# 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

DIMAP seminar
February 2019

# 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


DIMAP seminar
February 2019

# 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

DIMAP seminar
February 2019

# Introduction

Petri Nets, VASS, programs with no zero tests

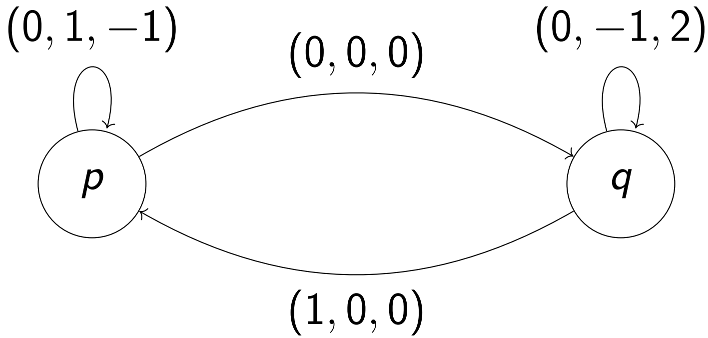
## Vector addition systems with states (VASS)

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

# Vector addition systems with states (VASS)

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

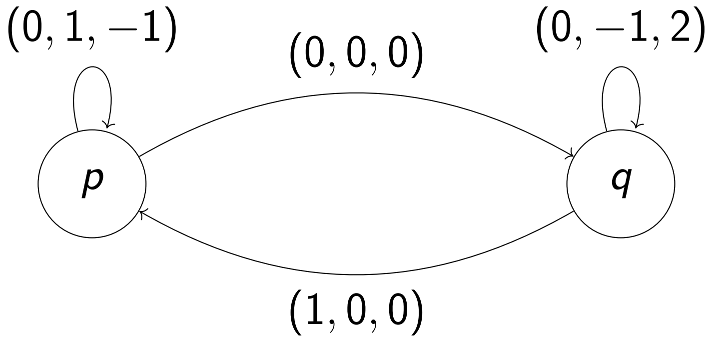
Example: $d = 3$, $Q = \{p, q\}$



# Vector addition systems with states (VASS)

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

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

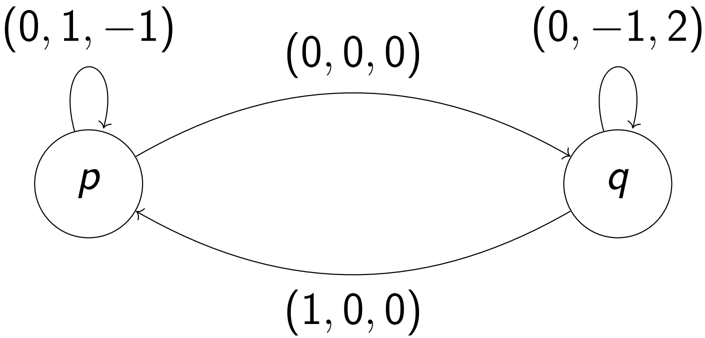



Configurations $p(\mathbf{v}) = (p, \mathbf{v}) \in Q \times \mathbb{N}^d$

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

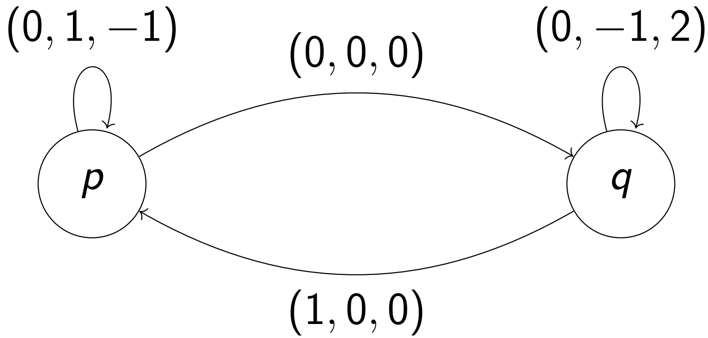
Example run:

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

# 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:

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

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

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 $\mathbf{v}' \geq \mathbf{v}$?

# 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

# Counter programs (with or without zero tests)

## Counter programs (with or without zero tests)

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

# Counter programs (with or without zero tests)

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

# Counter programs (with or without zero tests)

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

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

All variables are initialized to 0, and are never negative

Example

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

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

# Programs with no zero test = VASS

# Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

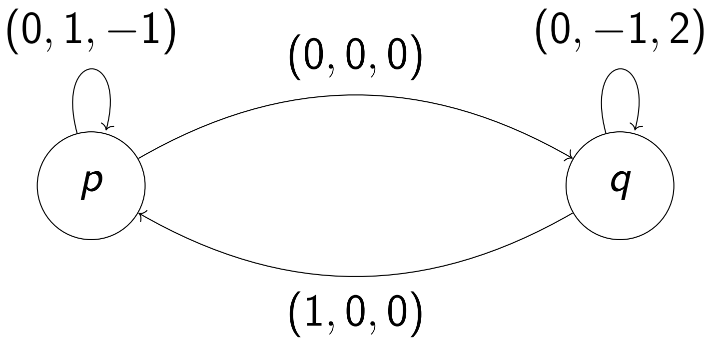
DECIDE: Does it have a complete run (executing **halt**)?

# 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**)?

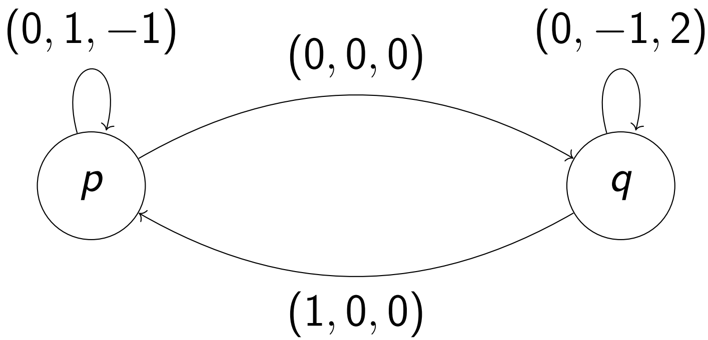


## Programs with no zero test = VASS

Reachability problem (for programs):

GIVEN: A counter program with no zero tests.

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



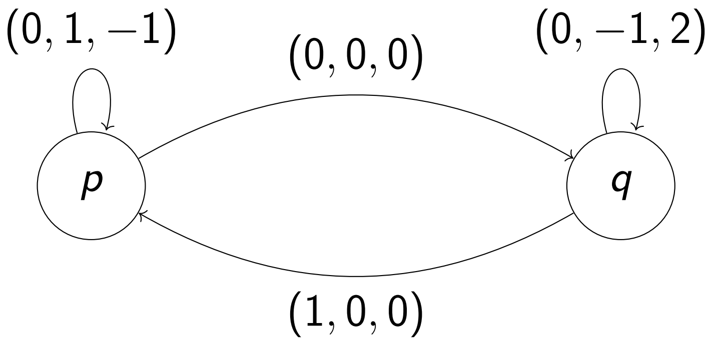

$p(0, 0, 1) \rightarrow^* p(1, 0, 2)$?

# 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**)?




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

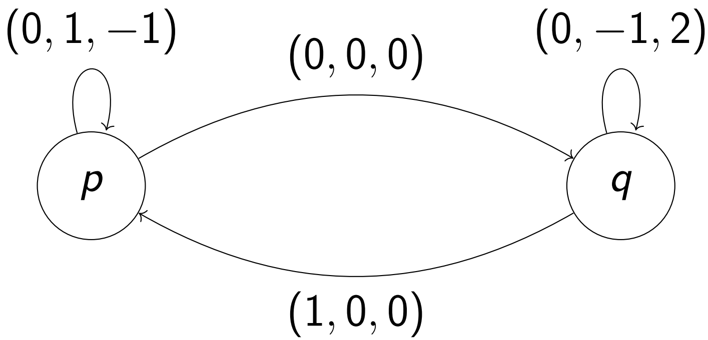
```
z += 1
loop
    loop
        y += 1    z −= 1
    loop
        y −= 1    z += 2
    x += 1
x −= 1    z −= 2
halt if x, y, z = 0.
```

# 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**)?




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

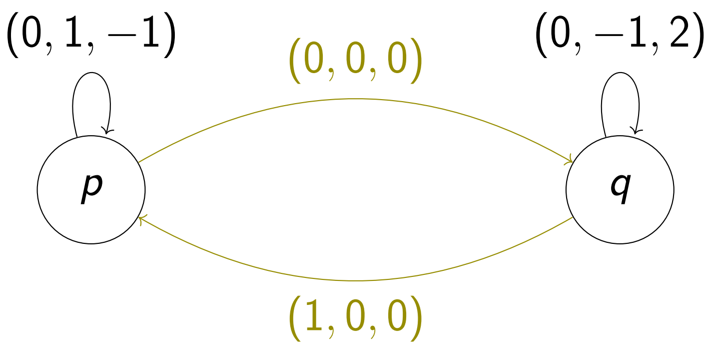
```
z += 1
loop
    loop
        y += 1    z −= 1
    loop
        y −= 1    z += 2
    x += 1
x −= 1    z −= 2
halt if x, y, z = 0.
```

# 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**)?




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

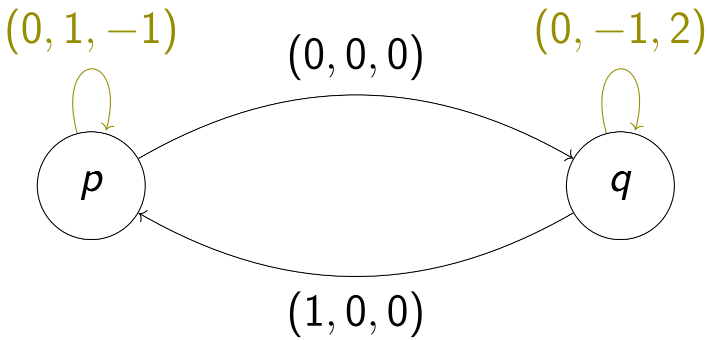
```
z += 1
loop
    loop
        y += 1    z −= 1
    loop
        y −= 1    z += 2
    x += 1
x −= 1    z −= 2
halt if x, y, z = 0.
```

# 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**)?




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

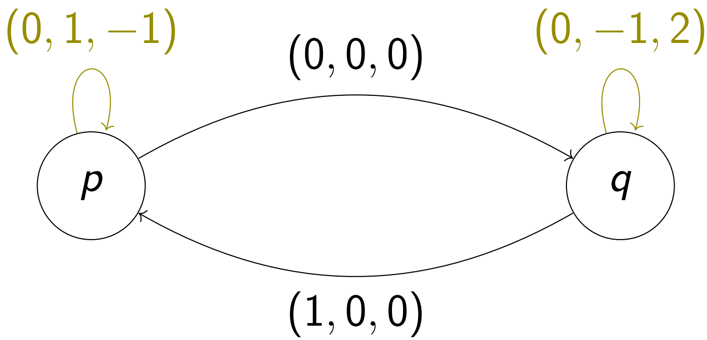
z += 1
**loop**
  **loop**
    y += 1  z −= 1
  **loop**
    y −= 1  z += 2
  x += 1
x −= 1  z −= 2
**halt if** x, y, z = 0.

# 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**)?




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

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) |
| 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

## 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** $x = 0$

# Programs with zero tests

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

Reachability becomes undecidable

# Programs with zero tests

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

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

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

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

Lipton encoded programs for $f = 2$-EXP

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

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

Input: programs with zero tests, s.t. counters bounded by $B$

## Encoding programs with zero tests and bounded counters

Input: programs with zero tests, s.t. counters bounded by $B$

We encode this into programs with no zero tests

# Encoding programs with zero tests and bounded counters

Input: programs with zero tests, s.t. counters bounded by $B$

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

Input: programs with zero tests, s.t. counters bounded by $B$

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 $\mathsf{x}_i$ add $\mathsf{x}_i'$

## Encoding programs with zero tests and bounded counters

Input: programs with zero tests, s.t. counters bounded by $B$

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 $\mathsf{x}_i$ add $\mathsf{x}_i'$

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

## Encoding programs with zero tests and bounded counters

Input: programs with zero tests, s.t. counters bounded by $B$

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 $\mathsf{x}_i$ add $\mathsf{x}_i'$

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

**loop**

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

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

## Encoding programs with zero tests and bounded counters

Input: programs with zero tests, s.t. counters bounded by $B$

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 $\mathsf{x}_i$ add $\mathsf{x}'_i$

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

## Encoding programs with zero tests and bounded counters

Input: programs with zero tests, s.t. counters bounded by $B$

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 $\mathsf{x}_i$ add $\mathsf{x}'_i$

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

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

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

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

## 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**

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

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

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

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

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

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

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

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

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

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

$x'_i = B - x_i$ ← holds because $b = 0$

Replace **test** $x_i = 0$ with

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

so a false zero test implies $d \neq 0$

This is the challenge

Extend **halt** with $b, d = 0$

# 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

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

If $B$ is fixed, just start the program with:

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

**loop**

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

# Recall what we wanted

$B$ – bound on the counters

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

If $B$ is fixed, just start the program with:

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

**loop**

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

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:

$\mathsf{b} \mathrel{+}= B$ ← “gadget for ratio $B$”
**loop**
    $\mathsf{c} \mathrel{+}= 1 \quad \mathsf{d} \mathrel{+}= B$

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

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

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

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

## 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

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

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

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

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

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

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.: $\mathsf{x} \mathrel{+}= \boxed{\mathsf{i}}$

```
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 $\mathsf{i}$, and $\mathsf{i}'$ auxiliary (guaranteed to be 0)

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

**loop**
$\quad \mathsf{x} \mathrel{+}= 1 \quad \mathsf{i} \mathrel{-}= 1 \quad \mathsf{i}' \mathrel{+}= 1$
**test** $\mathsf{i} = 0$
**loop**
$\quad \mathsf{i} \mathrel{+}= 1 \quad \mathsf{i}' \mathrel{-}= 1$
**test** $\mathsf{i}' = 0$

## How to use $B$-bounded $0$-tests?

Let $i \le B$ stored in $\mathsf{i}$, and $\mathsf{i}'$ auxiliary (guaranteed to be 0)

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

**loop**
    $\mathsf{x} \mathrel{+}= 1 \quad \mathsf{i} \mathrel{-}= 1 \quad \mathsf{i}' \mathrel{+}= 1$
**test** $\mathsf{i} = 0$
**loop**
    $\mathsf{i} \mathrel{+}= 1 \quad \mathsf{i}' \mathrel{-}= 1$
**test** $\mathsf{i}' = 0$

- 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** $<$*body*$>$

(b has no bound)

**loop**
    b $-=$ 1    b$'$ $+=$ 1
**loop**
    b$'$ $-=$ 1    b $+=$ 1
    $<$*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: $\mathrm{b} = B!,\ \mathrm{c} \geq 0,\ \mathrm{d} = \mathrm{c} \cdot \mathrm{b}$

Auxiliary variables $\mathrm{x}, \mathrm{y}, \mathrm{k}, \mathrm{i}$ (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**

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

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

$<$*main loop*$>$

**loop**

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

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

# High level description of the program

$B$ – previous bound

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

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

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

**loop**

$\quad \mathrm{c} \mathrel{+}= 1 \quad \mathrm{d} \mathrel{+}= 1 \quad \mathrm{x} \mathrel{+}= 1 \quad \mathrm{y} \mathrel{+}= 1$ ⟵ $\mathrm{c}, \mathrm{d}, \mathrm{x}, \mathrm{y} := c \cdot (B-1)!$

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

$<$*main loop*$>$

**loop**

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

**halt if** $\mathrm{y}, \mathrm{k} = 0$

## High level description of the program

$B$ – previous bound

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

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

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

**loop**

$\quad \mathrm{c} \mathrel{+}= 1 \quad \mathrm{d} \mathrel{+}= 1 \quad \mathrm{x} \mathrel{+}= 1 \quad \mathrm{y} \mathrel{+}= 1$ ⟵ $\mathrm{c}, \mathrm{d}, \mathrm{x}, \mathrm{y} := c \cdot (B-1)!$

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

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

**loop**

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

**halt if** $\mathrm{y}, \mathrm{k} = 0$

# 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)

b += 1,  k += $B$

**loop**

  c += 1  d += 1  x += 1  y += 1 ⟵ $\mathsf{c}, \mathsf{d}, \mathsf{x}, \mathsf{y} := c \cdot (B-1)!$

i += 1  k −= 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**

  x −= i  y −= 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**

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

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

<*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

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

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

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

# The main loop

Invariants

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

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

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

**loop**

  **loop**

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

    **loop at most** $b$ **times**

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

  **loop**

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

  **loop**

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

  **loop**

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

    **loop at most** $b$ **times**

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

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

## The main loop

Invariants

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

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

**loop**

    **loop**

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

        **loop at most** b **times**

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

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

    **loop**

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

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

    **loop**

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

    **loop**

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

        **loop at most** b **times**

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

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

# The main loop

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

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

**loop**

  **loop**

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

    **loop at most** $b$ **times**

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

  **loop**

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

  **loop**

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

  **loop**

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

    **loop at most** $b$ **times**

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

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

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

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

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

# Conclusion

## Conclusion

- Several applications and corollaries
  - coverability is different from reachability

## Conclusion

- Several applications and corollaries
  - coverability is different from reachability
  - satisfiability of FO2 on data words

# Conclusion

- Several applications and corollaries
  - coverability is different from reachability
  - satisfiability of FO2 on data words

- We can do $h$-EXPSPACE-hardness in dimension $h + 13$ (so fixed)

## Conclusion

- Several applications and corollaries
  - coverability is different from reachability
  - satisfiability of FO2 on data words

- We can do $h$-EXPSPACE-hardness in dimension $h + 13$ (so fixed)

Can we do Tower in fixed dimension?

# Conclusion

- Several applications and corollaries
  - coverability is different from reachability
  - satisfiability of FO2 on data words

- We can do $h$-EXPSPACE-hardness in dimension $h + 13$ (so fixed)

Can we do Tower in fixed dimension?

- The complexity is quite tight

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

# Conclusion

- Several applications and corollaries
  - coverability is different from reachability
  - satisfiability of FO2 on data words

- We can do $h$-EXPSPACE-hardness in dimension $h + 13$ (so fixed)

Can we do Tower in fixed dimension?

- The complexity is quite tight

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

(Don't tell Jérôme I wrote this)

# Conclusion

- Several applications and corollaries
  - coverability is different from reachability
  - satisfiability of FO2 on data words

- We can do $h$-EXPSPACE-hardness in dimension $h + 13$ (so fixed)

Can we do Tower in fixed dimension?

- The complexity is quite tight

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

(Don't tell Jérôme I wrote this)

- This originated from studying 1-Pushdown-VASS

# Conclusion

- Several applications and corollaries
  - coverability is different from reachability
  - satisfiability of FO2 on data words

- We can do $h$-EXPSPACE-hardness in dimension $h + 13$ (so fixed)

Can we do Tower in fixed dimension?

- The complexity is quite tight

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

(Don't tell Jérôme I wrote this)

- This originated from studying 1-Pushdown-VASS

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