The Reachability Problem for Petri Nets is Not Elementary

Wojciech Czerwiński[1], Sławomir Lasota[1], Ranko Lazić[2], Jérôme Leroux[3] and Filip Mazowiecki[3]

[1]University of Warsaw

[2]University of Warwick

[3]LaBRI

LSV, ENS Cachan 2018

The Reachability Problem for Petri Nets is Not Elementary

Wojciech Czerwiński[1], Sławomir Lasota[1], Ranko Lazić[2], Jérôme Leroux[3] and Filip Mazowiecki[3]

[1]University of Warsaw

[2]University of Warwick

[3]LaBRI

LSV, ENS Cachan 2018

[2018/10/08 18:43:46 (19)]

Introduction

Petri Nets, VASS, programs with no zero tests

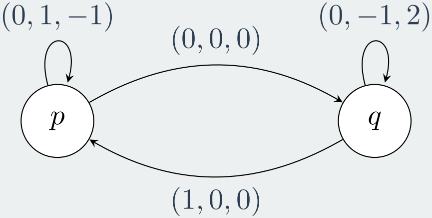
Vector addition systems with states (VASS)

(d, Q, T), where $T \subseteq Q \times \mathbb{Z}^d \times Q$

Vector addition systems with states (VASS)

(d, Q, T), where $T \subseteq Q \times \mathbb{Z}^d \times Q$

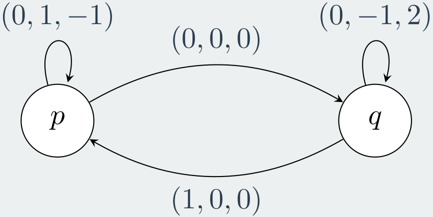
Example: $d = 3$, $Q = \{p, q\}$



Vector addition systems with states (VASS)

(d, Q, T), where $T \subseteq Q \times \mathbb{Z}^d \times Q$

Example: $d = 3$, $Q = \{p, q\}$

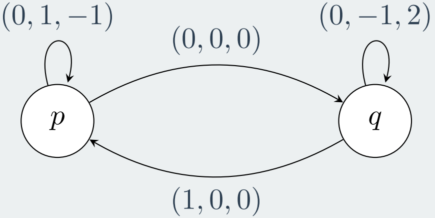


Configurations $p(\mathbf{v}) = (p, \mathbf{v}) \in Q \times \mathbb{N}^d$

Vector addition systems with states (VASS)

(d, Q, T), where $T \subseteq Q \times \mathbb{Z}^d \times Q$

Example: $d = 3$, $Q = \{p, q\}$



Configurations $p(\mathbf{v}) = (p, \mathbf{v}) \in Q \times \mathbb{N}^d$

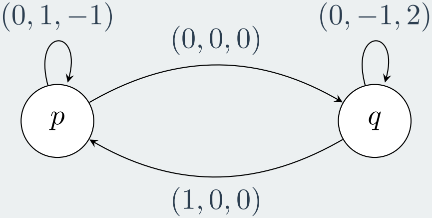
Example run:

$p(0, 0, 1) \to p(0, 1, 0) \to q(0, 1, 0) \to q(0, 0, 2) \to p(1, 0, 2)$

Vector addition systems with states (VASS)

(d, Q, T), where $T \subseteq Q \times \mathbb{Z}^d \times Q$

Example: $d = 3$, $Q = \{p, q\}$



Configurations $p(\mathbf{v}) = (p, \mathbf{v}) \in Q \times \mathbb{N}^d$

Example run:

$p(0, 0, 1) \to p(0, 1, 0) \to q(0, 1, 0) \to q(0, 0, 2) \to p(1, 0, 2)$

Notation: $p(0, 0, 1) \to^* p(1, 0, 2)$

Decision problems

Decision problems

Reachability problem:

GIVEN: VASS (d, Q, T) and configurations $p(\mathbf{u}), q(\mathbf{v})$

DECIDE: whether $p(\mathbf{u}) \rightarrow^{*} q(\mathbf{v})$?

Decision problems

Reachability problem:

GIVEN: VASS (d, Q, T) and configurations $p(\mathbf{u}), q(\mathbf{v})$

DECIDE: whether $p(\mathbf{u}) \to^* q(\mathbf{v})$?

Coverability problem:

GIVEN: VASS (d, Q, T) and configurations $p(\mathbf{u}), q(\mathbf{v})$

DECIDE: whether exists $\mathbf{v}'$ s.t. $p(\mathbf{u}) \to^* q(\mathbf{v}')$ and $\mathbf{v}' \geq \mathbf{v}$?

Decision problems

Reachability problem:

GIVEN: VASS (d, Q, T) and configurations $p(\mathbf{u}), q(\mathbf{v})$

DECIDE: whether $p(\mathbf{u}) \to^* q(\mathbf{v})$?

Coverability problem:

GIVEN: VASS (d, Q, T) and configurations $p(\mathbf{u}), q(\mathbf{v})$

DECIDE: whether exists $\mathbf{v}'$ s.t. $p(\mathbf{u}) \to^* q(\mathbf{v}')$ and $\mathbf{v}' \geq \mathbf{v}$?

- Coverability can be reduced to reachability

Decision problems

Reachability problem:

GIVEN: VASS (d, Q, T) and configurations $p(\mathbf{u}), q(\mathbf{v})$

DECIDE: whether $p(\mathbf{u}) \to^* q(\mathbf{v})$?

Coverability problem:

GIVEN: VASS (d, Q, T) and configurations $p(\mathbf{u}), q(\mathbf{v})$

DECIDE: whether exists $\mathbf{v}'$ s.t. $p(\mathbf{u}) \to^* q(\mathbf{v}')$ and $\mathbf{v}' \geq \mathbf{v}$?

- Coverability can be reduced to reachability
- We can assume $\mathbf{u} = \mathbf{v} = \mathbf{0}$

Counter programs with no zero tests

Counter programs with no zero tests

$\mathsf{x} \mathrel{+}= \mathsf{m}$	(add m to variable x)
$\mathsf{x} \mathrel{-}= \mathsf{m}$	(subtract m from variable x)
goto L **or** L'	(jump to either line L or line L')
test $\mathsf{x} = 0$	(continue if variable x is zero)
halt if $\mathsf{x}_1, \ldots, \mathsf{x}_l = 0$	(terminate if listed variables are zero).

Counter programs with no zero tests

x += m	(add m to variable x)
x −= m	(subtract m from variable x)
goto L **or** L'	(jump to either line L or line L')
~~**test** x $= 0$~~	~~(continue if variable x is zero)~~
halt if $x_1, \ldots, x_l = 0$	(terminate if listed variables are zero).

Counter programs with no zero tests

$\mathsf{x} \mathrel{+}= \mathsf{m}$	(add m to variable x)
$\mathsf{x} \mathrel{-}= \mathsf{m}$	(subtract m from variable x)
goto L **or** L'	(jump to either line L or line L')
~~**test** $\mathsf{x} = 0$~~	~~(continue if variable x is zero)~~
halt if $\mathsf{x}_1, \ldots, \mathsf{x}_l = 0$	(terminate if listed variables are zero).

All variables are initialized to 0, and are never negative

Counter programs with no zero tests

$\mathsf{x} \mathrel{+}= \mathsf{m}$	(add m to variable x)
$\mathsf{x} \mathrel{-}= \mathsf{m}$	(subtract m from variable x)
goto L **or** L'	(jump to either line L or line L')
~~**test** $\mathsf{x} = 0$~~	~~(continue if variable x is zero)~~
halt if $\mathsf{x}_1, \ldots, \mathsf{x}_l = 0$	(terminate if listed variables are zero).

All variables are initialized to 0, and are never negative

Example

1: $\mathsf{x}' \mathrel{+}= B$
2: **goto** 6 **or** 3
3: $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{x}' \mathrel{-}= 1$
4: $\mathsf{y} \mathrel{+}= 2$
5: **goto** 2
6: **halt if** $\mathsf{x}' = 0$.

Counter programs with no zero tests

$\mathsf{x} \mathrel{+}= \mathsf{m}$	(add m to variable x)
$\mathsf{x} \mathrel{-}= \mathsf{m}$	(subtract m from variable x)
goto L **or** L'	(jump to either line L or line L')
~~**test** $\mathsf{x} = 0$~~	~~(continue if variable x is zero)~~
halt if $\mathsf{x}_1, \ldots, \mathsf{x}_l = 0$	(terminate if listed variables are zero).

All variables are initialized to 0, and are never negative

Example

1: $\mathsf{x}' \mathrel{+}= B$
2: **goto** 6 **or** 3
3: $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{x}' \mathrel{-}= 1$
4: $\mathsf{y} \mathrel{+}= 2$
5: **goto** 2
6: **halt if** $\mathsf{x}' = 0$.

$\mathsf{x}' \mathrel{+}= B$
loop
    $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{x}' \mathrel{-}= 1$
    $\mathsf{y} \mathrel{+}= 2$
halt if $\mathsf{x}' = 0$.

Counter programs with no zero tests

$\mathsf{x} \mathrel{+}= \mathsf{m}$	(add m to variable x)
$\mathsf{x} \mathrel{-}= \mathsf{m}$	(subtract m from variable x)
goto L **or** L'	(jump to either line L or line L')
~~**test** $\mathsf{x} = 0$~~	~~(continue if variable x is zero)~~
halt if $\mathsf{x}_1, \ldots, \mathsf{x}_l = 0$	(terminate if listed variables are zero).

All variables are initialized to 0, and are never negative

Example

1: $\mathsf{x}' \mathrel{+}= B$
2: **goto** 6 **or** 3
3: $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{x}' \mathrel{-}= 1$
4: $\mathsf{y} \mathrel{+}= 2$
5: **goto** 2
6: **halt if** $\mathsf{x}' = 0$.

$\mathsf{x}' \mathrel{+}= B$
loop
    $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{x}' \mathrel{-}= 1$
    $\mathsf{y} \mathrel{+}= 2$
halt if $\mathsf{x}' = 0$.

A complete run ends with $\mathsf{x} = B$, $\mathsf{y} = 2B$

Programs with no zero test = VASS

Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

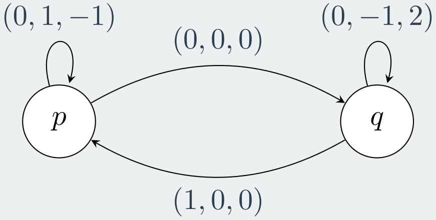
DECIDE: Does it have a complete run (executing **halt**)?

Programs with no zero test = VASS

Reachability problem (for programs):

Given: A counter program with no zero tests.

Decide: Does it have a complete run (executing **halt**)?

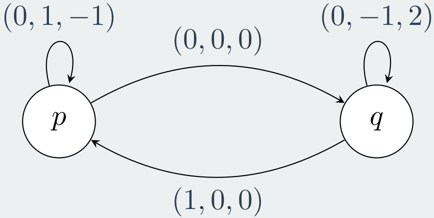


Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

DECIDE: Does it have a complete run (executing **halt**)?



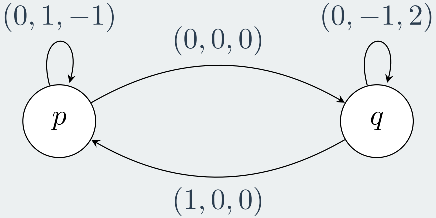
$p(0,0,1) \rightarrow^* p(1,0,2)$?

Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

DECIDE: Does it have a complete run (executing **halt**)?



$p(0,0,1) \rightarrow^* p(1,0,2)$?

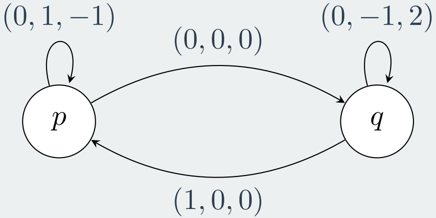
```
z += 1
loop
    loop
        y += 1    z −= 1
    loop
        y −= 1    z += 2
    x += 1
x −= 1    z −= 2
halt if x, y, z = 0.
```

Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

DECIDE: Does it have a complete run (executing **halt**)?



$p(0,0,1) \to^* p(1,0,2)$?

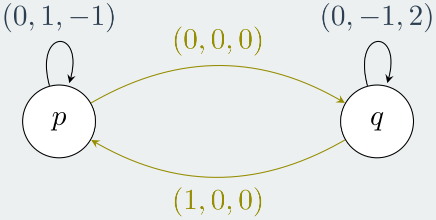
```
z += 1
loop
    loop
        y += 1    z -= 1
    loop
        y -= 1    z += 2
    x += 1
x -= 1    z -= 2
halt if x, y, z = 0.
```

Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

DECIDE: Does it have a complete run (executing **halt**)?



$p(0,0,1) \to^* p(1,0,2)$?

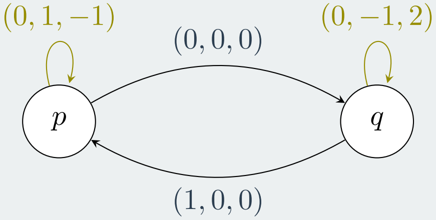
```
z += 1
loop
    loop
        y += 1    z -= 1
    loop
        y -= 1    z += 2
    x += 1
x -= 1    z -= 2
halt if x, y, z = 0.
```

Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

DECIDE: Does it have a complete run (executing **halt**)?



$p(0,0,1) \to^* p(1,0,2)$?

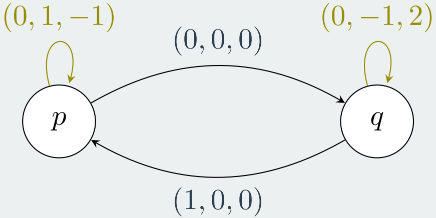
z += 1
loop
  loop
    y += 1 z −= 1
  loop
    y −= 1 z += 2
  x += 1
x −= 1 z −= 2
halt if $x, y, z = 0$.

Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

DECIDE: Does it have a complete run (executing **halt**)?



$p(0, 0, 1) \to^* p(1, 0, 2)$?

Coverability if **halt** is empty

```
z += 1
loop
    loop
        y += 1    z -= 1
    loop
        y -= 1    z += 2
    x += 1
x -= 1    z -= 2
halt if x, y, z = 0.
```

Reachability state of art

Reachability state of art

1976 — EXPSPACE-hard (Lipton)

Reachability state of art

1976 — EXPSPACE-hard (Lipton)

1981 — Decidable (Mayr)

Reachability state of art

1976 — EXPSPACE-hard (Lipton)

1981 — Decidable (Mayr)

1982 — Decidable (Kosaraju)

Reachability state of art

1976	EXPSPACE-hard (Lipton)
1981	Decidable (Mayr)
1982	Decidable (Kosaraju)
1992	Decidable (Lambert)

Reachability state of art

- 1976 — EXPSPACE-hard (Lipton)
- 1981 — Decidable (Mayr)
- 1982 — Decidable (Kosaraju)
- 1992 — Decidable (Lambert)
- 2009 — Decidable (Leroux)
- 2011 — Decidable (Leroux)
- 2012 — Decidable (Leroux)

Reachability state of art

1976 — EXPSPACE-hard (Lipton)

1981 — Decidable (Mayr)

1982 — Decidable (Kosaraju)

1992 — Decidable (Lambert)

2009 — Decidable (Leroux)

2011 — Decidable (Leroux)

2012 — Decidable (Leroux)

2015 — In $\mathbf{F}_{\omega^3}$ (Leroux and Schmitz)

Reachability state of art

1976	EXPSPACE-hard (Lipton)
1981	Decidable (Mayr)
1982	Decidable (Kosaraju)
1992	Decidable (Lambert)
2009	Decidable (Leroux)
2011	Decidable (Leroux)
2012	Decidable (Leroux)
2015	In $\mathbf{F}_{\omega^3}$ (Leroux and Schmitz)
2019?	Nonelementary (this talk)

Reachability state of art

1976	EXPSPACE-hard (Lipton)
1981	Decidable (Mayr)
1982	Decidable (Kosaraju)
1992	Decidable (Lambert)
2009	Decidable (Leroux)
2011	Decidable (Leroux)
2012	Decidable (Leroux)
2015	In $\mathbf{F}_{\omega^3}$ (Leroux and Schmitz)
2019?	Nonelementary (this talk)
2019?	In $\mathbf{F}_{\omega}$ (Leroux and Schmitz)

Reachability state of art

1976 — EXPSPACE-hard (Lipton)

Coverability
Rackoff 1978
EXPSPACE-complete

1981 — Decidable (Mayr)

1982 — Decidable (Kosaraju)

1992 — Decidable (Lambert)

2009 — Decidable (Leroux)

2011 — Decidable (Leroux)

2012 — Decidable (Leroux)

2015 — In $\mathbf{F}_{\omega^3}$ (Leroux and Schmitz)

2019? — Nonelementary (this talk)

2019? — In $\mathbf{F}_{\omega}$ (Leroux and Schmitz)

Reachability state of art

Coverability
Rackoff 1978
EXPSPACE-complete

1976	EXPSPACE-hard (Lipton)
1981	Decidable (Mayr)
1982	Decidable (Kosaraju)
1992	Decidable (Lambert)
2009	Decidable (Leroux)
2011	Decidable (Leroux)
2012	Decidable (Leroux)
2015	In $\mathbf{F}_{\omega^3}$ (Leroux and Schmitz)
2019?	Nonelementary (this talk)
2019?	In $\mathbf{F}_{\omega}$ (Leroux and Schmitz)

So **halt** is important

Outline

- High level idea of the proof
- Key combinatorial lemma
- The construction

Programs with zero tests

Additional command: **test** $\mathrm{x} = 0$

Programs with zero tests

Additional command: **test** $\mathrm{x} = 0$

Reachability becomes undecidable

Programs with zero tests

Additional command: **test** $\mathrm{x} = 0$

Reachability becomes undecidable

Let k – size of input

Programs with zero tests

Additional command: **test** $\mathrm{x} = 0$

Reachability becomes undecidable

Let k – size of input

Suppose counters are bounded by $B = f(k)$

Programs with zero tests

Additional command: **test** $\mathrm{x} = 0$

Reachability becomes undecidable

Let k – size of input

Suppose counters are bounded by $B = f(k)$

If f is n-EXP, i.e., $f(k) = 2^{\cdot^{\cdot^{\cdot^{2^k}}}} \Big\} n$ times.

Then reachability is $(n-1)$-EXPSPACE-complete

Programs with zero tests

Additional command: **test** $x = 0$

Reachability becomes undecidable

Let k – size of input

Suppose counters are bounded by $B = f(k)$

If f is n-EXP, i.e., $f(k) = \left.2^{\cdot^{\cdot^{\cdot^{2^k}}}}\right\} n$ times.

Then reachability is $(n-1)$-EXPSPACE-complete

Lipton encoded programs for $f = 2$-EXP

Programs with zero tests

Additional command: **test** $\mathrm{x} = 0$

Reachability becomes undecidable

Let k – size of input

Suppose counters are bounded by $B = f(k)$

If f is n-EXP, i.e., $f(k) = 2^{2^{\cdot^{\cdot^{\cdot^{2^k}}}}} \Big\} n$ times.

Then reachability is $(n-1)$-EXPSPACE-complete

Lipton encoded programs for $f = 2$-EXP

We can do it for any $f = n$-EXP

Encoding programs with zero tests and bounded counters

B – bound on the counters

Encoding programs with zero tests and bounded counters

B – bound on the counters

We encode this into programs with no zero tests

Encoding programs with zero tests and bounded counters

B – bound on the counters

We encode this into programs with no zero tests

Suppose we get (magically) three counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$

initialized to $\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Encoding programs with zero tests and bounded counters

B – bound on the counters

We encode this into programs with no zero tests

Suppose we get (magically) three counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$

initialized to $\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Encoding: for every x_i add x_i'

Encoding programs with zero tests and bounded counters

B – bound on the counters

We encode this into programs with no zero tests

Suppose we get (magically) three counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$

initialized to $\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Encoding: for every x_i add x'_i

Intuitively $\mathsf{x}_i + \mathsf{x}'_i = B$, so start with:

Encoding programs with zero tests and bounded counters

B – bound on the counters

We encode this into programs with no zero tests

Suppose we get (magically) three counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$

initialized to $\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Encoding: for every x_i add x_i'

Intuitively $\mathsf{x}_i + \mathsf{x}_i' = B$, so start with:

loop

$$\mathsf{x}_1' \mathrel{+}= 1 \quad \cdots \quad \mathsf{x}_l' \mathrel{+}= 1$$

$$\mathsf{b} \mathrel{-}= 1$$

Encoding programs with zero tests and bounded counters

B – bound on the counters

We encode this into programs with no zero tests

Suppose we get (magically) three counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$

initialized to $\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Encoding: for every x_i add x'_i

Intuitively $\mathsf{x}_i + \mathsf{x}'_i = B$, so start with:

loop

$$\mathsf{x}'_1 \mathrel{+}= 1 \quad \cdots \quad \mathsf{x}'_l \mathrel{+}= 1$$
$$\mathsf{b} \mathrel{-}= 1$$

Replace $\mathsf{x}_i \mathrel{+}= m$ with $\mathsf{x}_i \mathrel{+}= m \quad \mathsf{x}'_i \mathrel{-}= m$

Encoding programs with zero tests and bounded counters

B – bound on the counters

We encode this into programs with no zero tests

Suppose we get (magically) three counters $\mathsf{b}, \mathsf{c}, \mathsf{d}$

initialized to $\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Encoding: for every x_i add x_i'

Intuitively $\mathsf{x}_i + \mathsf{x}_i' = B$, so start with:

loop

$\quad \mathsf{x}_1' \mathrel{+}= 1 \quad \cdots \quad \mathsf{x}_l' \mathrel{+}= 1$

$\quad \mathsf{b} \mathrel{-}= 1$

Replace $\mathsf{x}_i \mathrel{+}= m$ with $\mathsf{x}_i \mathrel{+}= m \quad \mathsf{x}_i' \mathrel{-}= m$

Replace $\mathsf{x}_i \mathrel{-}= m$ with $\mathsf{x}_i \mathrel{-}= m \quad \mathsf{x}_i' \mathrel{+}= m$

Encoding (continued)

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

$\mathsf{x}_i' = B - \mathsf{x}_i$

Encoding (continued)

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ ⟵ c is "number of zero tests" $\cdot\, 2$

$\mathsf{x}'_i = B - \mathsf{x}_i$

Encoding (continued)

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ ← c is "number of zero tests" $\cdot\, 2$

$\mathsf{x}_i' = B - \mathsf{x}_i$

Replace **test** $\mathsf{x}_i = 0$ with

loop

$\quad \mathsf{x}_i \mathrel{+}= 1 \quad \mathsf{x}_i' \mathrel{-}= 1$

$\quad \mathsf{d} \mathrel{-}= 1$

$\mathsf{c} \mathrel{-}= 1$

loop

$\quad \mathsf{x}_i \mathrel{-}= 1 \quad \mathsf{x}_i' \mathrel{+}= 1$

$\quad \mathsf{d} \mathrel{-}= 1$

$\mathsf{c} \mathrel{-}= 1$

Encoding (continued)

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ ← c is "number of zero tests" $\cdot\, 2$

$\mathsf{x}'_i = B - \mathsf{x}_i$

Replace **test** $\mathsf{x}_i = 0$ with

loop

$\quad \mathsf{x}_i \mathrel{+}= 1 \quad \mathsf{x}'_i \mathrel{-}= 1$

$\quad \mathsf{d} \mathrel{-}= 1$

$\mathsf{c} \mathrel{-}= 1$

loop

$\quad \mathsf{x}_i \mathrel{-}= 1 \quad \mathsf{x}'_i \mathrel{+}= 1$

$\quad \mathsf{d} \mathrel{-}= 1$

$\mathsf{c} \mathrel{-}= 1$

Extend **halt** with $\mathsf{b}, \mathsf{d} = 0$

Encoding (continued)

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ ← c is "number of zero tests" $\cdot\, 2$

$\mathsf{x}_i' = B - \mathsf{x}_i$ ← holds because $\mathsf{b} = 0$

Replace **test** $\mathsf{x}_i = 0$ with

loop
  $\mathsf{x}_i \mathrel{+}= 1 \quad \mathsf{x}_i' \mathrel{-}= 1$
  $\mathsf{d} \mathrel{-}= 1$
$\mathsf{c} \mathrel{-}= 1$
loop
  $\mathsf{x}_i \mathrel{-}= 1 \quad \mathsf{x}_i' \mathrel{+}= 1$
  $\mathsf{d} \mathrel{-}= 1$
$\mathsf{c} \mathrel{-}= 1$

Extend **halt** with $\mathsf{b}, \mathsf{d} = 0$

Encoding (continued)

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ ← c is "number of zero tests" $\cdot\, 2$

$\mathsf{x}'_i = B - \mathsf{x}_i$ ← holds because $\mathsf{b} = 0$

Replace **test** $\mathsf{x}_i = 0$ with

$$\left.\begin{array}{l}
\textbf{loop} \\
\qquad \mathsf{x}_i \mathrel{+}= 1 \quad \mathsf{x}'_i \mathrel{-}= 1 \\
\qquad \mathsf{d} \mathrel{-}= 1 \\
\mathsf{c} \mathrel{-}= 1 \\
\textbf{loop} \\
\qquad \mathsf{x}_i \mathrel{-}= 1 \quad \mathsf{x}'_i \mathrel{+}= 1 \\
\qquad \mathsf{d} \mathrel{-}= 1 \\
\mathsf{c} \mathrel{-}= 1
\end{array}\right\}\ \text{c decreased by } 2 \text{ and d by at most } 2B$$

Extend **halt** with $\mathsf{b}, \mathsf{d} = 0$

Encoding (continued)

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ ← c is "number of zero tests" $\cdot\, 2$

$\mathsf{x}_i' = B - \mathsf{x}_i$ ← holds because $\mathsf{b} = 0$

Replace **test** $\mathsf{x}_i = 0$ with

$$\left.\begin{array}{l}
\textbf{loop} \\
\quad \mathsf{x}_i \mathrel{+}= 1 \quad \mathsf{x}_i' \mathrel{-}= 1 \\
\quad \mathsf{d} \mathrel{-}= 1 \\
\mathsf{c} \mathrel{-}= 1 \\
\textbf{loop} \\
\quad \mathsf{x}_i \mathrel{-}= 1 \quad \mathsf{x}_i' \mathrel{+}= 1 \\
\quad \mathsf{d} \mathrel{-}= 1 \\
\mathsf{c} \mathrel{-}= 1
\end{array}\right\}$$

c decreased by 2 and d by at most $2B$

so a false zero test implies $\mathsf{d} \neq 0$

Extend **halt** with $\mathsf{b}, \mathsf{d} = 0$

Encoding (continued)

B – bound on the counters

$\boxed{\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}}$ ⟵ c is "number of zero tests" $\cdot\, 2$

$\mathsf{x}'_i = B - \mathsf{x}_i$ ⟵ holds because $\mathsf{b} = 0$

Replace **test** $\mathsf{x}_i = 0$ with

loop
  $\mathsf{x}_i \mathrel{+}= 1 \quad \mathsf{x}'_i \mathrel{-}= 1$
  $\mathsf{d} \mathrel{-}= 1$
$\mathsf{c} \mathrel{-}= 1$
loop
  $\mathsf{x}_i \mathrel{-}= 1 \quad \mathsf{x}'_i \mathrel{+}= 1$
  $\mathsf{d} \mathrel{-}= 1$
$\mathsf{c} \mathrel{-}= 1$

} c decreased by 2 and d by at most $2B$

so a false zero test implies $\mathsf{d} \neq 0$

This is the challenge

Extend **halt** with $\mathsf{b}, \mathsf{d} = 0$

Key combinatorial lemma

(forget about VASS and programs for now)

Identity on numbers

k – input number

Identity on numbers

k – input number

Lemma (Trivial)

There exist $r_{k,0}, \ldots, r_{k,k},\ r_k,\ s_{k,0}, \ldots, s_{k,k},\ s_k > 0$

s.t. $\quad r_{k,i},\ r_k,\ s_{k,i},\ s_k = \mathcal{O}(k), \quad s_{k,i} > r_{k,i}, \quad s_k > r_k,$ and

$$\prod_{i=0}^{k} \frac{s_{k,i}}{r_{k,i}} = \frac{s_k}{r_k}$$

Identity on numbers

k – input number

Lemma (Trivial)

There exist $r_{k,0}, \ldots, r_{k,k},\ r_k,\ s_{k,0}, \ldots, s_{k,k},\ s_k > 0$

s.t. $\quad r_{k,i},\ r_k,\ s_{k,i},\ s_k = \mathcal{O}(k), \quad s_{k,i} > r_{k,i}, \quad s_k > r_k,$ and

$$\prod_{i=0}^{k} \frac{s_{k,i}}{r_{k,i}} = \frac{s_k}{r_k}$$

Solution:

$$\prod_{i=0}^{k} \frac{i+2}{i+1} = \frac{k+2}{1}$$

Identity on numbers

k – input number

Lemma (Trivial)

There exist $r_{k,0}, \ldots, r_{k,k},\ r_k,\ s_{k,0}, \ldots, s_{k,k},\ s_k > 0$

s.t. $\quad r_{k,i},\ r_k,\ s_{k,i},\ s_k = \mathcal{O}(k), \quad s_{k,i} > r_{k,i}, \quad s_k > r_k, \quad$ and

$$\prod_{i=0}^{k} \frac{s_{k,i}}{r_{k,i}} = \frac{s_k}{r_k}$$

Solution:

$$\prod_{i=0}^{k} \frac{i+2}{i+1} = \frac{k+2}{1}$$

We need something more complicated

Serious identity on numbers

k – input number

Serious identity on numbers

k – input number

Lemma (Key)

There exist $r_{k,0}, \ldots, r_{k,k},\ r_k,\ s_{k,0}, \ldots, s_{k,k},\ s_k > 0$

s.t. $r_{k,i},\ r_k,\ s_{k,i},\ s_k = \mathcal{O}(2^{poly(k)}),\quad s_{k,i} > r_{k,i},\quad s_k > r_k,$ and

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} = \frac{s_k}{r_k}$$

Serious identity on numbers

k – input number

Lemma (Key)

There exist $r_{k,0}, \ldots, r_{k,k},\ r_k,\ s_{k,0}, \ldots, s_{k,k},\ s_k > 0$

s.t. $\quad r_{k,i},\ r_k,\ s_{k,i},\ s_k = \mathcal{O}(2^{poly(k)}), \quad s_{k,i} > r_{k,i}, \quad s_k > r_k,$ and

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} = \frac{s_k}{r_k}$$

Wrong solution:

$$\prod_{i=0}^{k} \left(\frac{2^{k+1}+1}{2^{k+1}}\right)^{2^i} = \left(\frac{2^{k+1}+1}{2^{k+1}}\right)^{2^{k+1}-1} < e$$

Serious identity on numbers

k – input number

Lemma (Key)

There exist $r_{k,0}, \ldots, r_{k,k},\ r_k,\ s_{k,0}, \ldots, s_{k,k},\ s_k > 0$

s.t. $\quad r_{k,i},\ r_k,\ s_{k,i},\ s_k = \mathcal{O}(2^{poly(k)}), \quad s_{k,i} > r_{k,i}, \quad s_k > r_k,$ and

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} = \frac{s_k}{r_k}$$

Wrong solution:

$$\prod_{i=0}^{k} \left(\frac{2^{k+1}+1}{2^{k+1}} \right)^{2^i} = \left(\frac{2^{k+1}+1}{2^{k+1}} \right)^{2^{k+1}-1} < e$$

e is small

Serious identity on numbers

k – input number

Lemma (Key)

There exist $r_{k,0}, \ldots, r_{k,k},\ r_k,\ s_{k,0}, \ldots, s_{k,k},\ s_k > 0$

s.t. $\quad r_{k,i},\ r_k,\ s_{k,i},\ s_k = \mathcal{O}(2^{poly(k)}), \quad s_{k,i} > r_{k,i}, \quad s_k > r_k,$ and

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} = \frac{s_k}{r_k}$$

Wrong solution:

$$\prod_{i=0}^{k} \left(\frac{2^{k+1}+1}{2^{k+1}} \right)^{2^i} = \left(\frac{2^{k+1}+1}{2^{k+1}} \right)^{2^{k+1}-1} < e$$

e is small

but this is big

Proving the identity

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} = \frac{s_k}{r_k}$$

Proving the identity

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} = \frac{s_k}{r_k}$$

A solution: $\quad$ ($a_k, a_{k,i}$ are auxiliary)

$$a_{k,k-i} = a_k + 2^i \quad s_{k,k-i} = (a_k)^i a_{k,i} \quad r_{k,k-i} = a_k \prod_{j=0}^{i-1} a_{k,j}$$

$$a_k = 2^{2k} \quad s_k = \prod_{j=0}^{k} a_{k,j} \quad r_k = (a_k)^{k+1}$$

Proving the identity

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} = \frac{s_k}{r_k}$$

A solution: $(a_k, a_{k,i}$ are auxiliary)

$$a_{k,k-i} = a_k + 2^i \quad s_{k,k-i} = (a_k)^i a_{k,i} \quad r_{k,k-i} = a_k \prod_{j=0}^{i-1} a_{k,j}$$

$$a_k = 2^{2k} \quad s_k = \prod_{j=0}^{k} a_{k,j} \quad r_k = (a_k)^{k+1}$$

The point is these numbers are computable

Proving the identity

$$\prod_{i=0}^{k}\left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} = \frac{s_k}{r_k}$$

A solution: $(a_k, a_{k,i}$ are auxiliary)

$$a_{k,k-i} = a_k + 2^i \quad s_{k,k-i} = (a_k)^i a_{k,i} \quad r_{k,k-i} = a_k \prod_{j=0}^{i-1} a_{k,j}$$

$$a_k = 2^{2k} \quad s_k = \prod_{j=0}^{k} a_{k,j} \quad r_k = (a_k)^{k+1}$$

The point is these numbers are computable

Fix $y > 0$, you want $x > y$ s.t. $x = y \cdot \frac{s_k}{r_k}$

Proving the identity

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} = \frac{s_k}{r_k}$$

A solution: $(a_k, a_{k,i}$ are auxiliary)

$$\begin{array}{rclrclrcl} a_{k,k-i} & = & a_k + 2^i & s_{k,k-i} & = & (a_k)^i a_{k,i} & r_{k,k-i} & = & a_k \prod_{j=0}^{i-1} a_{k,j} \\ a_k & = & 2^{2k} & s_k & = & \prod_{j=0}^{k} a_{k,j} & r_k & = & (a_k)^{k+1} \end{array}$$

The point is these numbers are computable

Fix $y > 0$, you want $x > y$ s.t. $x = y \cdot \frac{s_k}{r_k}$

You start with $x = y$ and you multiply by $\frac{s_{k,i}}{r_{k,i}}$ (in order $i = k, \ldots, 0$)

Proving the identity

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} = \frac{s_k}{r_k}$$

A solution: $(a_k, a_{k,i}$ are auxiliary)

$$a_{k,k-i} = a_k + 2^i \quad s_{k,k-i} = (a_k)^i a_{k,i} \quad r_{k,k-i} = a_k \prod_{j=0}^{i-1} a_{k,j}$$

$$a_k = 2^{2k} \quad s_k = \prod_{j=0}^{k} a_{k,j} \quad r_k = (a_k)^{k+1}$$

The point is these numbers are computable

Fix $y > 0$, you want $x > y$ s.t. $x = y \cdot \frac{s_k}{r_k}$

You start with $x = y$ and you multiply by $\frac{s_{k,i}}{r_{k,i}}$ (in order $i = k, \ldots, 0$)

$$x = y \cdot \prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i}$$

Proving the identity

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} = \frac{s_k}{r_k}$$

A solution: $(a_k, a_{k,i}$ are auxiliary$)$

$$a_{k,k-i} = a_k + 2^i \quad s_{k,k-i} = (a_k)^i a_{k,i} \quad r_{k,k-i} = a_k \prod_{j=0}^{i-1} a_{k,j}$$

$$a_k = 2^{2k} \quad s_k = \prod_{j=0}^{k} a_{k,j} \quad r_k = (a_k)^{k+1}$$

The point is these numbers are computable

Fix $y > 0$, you want $x > y$ s.t. $x = y \cdot \frac{s_k}{r_k}$

You start with $x = y$ and you multiply by $\frac{s_{k,i}}{r_{k,i}}$ (in order $i = k, \ldots, 0$)

$$x = y \cdot \prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i}$$

Then $x, y = \Omega\left(2^{2^k}\right)$

Simple program using the identity

Input – $2^{poly(k)}$, $\quad r_k,\ r_{k,i},\ s_k,\ s_{k,i} \in \mathcal{O}(2^{poly(k)})$

Simple program using the identity

Input – $2^{poly(k)}$, $\quad r_k,\ r_{k,i},\ s_k,\ s_{k,i} \in \mathcal{O}(2^{poly(k)})$

x $+= 1$ $\quad$ y $+= 1$
loop
    x $+= 1$ $\quad$ y $+= 1$
for $i = k, \ldots, 0$ **do**
    loop exactly 2^i **times**
        loop
            x $-= r_{k,i}$ $\quad$ x$'$ $+= s_{k,i}$
        loop
            x$'$ $-= 1$ $\quad$ x $+= 1$
loop
    x $-= s_k$ $\quad$ y $-= r_k$
halt if y $= 0$

Simple program using the identity

Input – $2^{poly(k)}$, $\quad r_k,\ r_{k,i},\ s_k,\ s_{k,i} \in \mathcal{O}(2^{poly(k)})$

$\mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$
loop
$\quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ ⟵ Initialize x, y
for $i = k, \ldots, 0$ **do**
$\quad$ **loop exactly** 2^i **times**
$\quad\quad$ **loop**
$\quad\quad\quad \mathsf{x} \mathrel{-}= r_{k,i} \quad \mathsf{x}' \mathrel{+}= s_{k,i}$
$\quad\quad$ **loop**
$\quad\quad\quad \mathsf{x}' \mathrel{-}= 1 \quad \mathsf{x} \mathrel{+}= 1$
loop
$\quad \mathsf{x} \mathrel{-}= s_k \quad \mathsf{y} \mathrel{-}= r_k$
halt if $\mathsf{y} = 0$

Simple program using the identity

Input – $2^{poly(k)}$, $\quad r_k,\ r_{k,i},\ s_k,\ s_{k,i} \in \mathcal{O}(2^{poly(k)})$

$\mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$
loop
    $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ ← Initialize x, y
for $i = k, \ldots, 0$ **do**
    loop exactly 2^i **times**
        loop
            $\mathsf{x} \mathrel{-}= r_{k,i} \quad \mathsf{x}' \mathrel{+}= s_{k,i}$ ← Multiply weakly x by $\frac{s_{k,i}}{r_{k,i}}$
        loop
            $\mathsf{x}' \mathrel{-}= 1 \quad \mathsf{x} \mathrel{+}= 1$
loop
    $\mathsf{x} \mathrel{-}= s_k \quad \mathsf{y} \mathrel{-}= r_k$
halt if $\mathsf{y} = 0$

Simple program using the identity

Input – $2^{poly(k)}$, $\quad r_k,\ r_{k,i},\ s_k,\ s_{k,i} \in \mathcal{O}(2^{poly(k)})$

$\mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$
loop
  $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ ← Initialize x, y
for $i = k, \ldots, 0$ **do**
  loop exactly 2^i **times**
    loop
      $\mathsf{x} \mathrel{-}= r_{k,i} \quad \mathsf{x}' \mathrel{+}= s_{k,i}$ ← Multiply weakly x by $\frac{s_{k,i}}{r_{k,i}}$
    loop
      $\mathsf{x}' \mathrel{-}= 1 \quad \mathsf{x} \mathrel{+}= 1$ ← Put the value back to x
loop
  $\mathsf{x} \mathrel{-}= s_k \quad \mathsf{y} \mathrel{-}= r_k$
halt if $\mathsf{y} = 0$

Simple program using the identity

Input – $2^{poly(k)}$, $\quad r_k,\ r_{k,i},\ s_k,\ s_{k,i} \in \mathcal{O}(2^{poly(k)})$

```
x += 1    y += 1
loop
    x += 1    y += 1
for i = k, ..., 0 do
    loop exactly 2^i times
        loop
            x -= r_{k,i}    x' += s_{k,i}
        loop
            x' -= 1    x += 1
loop
    x -= s_k    y -= r_k
halt if y = 0
```

Initialize x, y

Multiply weakly x by $\frac{s_{k,i}}{r_{k,i}}$

Put the value back to x

Afterwards $\mathsf{x} \leq \mathsf{y} \cdot \frac{s_k}{r_k}$

Simple program using the identity

Input – $2^{poly(k)}$, $\quad r_k,\ r_{k,i},\ s_k,\ s_{k,i} \in \mathcal{O}(2^{poly(k)})$

$\mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$
loop
    $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ ← Initialize x, y
for $i = k, \ldots, 0$ **do**
    loop exactly 2^i **times**
        loop
            $\mathsf{x} \mathrel{-}= r_{k,i} \quad \mathsf{x}' \mathrel{+}= s_{k,i}$ ← Multiply weakly x by $\frac{s_{k,i}}{r_{k,i}}$
        loop
            $\mathsf{x}' \mathrel{-}= 1 \quad \mathsf{x} \mathrel{+}= 1$ ← Put the value back to x
loop
    $\mathsf{x} \mathrel{-}= s_k \quad \mathsf{y} \mathrel{-}= r_k$
halt if $\mathsf{y} = 0$

Afterwards $\mathsf{x} \leq \mathsf{y} \cdot \frac{s_k}{r_k}$

Actually $\mathsf{x} = \mathsf{y} \cdot \frac{s_k}{r_k}$

Simple program using the identity

Input – $2^{poly(k)}, \quad r_k,\ r_{k,i},\ s_k,\ s_{k,i} \in \mathcal{O}(2^{poly(k)})$

$\mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$
loop
  $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ ← Initialize x, y
for $i = k, \ldots, 0$ **do**
  loop exactly 2^i **times**
    loop
      $\mathsf{x} \mathrel{-}= r_{k,i} \quad \mathsf{x}' \mathrel{+}= s_{k,i}$ ← Multiply weakly x by $\frac{s_{k,i}}{r_{k,i}}$
    loop
      $\mathsf{x}' \mathrel{-}= 1 \quad \mathsf{x} \mathrel{+}= 1$ ← Put the value back to x

Afterwards $\mathsf{x} \leq \mathsf{y} \cdot \frac{s_k}{r_k}$

$$\left.\begin{array}{l} \textbf{loop} \\ \quad \mathsf{x} \mathrel{-}= s_k \quad \mathsf{y} \mathrel{-}= r_k \\ \textbf{halt if } \mathsf{y} = 0 \end{array}\right\} \text{Actually } \mathsf{x} = \mathsf{y} \cdot \frac{s_k}{r_k}$$

So x, y are initialized to multiples of $a \in \Omega(2^{2^k})$

The construction

(Now remember VASS, programs, the key lemma. . .)

Recall what we wanted

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Recall what we wanted

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

If B is fixed, just start the program with:

```
b += B
loop
    c += 1    d += B
```

Recall what we wanted

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

If B is fixed, just start the program with:

$\mathsf{b} \mathrel{+}= B$ ← "gadget for ratio B"

loop

$\quad \mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= B$

Recall what we wanted

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

If B is fixed, just start the program with:

$\mathsf{b} \mathrel{+}= B$ ⟵ "gadget for ratio B"

loop

    $\mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= B$

But in general we want $B = \left.2^{\cdot^{\cdot^{\cdot^{2^k}}}}\right\}n$ times.

Recall what we wanted

B – bound on the counters

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

If B is fixed, just start the program with:

$\mathsf{b}\ += B$ ⟵ "gadget for ratio B"
loop
  $\mathsf{c}\ += 1 \quad \mathsf{d}\ += B$

But in general we want $B = \left.2^{\cdot^{\cdot^{\cdot^{2^k}}}}\right\}n$ times.

For this we need the combinatorial lemma

Gadget for ratio $B = n$-EXP

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ gives us B-bounded 0-tests

Gadget for ratio $B = n$-EXP

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ gives us B-bounded 0-tests

Lemma (lifting the gadget)

Using a gadget for ratio B we can get a gadget for ratio $\approx 2^B$

Gadget for ratio $B = n$-EXP

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ gives us B-bounded 0-tests

Lemma (lifting the gadget)

Using a gadget for ratio B we can get a gadget for ratio $\approx 2^B$

A program with B-bounded 0-tests that ends with

$\mathsf{b} \approx 2^B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Gadget for ratio $B = n$-EXP

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ gives us B-bounded 0-tests

Lemma (lifting the gadget)

Using a gadget for ratio B we can get a gadget for ratio $\approx 2^B$

A program with B-bounded 0-tests that ends with

$\mathsf{b} \approx 2^B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

How to use the lemma:

Gadget for ratio $B = n$-EXP

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ gives us B-bounded 0-tests

Lemma (lifting the gadget)

Using a gadget for ratio B we can get a gadget for ratio $\approx 2^B$

A program with B-bounded 0-tests that ends with

$\mathsf{b} \approx 2^B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

How to use the lemma:

- By the previous slide we can start with B linear in the input

Gadget for ratio $B = n$-EXP

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ gives us B-bounded 0-tests

Lemma (lifting the gadget)

Using a gadget for ratio B we can get a gadget for ratio $\approx 2^B$

A program with B-bounded 0-tests that ends with

$\mathsf{b} \approx 2^B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

How to use the lemma:

- By the previous slide we can start with B linear in the input
- Afterwards lift the gadget n times

Gadget for ratio $B = n$-EXP

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ gives us B-bounded 0-tests

Lemma (lifting the gadget)

Using a gadget for ratio B we can get a gadget for ratio $\approx 2^B$

A program with B-bounded 0-tests that ends with

$\mathsf{b} \approx 2^B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

How to use the lemma:

- By the previous slide we can start with B linear in the input
- Afterwards lift the gadget n times

A program proving the lemma is what's left

How to use B-bounded 0-tests?

How to use B-bounded 0-tests?

Let $a \leq B$ computable by program $\mathcal{P}_a$ with B-bounded 0-tests

How to use B-bounded 0-tests?

Let $a \leq B$ computable by program $\mathcal{P}_a$ with B-bounded 0-tests (afterwards a stored in a)

How to use B-bounded 0-tests?

Let $a \leq B$ computable by program $\mathcal{P}_a$ with B-bounded 0-tests (afterwards a stored in a)

- We want e.g.: x $+=$ a

How to use B-bounded 0-tests?

Let $a \leq B$ computable by program $\mathcal{P}_a$ with B-bounded 0-tests (afterwards a stored in a)

- We want e.g.: x $+=$ a

$<\mathcal{P}_a$ with **halt** removed$>$ $\quad \rightarrow a$ computed in a
loop
    x $+= 1$
    a $-= 1$
test a $= 0$

How to use B-bounded 0-tests?

Let $a \leq B$ computable by program $\mathcal{P}_a$ with B-bounded 0-tests (afterwards a stored in a)

- We want e.g.: x $+=$ $\boxed{\text{a}}$

$<\mathcal{P}_a$ with **halt** removed$>$ $\quad \rightarrow a$ computed in a
loop
    x $+= 1$
    a $-= 1$
test a $= 0$

How to use B-bounded 0-tests?

Let $a \leq B$ computable by program $\mathcal{P}_a$ with B-bounded 0-tests (afterwards a stored in a)

- We want e.g.: x $+=$ $\boxed{\text{a}}$

```
<P_a with halt removed>        → a computed in a
loop
    x += 1
    a −= 1
test a = 0
```

- Also: **loop exactly** $\boxed{\text{a}}$ **times** $<$*body*$>$

How to use B-bounded 0-tests?

Let $a \leq B$ computable by program $\mathcal{P}_a$ with B-bounded 0-tests (afterwards a stored in a)

- We want e.g.: x $+=$ $\boxed{\text{a}}$

```
<P_a with halt removed>        → a computed in a
loop
    x += 1
    a −= 1
test a = 0
```

- Also: **loop exactly** $\boxed{\text{a}}$ **times** $<$*body*$>$

- Or: **loop at most** b **times** $<$*body*$>$

(b has no constraints)

How to use B-bounded 0-tests?

Let $a \leq B$ computable by program $\mathcal{P}_a$ with B-bounded 0-tests (afterwards a stored in a)

- We want e.g.: x $+= \boxed{\text{a}}$

$<\mathcal{P}_a$ with **halt** removed$>$ $\quad\rightarrow a$ computed in a
loop
  x $+= 1$
  a $-= 1$
test a $= 0$

- Also: **loop exactly** $\boxed{\text{a}}$ **times** $<$*body*$>$

- Or: **loop at most** b **times** $<$*body*$>$

(b has no constraints)

loop
  b $-= 1$ $\quad$ b$'$ $+= 1$
loop
  b$'$ $-= 1$ $\quad$ b $+= 1$
  $<$*body*$>$

What about the identity?

with B-bounded zero tests

What about the identity?

with B-bounded zero tests

Recall

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} = \frac{s_k}{r_k}$$

What about the identity?

with B-bounded zero tests

Recall

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} = \frac{s_k}{r_k}$$

We need k s.t. $s_{k,i},\ s_k,\ r_{k,i},\ r_k \leq B$

$(B \approx 2^k)$

What about the identity?

with B-bounded zero tests

Recall

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} = \frac{s_k}{r_k}$$

We need k s.t. $s_{k,i},\ s_k,\ r_{k,i},\ r_k \leq B$ so all r and s computable

$(B \approx 2^k)$

What about the identity?

with B-bounded zero tests

Recall

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} = \frac{s_k}{r_k}$$

We need k s.t. $s_{k,i},\ s_k,\ r_{k,i},\ r_k \leq B$ so all r and s computable

$(B \approx 2^k)$

Let

$$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i} \quad A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$$

What about the identity?

with B-bounded zero tests

Recall

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} = \frac{s_k}{r_k}$$

We need k s.t. $s_{k,i},\ s_k,\ r_{k,i},\ r_k \leq B$ so all r and s computable

$(B \approx 2^k)$

Let

$$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i} \quad A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$$

We also need k s.t. $B_k, A_k \approx 2^B$ (B_k will be the new ratio)

What about the identity?

with B-bounded zero tests

Recall

$$\prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} = \frac{s_k}{r_k}$$

We need k s.t. $s_{k,i},\ s_k,\ r_{k,i},\ r_k \leq B$ so all r and s computable
$(B \approx 2^k)$

Let

$$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i} \quad A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$$

We also need k s.t. $B_k, A_k \approx 2^B$ (B_k will be the new ratio)

What is k? Computable, assume some variable k $= k$

High level description of the program

Output: $\mathsf{b} = B_k,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

High level description of the program

$$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i}$$
$$A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$$

Output: $\mathsf{b} = B_k,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

High level description of the program

$$B_k = \prod_{i=0}^{k} \left(s_{k,i}\right)^{2^i}$$
$$A_k = \prod_{i=0}^{k} \left(r_{k,i}\right)^{2^i}$$

Output: $\mathsf{b} = B_k,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Two auxiliary variables x, y (to check correctness)

At some point they satisfy

$$\mathsf{x} \;=\; \mathsf{y} \cdot \prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}}\right)^{2^i} \;=\; \mathsf{y} \cdot \frac{s_k}{r_k}$$

High level description of the program

$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i}$
$A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$

Output: $\mathsf{b} = B_k,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Two auxiliary variables x, y (to check correctness)

At some point they satisfy

$$\mathsf{x} \ = \ \mathsf{y} \cdot \prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} \ = \ \mathsf{y} \cdot \frac{s_k}{r_k}$$

$\mathsf{b} \mathrel{+}= 1$
loop
    $\mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= 1 \quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$
<main loop>
loop
    $\mathsf{x} \mathrel{-}= \boxed{s_{\mathsf{k}}} \quad \mathsf{y} \mathrel{-}= \boxed{r_{\mathsf{k}}}$
halt if $\mathsf{y} = 0$

High level description of the program

$$B_k = \prod_{i=0}^{k} \left(s_{k,i}\right)^{2^i}$$
$$A_k = \prod_{i=0}^{k} \left(r_{k,i}\right)^{2^i}$$

Output: $\mathsf{b} = B_k,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Two auxiliary variables x, y (to check correctness)

At some point they satisfy

$$\mathsf{x} \ = \ \mathsf{y} \cdot \prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} \ = \ \mathsf{y} \cdot \frac{s_k}{r_k}$$

$\mathsf{b} \mathrel{+}= 1$
loop
    $\mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= 1 \quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1 \longleftarrow$ $\mathsf{c}, \mathsf{d}, \mathsf{x}, \mathsf{y} := c \cdot A_k$
<main loop>
loop
    $\mathsf{x} \mathrel{-}= \boxed{s_\mathsf{k}} \quad \mathsf{y} \mathrel{-}= \boxed{r_\mathsf{k}}$
halt if $\mathsf{y} = 0$

High level description of the program

$B_k = \prod_{i=0}^{k} \left(s_{k,i}\right)^{2^i}$
$A_k = \prod_{i=0}^{k} \left(r_{k,i}\right)^{2^i}$

Output: $\mathsf{b} = B_k, \ \mathsf{c} \geq 0, \ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Two auxiliary variables x, y (to check correctness)

At some point they satisfy

$$\mathsf{x} \ = \ \mathsf{y} \cdot \prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} \ = \ \mathsf{y} \cdot \frac{s_k}{r_k}$$

$\mathsf{b} \mathrel{+}= 1$
loop
  $\mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= 1 \quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ $\longleftarrow$ $\mathsf{c}, \mathsf{d}, \mathsf{x}, \mathsf{y} := c \cdot A_k$
$<$*main loop*$>$ $\leftarrow$ $\mathsf{c} := \mathsf{c}/A_k$, $\mathsf{d}, \mathsf{x} := d \cdot B_k / A_k$, $\mathsf{b} := B_k$
loop
  $\mathsf{x} \mathrel{-}= \boxed{s_{\mathsf{k}}} \quad \mathsf{y} \mathrel{-}= \boxed{r_{\mathsf{k}}}$
halt if $\mathsf{y} = 0$

High level description of the program

$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i}$
$A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$

Output: $\mathsf{b} = B_k,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$

Two auxiliary variables x, y (to check correctness)

At some point they satisfy

$$\mathsf{x} \ = \ \mathsf{y} \cdot \prod_{i=0}^{k} \left(\frac{s_{k,i}}{r_{k,i}} \right)^{2^i} \ = \ \mathsf{y} \cdot \frac{s_k}{r_k}$$

b $+= 1$
loop
  c $+= 1$ d $+= 1$ x $+= 1$ y $+= 1$ ← c, d, x, y $:= c \cdot A_k$
<main loop> ← c $:=$ c$/A_k$, d, x $:= d \cdot B_k / A_k$, b $:= B_k$
loop
  x $-= \boxed{s_{\mathsf{k}}}$ y $-= \boxed{r_{\mathsf{k}}}$
halt if y $= 0$

Invariant
b · c = d

The main loop

Invariant
$\mathsf{b} \cdot \mathsf{c} = \mathsf{d}$

$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i}$
$A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$

The main loop

Invariant
$b \cdot c = d$

$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i}$
$A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$

for $i = k, \ldots, 0$ **do**
  loop exactly $\boxed{2^i}$ **times**
    loop
      $c \mathrel{-}= \boxed{r_{k,i}}$ $\quad c' \mathrel{+}= 1$
      loop at most b **times**
        $d \mathrel{-}= \boxed{r_{k,i}}$ $d' \mathrel{+}= \boxed{s_{k,i}}$ $x \mathrel{-}= \boxed{r_{k,i}}$ $x' \mathrel{+}= \boxed{s_{k,i}}$
    loop
      $b \mathrel{-}= 1$ $\quad b' \mathrel{+}= \boxed{s_{k,i}}$
    loop
      $b' \mathrel{-}= 1$ $\quad b \mathrel{+}= 1$
    loop
      $c' \mathrel{-}= 1$ $\quad c \mathrel{+}= 1$
      loop at most b **times**
        $d' \mathrel{-}= 1$ $\quad d \mathrel{+}= 1$ $\quad x' \mathrel{-}= 1$ $\quad x \mathrel{+}= 1$

The main loop

Invariant
$b \cdot c = d$

$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i}$
$A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$

$c' := c/r_{k,i}$, $d' := d \cdot \frac{s_{k,i}}{r_{k,i}}$

for $i = k, \ldots, 0$ **do**
    loop exactly $\boxed{2^i}$ **times**
        loop
            $c \mathrel{-}= \boxed{r_{k,i}}$ $\quad c' \mathrel{+}= 1$
            loop at most b **times**
                $d \mathrel{-}= \boxed{r_{k,i}}$ $d' \mathrel{+}= \boxed{s_{k,i}}$ $x \mathrel{-}= \boxed{r_{k,i}}$ $x' \mathrel{+}= \boxed{s_{k,i}}$

        loop
            $b \mathrel{-}= 1$ $\quad b' \mathrel{+}= \boxed{s_{k,i}}$
        loop
            $b' \mathrel{-}= 1$ $\quad b \mathrel{+}= 1$
        loop
            $c' \mathrel{-}= 1$ $\quad c \mathrel{+}= 1$
            loop at most b **times**
                $d' \mathrel{-}= 1$ $\quad d \mathrel{+}= 1$ $\quad x' \mathrel{-}= 1$ $\quad x \mathrel{+}= 1$

The main loop

Invariant
$b \cdot c = d$

$B_k = \prod_{i=0}^{k} (s_{k,i})^{2^i}$
$A_k = \prod_{i=0}^{k} (r_{k,i})^{2^i}$

for $i = k, \ldots, 0$ **do**
  loop exactly $\boxed{2^i}$ **times**
    loop
      $c \mathrel{-}= \boxed{r_{k,i}}$ $\quad c' \mathrel{+}= 1$ $\qquad c' := c/r_{k,i},\ d' := d \cdot \frac{s_{k,i}}{r_{k,i}}$
      loop at most b **times**
        $d \mathrel{-}= \boxed{r_{k,i}}$ $d' \mathrel{+}= \boxed{s_{k,i}}$ $x \mathrel{-}= \boxed{r_{k,i}}$ $x' \mathrel{+}= \boxed{s_{k,i}}$
    loop
      $b \mathrel{-}= 1$ $\quad b' \mathrel{+}= \boxed{s_{k,i}}$ $\qquad b' := b \cdot s_{k,i}$
    loop
      $b' \mathrel{-}= 1$ $\quad b \mathrel{+}= 1$
    loop
      $c' \mathrel{-}= 1$ $\quad c \mathrel{+}= 1$
      loop at most b **times**
        $d' \mathrel{-}= 1$ $\quad d \mathrel{+}= 1$ $\quad x' \mathrel{-}= 1$ $\quad x \mathrel{+}= 1$

The main loop

Invariant
$b \cdot c = d$

$B_k = \prod_{i=0}^{k} \left(s_{k,i}\right)^{2^i}$
$A_k = \prod_{i=0}^{k} \left(r_{k,i}\right)^{2^i}$

for $i = k, \ldots, 0$ **do**
  loop exactly $\boxed{2^i}$ **times**
    loop
      $c \mathrel{-}= \boxed{r_{k,i}} \quad c' \mathrel{+}= 1$
      loop at most b **times**
        $d \mathrel{-}= \boxed{r_{k,i}} \; d' \mathrel{+}= \boxed{s_{k,i}} \; x \mathrel{-}= \boxed{r_{k,i}} \; x' \mathrel{+}= \boxed{s_{k,i}}$
    loop
      $b \mathrel{-}= 1 \quad b' \mathrel{+}= \boxed{s_{k,i}}$
    loop
      $b' \mathrel{-}= 1 \quad b \mathrel{+}= 1$
    loop
      $c' \mathrel{-}= 1 \quad c \mathrel{+}= 1$
      loop at most b **times**
        $d' \mathrel{-}= 1 \quad d \mathrel{+}= 1 \quad x' \mathrel{-}= 1 \quad x \mathrel{+}= 1$

$c' := c / r_{k,i}, \; d' := d \cdot \frac{s_{k,i}}{r_{k,i}}$

$b' := b \cdot s_{k,i}$

if any **loop** not maximal then $x < y \cdot \frac{s_k}{r_k}$

Conclusion

Conclusion

- Several applications and corollaries

e.g. satisfiability of FO2 on data words

Conclusion

- Several applications and corollaries

e.g. satisfiability of FO2 on data words

- We can do k-EXPSPACE-hardness in dimension $\mathcal{O}(k)$ (so fixed)

Conclusion

- Several applications and corollaries

e.g. satisfiability of FO2 on data words

- We can do k-EXPSPACE-hardness in dimension $\mathcal{O}(k)$ (so fixed)

Can we do Tower in fixed dimension?

Conclusion

- Several applications and corollaries

e.g. satisfiability of FO2 on data words

- We can do k-EXPSPACE-hardness in dimension $\mathcal{O}(k)$ (so fixed)

Can we do Tower in fixed dimension?

- The complexity is almost tight

Unless you believe in things between Tower ($\mathbf{F}_3$) and Ackermann ($\mathbf{F}_\omega$)

Conclusion

- Several applications and corollaries

e.g. satisfiability of FO2 on data words

- We can do k-EXPSPACE-hardness in dimension $\mathcal{O}(k)$ (so fixed)

Can we do Tower in fixed dimension?

- The complexity is almost tight

Unless you believe in things between Tower ($\mathbf{F}_3$) and Ackermann ($\mathbf{F}_\omega$)

- Can we improve lower bounds of Pushdown-VASS, BVASS...?

Conclusion

- Several applications and corollaries

e.g. satisfiability of FO2 on data words

- We can do k-EXPSPACE-hardness in dimension $\mathcal{O}(k)$ (so fixed)

Can we do Tower in fixed dimension?

- The complexity is almost tight

Unless you believe in things between Tower ($\mathbf{F}_3$) and Ackermann ($\mathbf{F}_\omega$)

- Can we improve lower bounds of Pushdown-VASS, BVASS...?

- This originated from studying 1-Pushdown-VASS

Conclusion

- Several applications and corollaries

e.g. satisfiability of FO2 on data words

- We can do k-EXPSPACE-hardness in dimension $\mathcal{O}(k)$ (so fixed)

Can we do Tower in fixed dimension?

- The complexity is almost tight

Unless you believe in things between Tower ($\mathbf{F}_3$) and Ackermann ($\mathbf{F}_\omega$)

- Can we improve lower bounds of Pushdown-VASS, BVASS...?

- This originated from studying 1-Pushdown-VASS

So maybe it's good to study restrictions of generalizations of etc...