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

Rennes 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

Rennes 2018

Introduction

Petri Nets, VASS, programs with no zero tests

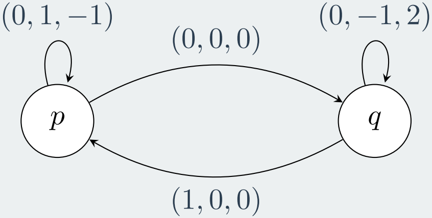
Vector addition systems with states (VASS)

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

Vector addition systems with states (VASS)

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

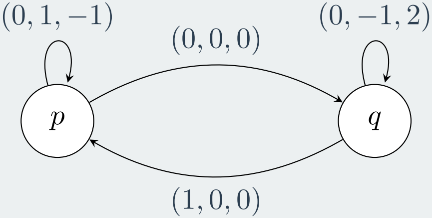
Example: $d = 3$, $Q = \{p, q\}$



Vector addition systems with states (VASS)

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

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

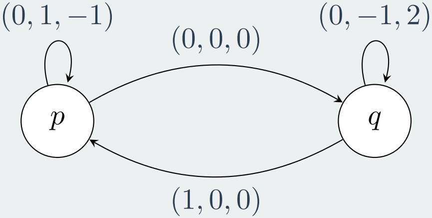


Configurations $p(\mathbf{v}) = (p, \mathbf{v}) \in Q \times \mathbb{N}^d$

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

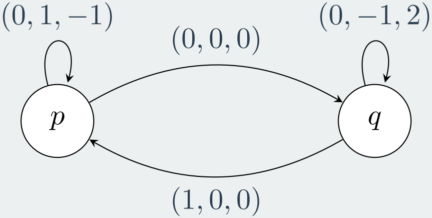
Example run:

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

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:

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

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

Decision problems

Decision problems

Reachability problem:

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

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

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

Coverability problem:

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

DECIDE: whether exists $\mathbf{v}'$ s.t. $p(\mathbf{u}) \rightarrow^* q(\mathbf{v}')$ 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 or without zero tests)

Counter programs (with or without 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 or without 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 $\mathsf{x}_1, \ldots, \mathsf{x}_l = 0$	(terminate if listed variables are zero).

Counter programs (with or without 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 or without 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 or without 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 or without 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
$\quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{x}' \mathrel{-}= 1$
$\quad \mathsf{y} \mathrel{+}= 2$
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.

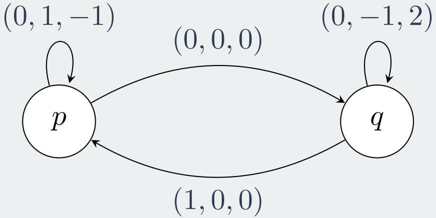
DECIDE: Does it have a complete run (executing **halt**)?

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**)?

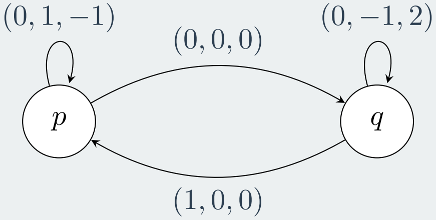


Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

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



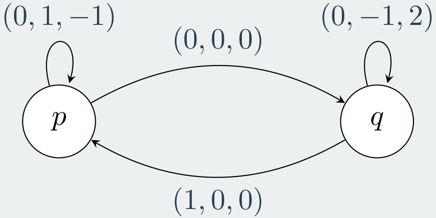
$p(0,0,1) \to^* p(1,0,2)$?

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**)?



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

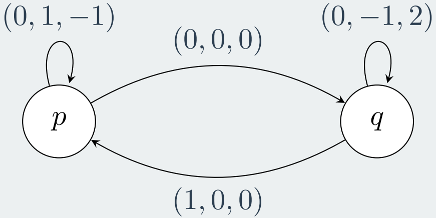
z += 1
loop
    loop
        y += 1   z −= 1
    loop
        y −= 1   z += 2
    x += 1
x −= 1   z −= 2
halt if x, y, z = 0.

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**)?



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

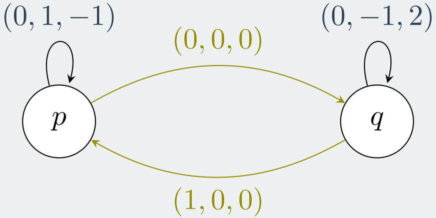
z += 1
loop
  loop
    y += 1 z −= 1
  loop
    y −= 1 z += 2
  x += 1
x −= 1 z −= 2
halt if x, y, z = 0.

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**)?



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

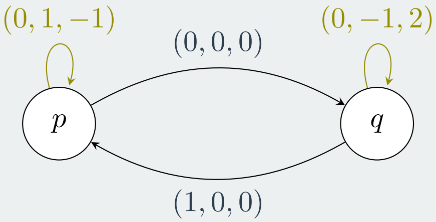
```
z += 1
loop
    loop
        y += 1    z -= 1
    loop
        y -= 1    z += 2
    x += 1
x -= 1    z -= 2
halt if x, y, z = 0.
```

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**)?



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

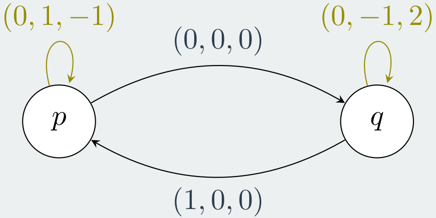
z += 1
loop
    loop
        y += 1   z −= 1
    loop
        y −= 1   z += 2
    x += 1
x −= 1   z −= 2
halt if $x, y, z = 0$.

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**)?



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

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

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)

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)

Coverability
Rackoff 1978
EXPSPACE-complete

So **halt** is important

Outline

- High level idea of the proof
- Key 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** $\mathrm{x} = 0$

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

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., $f(k) = \left.2^{\cdot^{\cdot^{\cdot^{2^k}}}}\right\} n \text{ times.}$

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

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 $2B$

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

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

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ ← 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
$\quad \mathsf{x}_i \mathrel{+}= 1 \quad \mathsf{x}_i' \mathrel{-}= 1$
$\quad \mathsf{d} \mathrel{-}= 1$
$\mathsf{c} \mathrel{-}= 1$
loop
$\quad \mathsf{x}_i \mathrel{-}= 1 \quad \mathsf{x}_i' \mathrel{+}= 1$
$\quad \mathsf{d} \mathrel{-}= 1$
$\mathsf{c} \mathrel{-}= 1$

} 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 $\mathsf{b}, \mathsf{d} = 0$

The main construction

to obtain b, c and d

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:

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

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

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

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 ← "gadget for ratio B"
loop
    c $+=$ 1 d $+=$ 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:

$\mathsf{b} \mathrel{+}= B$ ← "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.

For this we need an iterative construction

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 $B = \left.2^{\cdot^{\cdot^{2^k}}}\right\} n$ times.

For this we need an iterative construction

Some variables will be bounded and allowed to be 0-tested

Gadget for ratio $B = n$-EXP

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ allows for 0-tests on variables bounded by B

Gadget for ratio $B = n$-EXP

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ allows for 0-tests on variables bounded by B

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

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ allows for 0-tests on variables bounded by B

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

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ allows for 0-tests on variables bounded by B

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

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ allows for 0-tests on variables bounded by B

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

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ allows for 0-tests on variables bounded by B

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

$\mathsf{b} = B,\ \mathsf{c} \geq 0,\ \mathsf{d} = \mathsf{c} \cdot \mathsf{b}$ allows for 0-tests on variables bounded by B

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 $i \leq B$ stored in i, and i′ auxiliary (guaranteed to be 0)

How to use B-bounded 0-tests?

Let $i \leq B$ stored in i, and i$'$ auxiliary (guaranteed to be 0)

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

How to use B-bounded 0-tests?

Let $i \leq B$ stored in i, and i$'$ auxiliary (guaranteed to be 0)

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

loop
    x $+= 1$ i $-= 1$ i$' += 1$
test i $= 0$
loop
    i $+= 1$ i$' -= 1$
test i$' = 0$

How to use B-bounded 0-tests?

Let $i \leq B$ stored in i, and i$'$ auxiliary (guaranteed to be 0)

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

```
loop
    x += 1    i −= 1    i′ += 1
test i = 0
loop
    i += 1    i′ −= 1
test i′ = 0
```

How to use B-bounded 0-tests?

Let $i \leq B$ stored in i, and i' auxiliary (guaranteed to be 0)

- We want e.g.: $\mathsf{x} \mathrel{+}= \boxed{\mathsf{i}}$ or $\mathsf{x} \mathrel{-}= \boxed{\mathsf{i}}$

```
loop
    x += 1    i -= 1    i' += 1
test i = 0
loop
    i += 1    i' -= 1
test i' = 0
```

How to use B-bounded 0-tests?

Let $i \leq B$ stored in i, and i$'$ auxiliary (guaranteed to be 0)

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

loop
  x $+= 1$ i $-= 1$ i$' += 1$
test i $= 0$
loop
  i $+= 1$ i$' -= 1$
test i$' = 0$

- Or **loop at most** b **times** $<\mathit{body}>$

(b has no bound)

How to use B-bounded 0-tests?

Let $i \leq B$ stored in i, and i$'$ auxiliary (guaranteed to be 0)

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

loop
  x $+= 1$ i $-= 1$ i$'$ $+= 1$
test i $= 0$
loop
  i $+= 1$ i$'$ $-= 1$
test i$'$ $= 0$

- Or **loop at most** b **times** $<\!\mathit{body}\!>$

(b has no bound)

loop
  b $-= 1$ b$'$ $+= 1$
loop
  b$'$ $-= 1$ b $+= 1$
  $<\!\mathit{body}\!>$

High level description of the program

B – previous bound

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

High level description of the program

B – previous bound

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

Auxiliary variables $\mathsf{x}, \mathsf{y}, \mathsf{k}, \mathsf{i}$ (to check correctness)

High level description of the program

B – previous bound

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

Auxiliary variables $\mathsf{x}, \mathsf{y}, \mathsf{k}, \mathsf{i}$ (to check correctness)

$\mathsf{b} \mathrel{+}= 1, \quad \mathsf{k} \mathrel{+}= B$

loop

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

$\mathsf{i} \mathrel{+}= 1 \quad \mathsf{k} \mathrel{-}= 1$

<*main loop*>

loop

  $\mathsf{x} \mathrel{-}= \mathsf{i} \quad \mathsf{y} \mathrel{-}= 1$

halt if $\mathsf{y}, \mathsf{k} = 0$

High level description of the program

B – previous bound

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

Auxiliary variables $\mathsf{x}, \mathsf{y}, \mathsf{k}, \mathsf{i}$ (to check correctness)

$\mathsf{b} \mathrel{+}= 1, \quad \mathsf{k} \mathrel{+}= B$

loop

$\quad \mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= 1 \quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ $\longleftarrow$ $\mathsf{c}, \mathsf{d}, \mathsf{x}, \mathsf{y} := c \cdot B!$

$\mathsf{i} \mathrel{+}= 1 \quad \mathsf{k} \mathrel{-}= 1$

$<$*main loop*$>$

loop

$\quad \mathsf{x} \mathrel{-}= \mathsf{i} \quad \mathsf{y} \mathrel{-}= 1$

halt if $\mathsf{y}, \mathsf{k} = 0$

High level description of the program

B – previous bound

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

Auxiliary variables $\mathsf{x}, \mathsf{y}, \mathsf{k}, \mathsf{i}$ (to check correctness)

$\mathsf{b} \mathrel{+}= 1, \quad \mathsf{k} \mathrel{+}= B$

loop

$\quad \mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= 1 \quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ ⟵ $\mathsf{c}, \mathsf{d}, \mathsf{x}, \mathsf{y} := c \cdot B!$

$\mathsf{i} \mathrel{+}= 1 \quad \mathsf{k} \mathrel{-}= 1$

<*main loop*> ⟵ $\mathsf{c} := \mathsf{c}/(B-1)!, \quad \mathsf{d}, \mathsf{x} := d \cdot B, \quad \mathsf{b} := b \cdot B!, \quad \mathsf{k} = 0, \quad \mathsf{i} = B$

loop

$\quad \mathsf{x} \mathrel{-}= \mathsf{i} \quad \mathsf{y} \mathrel{-}= 1$

halt if $\mathsf{y}, \mathsf{k} = 0$

High level description of the program

B – previous bound

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

Auxiliary variables $\mathsf{x}, \mathsf{y}, \mathsf{k}, \mathsf{i}$ (to check correctness)

$\mathsf{b} \mathrel{+}= 1, \quad \mathsf{k} \mathrel{+}= B$
loop
$\quad \mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= 1 \quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ ⟵ $\mathsf{c}, \mathsf{d}, \mathsf{x}, \mathsf{y} := c \cdot B!$
$\mathsf{i} \mathrel{+}= 1 \quad \mathsf{k} \mathrel{-}= 1$
<main loop> ⟵ $\mathsf{c} := \mathsf{c}/(B-1)!, \quad \mathsf{d}, \mathsf{x} := d \cdot B, \quad \mathsf{b} := b \cdot B!, \quad \mathsf{k} = 0, \quad \mathsf{i} = B$
loop
$\quad \mathsf{x} \mathrel{-}= \mathsf{i} \quad \mathsf{y} \mathrel{-}= 1$
halt if $\mathsf{y}, \mathsf{k} = 0$

Invariants
$\mathsf{i} + \mathsf{k} = B, \quad \mathsf{b} \cdot \mathsf{c} = \mathsf{d}$

High level description of the program

B – previous bound

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

Auxiliary variables $\mathsf{x}, \mathsf{y}, \mathsf{k}, \mathsf{i}$ (to check correctness)

$\mathsf{b} \mathrel{+}= 1, \quad \mathsf{k} \mathrel{+}= B$
loop
$\quad \mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= 1 \quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{y} \mathrel{+}= 1$ ⟵ $\mathsf{c}, \mathsf{d}, \mathsf{x}, \mathsf{y} := c \cdot B!$
$\mathsf{i} \mathrel{+}= 1 \quad \mathsf{k} \mathrel{-}= 1$
<main loop> ⟵ $\mathsf{c} := \mathsf{c}/(B-1)!, \quad \mathsf{d}, \mathsf{x} := d \cdot B, \quad \mathsf{b} := b \cdot B!, \quad \mathsf{k} = 0, \quad \mathsf{i} = B$
loop
$\quad \mathsf{x} \mathrel{-}= \mathsf{i} \quad \mathsf{y} \mathrel{-}= 1$
halt if $\mathsf{y}, \mathsf{k} = 0$

Invariants
$\mathsf{i} + \mathsf{k} = B, \quad \mathsf{b} \cdot \mathsf{c} = \mathsf{d}$

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

The main loop

Invariants
$\mathsf{i} + \mathsf{k} = B, \quad \mathsf{b} \cdot \mathsf{c} = \mathsf{d}$

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

The main loop

Invariants

$\mathsf{i} + \mathsf{k} = B, \quad \mathsf{b} \cdot \mathsf{c} = \mathsf{d}$

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

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

The main loop

Invariants

$\mathsf{i} + \mathsf{k} = B, \quad \mathsf{b} \cdot \mathsf{c} = \mathsf{d}$

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

$\mathsf{c}' := \mathsf{c} \cdot \frac{1}{\mathsf{i}},\ \mathsf{d}' := \mathsf{d} \cdot \frac{\mathsf{i}+1}{\mathsf{i}}$

loop

  loop

    $\mathsf{c} \mathrel{-}= \boxed{\mathsf{i}} \quad \mathsf{c}' \mathrel{+}= 1$

    loop at most b **times**

      $\mathsf{d} \mathrel{-}= \boxed{\mathsf{i}} \quad \mathsf{d}' \mathrel{+}= \boxed{\mathsf{i}+1} \quad \mathsf{x} \mathrel{-}= \boxed{\mathsf{i}} \quad \mathsf{x}' \mathrel{+}= \boxed{\mathsf{i}+1}$

  loop

    $\mathsf{b} \mathrel{-}= 1 \quad \mathsf{b}' \mathrel{+}= \boxed{\mathsf{i}+1}$

  loop

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

  loop

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

    loop at most b **times**

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

  $\mathsf{k} \mathrel{-}= 1 \quad \mathsf{i} \mathrel{+}= 1$

The main loop

Invariants

$\mathsf{i} + \mathsf{k} = B, \quad \mathsf{b} \cdot \mathsf{c} = \mathsf{d}$

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

loop

 loop

  $\mathsf{c} \mathrel{-}= \boxed{\mathsf{i}} \quad \mathsf{c}' \mathrel{+}= 1$

  loop at most b **times**

   $\mathsf{d} \mathrel{-}= \boxed{\mathsf{i}} \quad \mathsf{d}' \mathrel{+}= \boxed{\mathsf{i}+1} \quad \mathsf{x} \mathrel{-}= \boxed{\mathsf{i}} \quad \mathsf{x}' \mathrel{+}= \boxed{\mathsf{i}+1}$

 loop

  $\mathsf{b} \mathrel{-}= 1 \quad \mathsf{b}' \mathrel{+}= \boxed{\mathsf{i}+1}$

 loop

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

 loop

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

  loop at most b **times**

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

 $\mathsf{k} \mathrel{-}= 1 \quad \mathsf{i} \mathrel{+}= 1$

$\mathsf{c}' := \mathsf{c} \cdot \frac{1}{\mathsf{i}}, \ \mathsf{d}' := \mathsf{d} \cdot \frac{\mathsf{i}+1}{\mathsf{i}}$

$\mathsf{b}' := \mathsf{b} \cdot (\mathsf{i}+1)$

The main loop

Invariants

$\mathsf{i} + \mathsf{k} = B, \quad \mathsf{b} \cdot \mathsf{c} = \mathsf{d}$

$\prod_{i=1}^{k-1} \frac{i+1}{i} \;=\; k$

loop

  loop

    $\mathsf{c} \mathrel{-}= \boxed{\mathsf{i}} \quad \mathsf{c}' \mathrel{+}= 1$

    loop at most b **times**

      $\mathsf{d} \mathrel{-}= \boxed{\mathsf{i}} \quad \mathsf{d}' \mathrel{+}= \boxed{\mathsf{i}+1} \quad \mathsf{x} \mathrel{-}= \boxed{\mathsf{i}} \quad \mathsf{x}' \mathrel{+}= \boxed{\mathsf{i}+1}$

  loop

    $\mathsf{b} \mathrel{-}= 1 \quad \mathsf{b}' \mathrel{+}= \boxed{\mathsf{i}+1}$

  loop

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

  loop

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

    loop at most b **times**

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

  $\mathsf{k} \mathrel{-}= 1 \quad \mathsf{i} \mathrel{+}= 1$

$\mathsf{c}' := \mathsf{c} \cdot \frac{1}{\mathsf{i}}, \ \mathsf{d}' := \mathsf{d} \cdot \frac{\mathsf{i}+1}{\mathsf{i}}$

$\mathsf{b}' := \mathsf{b} \cdot (\mathsf{i}+1)$

if any **loop** not maximal then $\mathsf{x} < \mathsf{y} \cdot B$

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. . .