Polynomial-Space Completeness of Reachability for Succinct Branching VASS in Dimension One

Diego Figueira[1] Ranko Lazić [2] Jérôme Leroux [1]
Filip Mazowiecki [3] Grégoire Sutre [1]

[1]LaBRI, CNRS

[2]University of Warwick

[3]University of Oxford

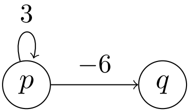
ICALP 2017
Warsaw

BVASS

Recall VASS

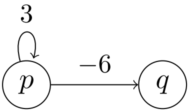
BVASS

Recall VASS



BVASS

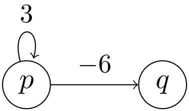
Recall VASS



$(d = 1)$

BVASS

Recall VASS



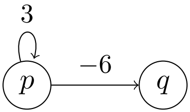
$(d = 1)$

Computations are words:

$p, 0 \xrightarrow{3} p, 3 \xrightarrow{3} p, 6 \xrightarrow{-6} q, 0$

BVASS

Recall VASS



$(d = 1)$

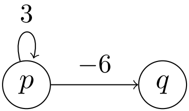
Computations are words:

$p, 0 \xrightarrow{3} p, 3 \xrightarrow{3} p, 6 \xrightarrow{-6} q, 0$

States Q, transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$, configurations $Q \times \mathbb{N}^d$

BVASS

Recall VASS



$(d = 1)$

Computations are words:

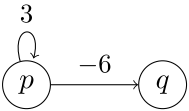
$p, 0 \xrightarrow{3} p, 3 \xrightarrow{3} p, 6 \xrightarrow{-6} q, 0$

States Q, transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$, configurations $Q \times \mathbb{N}^d$

BVASS: states Q, transitions $T \subseteq Q^2 \times \mathbb{Z}^d \times Q$, configurations $Q \times \mathbb{N}^d$

BVASS

Recall VASS



$(d = 1)$

Computations are words:

$p, 0 \xrightarrow{3} p, 3 \xrightarrow{3} p, 6 \xrightarrow{-6} q, 0$

States Q, transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$, configurations $Q \times \mathbb{N}^d$

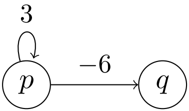
BVASS: states Q, transitions $T \subseteq Q^2 \times \mathbb{Z}^d \times Q$, configurations $Q \times \mathbb{N}^d$

Computations are binary trees:

- leaves $(q_I, \vec{0})$

BVASS

Recall VASS



$(d = 1)$

Computations are words:

$p, 0 \xrightarrow{3} p, 3 \xrightarrow{3} p, 6 \xrightarrow{-6} q, 0$

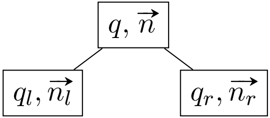
States Q, transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$, configurations $Q \times \mathbb{N}^d$

BVASS: states Q, $\underline{\text{transitions } T \subseteq Q^2 \times \mathbb{Z}^d \times Q}$, configurations $Q \times \mathbb{N}^d$

Computations are binary trees:

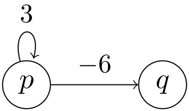
- leaves $(q_I, \vec{0})$
- inner nodes

$(q_l, q_r, \vec{z}, q) \in T$



BVASS

Recall VASS



$(d = 1)$

Computations are words:

$p, 0 \xrightarrow{3} p, 3 \xrightarrow{3} p, 6 \xrightarrow{-6} q, 0$

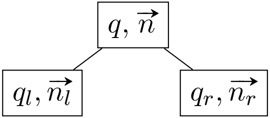
States Q, transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$, configurations $Q \times \mathbb{N}^d$

BVASS: states Q, $\underline{\text{transitions } T \subseteq Q^2 \times \mathbb{Z}^d \times Q}$, configurations $Q \times \mathbb{N}^d$

Computations are binary trees:

- leaves $(q_I, \vec{0})$
- inner nodes

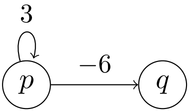
$(q_l, q_r, \vec{z}, q) \in T$



$\vec{n} = \vec{n_l} + \vec{z} + \vec{n_r}$

BVASS

Recall VASS



$(d = 1)$

unary/binary

Computations are words:

$p, 0 \xrightarrow{3} p, 3 \xrightarrow{3} p, 6 \xrightarrow{-6} q, 0$

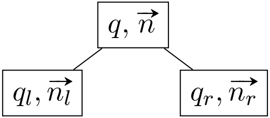
States Q, transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$, configurations $Q \times \mathbb{N}^d$

BVASS: states Q, transitions $T \subseteq Q^2 \times \mathbb{Z}^d \times Q$, configurations $Q \times \mathbb{N}^d$

Computations are binary trees:

- leaves $(q_I, \vec{0})$
- inner nodes

$(q_l, q_r, \vec{z}, q) \in T$



$\vec{n} = \vec{n_l} + \vec{z} + \vec{n_r}$

Example

$(d = 1)$

Example

$(d = 1)$

Fix n, b, where $0 \leq b \leq 2^n$

Example

$(d = 1)$

Fix n, b, where $0 \leq b \leq 2^n$

- states $Q = \{q_1 \ldots q_n\} \cup \{q_I, q_F\}$

Example

$(d = 1)$

Fix n, b, where $0 \leq b \leq 2^n$

- states $Q = \{q_1 \ldots q_n\} \cup \{q_I, q_F\}$
- three types of transitions:

Example

$(d = 1)$

Fix n, b, where $0 \leq b \leq 2^n$

- states $Q = \{q_1 \ldots q_n\} \cup \{q_I, q_F\}$
- three types of transitions:
  - $(q_I, q_I, 0, q_1), (q_I, q_I, 1, q_1)$ (initialize)

Example

$(d = 1)$

Fix n, b, where $0 \leq b \leq 2^n$

- states $Q = \{q_1 \dots q_n\} \cup \{q_I, q_F\}$
- three types of transitions:
  - $(q_I, q_I, 0, q_1), (q_I, q_I, 1, q_1)$ (initialize)
  - $(q_i, q_i, 0, q_{i+1})$ for all $i < n$ (build tree)

Example

$(d = 1)$

Fix n, b, where $0 \leq b \leq 2^n$

- states $Q = \{q_1 \ldots q_n\} \cup \{q_I, q_F\}$
- three types of transitions:
  - $(q_I, q_I, 0, q_1), (q_I, q_I, 1, q_1)$ (initialize)
  - $(q_i, q_i, 0, q_{i+1})$ for all $i < n$ (build tree)
  - $(q_n, q_n, -b, q_F)$ (check value)

Example

$(d = 1)$

Fix n, b, where $0 \leq b \leq 2^n$

- states $Q = \{q_1 \ldots q_n\} \cup \{q_I, q_F\}$
- three types of transitions:
  - $(q_I, q_I, 0, q_1), (q_I, q_I, 1, q_1)$ (initialize)
  - $(q_i, q_i, 0, q_{i+1})$ for all $i < n$ (build tree)
  - $(q_n, q_n, -b, q_F)$ (check value)

$n = 2$, $b = 3$

Example

$(d = 1)$

Fix n, b, where $0 \leq b \leq 2^n$

- states $Q = \{q_1 \ldots q_n\} \cup \{q_I, q_F\}$
- three types of transitions:
  - $(q_I, q_I, 0, q_1), (q_I, q_I, 1, q_1)$ (initialize)
  - $(q_i, q_i, 0, q_{i+1})$ for all $i < n$ (build tree)
  - $(q_n, q_n, -b, q_F)$ (check value)

$n = 2$, $b = 3$

goal: $(q_F, 0)$

Example

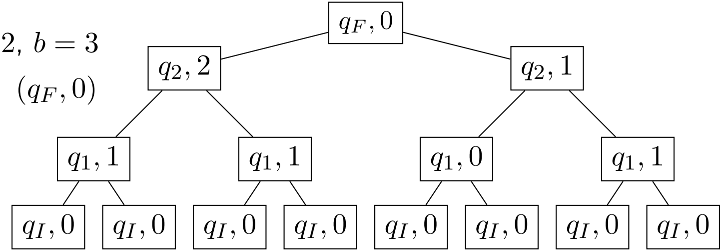
$(d = 1)$

Fix n, b, where $0 \le b \le 2^n$

- states $Q = \{q_1 \ldots q_n\} \cup \{q_I, q_F\}$
- three types of transitions:
  - $(q_I, q_I, 0, q_1), (q_I, q_I, 1, q_1)$ (initialize)
  - $(q_i, q_i, 0, q_{i+1})$ for all $i < n$ (build tree)
  - $(q_n, q_n, -b, q_F)$ (check value)

$n = 2$, $b = 3$

goal: $(q_F, 0)$



BVASS state of the art

Input: BVASS $\mathcal{B}$, configuration $(q, \vec{n})$

Problem: reachability of $(q, \vec{n})$

BVASS state of the art

Input: BVASS $\mathcal{B}$, configuration $(q, \vec{n})$

Problem: reachability of $(q, \vec{n})$

Decidability: open, even for $d = 2$

BVASS state of the art

Input: BVASS $\mathcal{B}$, configuration $(q, \vec{n})$

Problem: reachability of $(q, \vec{n})$

Decidability: open, even for $d = 2$

Connections with:

BVASS state of the art

Input: BVASS $\mathcal{B}$, configuration $(q, \vec{n})$

Problem: reachability of $(q, \vec{n})$

Decidability: open, even for $d = 2$

Connections with:

- FO^2 on data trees [Bojańczyk et al., 2009]

BVASS state of the art

Input: BVASS $\mathcal{B}$, configuration $(q, \vec{n})$

Problem: reachability of $(q, \vec{n})$

Decidability: open, even for $d = 2$

Connections with:

- FO^2 on data trees [Bojańczyk et al., 2009]
- linear logic [de Groote et al., 2004]

BVASS state of the art

Input: BVASS $\mathcal{B}$, configuration $(q, \vec{n})$

Problem: reachability of $(q, \vec{n})$

Decidability: open, even for $d = 2$

Connections with:

- FO^2 on data trees [Bojańczyk et al., 2009]
- linear logic [de Groote et al., 2004]
- recursively parallel programs [Bouajjani and Emmi, 2013]

BVASS state of the art

Input: BVASS $\mathcal{B}$, configuration $(q, \vec{n})$

Problem: reachability of $(q, \vec{n})$

Decidability: open, even for $d = 2$

Connections with:

- FO^2 on data trees [Bojańczyk et al., 2009]
- linear logic [de Groote et al., 2004]
- recursively parallel programs [Bouajjani and Emmi, 2013]

  ⋮

BVASS state of the art

Input: BVASS $\mathcal{B}$, configuration $(q, \vec{n})$

Problem: reachability of $(q, \vec{n})$

Decidability: open, even for $d = 2$

Connections with:

- FO^2 on data trees [Bojańczyk et al., 2009]
- linear logic [de Groote et al., 2004]
- recursively parallel programs [Bouajjani and Emmi, 2013]

  ⋮

Other problems:

Coverability, boundedness – 2EXPTIME-complete [Demri et al., 2013]

1-BVASS state of the art

Reachability for $d = 1$

1-BVASS state of the art

Reachability for $d = 1$

Unary encoding – $\mathrm{PT{\scriptsize IME}}$-complete [Göller et al., 2016]

1-BVASS state of the art

Reachability for $d = 1$

Unary encoding – $\text{PT}\textsc{ime}$-complete [Göller et al., 2016]

Status

	unary	binary
1-VASS	NL-complete	NP-complete
1-BVASS	P-complete	

1-BVASS state of the art

Reachability for $d = 1$

Unary encoding – PTIME-complete [Göller et al., 2016]

Status

	unary	binary
1-VASS	NL-complete	NP-complete
1-BVASS	P-complete	

NP-hard

in EXPTIME

1-BVASS state of the art

Reachability for $d = 1$

Unary encoding – PTIME-complete [Göller et al., 2016]

Status

	unary	binary
1-VASS	NL-complete	NP-complete
1-BVASS	P-complete	PSPACE-complete

NP-hard

in EXPTIME

1-BVASS state of the art

Reachability for $d = 1$

Unary encoding – PTime-complete [Göller et al., 2016]

Status

	unary	binary
1-VASS	NL-complete	NP-complete
1-BVASS	P-complete	PSpace-complete

NP-hard

in ExpTime

Easy to remember

1-BVASS state of the art

Reachability for $d = 1$

Unary encoding – PTIME-complete [Göller et al., 2016]

Status

	unary	binary
1-VASS	NL-complete	NP-complete
1-BVASS	P-complete	PSPACE-complete

NP-hard

in EXPTIME

Easy to remember

. . . but misleading: branching is not alternation

1-BVASS state of the art

Reachability for $d = 1$

Unary encoding – PTIME-complete [Göller et al., 2016]

Status

	unary	binary
1-VASS	NL-complete	NP-complete
1-BVASS	P-complete	PSPACE-complete

NP-hard (binary: 1-VASS → 1-BVASS)

in EXPTIME (1-BVASS: unary → binary)

Easy to remember

. . . but misleading: branching is not alternation

Connections with:

- Timed pushdown systems [Clemente et al., 2017]

1-BVASS upper bound

1-BVASS $\mathcal{B}$, is (q, n) reachable?

1-BVASS upper bound

$1\text{-BVASS}\ \mathcal{B}$, is (q, n) reachable?

Lemma (small witness)

If (q, n) is reachable then there is a computation with size bounded by $N = poly(n) \cdot exp(|B|)$.

1-BVASS upper bound

1-BVASS $\mathcal{B}$, is (q, n) reachable?

Lemma (small witness)

If (q, n) is reachable then there is a computation with size bounded by $N = poly(n) \cdot exp(|B|)$.

(easy for 1-VASS)

1-BVASS upper bound

1-BVASS $\mathcal{B}$, is (q, n) reachable?

Lemma (small witness)

If (q, n) is reachable then there is a computation with size bounded by $N = poly(n) \cdot exp(|B|)$.

(easy for 1-VASS)

Lemma $\implies$ reachability reduces to:

non-emptiness of tree-automaton, states $Q \times \{0 \dots N\}$

1-BVASS upper bound

1-BVASS $\mathcal{B}$, is (q, n) reachable?

Lemma (small witness)

If (q, n) is reachable then there is a computation with size bounded by $N = poly(n) \cdot exp(|B|)$.

(easy for 1-VASS)

Lemma $\implies$ reachability reduces to:

non-emptiness of tree-automaton, states $Q \times \{0 \ldots N\}$

Only exponential trees

1-BVASS upper bound

1-BVASS $\mathcal{B}$, is (q, n) reachable?

Lemma (small witness)

If (q, n) is reachable then there is a computation with size bounded by $N = poly(n) \cdot exp(|B|)$.

(easy for 1-VASS)

Lemma $\implies$ reachability reduces to:

non-emptiness of tree-automaton, states $Q \times \{0 \ldots N\}$

Only exponential trees

Guess a traversal

1-BVASS upper bound

1-BVASS $\mathcal{B}$, is (q, n) reachable?

Lemma (small witness)

If (q, n) is reachable then there is a computation with size bounded by $N = poly(n) \cdot exp(|B|)$.

(easy for 1-VASS)

Lemma $\implies$ reachability reduces to:

non-emptiness of tree-automaton, states $Q \times \{0 \ldots N\}$

Only exponential trees

Guess a traversal

(remembering only polynomially many ancestors)

1-BVASS upper bound

1-BVASS $\mathcal{B}$, is (q, n) reachable?

Lemma (small witness)

If (q, n) is reachable then there is a computation with size bounded by $N = poly(n) \cdot exp(|B|)$.

(easy for 1-VASS)

Lemma $\implies$ reachability reduces to:

non-emptiness of tree-automaton, states $Q \times \{0 \dots N\}$

Only exponential trees

Guess a traversal

(remembering only polynomially many ancestors)

So in PSPACE

1-BVASS upper bound

1-BVASS $\mathcal{B}$, is (q, n) reachable?

Core of the paper

Lemma (small witness)

If (q, n) is reachable then there is a computation with size bounded by $N = poly(n) \cdot exp(|B|)$.

(easy for 1-VASS)

Lemma $\implies$ reachability reduces to:

non-emptiness of tree-automaton, states $Q \times \{0 \ldots N\}$

Only exponential trees

Guess a traversal

(remembering only polynomially many ancestors)

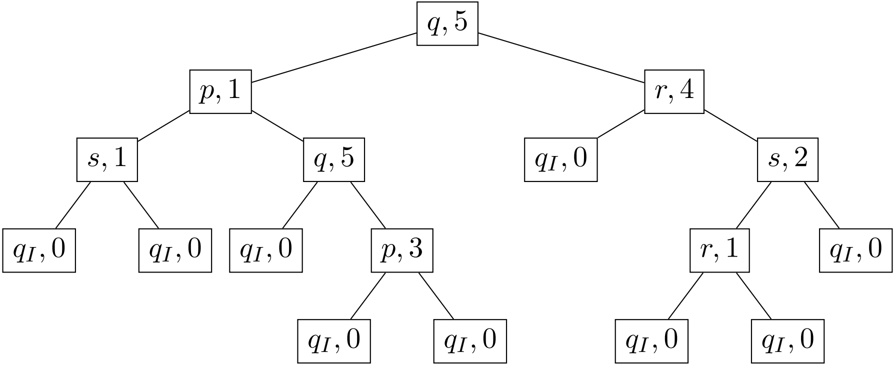
So in PSPACE

Cycles

State repetition on paths

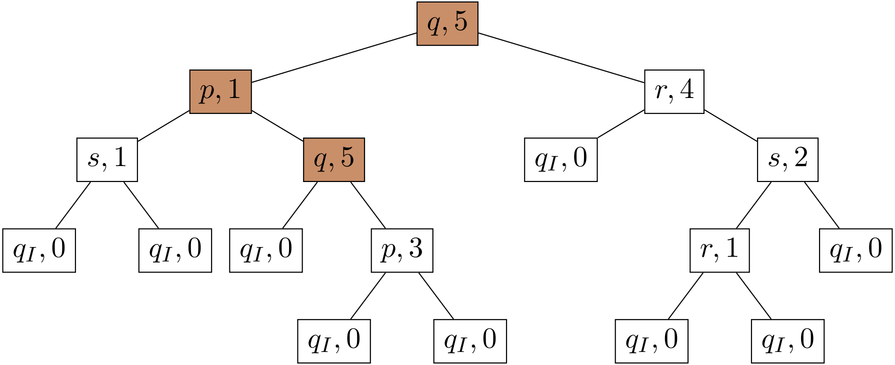
Cycles

State repetition on paths



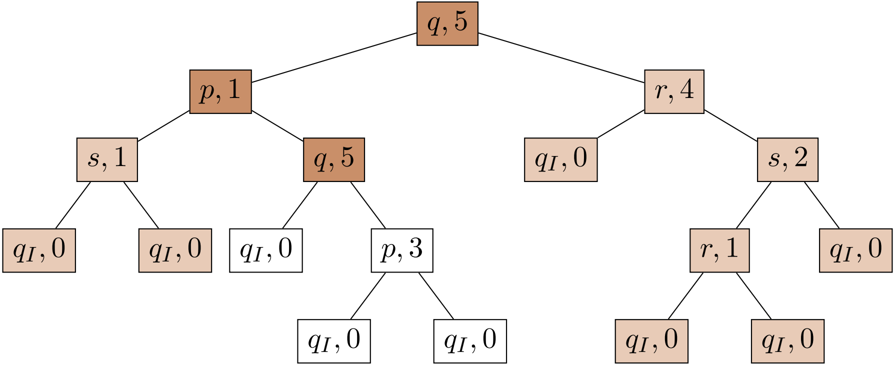
Cycles

State repetition on paths



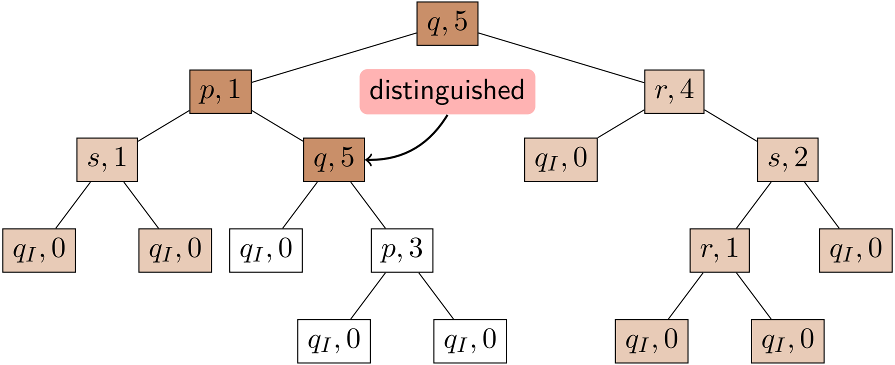
Cycles

State repetition on paths



Cycles

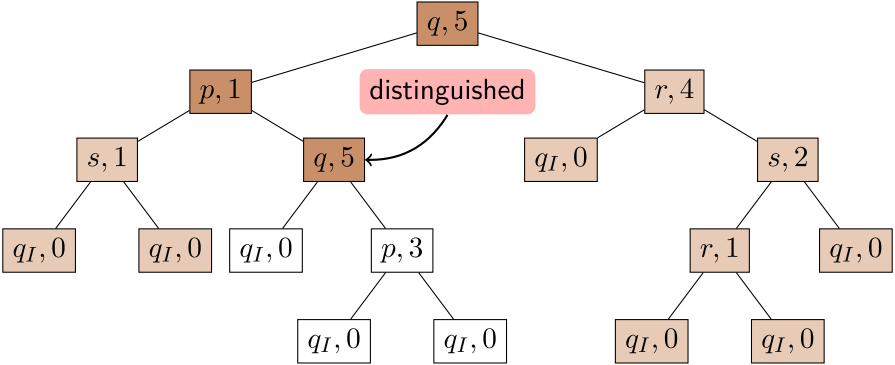
State repetition on paths



Cycle: run with a distinguished leaf

Cycles

State repetition on paths

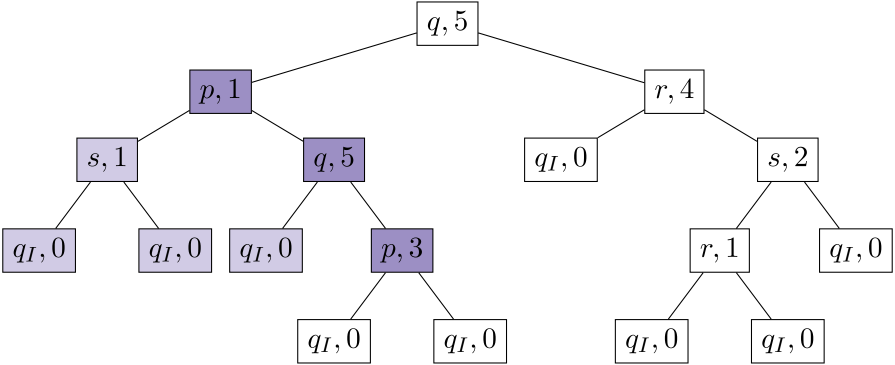


Cycle: run with a distinguished leaf

Zero cycle

Cycles

State repetition on paths

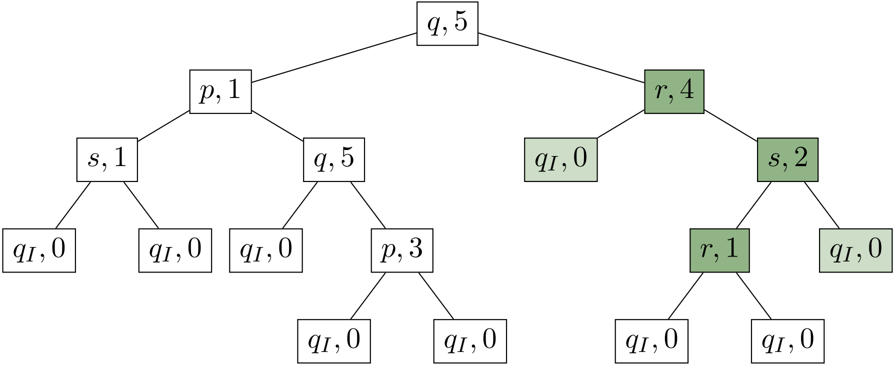


Cycle: run with a distinguished leaf

Decreasing cycle (value -2)

Cycles

State repetition on paths

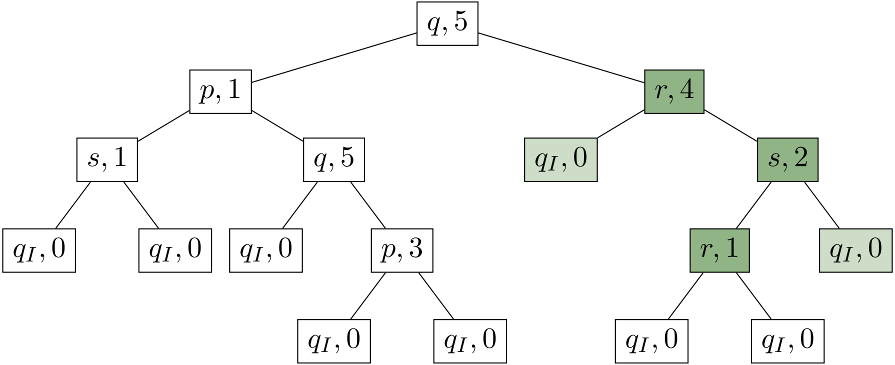


Cycle: run with a distinguished leaf

Increasing cycle (value 3)

Cycles

State repetition on paths

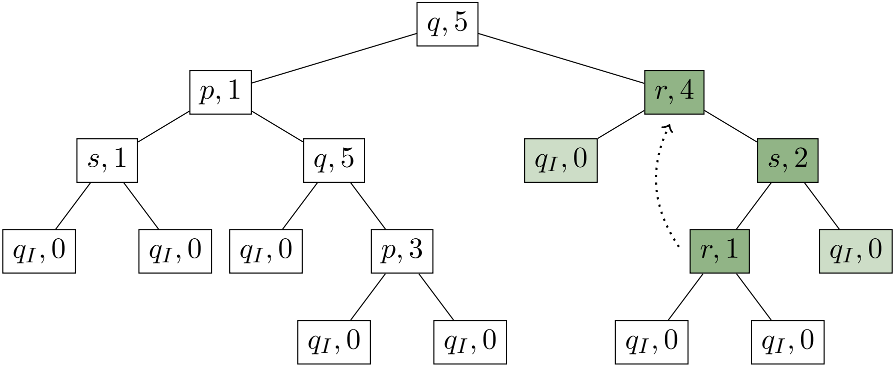


Cycle: run with a distinguished leaf

How to remove a cycle?

Cycles

State repetition on paths

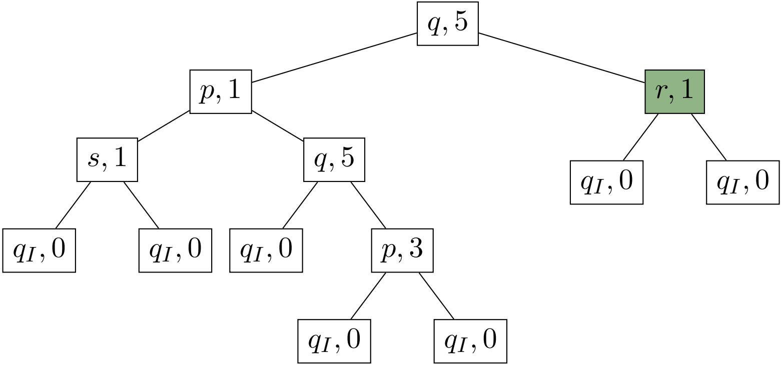


Cycle: run with a distinguished leaf

How to remove a cycle?

Cycles

State repetition on paths

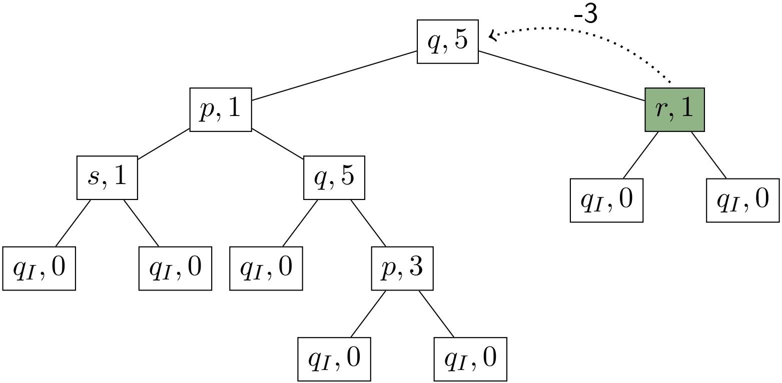


Cycle: run with a distinguished leaf

How to remove a cycle?

Cycles

State repetition on paths

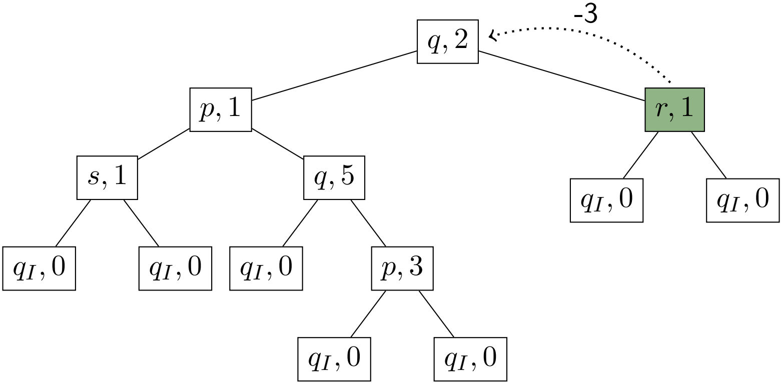


Cycle: run with a distinguished leaf

How to remove a cycle?

Cycles

State repetition on paths

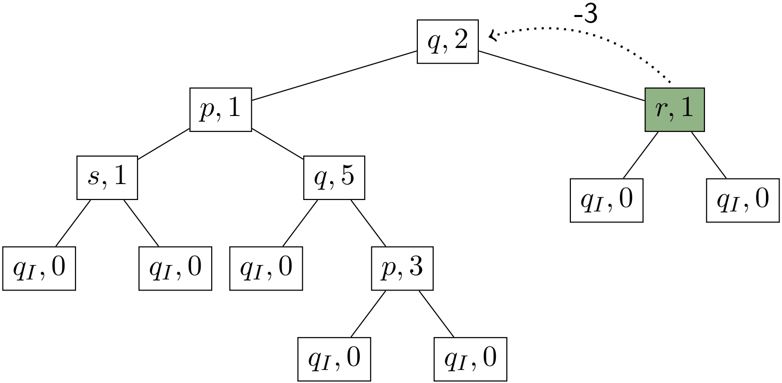


Cycle: run with a distinguished leaf

How to remove a cycle?

Cycles

State repetition on paths



Cycle: run with a distinguished leaf

How to remove a cycle? Removing zero and decreasing cycles is safe

Coverability modulo d

Is (q, n) coverable?

Coverability modulo d

Is (q, n) coverable?

Definition (coverability):

Coverability modulo d

Is (q, n) coverable?

Definition (coverability):

Is (q, x) reachable, for some $x \geq n$

Coverability modulo d

Is (q, n) coverable?

d-coverability

Definition (coverability):

Is (q, x) reachable, for some $x \geq n$

$x \geq n$ and $x \equiv n \mod d$

Coverability modulo d

Is (q, n) coverable?

d-coverability

Definition (coverability):

Is (q, x) reachable, for some $x \geq n$

$x \geq n$ and $x \equiv n \mod d$

Lemma (small witness for coverability)

If (q, n) is d-coverable then there is a small computation.

Coverability modulo d

Is (q, n) coverable?

d-coverability

Definition (coverability):

Is (q, x) reachable, for some $x \geq n$

$x \geq n$ and $x \equiv n \mod d$

Lemma (small witness for coverability)

If (q, n) is d-coverable then there is a small computation.

Proof idea for coverability:

Coverability modulo d

Is (q, n) coverable?

d-coverability

Definition (coverability):

Is (q, x) reachable, for some $x \geq n$

$x \geq n$ and $x \equiv n \mod d$

Lemma (small witness for coverability)

If (q, n) is d-coverable then there is a small computation.

Proof idea for coverability:

- Remove decreasing cycles and zero cycles

Coverability modulo d

Is (q, n) coverable?

d-coverability

Definition (coverability):

Is (q, x) reachable, for some $x \geq n$

$x \geq n$ and $x \equiv n \mod d$

Lemma (small witness for coverability)

If (q, n) is d-coverable then there is a small computation.

Proof idea for coverability:

- Remove decreasing cycles and zero cycles
- Two cases:

Coverability modulo d

Is (q, n) coverable?

d-coverability

Definition (coverability):

Is (q, x) reachable, for some $x \geq n$

$x \geq n$ and $x \equiv n \mod d$

Lemma (small witness for coverability)

If (q, n) is d-coverable then there is a small computation.

Proof idea for coverability:

- Remove decreasing cycles and zero cycles
- Two cases:

  No cycles – computation depth $\leq |Q|$

Coverability modulo d

Is (q, n) coverable?

d-coverability

Definition (coverability):

Is (q, x) reachable, for some $x \geq n$

$x \geq n$ and $x \equiv n \mod d$

Lemma (small witness for coverability)

If (q, n) is d-coverable then there is a small computation.

Proof idea for coverability:

- Remove decreasing cycles and zero cycles
- Two cases:

  No cycles – computation depth $\leq |Q|$

  Any increasing cycle – reduced to state reachability

Reachability

Is (q, n) reachable?

Reachability

Is (q, n) reachable?

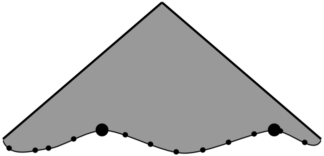
Witness representing computations

Reachability

Is (q, n) reachable?

Witness representing computations

Partial run



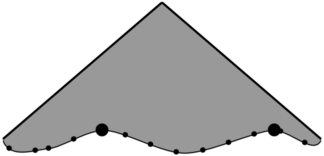
Reachability

Is (q, n) reachable?

Witness representing computations

Partial run

- proper leaves $(q_I, 0)$ $\bullet$



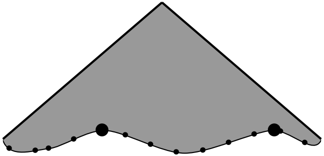
Reachability

Is (q, n) reachable?

Witness representing computations

Partial run

- proper leaves $(q_I, 0)$ •
- reachable nodes ●



Reachability

Is (q, n) reachable?

Witness representing computations

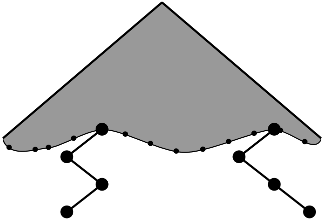
Partial run

- proper leaves $(q_I, 0)$ •

- reachable nodes ●

Decreasing simple cycles

- for every ●



Reachability

Is (q, n) reachable?

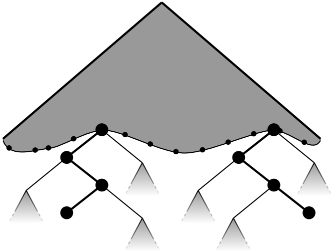
Witness representing computations

Partial run

- proper leaves $(q_I, 0)$ •
- reachable nodes ●

Decreasing simple cycles

- for every ●



Cycles are implicit

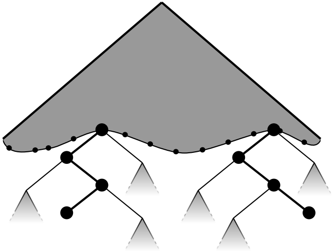
Reachability

Is (q, n) reachable?

Witness representing computations

Partial run

- proper leaves $(q_I, 0)$ •
- reachable nodes ●



Decreasing simple cycles

- for every ●

Cycles are implicit

- inductive construction

Reachability

Is (q, n) reachable?

Witness representing computations

Partial run

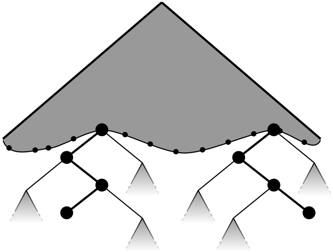
- proper leaves $(q_I, 0)$ •
- reachable nodes ●

Decreasing simple cycles

- for every ●

Cycles are implicit

- inductive construction
  (on bottom full runs)



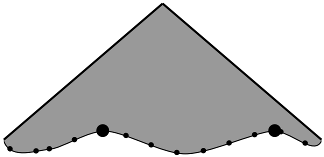
How to get the computation tree?

Given a witness

How to get the computation tree?

Given a witness

Take the partial run

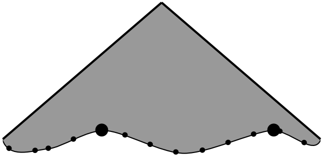


How to get the computation tree?

Given a witness

Take the partial run

- $-d_1, -d_2$ cycles values



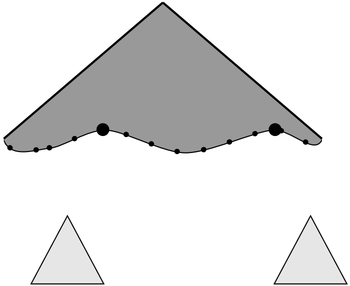
How to get the computation tree?

Given a witness

Take the partial run

- $-d_1, -d_2$ cycles values

Build d_i-coverability runs



How to get the computation tree?

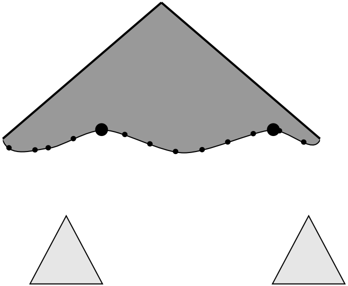
Given a witness

Take the partial run

- $-d_1, -d_2$ cycles values

Build d_i-coverability runs

(small by lemma)



How to get the computation tree?

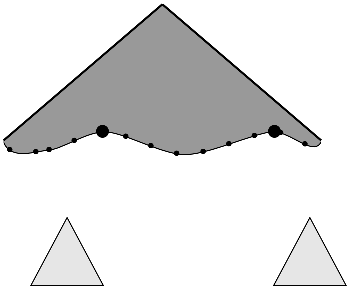
Given a witness

Take the partial run

- $-d_1, -d_2$ cycles values

Build d_i-coverability runs
(small by lemma)

Adjust values



How to get the computation tree?

Given a witness

Take the partial run

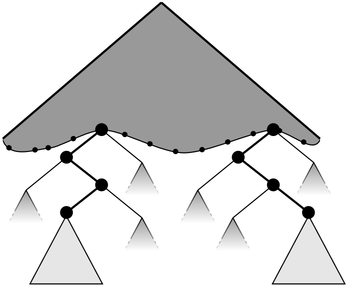
- $-d_1, -d_2$ cycles values

Build d_i-coverability runs

(small by lemma)

Adjust values

(with decreasing cycles)



How to get the computation tree?

Given a witness

Take the partial run

- $-d_1, -d_2$ cycles values

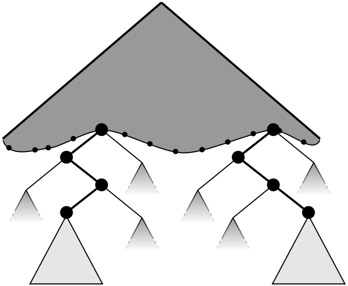
Build d_i-coverability runs

(small by lemma)

Adjust values

(with decreasing cycles)

Proceed by induction



How to get the computation tree?

Given a witness

Take the partial run

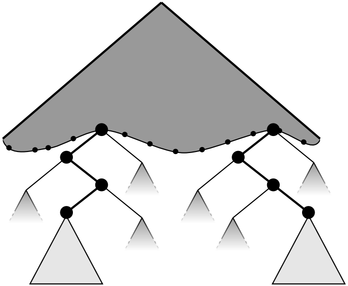
- $-d_1, -d_2$ cycles values

Build d_i-coverability runs

(small by lemma)

Adjust values

(with decreasing cycles)



Proceed by induction

Size: $\mathcal{O}(M^d)$, M – max partial run size, d – depth

Size of the run

$\mathcal{O}(M^d)$, M – max partial run size, d – depth

Size of the run

$\mathcal{O}(M^d)$, M – max partial run size, d – depth

To prove the (small witness) lemma:

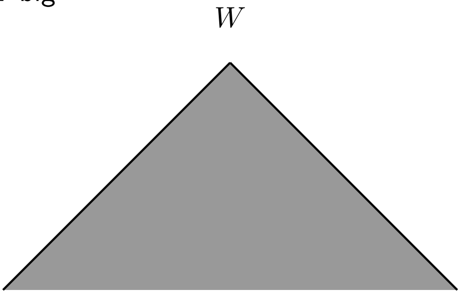
$d \leq |Q|$ and M exponential

Size of the run

$\mathcal{O}(M^d)$, M – max partial run size, d – depth

To prove the (small witness) lemma:
$d \leq |Q|$ and M exponential

Start with a full run: $d = 1$ and M big



Size of the run

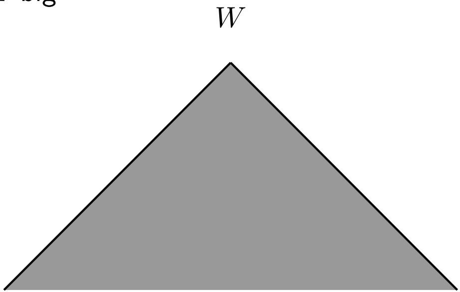
$\mathcal{O}(M^d)$, M – max partial run size, d – depth

To prove the (small witness) lemma:

$d \leq |Q|$ and M exponential

Start with a full run: $d = 1$ and M big

Two operations on a witness W:



Size of the run

$\mathcal{O}(M^d)$, M – max partial run size, d – depth

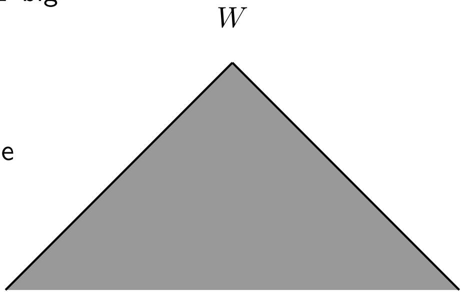
To prove the (small witness) lemma:

$d \leq |Q|$ and M exponential

Start with a full run: $d = 1$ and M big

Two operations on a witness W:

- $O_1(W)$ remove a negative cycle



Size of the run

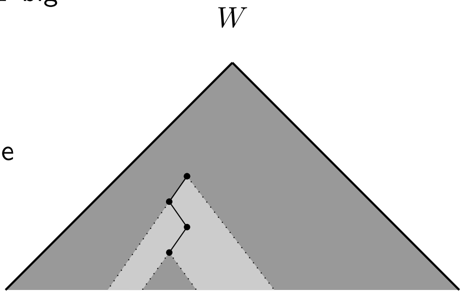
$\mathcal{O}(M^d)$, M – max partial run size, d – depth

To prove the (small witness) lemma:
$d \leq |Q|$ and M exponential

Start with a full run: $d = 1$ and M big

Two operations on a witness W:

- $O_1(W)$ remove a negative cycle



Size of the run

$\mathcal{O}(M^d)$, M – max partial run size, d – depth

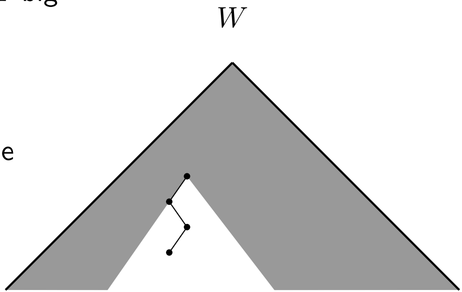
To prove the (small witness) lemma:

$d \leq |Q|$ and M exponential

Start with a full run: $d = 1$ and M big

Two operations on a witness W:

- $O_1(W)$ remove a negative cycle



Size of the run

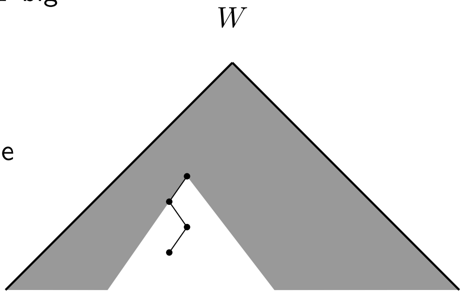
$\mathcal{O}(M^d)$, M – max partial run size, d – depth

To prove the (small witness) lemma:
$d \leq |Q|$ and M exponential

Start with a full run: $d = 1$ and M big

Two operations on a witness W:

- $O_1(W)$ remove a negative cycle
- $O_2(W)$ collapse depth



Size of the run

$\mathcal{O}(M^d)$, M – max partial run size, d – depth

To prove the (small witness) lemma:

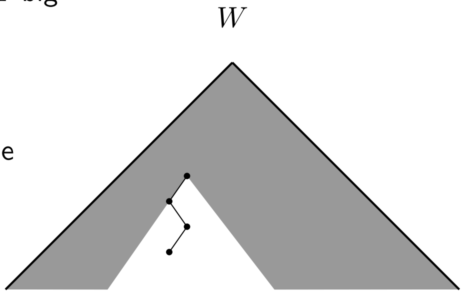
$d \leq |Q|$ and M exponential

Start with a full run: $d = 1$ and M big

Two operations on a witness W:

- $O_1(W)$ remove a negative cycle
- $O_2(W)$ collapse depth

if $d > |Q|$ then collapse



Obtaining a small witness

$O_1(W)$ decreases partial runs,

$O_2(W)$ decreases depth

Obtaining a small witness

$O_1(W)$ decreases partial runs,

$O_2(W)$ decreases depth

Define a WQO $\preceq$ on witnesses

Obtaining a small witness

$O_1(W)$ decreases partial runs,

$O_2(W)$ decreases depth

Define a WQO $\preceq$ on witnesses

$O_1(W) \prec W$ and $O_2(W) \prec W$

Obtaining a small witness

$O_1(W)$ decreases partial runs,

$O_2(W)$ decreases depth

Define a WQO $\preceq$ on witnesses

$O_1(W) \prec W$ and $O_2(W) \prec W$

Perform O_1 and O_2 if possible

Obtaining a small witness

$O_1(W)$ decreases partial runs,

$O_2(W)$ decreases depth

Define a WQO $\preceq$ on witnesses

$O_1(W) \prec W$ and $O_2(W) \prec W$

Perform O_1 and O_2 if possible

WQO guarantees termination

Obtaining a small witness

$O_1(W)$ decreases partial runs,

$O_2(W)$ decreases depth

Define a WQO $\preceq$ on witnesses

$O_1(W) \prec W$ and $O_2(W) \prec W$

Perform O_1 and O_2 if possible

WQO guarantees termination

In the end:

- $d \leq |Q|$

Obtaining a small witness

$O_1(W)$ decreases partial runs,

$O_2(W)$ decreases depth

Define a WQO $\preceq$ on witnesses

$O_1(W) \prec W$ and $O_2(W) \prec W$

Perform O_1 and O_2 if possible

WQO guarantees termination

In the end:

- $d \leq |Q|$
- max partial runs without decreasing cycles

Obtaining a small witness

$O_1(W)$ decreases partial runs,

$O_2(W)$ decreases depth

Define a WQO $\preceq$ on witnesses

$O_1(W) \prec W$ and $O_2(W) \prec W$

Perform O_1 and O_2 if possible

WQO guarantees termination

In the end:

- $d \leq |Q|$
- max partial runs without decreasing cycles (essentially small)

Lower bound

PSPACE-hardness

Lower bound

PSPACE-hardness

Take an alternating PTIME Turing Machine

Lower bound

PSpace-hardness

Take an alternating PTime Turing Machine

- time and space bound: N

Lower bound

PSPACE-hardness

Take an alternating PTIME Turing Machine

- time and space bound: N

Build a 1-BVASS

Lower bound

PSPACE-hardness

Take an alternating PTIME Turing Machine

- time and space bound: N

Build a 1-BVASS

Machine tape encoded in the counter

Lower bound

PSPACE-hardness

Take an alternating PTIME Turing Machine

- time and space bound: N

Build a 1-BVASS

Machine tape encoded in the counter

Example $N = 4$, tape 1001

$$\boxed{(\underline{1}000)(\underline{0}000)(\underline{0}000)(\underline{1}000)}$$

Lower bound

PSPACE-hardness

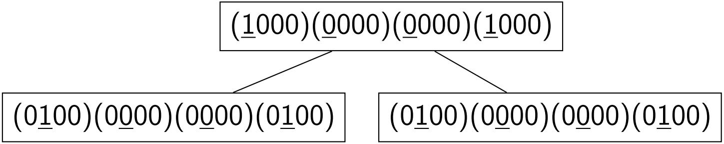
Take an alternating PTIME Turing Machine

- time and space bound: N

Build a 1-BVASS

Machine tape encoded in the counter

Example $N = 4$, tape 1001



alternation

Lower bound

PSPACE-hardness

Take an alternating PTIME Turing Machine

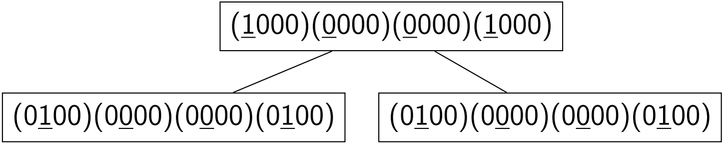
- time and space bound: N

Build a 1-BVASS

Machine tape encoded in the counter

Example $N = 4$, tape 1001

alternation



Branching with equal values

Lower bound

PSPACE-hardness

Take an alternating PTIME Turing Machine

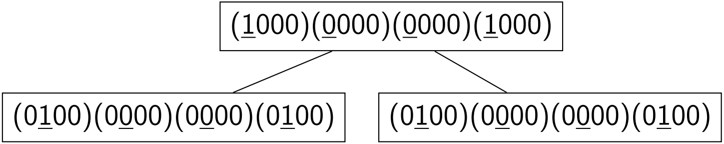
- time and space bound: N

Build a 1-BVASS

Machine tape encoded in the counter

Example $N = 4$, tape 1001

alternation



Branching with equal values

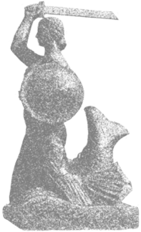
Possible for height N

Conclusions

- Reachability of BVASS?

Conclusions

- Reachability of BVASS?
- At least in dimension 2?



Conclusions

- Reachability of BVASS?
- At least in dimension 2?
- Bounded 1-VASS: PSPACE-complete [Fearnley and Jurdziński, 2013]

Conclusions

- Reachability of BVASS?
- At least in dimension 2?
- Bounded 1-VASS: PSPACE-complete [Fearnley and Jurdziński, 2013]
- Bounded 1-BVASS?

Conclusions

- Reachability of BVASS?
- At least in dimension 2?
- Bounded 1-VASS: PSPACE-complete [Fearnley and Jurdziński, 2013]
- Bounded 1-BVASS?
  in EXPTIME and PSPACE-hard