Lecture 8

NP-hardness of Skolem and

undecidability for $\min, +$ weighted automata

Skolem problem for LRS

Over $(\mathbb{Q}, +, \cdot, 0, 1)$, is there n such that $u_n = 0$?

Skolem problem for LRS

Over $(\mathbb{Q}, +, \cdot, 0, 1)$, is there n such that $u_n = 0$?

Theorem (Blondel and Portier 2002; Akshay et al. 2020)

The Skolem problem is NP-hard

Skolem problem for LRS

Over $(\mathbb{Q}, +, \cdot, 0, 1)$, is there n such that $u_n = 0$?

Theorem (Blondel and Portier 2002; Akshay et al. 2020)

The Skolem problem is NP-hard

Proof.

We reduce from 3-SAT. Fix: $x_1, \ldots, x_s$

Skolem problem for LRS

Over $(\mathbb{Q}, +, \cdot, 0, 1)$, is there n such that $u_n = 0$?

Theorem (Blondel and Portier 2002; Akshay et al. 2020)

The Skolem problem is NP-hard

Proof.

We reduce from 3-SAT. Fix: $x_1, \ldots, x_s$

Let $p_1, \ldots, p_s$ be the first s prime numbers

For every $j \in \{1, \ldots, s\}$ we define

$$u_n^j = \begin{cases} 0 & \text{for } 1 \leqslant n < p_j \\ 1 & \text{for } n = p_j \\ u_{n-p_i}^i & \text{for } n > p_j \end{cases}$$

NP hardness proof (1)

Let $\varphi = C_1 \wedge \ldots \wedge C_m$

and $C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3}$

NP hardness proof (1)

Let $\varphi = C_1 \wedge \ldots \wedge C_m$

and $C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3}$

For every i_l, where $i \in \{1, 2, 3\}$ let

$$y^{i_l} = \begin{cases} 1 - u^k & \text{if } v_{i_l} = x_k \text{ for some } k \in \{1, \ldots, s\} \\ u^k & \text{if } v_{i_l} = \neg x_k \text{ for some } k \in \{1, \ldots, s\} \end{cases}$$

NP hardness proof (1)

Let $\varphi = C_1 \wedge \ldots \wedge C_m$

and $C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3}$

For every i_l, where $i \in \{1, 2, 3\}$ let

$$y^{i_l} = \begin{cases} 1 - u^k & \text{if } v_{i_l} = x_k \text{ for some } k \in \{1, \ldots, s\} \\ u^k & \text{if } v_{i_l} = \neg x_k \text{ for some } k \in \{1, \ldots, s\} \end{cases}$$

Define the sequences $y^i = y^{i_1} y^{i_2} y^{i_3}$ for all $i \in \{1, \ldots, m\}$

And $y = y^1 + \ldots + y^m$

NP hardness proof (1)

Let $\varphi = C_1 \wedge \ldots \wedge C_m$

and $C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3}$

For every i_l, where $i \in \{1, 2, 3\}$ let

$$y^{i_l} = \begin{cases} 1 - u^k & \text{if } v_{i_l} = x_k \text{ for some } k \in \{1, \ldots, s\} \\ u^k & \text{if } v_{i_l} = \neg x_k \text{ for some } k \in \{1, \ldots, s\} \end{cases}$$

Define the sequences $y^i = y^{i_1} y^{i_2} y^{i_3}$ for all $i \in \{1, \ldots, m\}$

And $y = y^1 + \ldots + y^m$

Let $f : \mathbb{N} \to \{0, 1\}^s$ defined as

$$f(n) = (a_1, \ldots, a_s)$$

where $a_j = 1 \iff p_j | n$

NP hardness proof (2)

$f(n) = (a_1, \ldots, a_s)$ is an evaluation of $x_1, \ldots, x_s$

Fact

$y_n^i = 0$ iff $f(n)$ satisfies C_i

NP hardness proof (2)

$f(n) = (a_1, \ldots, a_s)$ is an evaluation of $x_1, \ldots, x_s$

Fact

$y_n^i = 0$ iff $f(n)$ satisfies C_i

Theorem

φ is satisfiable iff there is n s.t. $y_n = 0$.

■

Weighted automata

Decision problems:

Containment: Given $\mathcal{A}$ and $\mathcal{B}$ does $\mathcal{A}(w) \leqslant \mathcal{B}(w)$ hold for all w

Equivalence: Given $\mathcal{A}$ and $\mathcal{B}$ does $\mathcal{A}(w) = \mathcal{B}(w)$ hold for all w

Boundedness: Given $\mathcal{A}$ and c does $\mathcal{A}(w) \leqslant c$ hold for all w

(given $\mathcal{A}$ and c is there a word w s.t. $\mathcal{A}(w) > c$)

Boundedness (2): Given $\mathcal{A}$ and c is there a word w s.t. $\mathcal{A}(w) = c$?

Weighted automata

Decision problems:

Containment: Given $\mathcal{A}$ and $\mathcal{B}$ does $\mathcal{A}(w) \leqslant \mathcal{B}(w)$ hold for all w

Equivalence: Given $\mathcal{A}$ and $\mathcal{B}$ does $\mathcal{A}(w) = \mathcal{B}(w)$ hold for all w

Boundedness: Given $\mathcal{A}$ and c does $\mathcal{A}(w) \leqslant c$ hold for all w

(given $\mathcal{A}$ and c is there a word w s.t. $\mathcal{A}(w) > c$)

Boundedness (2): Given $\mathcal{A}$ and c is there a word w s.t. $\mathcal{A}(w) = c$?

- These definitions make sense if $\leqslant$ makes sense in the semiring

Weighted automata

Decision problems:

Containment: Given $\mathcal{A}$ and $\mathcal{B}$ does $\mathcal{A}(w) \leqslant \mathcal{B}(w)$ hold for all w

Equivalence: Given $\mathcal{A}$ and $\mathcal{B}$ does $\mathcal{A}(w) = \mathcal{B}(w)$ hold for all w

Boundedness: Given $\mathcal{A}$ and c does $\mathcal{A}(w) \leqslant c$ hold for all w

(given $\mathcal{A}$ and c is there a word w s.t. $\mathcal{A}(w) > c$)

Boundedness (2): Given $\mathcal{A}$ and c is there a word w s.t. $\mathcal{A}(w) = c$?

- These definitions make sense if $\leqslant$ makes sense in the semiring
- Decidability of containment $\implies$ decidability of boundedness: define $\mathcal{B}(w) = c$ for all c.

Decision problems

- Over $(\min, +)$ containment and equivalence are interreducible

  decidable containment $\implies$ decidable equivalence: check $\mathcal{A} \leqslant \mathcal{B}$ and $\mathcal{B} \leqslant \mathcal{A}$

Decision problems

- Over $(\min, +)$ containment and equivalence are interreducible

  decidable containment $\implies$ decidable equivalence: check $\mathcal{A} \leqslant \mathcal{B}$ and $\mathcal{B} \leqslant \mathcal{A}$

  decidable equivalence $\implies$ decidable containment: let $\mathcal{C} = \min(\mathcal{A}, \mathcal{B})$

  $\mathcal{C} = \mathcal{A}$ is equivalent to $\mathcal{A} \leqslant \mathcal{B}$

Decision problems

- Over $(\min, +)$ containment and equivalence are interreducible

  decidable containment $\implies$ decidable equivalence: check $\mathcal{A} \leqslant \mathcal{B}$ and $\mathcal{B} \leqslant \mathcal{A}$

  decidable equivalence $\implies$ decidable containment: let $\mathcal{C} = \min(\mathcal{A}, \mathcal{B})$
  $\mathcal{C} = \mathcal{A}$ is equivalent to $\mathcal{A} \leqslant \mathcal{B}$

- But over $(\mathbb{Q}, +, \cdot, 0, 1)$ containment is undecidable
  (because boundedness is undecidable)

  While equivalence is decidable
  (possibly a proof in two weeks)

Weighted automata over min, $+$

Theorem (Krob 1994, Almagor et al. 2011)

The boundedness problem for weighted automata over $(\mathbb{Z}_{+\infty}, \text{min}, +, \infty, 0)$ is undecidable.

Weighted automata over $\min, +$

Theorem (Krob 1994, Almagor et al. 2011)

The boundedness problem for weighted automata over $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$ is undecidable.

Proof.

Reduce from a two counter Minsky machine.

Given $\mathcal{M}$ is there a halting run ending with 0 in the counters?

Weighted automata over min, $+$

Theorem (Krob 1994, Almagor et al. 2011)

The boundedness problem for weighted automata over $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$ is undecidable.

Proof.

Reduce from a two counter Minsky machine.

Given $\mathcal{M}$ is there a halting run ending with 0 in the counters?

- $\mathcal{M}$ is a sequence of lines $l_1, \ldots, l_n$ with commands

  Possible commands for $c \in \{x, y\}$

  $INC(c), \quad DEC(c)$

  $GOTO\ l_i, \quad IF\ c = 0\ GOTO\ l_i\ ELSE\ GOTO\ l_j$

  $HALT$

Weighted automata over $\min, +$

Theorem (Krob 1994, Almagor et al. 2011)

The boundedness problem for weighted automata over $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$ is undecidable.

Proof.

Reduce from a two counter Minsky machine.

Given $\mathcal{M}$ is there a halting run ending with 0 in the counters?

- $\mathcal{M}$ is a sequence of lines $l_1, \ldots, l_n$ with commands

  Possible commands for $c \in \{x, y\}$

  $INC(c), \quad DEC(c)$

  $GOTO\ l_i, \quad IF\ c = 0\ GOTO\ l_i\ ELSE\ GOTO\ l_j$

  $HALT$

  One can assume that counters can never drop below 0

Boundedness undecidability (1)

Weighted automaton $\mathcal{A}$, alphabet is

$\Sigma = \{INC(x), DEC(x), INC(y), DEC(y)\} \cup \{GOTO\ l_1, \ldots, GOTO\ l_n\}$

Boundedness undecidability (1)

Weighted automaton $\mathcal{A}$, alphabet is

$\Sigma = \{INC(x), DEC(x), INC(y), DEC(y)\} \cup \{GOTO\ l_1, \ldots, GOTO\ l_n\}$

Is there a word w with value $\mathcal{A}(w) = 1$ (or value $\mathcal{A}(w) \geqslant 1$)?

Boundedness undecidability (1)

Weighted automaton $\mathcal{A}$, alphabet is

$\Sigma = \{INC(x), DEC(x), INC(y), DEC(y)\} \cup \{GOTO\ l_1, \ldots, GOTO\ l_n\}$

Is there a word w with value $\mathcal{A}(w) = 1$ (or value $\mathcal{A}(w) \geqslant 1$)?

Rough idea: if the input word gives an accepting run in $\mathcal{M}$ then $\mathcal{A}$ outputs 1

Otherwise: there will be a run in $\mathcal{A}$ of value < 1 and $\mathcal{A}$ outputs < 1

Boundedness undecidability (1)

Weighted automaton $\mathcal{A}$, alphabet is

$\Sigma = \{INC(x), DEC(x), INC(y), DEC(y)\} \cup \{GOTO\ l_1, \ldots, GOTO\ l_n\}$

Is there a word w with value $\mathcal{A}(w) = 1$ (or value $\mathcal{A}(w) \geqslant 1$)?

Rough idea: if the input word gives an accepting run in $\mathcal{M}$ then $\mathcal{A}$ outputs 1

Otherwise: there will be a run in $\mathcal{A}$ of value < 1 and $\mathcal{A}$ outputs < 1

- There will be 5 components (to detect different errors)

Boundedness undecidability (1)

Weighted automaton $\mathcal{A}$, alphabet is

$\Sigma = \{INC(x), DEC(x), INC(y), DEC(y)\} \cup \{GOTO\ l_1, \ldots, GOTO\ l_n\}$

Is there a word w with value $\mathcal{A}(w) = 1$ (or value $\mathcal{A}(w) \geqslant 1$)?

Rough idea: if the input word gives an accepting run in $\mathcal{M}$ then $\mathcal{A}$ outputs 1

Otherwise: there will be a run in $\mathcal{A}$ of value < 1 and $\mathcal{A}$ outputs < 1

- There will be 5 components (to detect different errors)

Each has states $\{q_1, \ldots, q_n\}$ (one for each l_i)

And shared states $\{q_{freeze}, q_{halt}\}$

Almost all initial/final weights are 0

Boundedness undecidability (1)

Weighted automaton $\mathcal{A}$, alphabet is

$\Sigma = \{INC(x), DEC(x), INC(y), DEC(y)\} \cup \{GOTO\ l_1, \ldots, GOTO\ l_n\}$

Is there a word w with value $\mathcal{A}(w) = 1$ (or value $\mathcal{A}(w) \geqslant 1$)?

Rough idea: if the input word gives an accepting run in $\mathcal{M}$ then $\mathcal{A}$ outputs 1

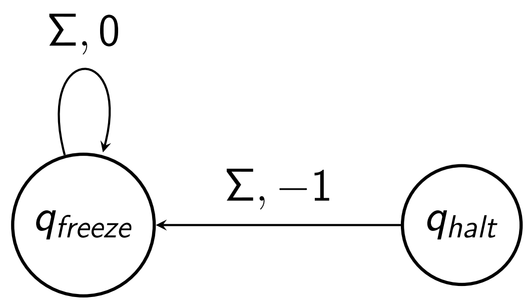
Otherwise: there will be a run in $\mathcal{A}$ of value < 1 and $\mathcal{A}$ outputs < 1

- There will be 5 components (to detect different errors)

Each has states $\{q_1, \ldots, q_n\}$ (one for each l_i)

And shared states $\{q_{freeze}, q_{halt}\}$

Almost all initial/final weights are 0



Boundedness undecidability (2)

1. Command checker

Check if (when ignoring the counters) the run is ok

Boundedness undecidability (2)

1. Command checker

Check if (when ignoring the counters) the run is ok

- Suppose you're in state q_i and read letter a

  If command i is $INC(c)$ then if $a = q_i$ add a transition $(q_i, a, 0, q_{i+1})$

Boundedness undecidability (2)

1. Command checker

Check if (when ignoring the counters) the run is ok

- Suppose you're in state q_i and read letter a

  If command i is $INC(c)$ then if $a = q_i$ add a transition $(q_i, a, 0, q_{i+1})$

  If command i is $IF\ c = 0\ GOTO\ l_j\ ELSE\ GOTO\ l_{j'}$

  then if $a = GOTO\ l_k$ for $k \in \{j, j'\}$ add two transitions (for each k) $(q_i, a, 0, q_k)$

Boundedness undecidability (2)

1. Command checker

Check if (when ignoring the counters) the run is ok

- Suppose you're in state q_i and read letter a

  If command i is $INC(c)$ then if $a = q_i$ add a transition $(q_i, a, 0, q_{i+1})$

  If command i is $IF\ c = 0\ GOTO\ l_j\ ELSE\ GOTO\ l_{j'}$

  then if $a = GOTO\ l_k$ for $k \in \{j, j'\}$ add two transitions (for each k) $(q_i, a, 0, q_k)$

  If a “doesn't match” q_i then add a transition $(q_i, a, 0, q_{freeze})$

Boundedness undecidability (2)

1. Command checker

Check if (when ignoring the counters) the run is ok

- Suppose you're in state q_i and read letter a

  If command i is $INC(c)$ then if $a = q_i$ add a transition $(q_i, a, 0, q_{i+1})$

  If command i is $IF\ c = 0\ GOTO\ l_j\ ELSE\ GOTO\ l_{j'}$

  then if $a = GOTO\ l_k$ for $k \in \{j, j'\}$ add two transitions (for each k) $(q_i, a, 0, q_k)$

  If a "doesn't match" q_i then add a transition $(q_i, a, 0, q_{freeze})$

- If i-th command is HALT and $a = HALT$ then add $(q_i, a, 1, q_{halt})$

Boundedness undecidability (2)

1. Command checker

Check if (when ignoring the counters) the run is ok

- Suppose you're in state q_i and read letter a

  If command i is $INC(c)$ then if $a = q_i$ add a transition $(q_i, a, 0, q_{i+1})$

  If command i is $IF\ c = 0\ GOTO\ l_j\ ELSE\ GOTO\ l_{j'}$

  then if $a = GOTO\ l_k$ for $k \in \{j, j'\}$ add two transitions (for each k) $(q_i, a, 0, q_k)$

  If a "doesn't match" q_i then add a transition $(q_i, a, 0, q_{freeze})$

- If i-th command is HALT and $a = HALT$ then add $(q_i, a, 1, q_{halt})$

Note that a correct run will have weight 1 and others will have weight 0

Boundedness undecidability (3)

2., 3. Positive jump checker

One for each $c \in \{x, y\}$

The transitions are almost like before

On a fresh copy of $q_1, \ldots, q_k$

Boundedness undecidability (3)

2., 3. Positive jump checker

One for each $c \in \{x, y\}$

The transitions are almost like before

On a fresh copy of $q_1, \ldots, q_k$

- Here in states if $\mathit{IF}\ c = 0\ \mathit{GOTO}\ l_j\ \mathit{ELSE}\ \mathit{GOTO}\ l_k$ if we read $\mathit{GOTO}\ l_k$ we want to know that counter c was positive

Boundedness undecidability (3)

2., 3. Positive jump checker

One for each $c \in \{x, y\}$

The transitions are almost like before

On a fresh copy of $q_1, \dots, q_k$

- Here in states if $IF\ c = 0\ GOTO\ l_j\ ELSE\ GOTO\ l_k$ if we read $GOTO\ l_k$ we want to know that counter c was positive
- When reading $INC(c)$ change weight to 1 and when reading $DEC(c)$ change to -1

Boundedness undecidability (3)

2., 3. Positive jump checker

One for each $c \in \{x, y\}$

The transitions are almost like before

On a fresh copy of $q_1, \ldots, q_k$

- Here in states if $IF\ c = 0\ GOTO\ l_j\ ELSE\ GOTO\ l_k$ if we read $GOTO\ l_k$ we want to know that counter c was positive
- When reading $INC(c)$ change weight to 1 and when reading $DEC(c)$ change to -1

Other weights (that don't go to q_{freeze}) remain 0 and $HALT$ remains 1

Boundedness undecidability (3)

2., 3. Positive jump checker

One for each $c \in \{x, y\}$

The transitions are almost like before

On a fresh copy of $q_1, \ldots, q_k$

- Here in states if $IF\ c = 0\ GOTO\ l_j\ ELSE\ GOTO\ l_k$ if we read $GOTO\ l_k$

  we want to know that counter c was positive

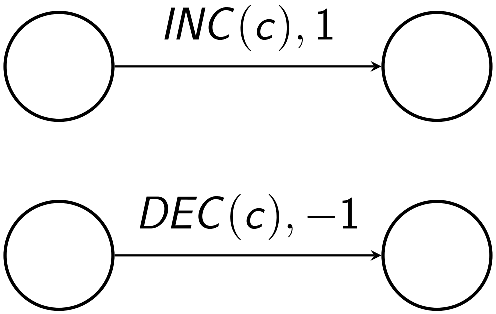
- When reading $INC(c)$ change weight to 1

  and when reading $DEC(c)$ change to -1

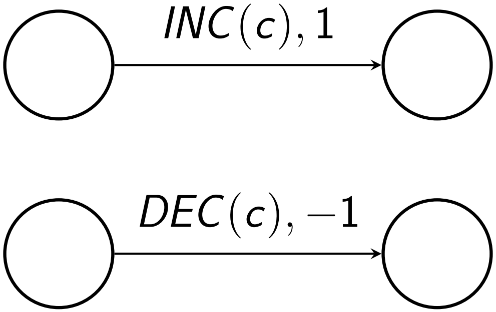
  Other weights (that don't go to q_{freeze}) remain 0 and $HALT$ remains 1

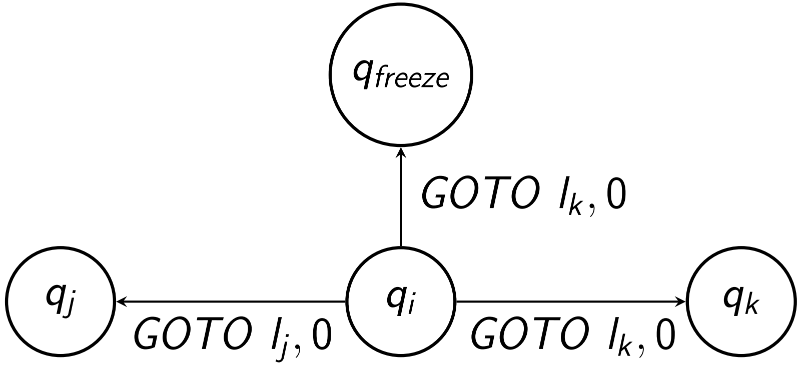
- Positivity checks with q_{freeze}

Boundedness undecidability (4)

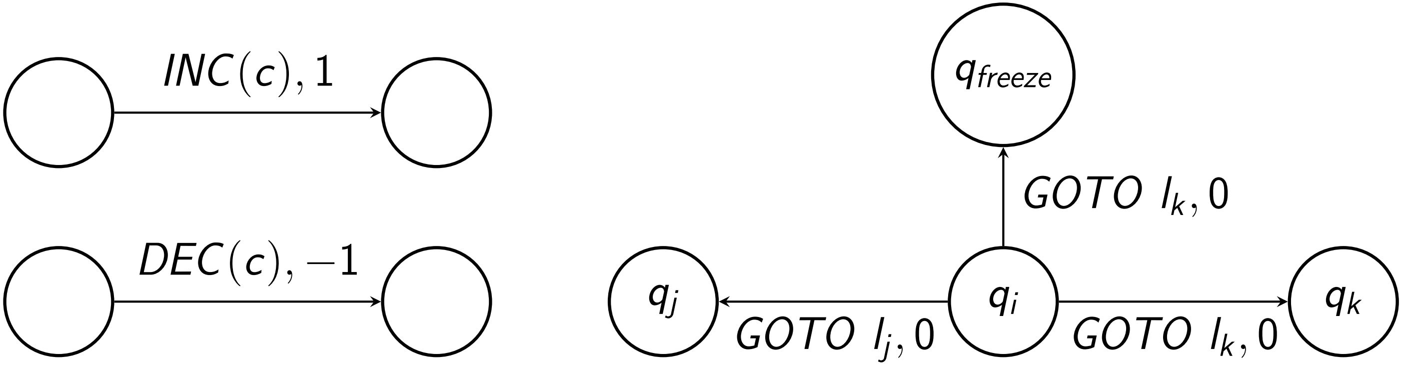


Boundedness undecidability (4)





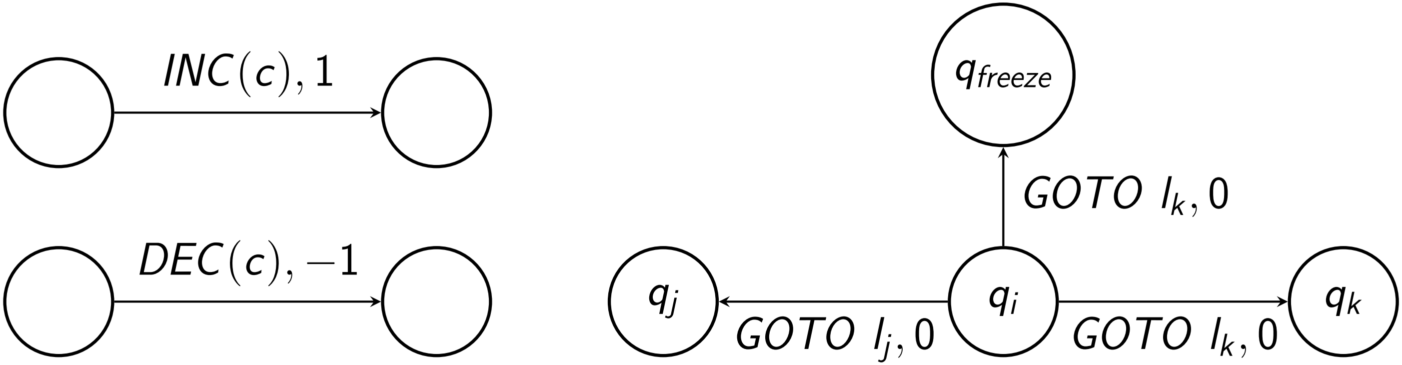
Boundedness undecidability (4)



- If the counter is positive then the run in freeze will have positive value

  If it's not positive then it will have value at most 0

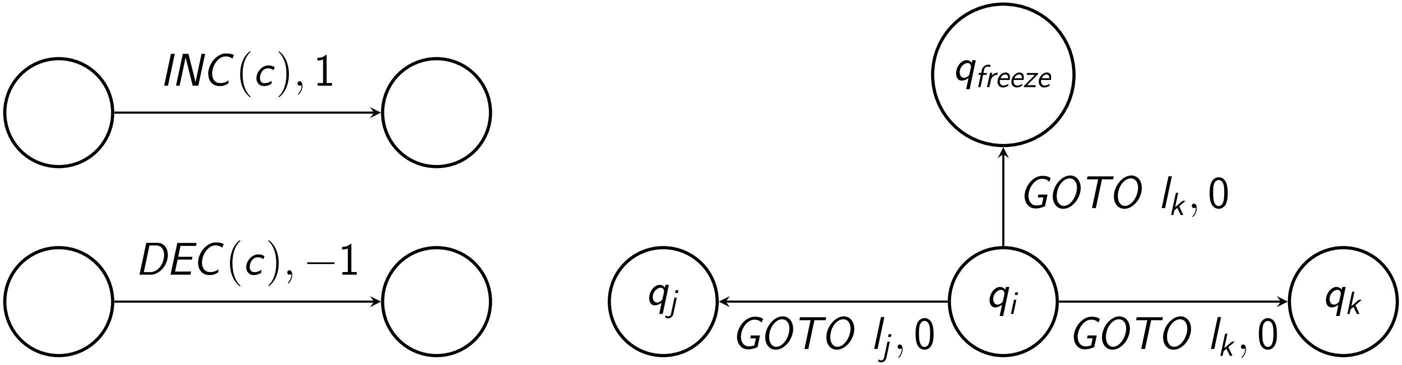
Boundedness undecidability (4)



- If the counter is positive then the run in freeze will have positive value
  If it's not positive then it will have value at most 0

- If all $GOTO\ l_k, 0$ are correct then all runs in q_{freeze} have positive value
  If any is wrong then at least one has value at most 0

Boundedness undecidability (4)



- If the counter is positive then the run in freeze will have positive value
  If it's not positive then it will have value at most 0
- If all $GOTO\ l_k, 0$ are correct then all runs in q_{freeze} have positive value
  If any is wrong then at least one has value at most 0
- Since we take min of all values this is good

Boundedness undecidability (5)

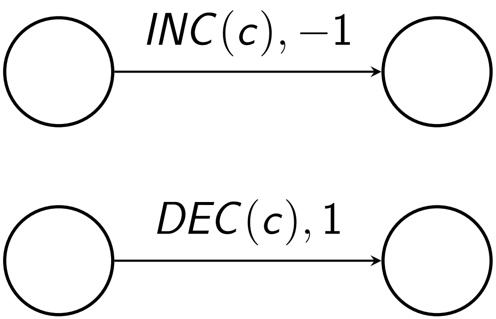
- **4., 5. Zero test checker**

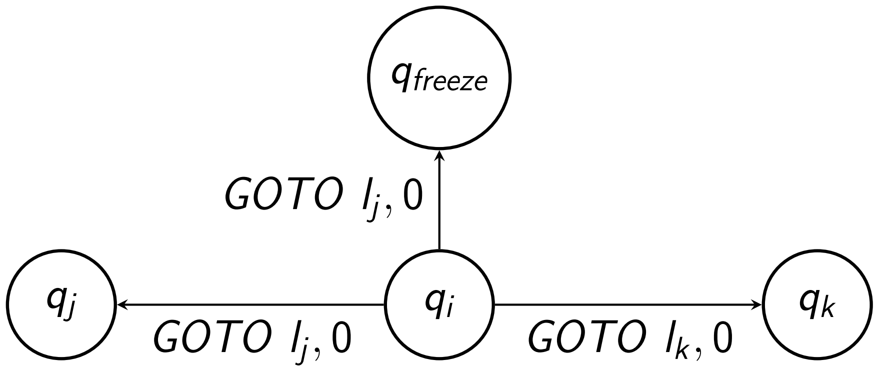
  Here in states if *IF* $c = 0$ *GOTO* l_j *ELSE GOTO* l_k if we read *GOTO* l_j we want to know that counter c was zero

Boundedness undecidability (5)

- **4., 5. Zero test checker**

  Here in states if $IF\ c = 0\ GOTO\ l_j\ ELSE\ GOTO\ l_k$ if we read $GOTO\ l_j$ we want to know that counter c was zero

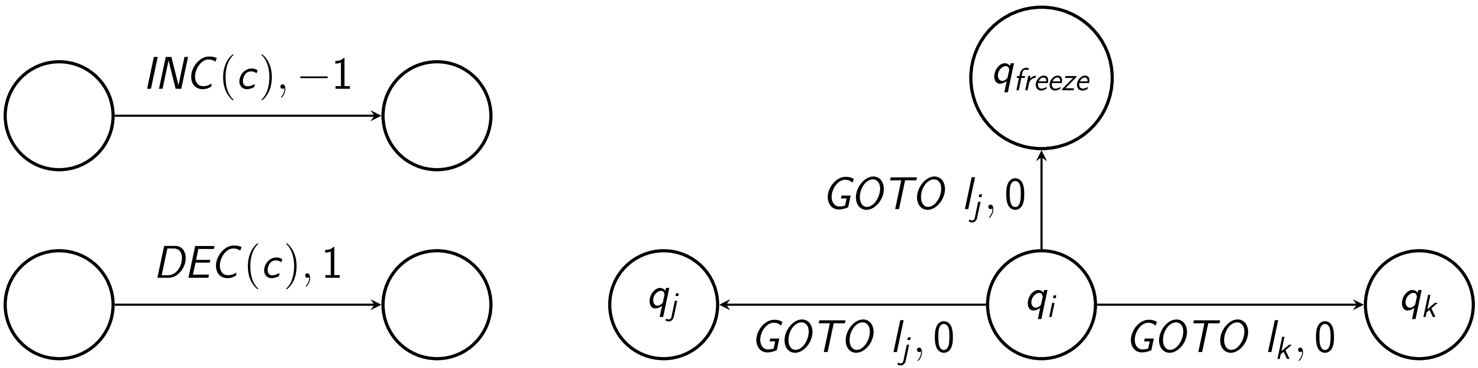




Boundedness undecidability (5)

- **4., 5. Zero test checker**

  Here in states if *IF* $c = 0$ *GOTO* l_j *ELSE GOTO* l_k if we read *GOTO* l_j we want to know that counter c was zero

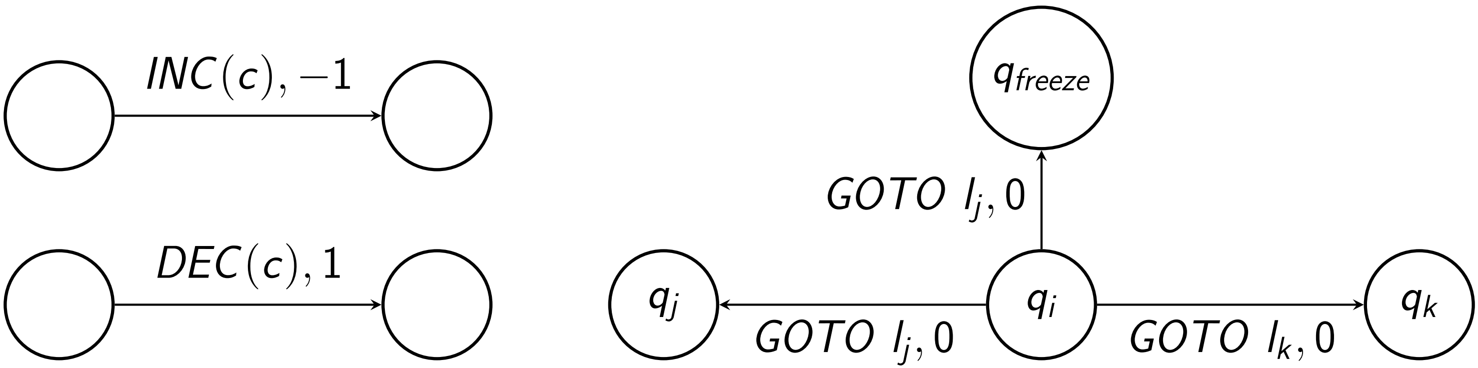


- Change the initial weight of q_1 to 1
- And change the weight when reading *HALT* to 0

Boundedness undecidability (5)

- **4., 5. Zero test checker**

  Here in states if *IF* $c = 0$ *GOTO* l_j *ELSE GOTO* l_k if we read *GOTO* l_j we want to know that counter c was zero



- Change the initial weight of q_1 to 1
- And change the weight when reading *HALT* to 0

- When any zero test is wrong there is a run in q_{freeze} with value < 1

Containment over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$

Theorem (Almagor et al. 2011)

The containment problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is undecidable.

Containment over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$

Theorem (Almagor et al. 2011)

The containment problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is undecidable.

Proof.

Recall that boundedness for $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$ is undecidable

Hence containment too.

Containment over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$

Theorem (Almagor et al. 2011)

The containment problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is undecidable.

Proof.

Recall that boundedness for $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$ is undecidable

Hence containment too.

Given an automaton $\mathcal{A}$ over $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$

let $\mathcal{A}^{+c}$ be defined that we add c to all initial, final weights and transitions

Containment over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$

Theorem (Almagor et al. 2011)

The containment problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is undecidable.

Proof.

Recall that boundedness for $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$ is undecidable

Hence containment too.

Given an automaton $\mathcal{A}$ over $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$

let $\mathcal{A}^{+c}$ be defined that we add c to all initial, final weights and transitions

Choose c such that $-c$ is the smallest negative value (or 0) in $\mathcal{A}$ and $\mathcal{B}$

Containment over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$

Theorem (Almagor et al. 2011)

The containment problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is undecidable.

Proof.

Recall that boundedness for $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$ is undecidable

Hence containment too.

Given an automaton $\mathcal{A}$ over $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$

let $\mathcal{A}^{+c}$ be defined that we add c to all initial, final weights and transitions

Choose c such that $-c$ is the smallest negative value (or 0) in $\mathcal{A}$ and $\mathcal{B}$

And observe that $\mathcal{A} \leqslant \mathcal{B}$ iff $\mathcal{A}^{+c} \leqslant \mathcal{B}^{+c}$

■

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (1)

Theorem

The boundedness problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is PSPACE-complete

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (1)

Theorem

The boundedness problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is PSPACE-complete

Proof.

Consider the problem given: $\mathcal{A}$ and $c \in \mathbb{N}$

does there exist w s.t. $\mathcal{A}(w) \geqslant c$

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (1)

Theorem

The boundedness problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is PSPACE-complete

Proof.

Consider the problem given: $\mathcal{A}$ and $c \in \mathbb{N}$

does there exist w s.t. $\mathcal{A}(w) \geqslant c$

Lemma

If there is such a w then $|w| \leqslant (c+1)^{|Q|}$

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (1)

Theorem

The boundedness problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is PSPACE-complete

Proof.

Consider the problem given: $\mathcal{A}$ and $c \in \mathbb{N}$

does there exist w s.t. $\mathcal{A}(w) \geqslant c$

Lemma

If there is such a w then $|w| \leqslant (c+1)^{|Q|}$

For a matrix M over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ we write M^c

by replacing every entry $> c$ with c

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (1)

Theorem

The boundedness problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is PSPACE-complete

Proof.

Consider the problem given: $\mathcal{A}$ and $c \in \mathbb{N}$

does there exist w s.t. $\mathcal{A}(w) \geqslant c$

Lemma

If there is such a w then $|w| \leqslant (c+1)^{|Q|}$

For a matrix M over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ we write M^c

by replacing every entry $> c$ with c

Let $w = w_1 \dots w_n$ and let $M_i = M_{w_1} \dots M_{w_i}$

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (1)

Theorem

The boundedness problem over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ is PSPACE-complete

Proof.

Consider the problem given: $\mathcal{A}$ and $c \in \mathbb{N}$

does there exist w s.t. $\mathcal{A}(w) \geqslant c$

Lemma

If there is such a w then $|w| \leqslant (c+1)^{|Q|}$

For a matrix M over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ we write M^c

by replacing every entry $> c$ with c

Let $w = w_1 \dots w_n$ and let $M_i = M_{w_1} \dots M_{w_i}$

If $M_i^c = M_j^c$ then we can shorten the witness

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (2)

- By Lemma we can guess the witness in PSPACE

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (2)

- By Lemma we can guess the witness in PSPACE

- PSPACE-hardness

  Recall that universality is PSPACE-complete for finite automata

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (2)

- By Lemma we can guess the witness in PSPACE

- PSPACE-hardness

  Recall that universality is PSPACE-complete for finite automata

  Given a finite automaton $\mathcal{A}$ just define a weighted automaton $\mathcal{B}$

  With all transition weights 0 and inital/final weights 0

Boundedness over $(\mathbb{N}_{+\infty}, \min, +, \infty, 0)$ (2)

- By Lemma we can guess the witness in PSPACE

- PSPACE-hardness

  Recall that universality is PSPACE-complete for finite automata

  Given a finite automaton $\mathcal{A}$ just define a weighted automaton $\mathcal{B}$

  With all transition weights 0 and inital/final weights 0

  Then there exists a words w s.t. $\mathcal{B}(w) > 0$ iff $\mathcal{B}(w) = \infty$

  So iff $\mathcal{A}$ does not accept w

$\blacksquare$

Concluding remarks

- For weighted automata most problems are either undecidable, or easily decidable

Concluding remarks

- For weighted automata most problems are either undecidable, or easily decidable

  Two rules of a thumb

- If something is undecidable then most often it is undecidable for

  linear ambiguous class

  Like boundedness for $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$

Concluding remarks

- For weighted automata most problems are either undecidable, or easily decidable

  Two rules of a thumb

- If something is undecidable then most often it is undecidable for
  linear ambiguous class
  Like boundedness for $(\mathbb{Z}_{+\infty}, \min, +, \infty, 0)$

- For finitely ambiguous usually problems become decidable