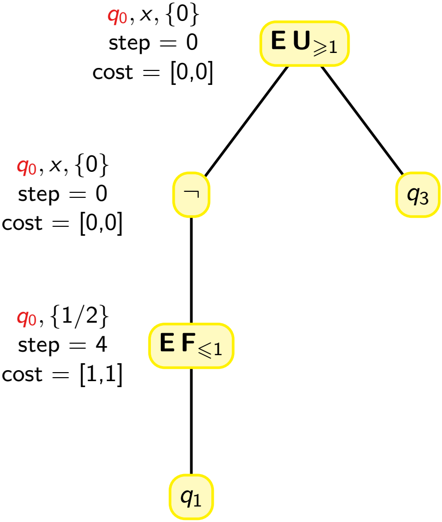


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

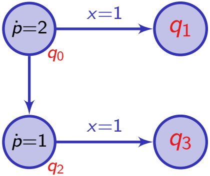


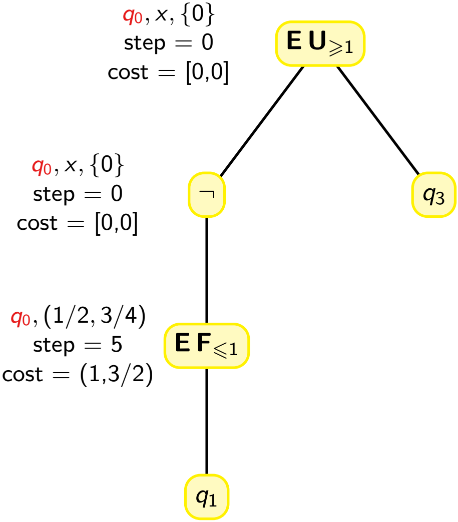


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

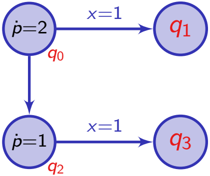


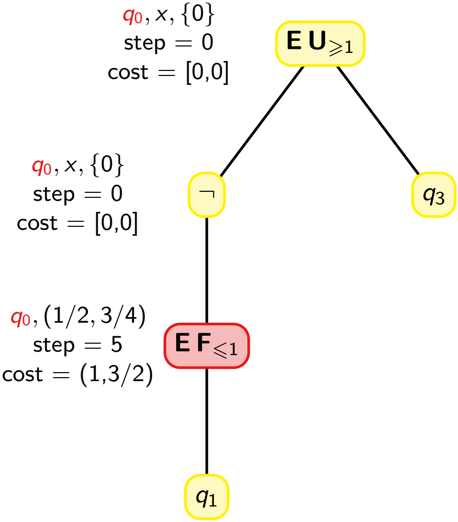


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

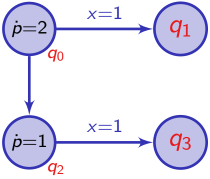


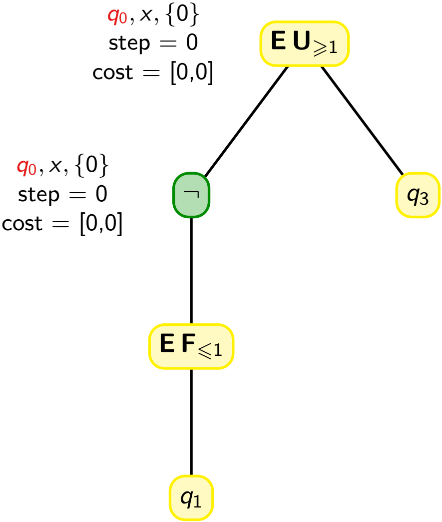


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

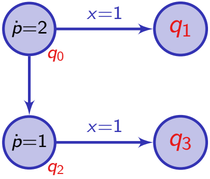




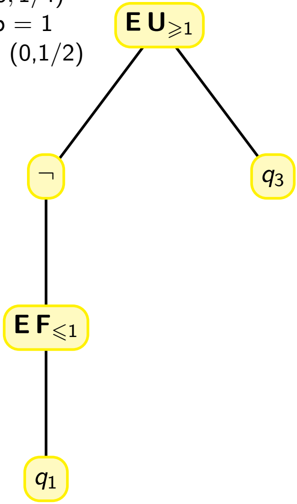
Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]



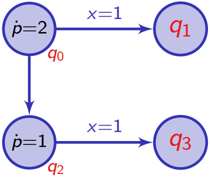
$q_0, (0, 1/4)$
step $= 1$
cost $=$ (0,1/2)

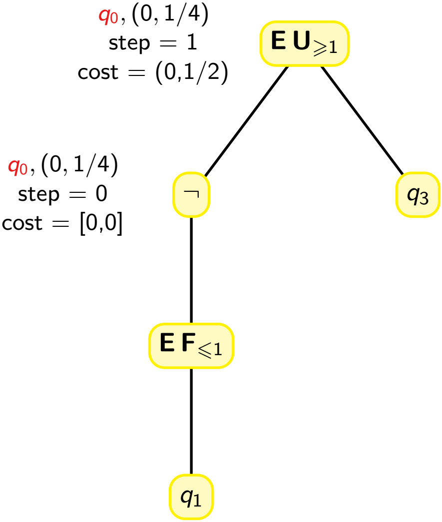


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

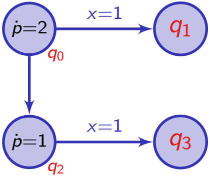


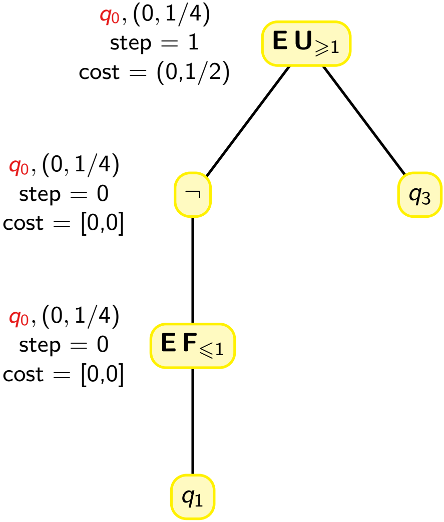


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

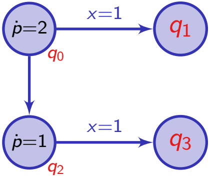




Skip

Execution of the PSPACE algorithm

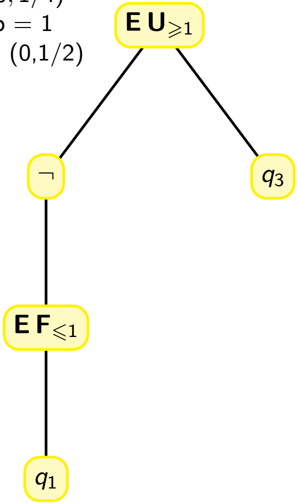
Same idea as [HKV96]



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

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

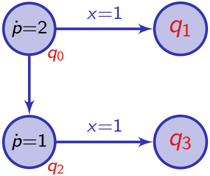
$q_0, \{1/4\}$
step $= 1$
cost $=$ (0,1/2)

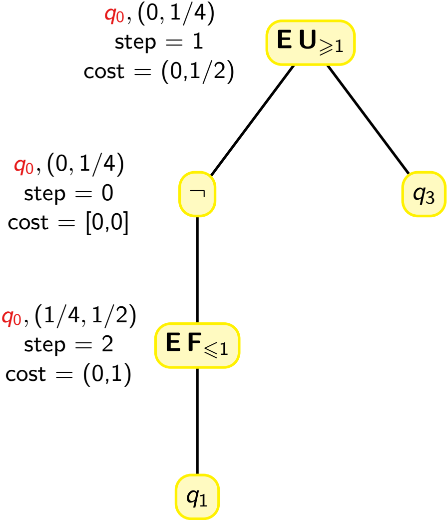


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

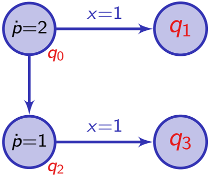


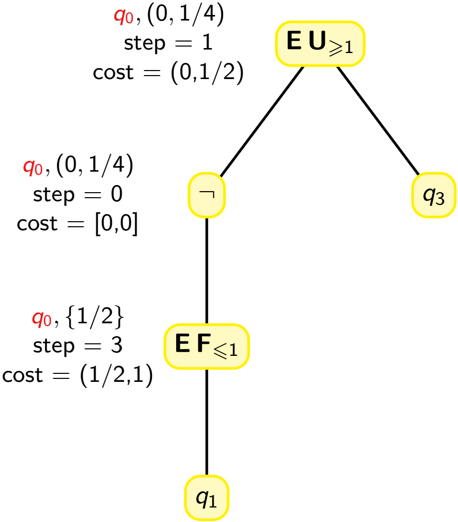


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]



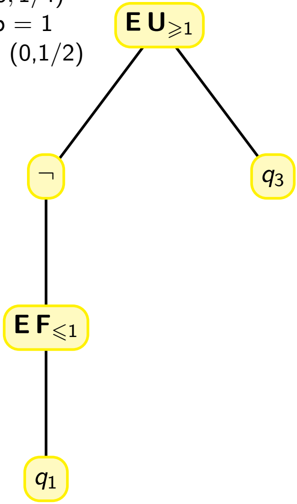


Skip

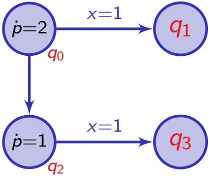
Execution of the PSPACE algorithm

Same idea as [HKV96]

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



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

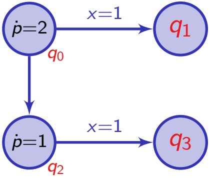


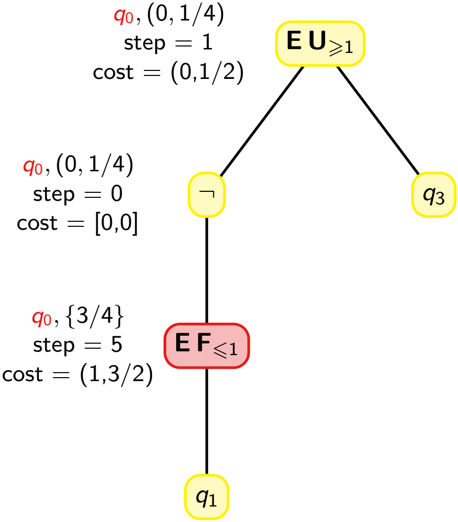
$q_0, \{3/4\}$
step $= 5$
cost $= (1,3/2)$

Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

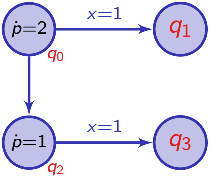


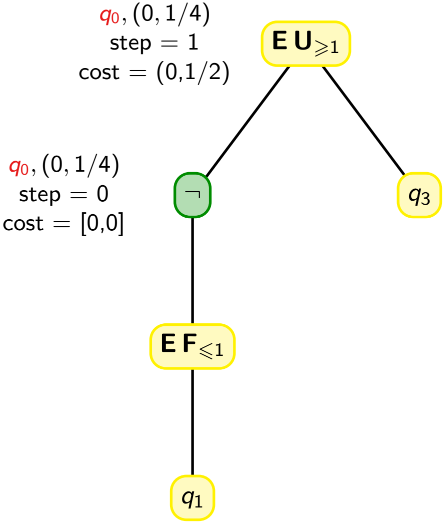


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]



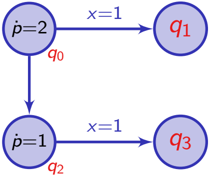


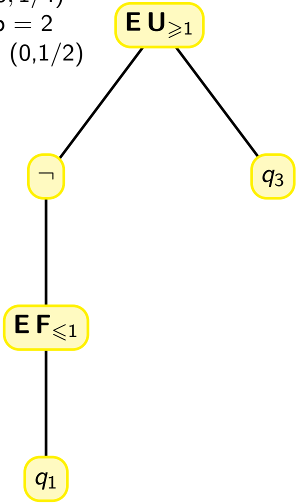
Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

$q_2, (0, 1/4)$
step $= 2$
cost $= (0,1/2)$

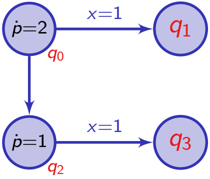




Skip

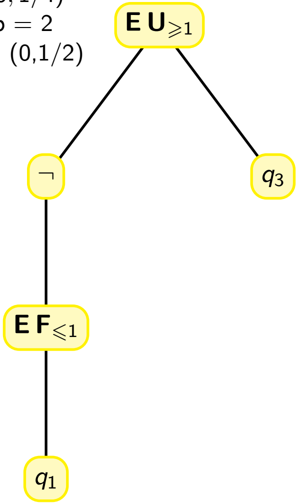
Execution of the PSPACE algorithm

Same idea as [HKV96]



$q_2, (0, 1/4)$
step $= 2$
cost $= (0,1/2)$

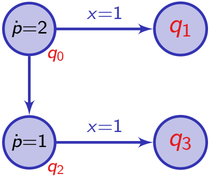
$q_2, (0, 1/4)$
step $= 0$
cost $= [0,0]$

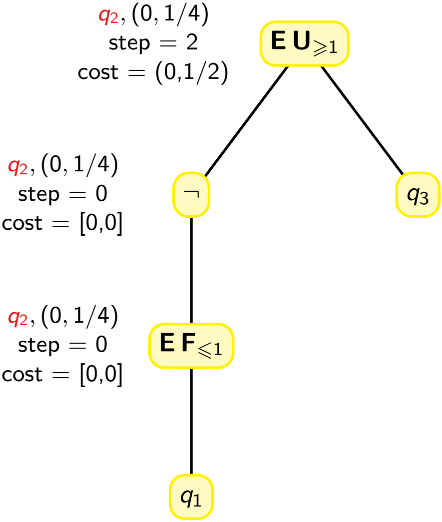


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

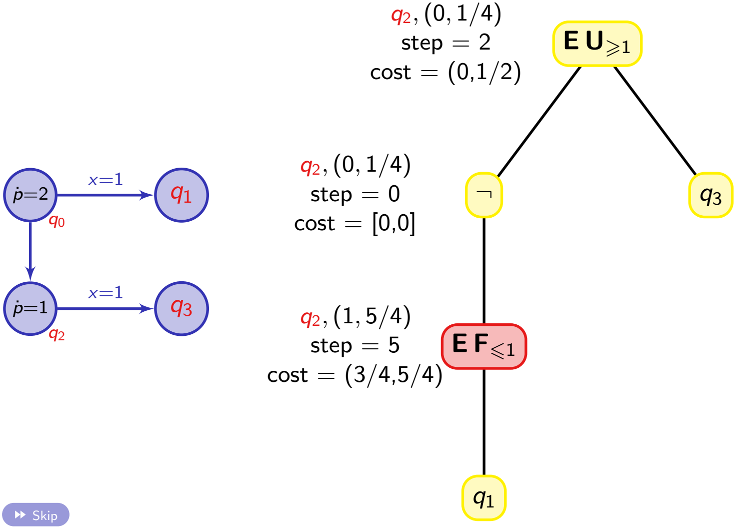




Skip

Execution of the PSPACE algorithm

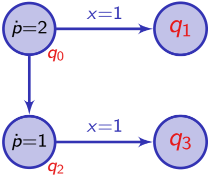
Same idea as [HKV96]

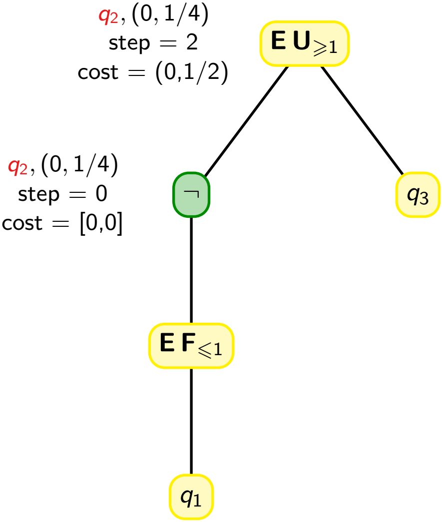


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]



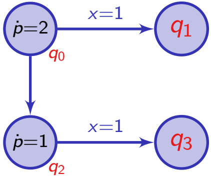


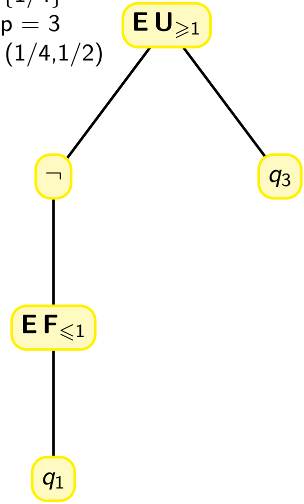
Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

$q_2, \{1/4\}$
step $= 3$
cost $= (1/4, 1/2)$



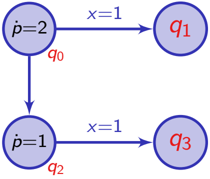


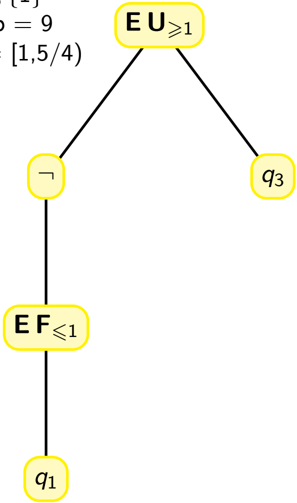
Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

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

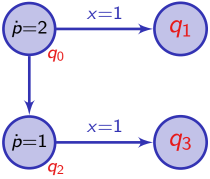


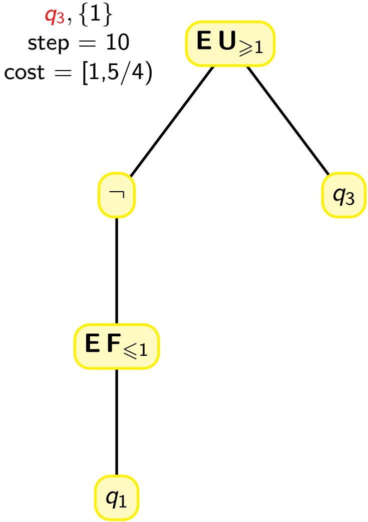


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

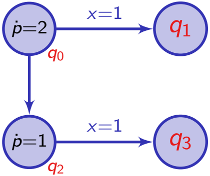


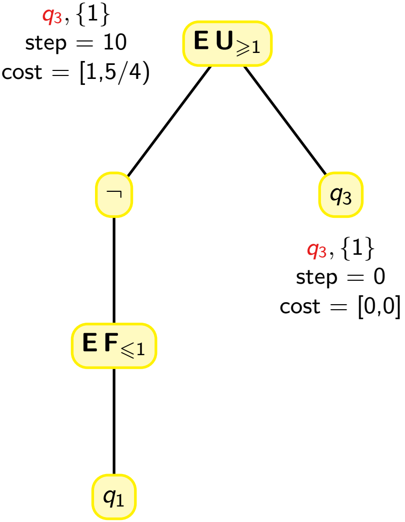


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

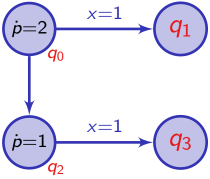


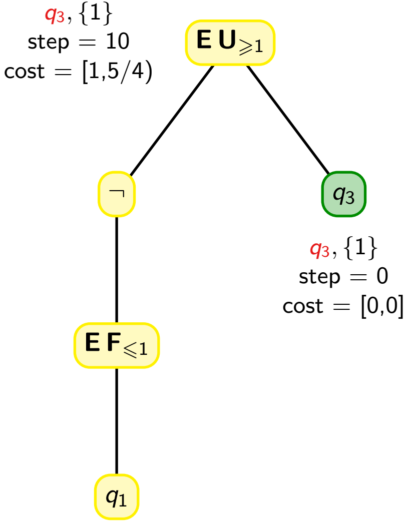


Skip

Execution of the PSPACE algorithm

Same idea as [HKV96]

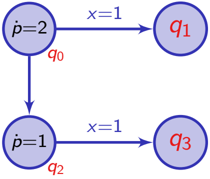


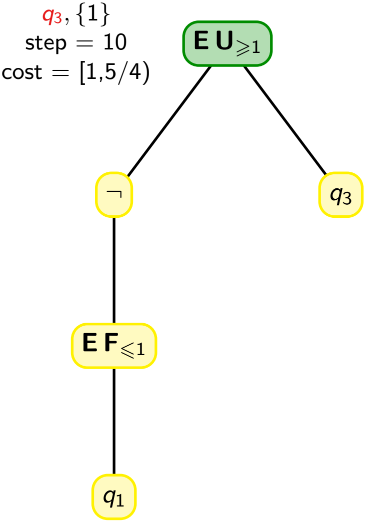


Skip

Execution of the PSPACE algorithm

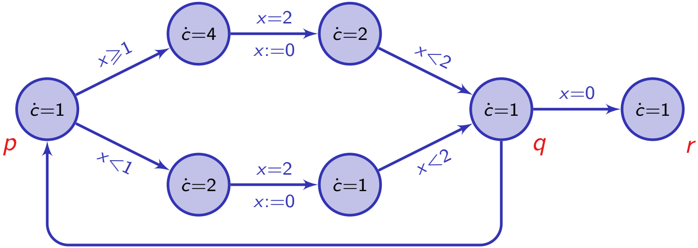
Same idea as [HKV96]



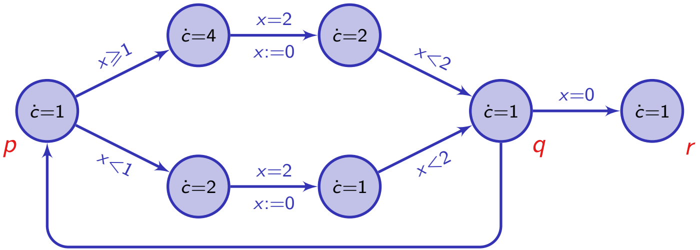


Skip

Explosion of the number of a_i's

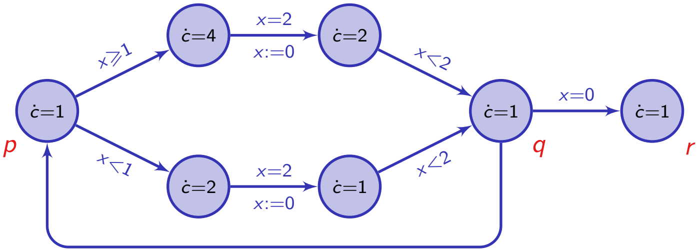


Explosion of the number of a_i's



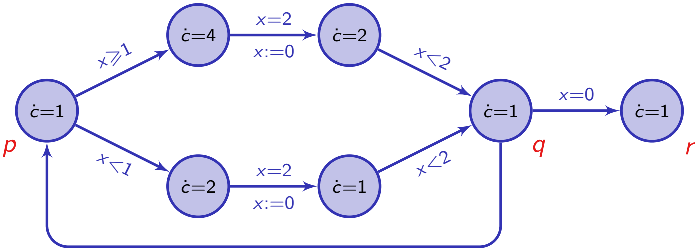
- value of x in state p: $x_0, x_1 x_2 x_3 \ldots x_n$

Explosion of the number of a_i's



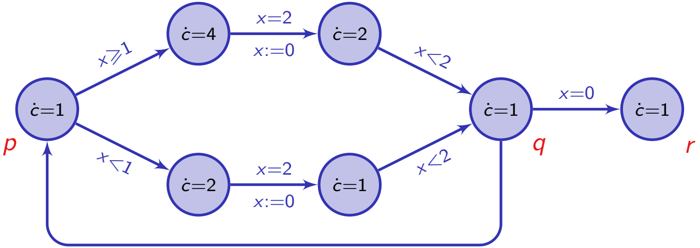
- value of x in state p: $x_0, x_1 x_2 x_3 \ldots x_n$ + cost 4 between p and q

Explosion of the number of a_i's



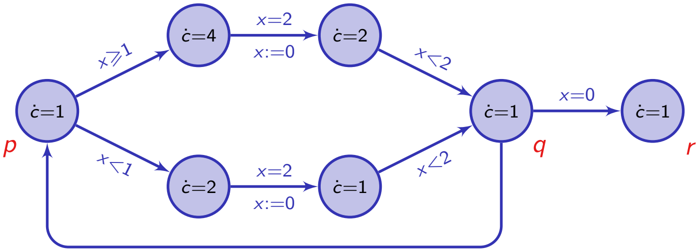
- value of x in state p: $x_0, x_1x_2x_3\dots x_n$ + cost 4 between p and q
  $\rightarrow$ value of x in state q: $x_1, x_2x_3\dots x_n$

Explosion of the number of a_i's



- value of x in state p: $x_0, x_1x_2x_3 \ldots x_n$ + cost 4 between p and q
  $\rightarrow$ value of x in state q: $x_1, x_2x_3 \ldots x_n$
- $\varphi(X) = \mathbf{E}\Big((p \vee q)\mathbf{U}_{=0}(\neg p \wedge \mathbf{E}\left(\neg q \mathbf{U}_{=4}(q \wedge X)\right))\Big)$

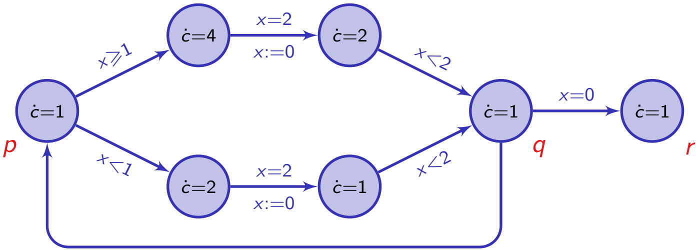
Explosion of the number of a_i's



- value of x in state p: $x_0, x_1 x_2 x_3 \ldots x_n$ + cost 4 between p and q
  $\rightarrow$ value of x in state q: $x_1, x_2 x_3 \ldots x_n$

- $\varphi(X) = \mathbf{E}\left((p \vee q)\mathbf{U}_{=0}(\neg p \wedge \mathbf{E}\,(\neg q\mathbf{U}_{=4}(q \wedge X)))\right)$
  - $p, x \models \varphi(\mathbf{E}\,\mathbf{F}_{=0} r)$ iff $x \in \{0, 1\}$

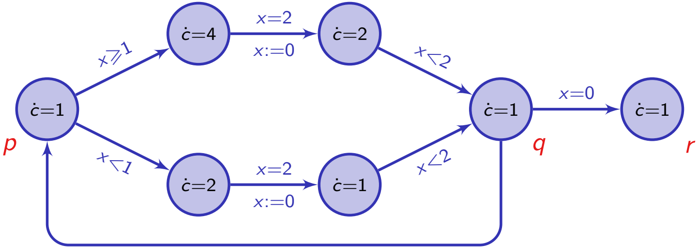
Explosion of the number of a_i's



- value of x in state p: $x_0, x_1 x_2 x_3 \ldots x_n$ + cost 4 between p and q
  $\rightarrow$ value of x in state q: $x_1, x_2 x_3 \ldots x_n$

- $\varphi(X) = \mathbf{E}\Big((p \vee q)\mathbf{U}_{=0}(\neg p \wedge \mathbf{E}\,(\neg q\mathbf{U}_{=4}(q \wedge X)))\Big)$
  - $p, x \models \varphi(\mathbf{E}\,\mathbf{F}_{=0} r)$ iff $x \in \{0, 1\}$
  - $p, x \models \varphi\Big(\varphi(\mathbf{E}\,\mathbf{F}_{=0} r)\Big)$ iff $x \in \{0, 1/2, 1, 3/2\}$

Explosion of the number of a_i's



- value of x in state p: $x_0, x_1x_2x_3\dots x_n$ + cost 4 between p and q
  $\rightarrow$ value of x in state q: $x_1, x_2x_3\dots x_n$

- $\varphi(X) = \mathbf{E}\Big((p \vee q)\mathbf{U}_{=0}(\neg p \wedge \mathbf{E}\,(\neg q\mathbf{U}_{=4}(q \wedge X)))\Big)$
  - $p, x \models \varphi(\mathbf{E}\,\mathbf{F}_{=0} r)$ iff $x \in \{0, 1\}$
  - $p, x \models \varphi\Big(\varphi(\mathbf{E}\,\mathbf{F}_{=0} r)\Big)$ iff $x \in \{0, 1/2, 1, 3/2\}$
  - $p, x \models \varphi^n(\mathbf{E}\,\mathbf{F}_{=0} r)$ iff $x \in \{k/2^{n-1} \mid 0 \leqslant k < 2^n\}$

Conclusion

- Priced (weighted) timed automata: a natural model for modelling resource consumption in timed systems

Conclusion

- Priced (weighted) timed automata: a natural model for modelling resource consumption in timed systems
- Unfortunately: costs are expensive!

Conclusion

- Priced (weighted) timed automata: a natural model for modelling resource consumption in timed systems
- Unfortunately: costs are expensive!
  - mostly undecidable for three clocks or more

Conclusion

- Priced (weighted) timed automata: a natural model for modelling resource consumption in timed systems

- Unfortunately: costs are expensive!
  - mostly undecidable for three clocks or more
  - rather involved algorithm for deciding WCTL for one clock
    - however, only a PSPACE theoretical complexity

Conclusion

- Priced (weighted) timed automata: a natural model for modelling resource consumption in timed systems
- Unfortunately: costs are expensive!
  - mostly undecidable for three clocks or more
  - rather involved algorithm for deciding WCTL for one clock
    - however, only a PSPACE theoretical complexity
  - $\text{WCTL}^\star$ and WMTL are undecidable already for one clock