Targeting Completeness: Using Closed Forms for Size Bounds of Integer Programs

14th International Symposium on Frontiers of Combining Systems

Nils Lommen and Jürgen Giesl

Motivation

Goal: Infer (upper) size and time bounds for "real-world" programs

$$\texttt{while } (\mathtt{x}_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ \mathtt{x}_2 + \mathtt{x}_1^2 \end{bmatrix}$$
$$\texttt{end}$$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Motivation

Goal: Infer (upper) size and time bounds for "real-world" programs

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ x_2 + x_1^2 \end{bmatrix}$$
$$\texttt{end}$$

- How large are the variables?

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Motivation

Goal: Infer (upper) size and time bounds for "real-world" programs

$$
\begin{aligned}
&\mathtt{while}\ (x_1 > 0)\ \mathtt{do}\\
&\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ x_2 + x_1^2 \end{bmatrix}\\
&\mathtt{end}\\
&\mathtt{while}\ (x_2 > 0)\ \mathtt{do}\\
&\quad \begin{bmatrix} x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_2 - 1 \end{bmatrix}\\
&\mathtt{end}
\end{aligned}
$$

- How large are the variables?

Motivation

Goal: Infer (upper) size and time bounds for "real-world" programs

$$
\begin{array}{l}
\texttt{while}\ (x_1 > 0)\ \texttt{do} \\
\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ x_2 + x_1^2 \end{bmatrix} \\
\texttt{end} \\
\texttt{while}\ (x_2 > 0)\ \texttt{do} \\
\quad \begin{bmatrix} x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_2 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- How large are the variables?
- How often do we execute the second loop?

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Motivation

Goal: Infer (upper) size and time bounds for "real-world" programs

$$
\begin{array}{l}
\texttt{while}\ (\mathtt{x}_1 > 0)\ \texttt{do} \\
\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ \mathtt{x}_2 + \mathtt{x}_1^2 \end{bmatrix} \\
\texttt{end} \\
\texttt{while}\ (\mathtt{x}_2 > 0)\ \texttt{do} \\
\quad \begin{bmatrix} \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_2 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- How large are the variables?
- How often do we execute the second loop?
  - Maximal "size" of $\mathtt{x}_2$ times

Motivation

Goal: Infer (upper) size and time bounds for "real-world" programs

$$\texttt{while } (x_1 > 0) \texttt{ do} \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ x_2 + x_1^2 \end{bmatrix} \quad \texttt{end}$$

$$\texttt{while } (x_2 > 0) \texttt{ do} \quad \begin{bmatrix} x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_2 - 1 \end{bmatrix} \quad \texttt{end}$$

- How large are the variables?
- How often do we execute the second loop?
  - Maximal "size" of x_2 times
  - Existing tools usually fail with non-linear arithmetic.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Motivation

Goal: Infer (upper) size and time bounds for "real-world" programs

$$\texttt{while } (x_1 > 0) \texttt{ do} \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ x_2 + x_1^2 \end{bmatrix} \quad \texttt{end}$$

$$\texttt{while } (x_2 > 0) \texttt{ do} \quad \begin{bmatrix} x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_2 - 1 \end{bmatrix} \quad \texttt{end}$$

- How large are the variables?
- How often do we execute the second loop?
  - Maximal "size" of x_2 times
  - Existing tools usually fail with non-linear arithmetic.
  - Can compute non-linear size and time bounds for prs loops.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Motivation

Goal: Infer (upper) size and time bounds for "real-world" programs

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ x_2 + x_1^2 \end{bmatrix}$$
$$\texttt{end}$$
$$\texttt{while } (x_2 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_2 - 1 \end{bmatrix}$$
$$\texttt{end}$$

- How large are the variables?
- How often do we execute the second loop?
  - Maximal "size" of x_2 times
  - Existing tools usually fail with non-linear arithmetic.
  - Can compute non-linear size and time bounds for prs loops.
  - Approach is complete for a large class of programs.

Motivation

Goal: Infer (upper) size and time bounds for "real-world" programs

$$\begin{array}{l}\texttt{while}\ (x_1 > 0)\ \texttt{do}\\ \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ x_2 + x_1^2 \end{bmatrix}\\ \texttt{end}\\ \texttt{while}\ (x_2 > 0)\ \texttt{do}\\ \quad \begin{bmatrix} x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_2 - 1 \end{bmatrix}\\ \texttt{end}\end{array}$$

- How large are the variables?
- How often do we execute the second loop?
  - Maximal "size" of x_2 times
  - Existing tools usually fail with non-linear arithmetic.
  - Can compute non-linear size and time bounds for prs loops.
  - Approach is complete for a large class of programs.

- Size bound computations are implemented in the automatic complexity analysis tool KoAT

Overview

Goal: Infer (upper) size and time bounds for "real-world" programs

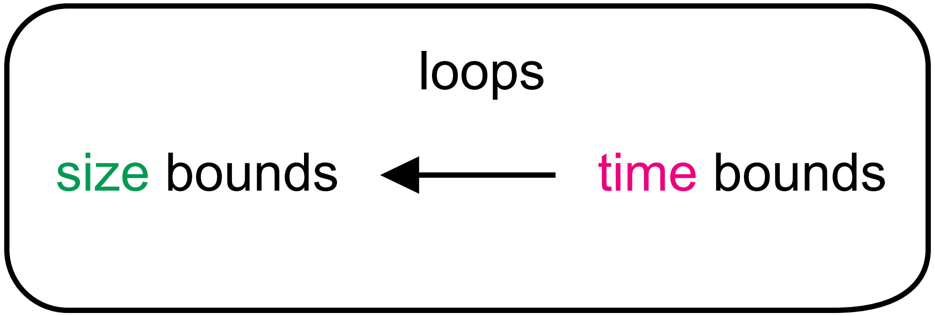
loops

size bounds

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

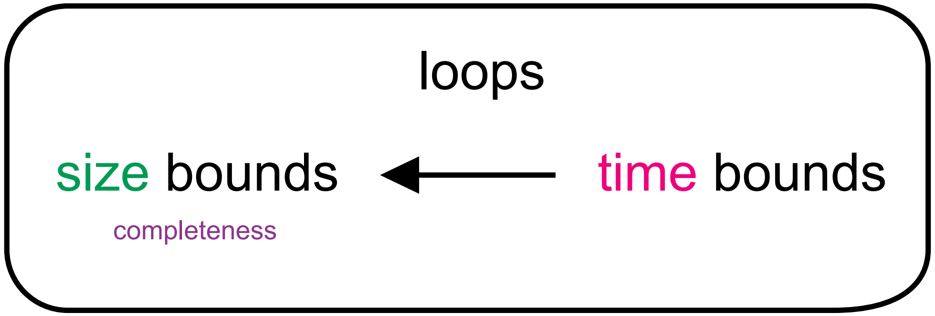
Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

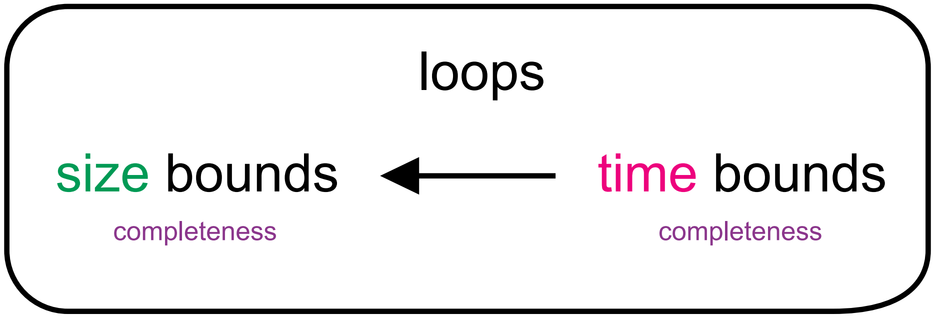
Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

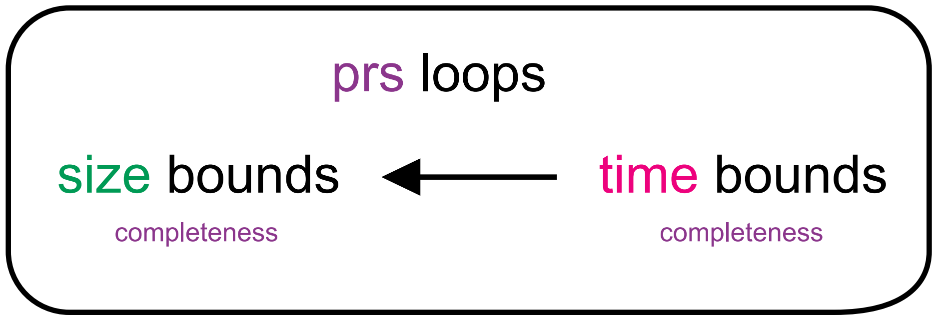
Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

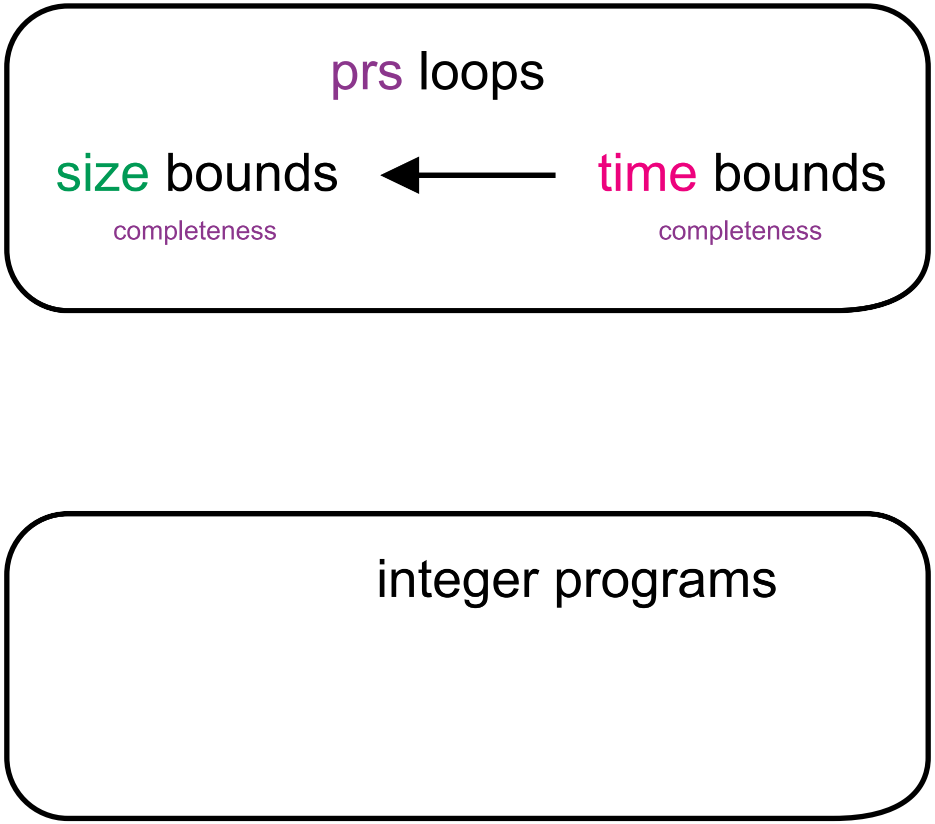
Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

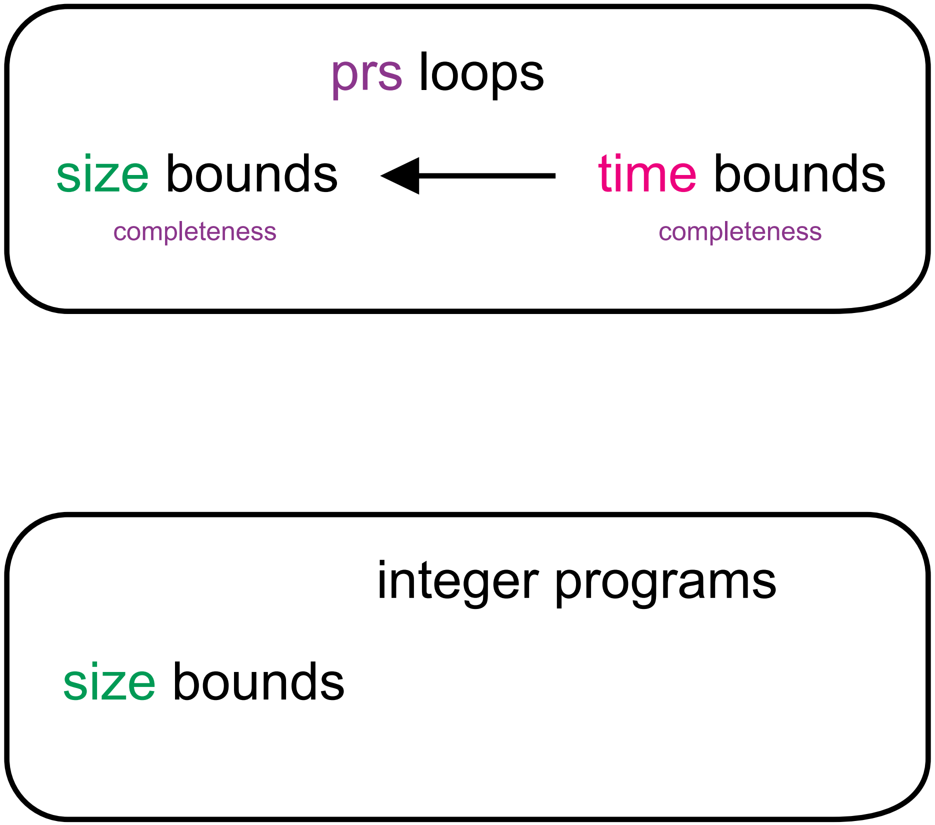
Goal: Infer (upper) size and time bounds for "real-world" programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

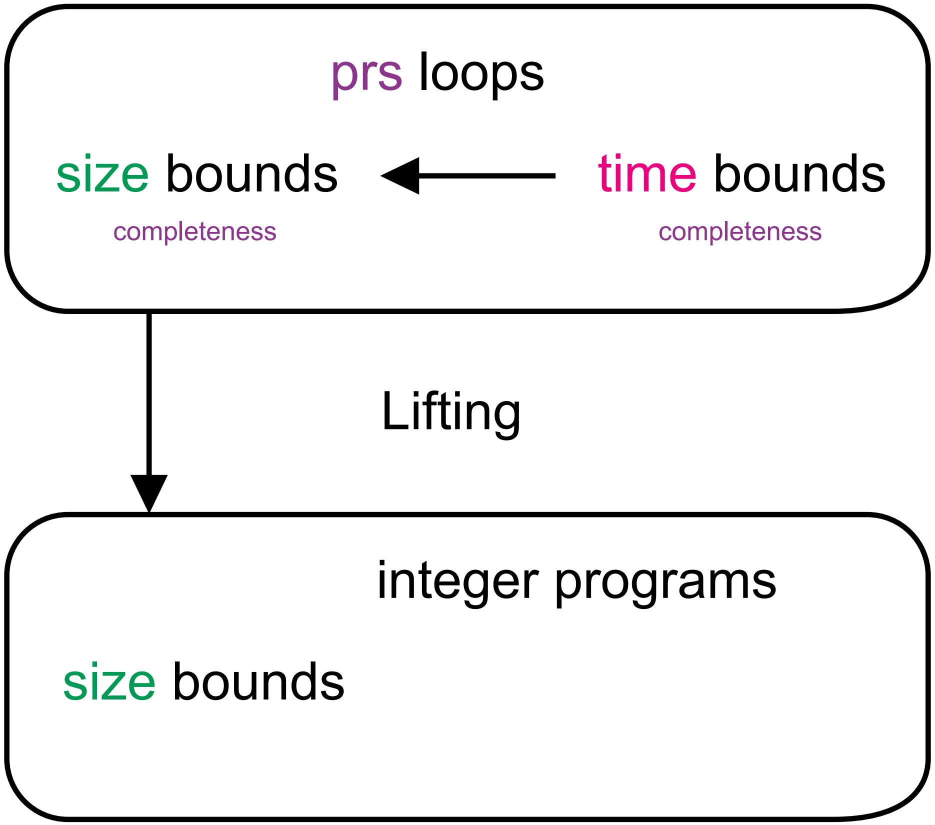
Goal: Infer (upper) size and time bounds for “real-world” programs



FroCoS ’23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

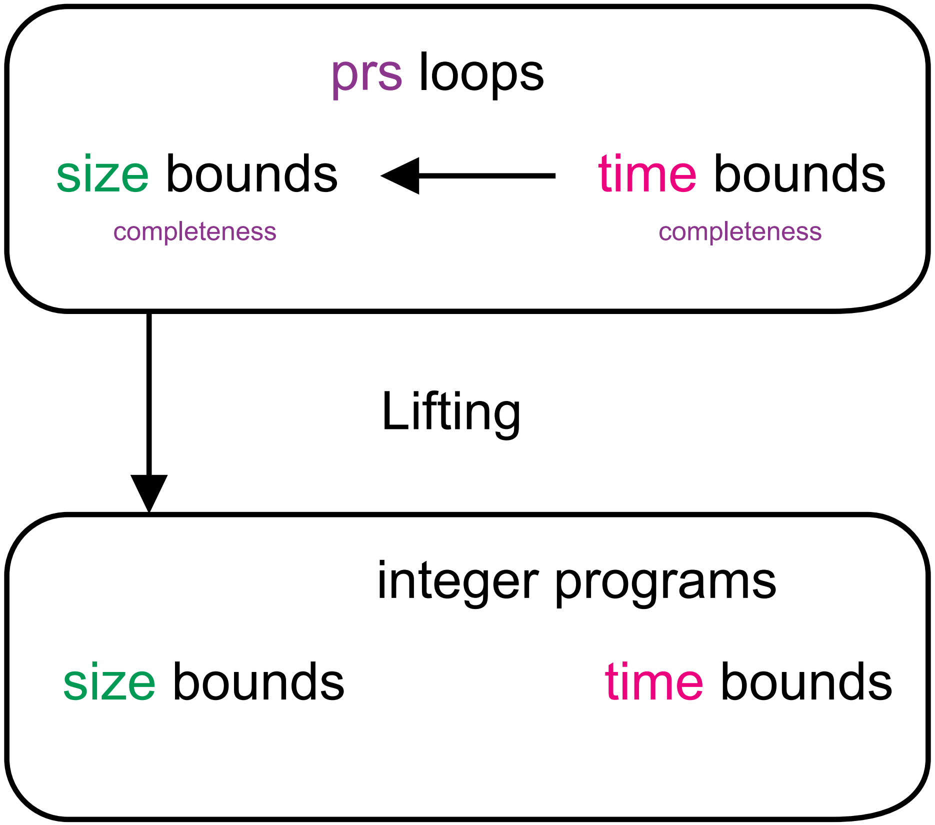
Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

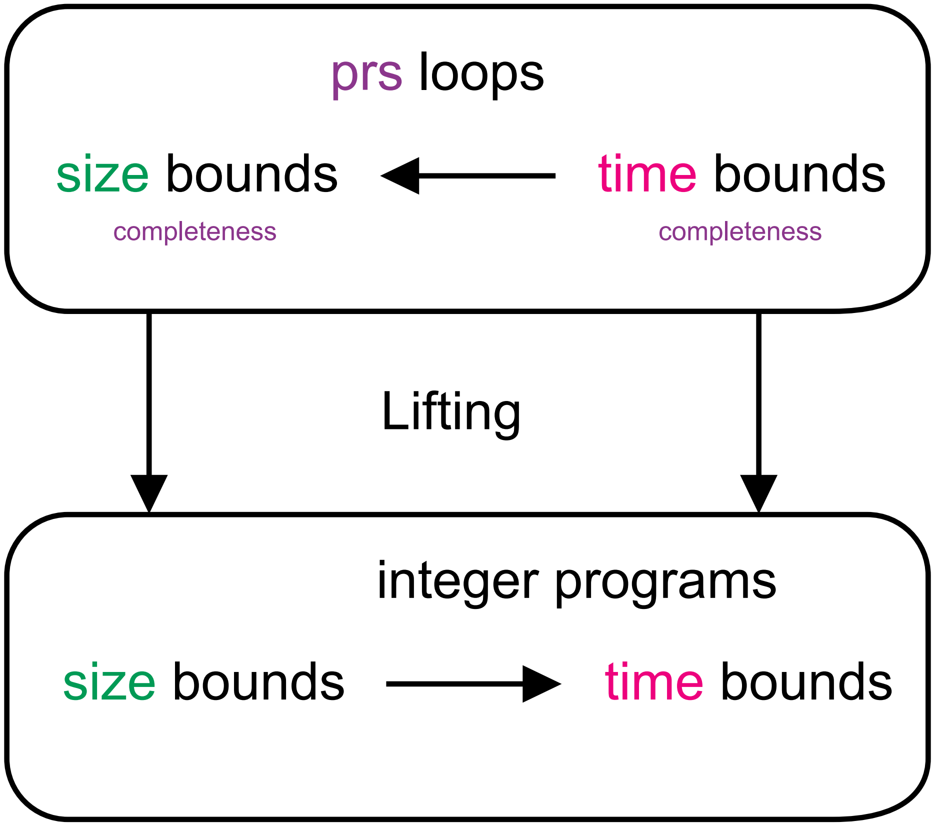
Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

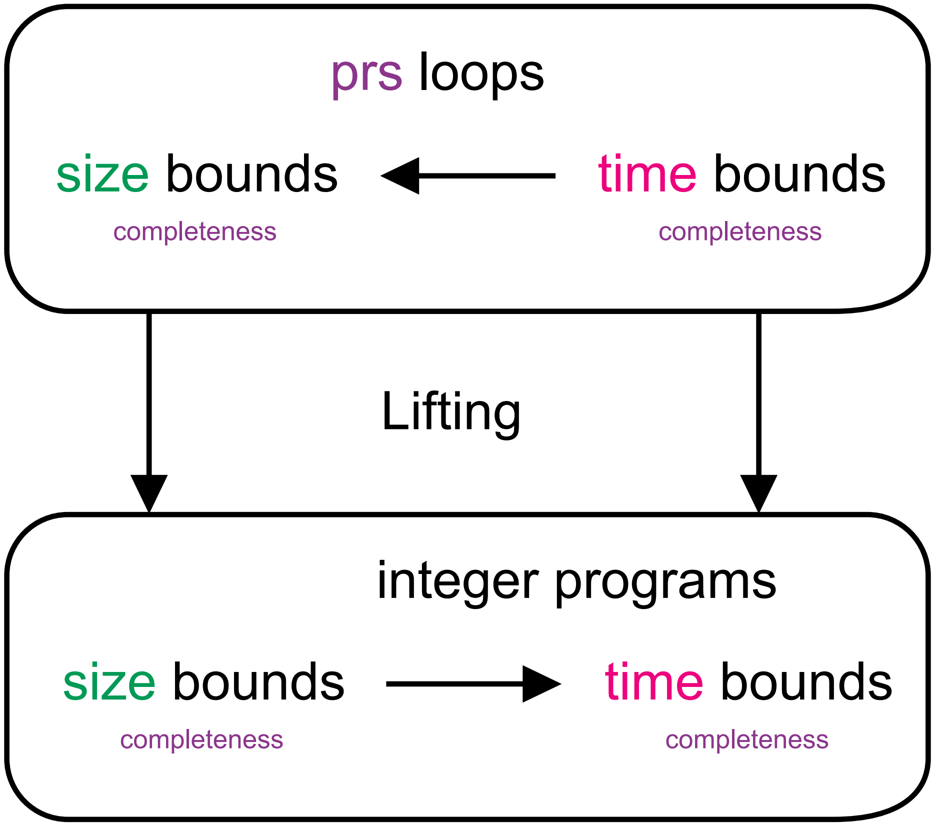
Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

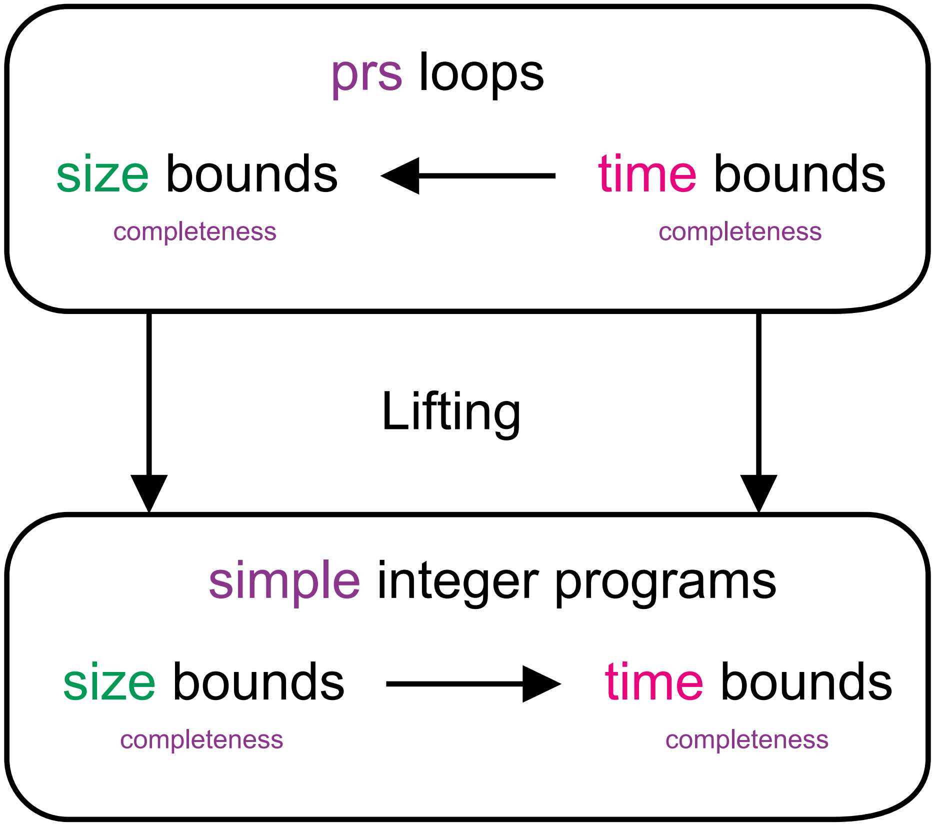
Goal: Infer (upper) size and time bounds for "real-world" programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

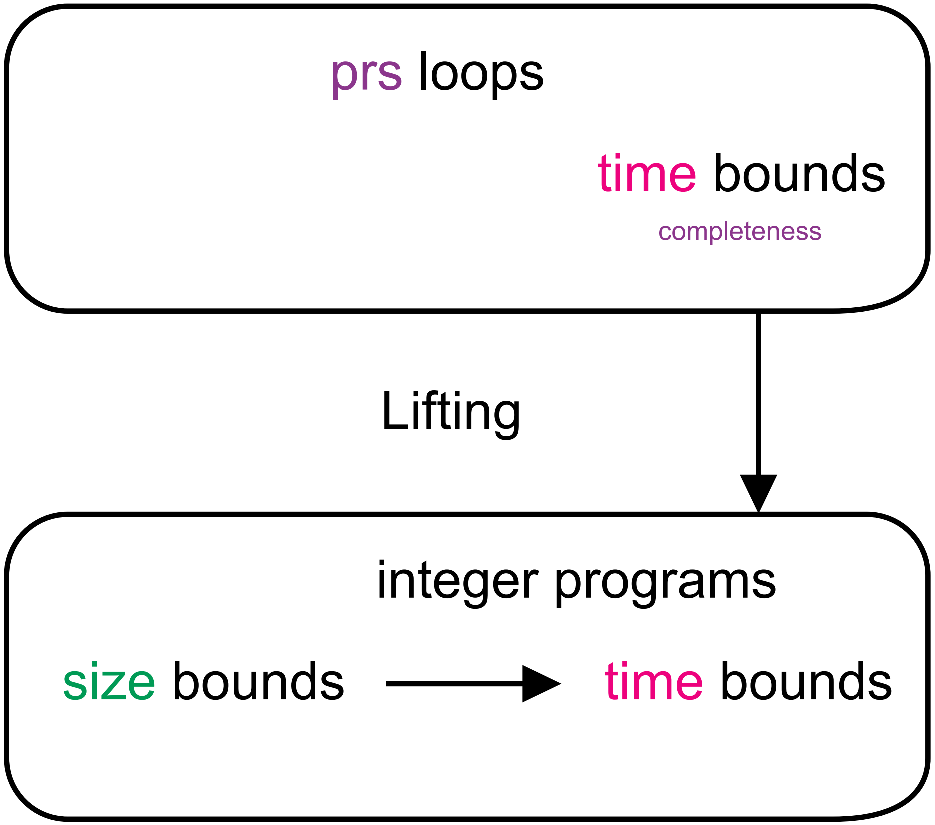
Overview

Goal: Infer (upper) size and time bounds for “real-world” programs



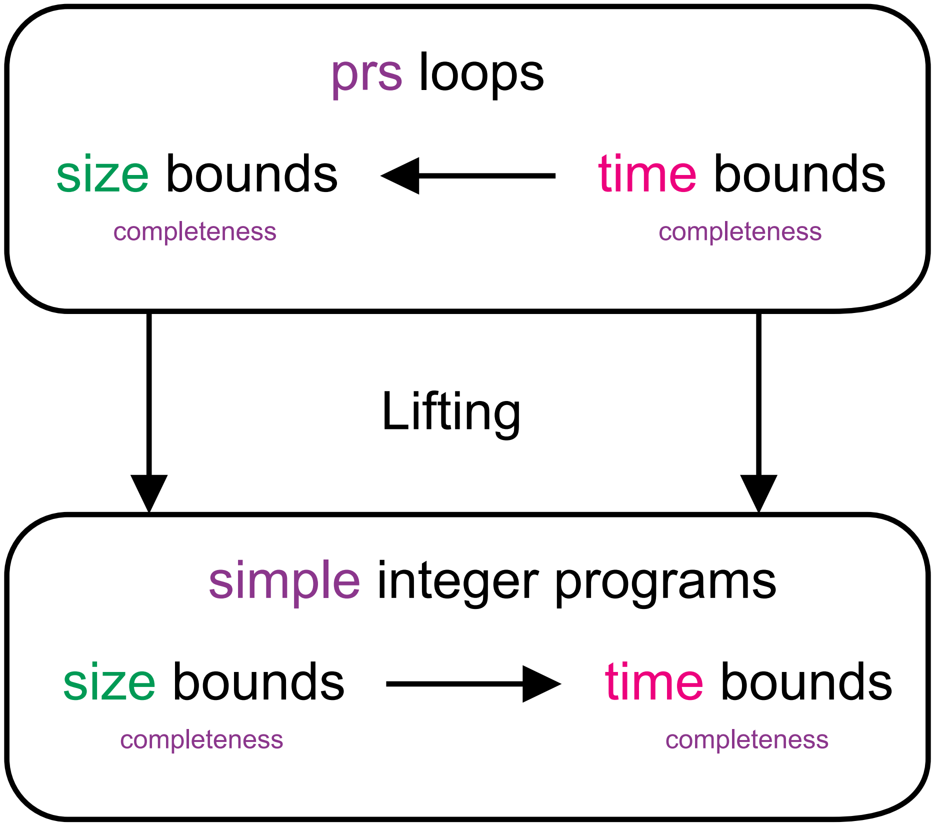
Overview

Goal: Infer (upper) size and time bounds for "real-world" programs



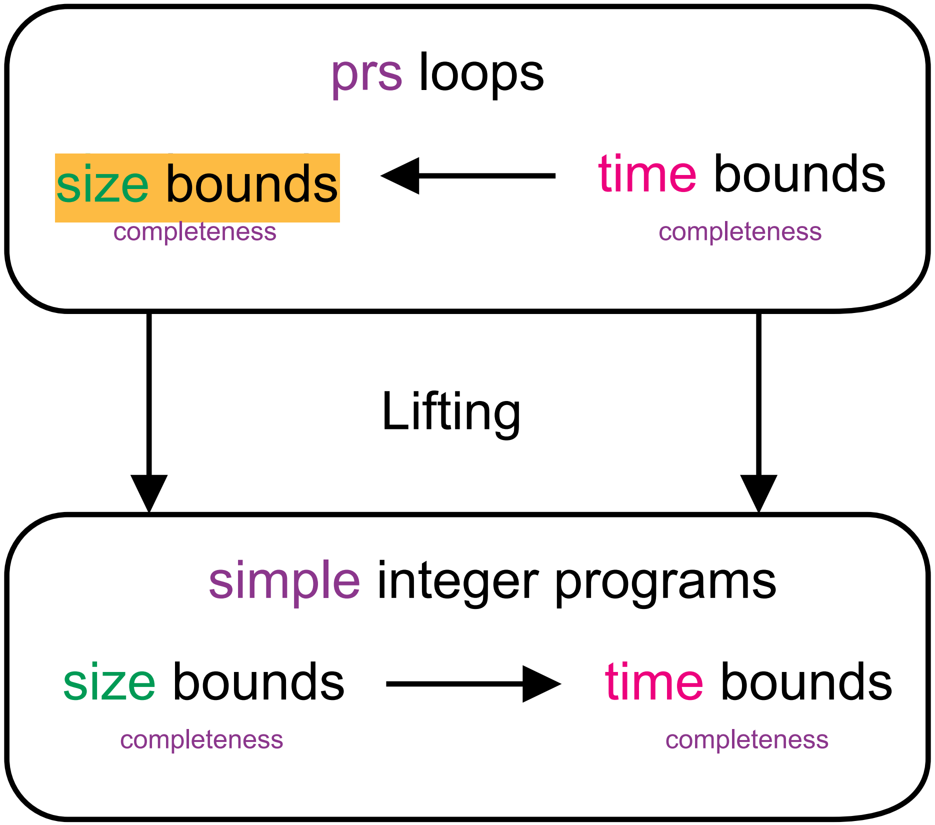
Overview

Goal: Infer (upper) size and time bounds for "real-world" programs



Overview

Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

$$\mathtt{while}\ (\mathtt{x}_1 > 0)\ \mathtt{do}$$

$$\begin{bmatrix}\mathtt{x}_1 \\ \mathtt{x}_2\end{bmatrix} \leftarrow \begin{bmatrix}\mathtt{x}_1 - 1 \\ \mathtt{x}_2 + \mathtt{x}_1^2\end{bmatrix}$$

$$\mathtt{end}$$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x1 > 0) do
```

$$\begin{bmatrix}\mathtt{x}_1\\ \mathtt{x}_2\end{bmatrix} \leftarrow \begin{bmatrix}\mathtt{x}_1 - 1\\ \mathtt{x}_2 + \mathtt{x}_1^2\end{bmatrix}$$

```
end
```

- Compute closed form for $\mathtt{x}_1$.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

$$\texttt{while } (\mathtt{x}_1 > 0) \texttt{ do} \quad \begin{bmatrix}\mathtt{x}_1\\ \mathtt{x}_2\end{bmatrix} \leftarrow \begin{bmatrix}\mathtt{x}_1 - 1\\ \mathtt{x}_2 + \mathtt{x}_1^2\end{bmatrix} \quad \texttt{end}$$

- Compute closed form for $\mathtt{x}_1$.

- Closed form: $\mathtt{cl}_{x_1}^{n} = x_1 - n$

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ \mathtt{x}_2 + \mathtt{x}_1^2 \end{bmatrix}$$

```
end
```

- Compute closed form for $\mathtt{x}_1$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.

- Closed form: $\mathtt{cl}_{x_1}^{n} = x_1 - n$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
    [x₁]    [x₁ - 1   ]
    [x₂] ← [x₂ + x₁² ]
end
```

- Compute closed form for $\mathtt{x}_1$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.

- Closed form: $\mathtt{cl}_{x_1}^{n} = x_1 - n$
- Over-approximation: $x_1 + n$

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
```
$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ \mathtt{x}_2 + \mathtt{x}_1^2 \end{bmatrix}$$
```
end
```

- Compute closed form for $\mathtt{x}_1$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}_{x_1}^{n} = x_1 - n$
- Over-approximation: $x_1 + n$
- Size bound:

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
```
$$\begin{bmatrix}\mathtt{x}_1\\ \mathtt{x}_2\end{bmatrix} \leftarrow \begin{bmatrix}\mathtt{x}_1 - 1\\ \mathtt{x}_2 + \mathtt{x}_1^2\end{bmatrix}$$
```
end
```

- Compute closed form for $\mathtt{x}_1$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}_{x_1}^{n} = x_1 - n$
- Over-approximation: $x_1 + n$
- Size bound: $x_1 + x_1$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
    [x₁]  ←  [x₁ − 1   ]
    [x₂]     [x₂ + x₁² ]
end
```

- Compute closed form for $\mathtt{x}_1$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}_{x_1}^{n} = x_1 - n$
- Over-approximation: $x_1 + n$
- Size bound: $x_1 + x_1 = 2 \cdot x_1$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
   [x₁]    [x₁ − 1    ]
   [x₂] ← [x₂ + x₁²  ]
end
```

- Compute closed form for $\mathtt{x}_1$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}_{x_1}^{n} = x_1 - n$
- Over-approximation: $x_1 + n$
- Size bound: $x_1 + x_1 = 2 \cdot x_1$

$\Rightarrow$ for an initial configuration $x_1 = -5$:

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
    [x₁]    [x₁ − 1    ]
    [x₂] ←  [x₂ + x₁²  ]
end
```

- Compute closed form for $\mathtt{x}_1$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}_{x_1}^{n} = x_1 - n$
- Over-approximation: $x_1 + n$
- Size bound: $x_1 + x_1 = 2 \cdot x_1$

$\Rightarrow$ for an initial configuration $x_1 = -5$: $2 \cdot |-5| = 10$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

$$\texttt{while } (\mathtt{x}_1 > 0) \texttt{ do} \quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ \mathtt{x}_2 + \mathtt{x}_1^2 \end{bmatrix} \quad \texttt{end}$$

- Compute closed form for $\mathtt{x}_2$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ \mathtt{x}_2 + \mathtt{x}_1^2 \end{bmatrix}$$

```
end
```

- Compute closed form for $\mathtt{x}_2$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_2} = x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3})$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
    [x₁]    [x₁ − 1  ]
    [x₂] ← [x₂ + x₁²]
end
```

- Compute closed form for $\mathtt{x}_2$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_2} = x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3})$
- Over-approximation: $x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 + x_1 \cdot n + \frac{n}{2} + \frac{n^2}{3})$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_1$ and $\mathtt{x}_2$

```
while (x₁ > 0) do
  [x₁]    [x₁ − 1   ]
  [x₂] ← [x₂ + x₁² ]
end
```

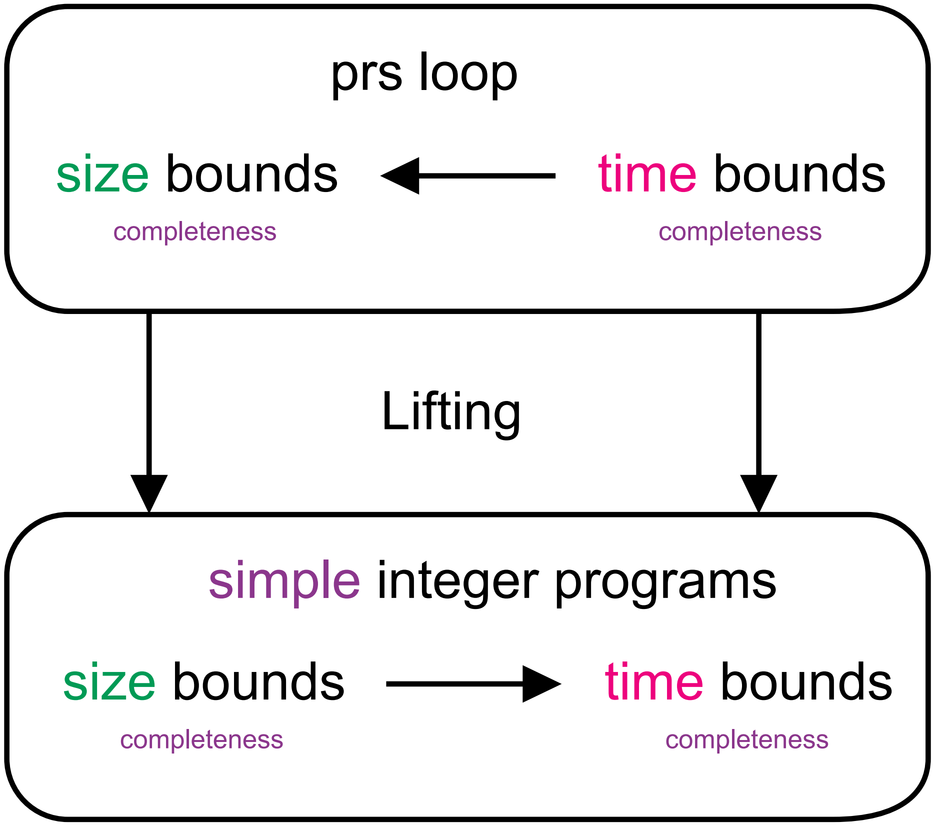
- Compute closed form for $\mathtt{x}_2$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_2} = x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3})$
- Over-approximation: $x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 + x_1 \cdot n + \frac{n}{2} + \frac{n^2}{3})$
- Size bound: $x_2 + x_1 \cdot (\frac{1}{6} + x_1 + x_1^2 + x_1 \cdot x_1 + \frac{x_1}{2} + \frac{{x_1}^2}{3})$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

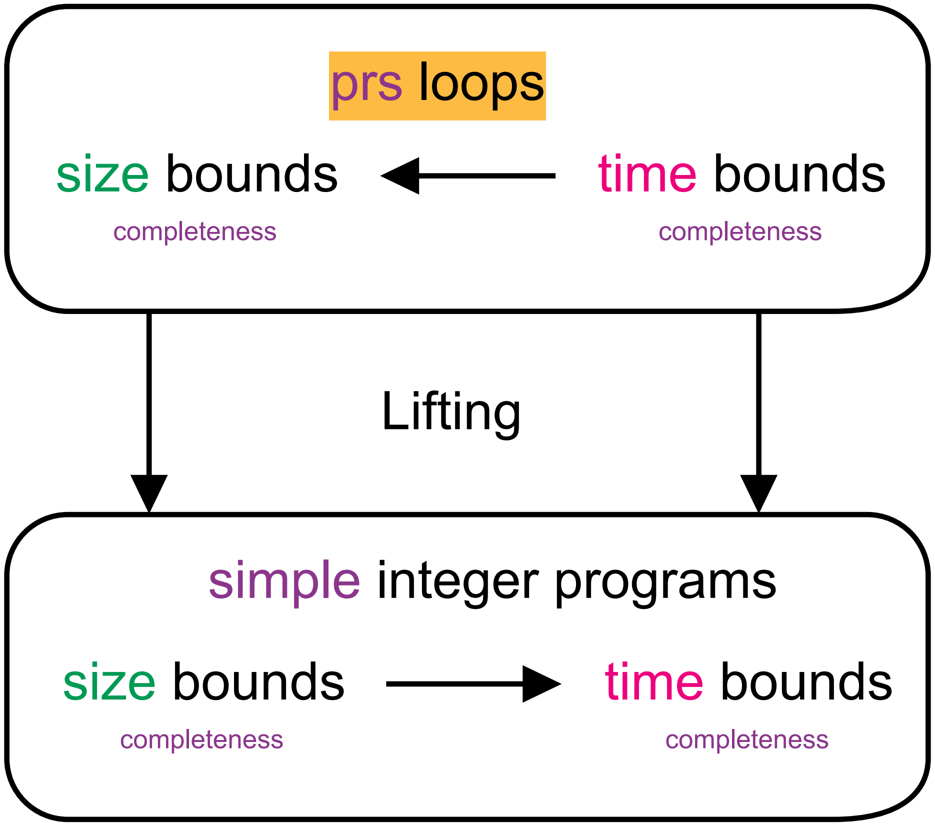
Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

Goal: Infer (upper) size and time bounds for "real-world" programs



Periodic Rational Solvable Loops

```
while (τ) do


end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \qquad \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

$$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

$$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, …) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

$$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix

- Variable value depends at most linearly on its previous value.

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

$$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

$$
\texttt{while } (\tau) \texttt{ do} \quad \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} + \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix} \quad \texttt{end}
$$

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:
  $$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$
- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\texttt{x} \leftarrow \texttt{x}^2$ (so the value is $x^{(2^n)}$)

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```
$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} + \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix}$$
```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:
  $$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$
- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} + \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

  $$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (x1 > 0) do
   [ ] ← [ ][ ] + [ ]
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:
  $$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$
- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (x₁ > 0) do
```
$$\begin{bmatrix} x_1 \\ \\ \\ \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} x_1 \\ \\ \\ \end{bmatrix} + \begin{bmatrix} -1 \\ \\ \\ \end{bmatrix}$$
```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, …) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:
  $$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$
- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (x₁ > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ \\ \\ \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ \\ \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ \\ \\ \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

$$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.

Periodic Rational Solvable Loops

```
while (x_1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

$$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} \mathcal{S}_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $x \leftarrow x^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

    $\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n$ with $\alpha_j \in \overline{\mathbb{Q}}[x_1, \ldots, x_d]$, $a_j \in \mathbb{N}$ and $b_j \in \overline{\mathbb{Q}}$

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \ldots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

while $(\mathtt{x}_1 > 0)$ do

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

end

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \ldots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

while $(\mathtt{x}_1 > 0)$ do

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

end

- closed form for $\mathtt{x}_2$:

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \ldots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

while $(\mathtt{x}_1 > 0)$ do

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

end

- closed form for $\mathtt{x}_2$:

$$x_2 + n \cdot \left(\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3}\right)$$

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \ldots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

$$
\begin{array}{l}
\texttt{while } (\mathtt{x}_1 > 0) \texttt{ do} \\
\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- closed form for $\mathtt{x}_2$:

$$x_2 + n \cdot \left(\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3}\right)$$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \ldots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

$$\begin{aligned}&\texttt{while } (\texttt{x}_1 > 0) \texttt{ do}\\ &\quad \begin{bmatrix}\texttt{x}_1\\ \texttt{x}_2\\ \texttt{x}_3\\ \texttt{x}_4\end{bmatrix} \leftarrow \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 3 & 2\\ 0 & 0 & -5 & -3\end{bmatrix}\begin{bmatrix}\texttt{x}_1\\ \texttt{x}_2\\ \texttt{x}_3\\ \texttt{x}_4\end{bmatrix} + \begin{bmatrix}-1\\ \texttt{x}_1^2\\ 0\\ 0\end{bmatrix}\\ &\texttt{end}\end{aligned}$$

- closed form for $\texttt{x}_2$:

$$x_2 + n \cdot \left(\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3}\right)$$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \ldots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- closed form for $\mathtt{x}_2$:

$$x_2 + n \cdot \left(\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3}\right)$$

- closed form for $\mathtt{x}_3$:

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \ldots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- closed form for $\mathtt{x}_2$:
  $x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3})$
- closed form for $\mathtt{x}_3$:
  $\frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$ for a linear polynomial α.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \dots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

```
while (x_1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- closed form for x_2:
  $x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3})$
- closed form for x_3:
  $\frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$ for a linear polynomial α.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \dots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- closed form for $\mathtt{x}_2$:
  $x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3})$
- closed form for $\mathtt{x}_3$:
  $\frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$ for a linear polynomial α.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \dots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- closed form for $\mathtt{x}_2$:
  $x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3})$
- closed form for $\mathtt{x}_3$:
  $\frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$ for a linear polynomial α.

- How to handle algebraic $\overline{\mathbb{Q}} \setminus \mathbb{Q}$ numbers?

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \dots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- closed form for $\mathtt{x}_2$:
  $x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3})$
- closed form for $\mathtt{x}_3$:
  $\frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$ for a linear polynomial α.

- How to handle algebraic $\overline{\mathbb{Q}} \setminus \mathbb{Q}$ numbers?
- When do we have polynomial size bounds?

FroCoS ’23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Closed Forms: PRS Loops

- Closed forms are computable for all prs loops.
  - poly-exponential expressions:

$$\sum_j \alpha_j \cdot n^{a_j} \cdot b_j^n \text{ with } \alpha_j \in \overline{\mathbb{Q}}[x_1, \dots, x_d],\ a_j \in \mathbb{N} \text{ and } b_j \in \overline{\mathbb{Q}}$$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} + \begin{bmatrix} -1 \\ \mathtt{x}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- closed form for $\mathtt{x}_2$:
  $x_2 + n \cdot (\frac{1}{6} + x_1 + x_1^2 - x_1 \cdot n - \frac{n}{2} + \frac{n^2}{3})$
- closed form for $\mathtt{x}_3$:
  $\frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$ for a linear polynomial α.

- How to handle algebraic $\overline{\mathbb{Q}} \setminus \mathbb{Q}$ numbers?
- When do we have polynomial size bounds?
  - When are (polynomial) time bounds computable?

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

$$\begin{aligned}&\mathtt{while}\ (\mathtt{x}_1 > 0)\ \mathtt{do}\\&\quad\begin{bmatrix}\mathtt{x}_1\\\mathtt{x}_3\\\mathtt{x}_4\end{bmatrix} \leftarrow \begin{bmatrix}\mathtt{x}_1 - 1\\3\mathtt{x}_3 + 2\mathtt{x}_4\\-5\mathtt{x}_3 - 3\mathtt{x}_4\end{bmatrix}\\&\mathtt{end}\end{aligned}$$

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x₁ > 0) do
    [x₁]     [   x₁ − 1    ]
    [x₃]  ←  [ 3x₃ + 2x₄   ]
    [x₄]     [ −5x₃ − 3x₄  ]
end
```

$$\texttt{while } (\mathtt{x}_1 > 0) \texttt{ do} \quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix} \quad \texttt{end}$$

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}_{x_3}^n = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

$$
\begin{array}{l}
\texttt{while } (\mathtt{x}_1 > 0) \texttt{ do} \\
\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}_{x_3}^{n} = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation:

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix}$$

```
end
```

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_3} = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation: $\frac{1}{2} \cdot |\alpha| \cdot (|-\mathrm{i}|)^n + \frac{1}{2} \cdot |\overline{\alpha}| \cdot |\mathrm{i}|^n$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x1 > 0) do
    [x1]     [   x1 - 1    ]
    [x3]  ←  [ 3x3 + 2x4   ]
    [x4]     [ -5x3 - 3x4  ]
end
```

$$\texttt{while } (\mathtt{x}_1 > 0) \texttt{ do} \quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix} \quad \texttt{end}$$

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_3} = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation: $\frac{1}{2} \cdot |\alpha| \cdot (|-\mathrm{i}|)^n + \frac{1}{2} \cdot |\overline{\alpha}| \cdot |\mathrm{i}|^n = \frac{1}{2} \cdot |\alpha| + \frac{1}{2} \cdot |\overline{\alpha}|$

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix}$$

```
end
```

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_3} = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation: $\frac{1}{2} \cdot |\alpha| \cdot (|-\mathrm{i}|)^n + \frac{1}{2} \cdot |\overline{\alpha}| \cdot |\mathrm{i}|^n = |\alpha|$

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix}$$

```
end
```

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}_{x_3}^{n} = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation: $\frac{1}{2} \cdot |\alpha| \cdot (|-\mathrm{i}|)^n + \frac{1}{2} \cdot |\overline{\alpha}| \cdot |\mathrm{i}|^n = |\alpha|$
- Size bound: $|\alpha| = 4 \cdot x_3 + 2 \cdot x_4$

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix}$$

```
end
```

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_3} = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation: $\frac{1}{2} \cdot |\alpha| \cdot (|-\mathrm{i}|)^n + \frac{1}{2} \cdot |\overline{\alpha}| \cdot |\mathrm{i}|^n = |\alpha|$
- Size bound: $|\alpha| = 4 \cdot x_3 + 2 \cdot x_4$

- How to handle algebraic $\overline{\mathbb{Q}} \setminus \mathbb{Q}$ numbers?

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix}$$

```
end
```

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_3} = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation: $\frac{1}{2} \cdot |\alpha| \cdot (|-\mathrm{i}|)^n + \frac{1}{2} \cdot |\overline{\alpha}| \cdot |\mathrm{i}|^n = |\alpha|$
- Size bound: $|\alpha| = 4 \cdot x_3 + 2 \cdot x_4$

- How to handle algebraic $\overline{\mathbb{Q}} \setminus \mathbb{Q}$ numbers? Take absolute value!

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x₁ > 0) do
```
$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix}$$
```
end
```

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_3} = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation: $\frac{1}{2} \cdot |\alpha| \cdot (|-\mathrm{i}|)^n + \frac{1}{2} \cdot |\overline{\alpha}| \cdot |\mathrm{i}|^n = |\alpha|$
- Size bound: $|\alpha| = 4 \cdot x_3 + 2 \cdot x_4$

- How to handle algebraic $\overline{\mathbb{Q}} \setminus \mathbb{Q}$ numbers? Take absolute value!
- When do we have polynomial size bounds?

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix}$$

```
end
```

- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}_{x_3}^n = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation: $\frac{1}{2} \cdot |\alpha| \cdot (|-\mathrm{i}|)^n + \frac{1}{2} \cdot |\overline{\alpha}| \cdot |\mathrm{i}|^n = |\alpha|$
- Size bound: $|\alpha| = 4 \cdot x_3 + 2 \cdot x_4$

- How to handle algebraic $\overline{\mathbb{Q}} \setminus \mathbb{Q}$ numbers? Take absolute value!
- When do we have polynomial size bounds?
  - All eigenvalues λ are *unit*: $|\lambda| \leq 1$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Bounds by Closed Forms

Goal: Infer (absolute) size bound for $\mathtt{x}_3$

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 - 3\mathtt{x}_4 \end{bmatrix}$$

```
end
```

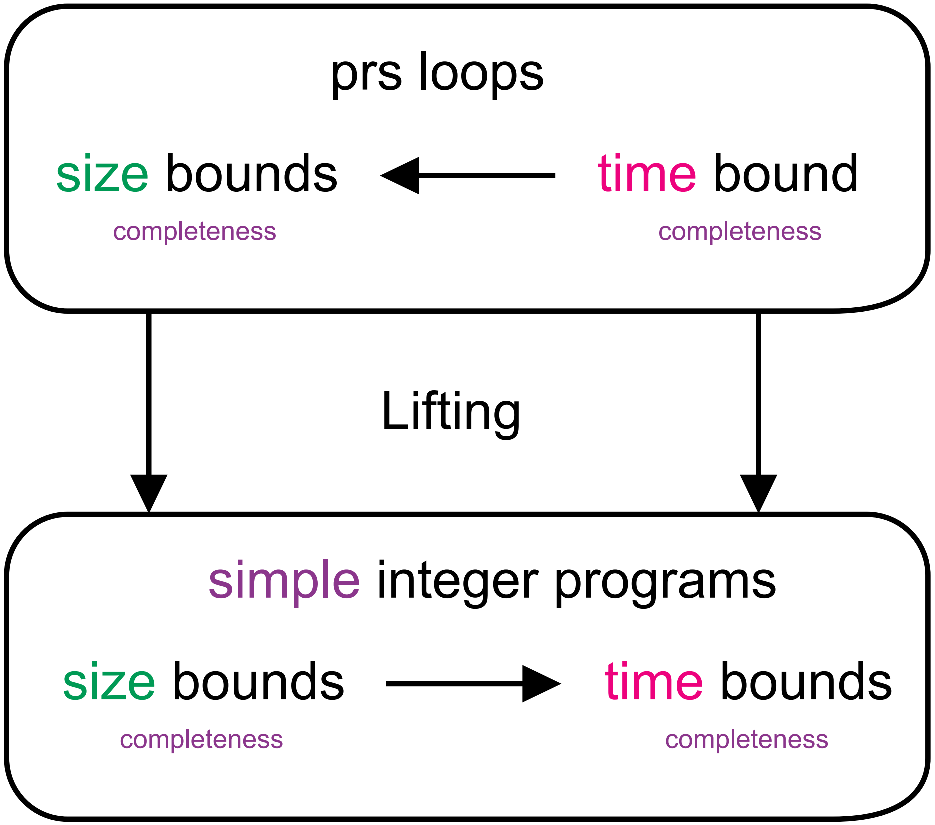
- Compute closed form for $\mathtt{x}_3$.
- Over-approximate closed form to non-negative, weakly monotonic increasing expression.
- Replace n by an over-approximation of the runtime.

- Closed form: $\mathtt{cl}^n_{x_3} = \frac{1}{2} \cdot \alpha \cdot (-\mathrm{i})^n + \frac{1}{2} \cdot \overline{\alpha} \cdot \mathrm{i}^n$
- Over-approximation: $\frac{1}{2} \cdot |\alpha| \cdot (|-\mathrm{i}|)^n + \frac{1}{2} \cdot |\overline{\alpha}| \cdot |\mathrm{i}|^n = |\alpha|$
- Size bound: $|\alpha| = 4 \cdot x_3 + 2 \cdot x_4$

- How to handle algebraic $\overline{\mathbb{Q}} \setminus \mathbb{Q}$ numbers? Take absolute value!
- When do we have polynomial size bounds?
  - All eigenvalues λ are *unit*: $|\lambda| \leq 1$
  - When are (polynomial) time bounds computable?

Overview

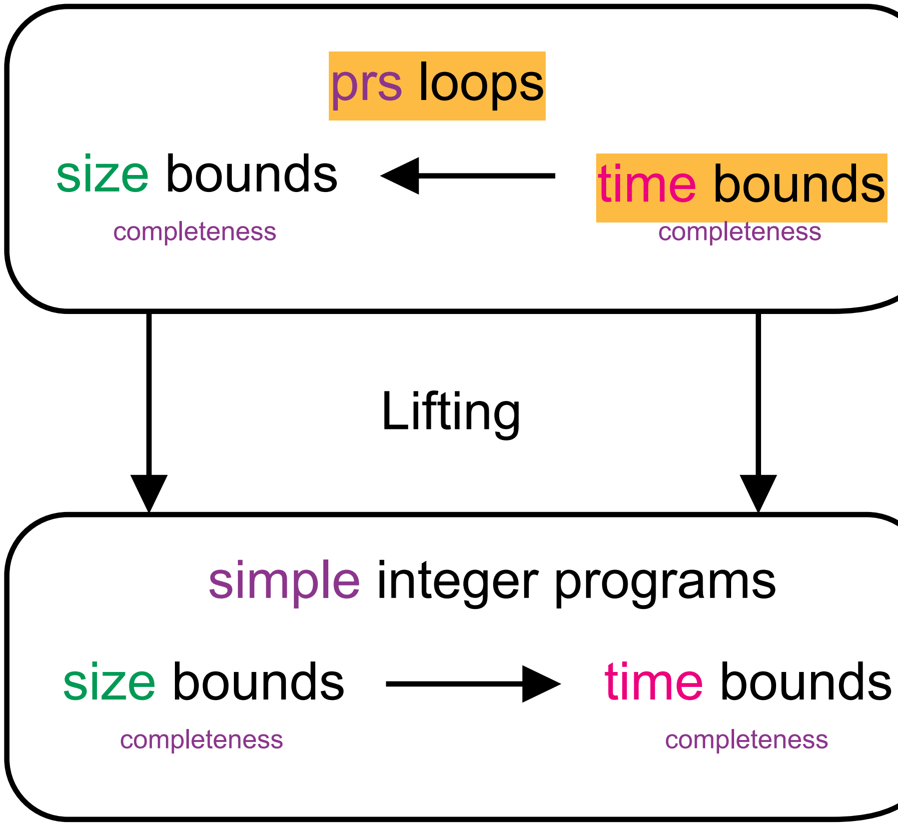
Goal: Infer (upper) size and time bounds for "real-world" programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

Goal: Infer (upper) size and time bounds for “real-world” programs



FroCoS ’23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} + \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

$$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} + \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

$$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.
- Periodic rational: there exists $n \in \mathbb{N}$ s.t. $\lambda^n \in \mathbb{Q}$ for $\lambda \in \overline{\mathbb{Q}}$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} + \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:

$$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$

- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.
- Periodic rational: there exists $n \in \mathbb{N}$ s.t. $\lambda^n \in \mathbb{Q}$ for $\lambda \in \overline{\mathbb{Q}}$
  $\sqrt{3}$ and i as $(\sqrt{3})^2 \in \mathbb{Q}$ and $\mathrm{i}^2 \in \mathbb{Q}$ ✓

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} + \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix}$$

```
end
```

- τ built from $\wedge$, $\vee$, ($\neg$, ...) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:
  $$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$
- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.
- Periodic rational: there exists $n \in \mathbb{N}$ s.t. $\lambda^n \in \mathbb{Q}$ for $\lambda \in \overline{\mathbb{Q}}$
  $\sqrt{3}$ and i as $(\sqrt{3})^2 \in \mathbb{Q}$ and $\mathrm{i}^2 \in \mathbb{Q}$ ✓ $2 + 3\mathrm{i}$ ✗

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Periodic Rational Solvable Loops

```
while (τ) do
```

$$\begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} \leftarrow \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_d \end{bmatrix} + \begin{bmatrix} p_1 \\ \vdots \\ p_d \end{bmatrix}$$

```
end
```

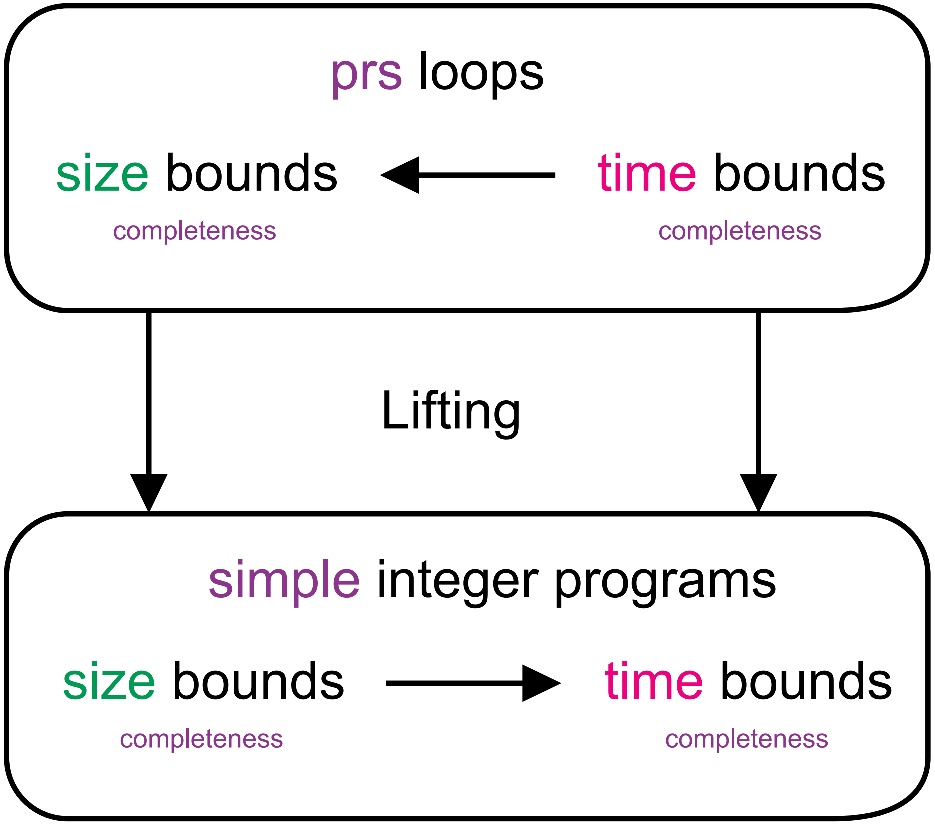
- τ built from $\wedge$, $\vee$, ($\neg$, …) and polynomial inequations over $\mathbb{Z}$
- Partition variables into blocks:
  $$\mathcal{S}_1 \uplus \cdots \uplus \mathcal{S}_d$$
- $A_i \in \mathbb{Z}^{|\mathcal{S}_i| \times |\mathcal{S}_i|}$ integer matrix with periodic rational eigenvalues
- $p_i \in \mathbb{Z}[\bigcup_{j<i} S_j]^{|\mathcal{S}_i|}$ polynomials

- Variable value depends at most linearly on its previous value.
  - Prevent super-exponential growth: $\mathtt{x} \leftarrow \mathtt{x}^2$ (so the value is $x^{(2^n)}$)
- Non-linear dependencies only of variables from blocks with lower indices
- Solve recurrence to obtain closed form.
- Periodic rational: there exists $n \in \mathbb{N}$ s.t. $\lambda^n \in \mathbb{Q}$ for $\lambda \in \overline{\mathbb{Q}}$
  $\sqrt{3}$ and i as $(\sqrt{3})^2 \in \mathbb{Q}$ and $\mathrm{i}^2 \in \mathbb{Q}$ ✓ $2 + 3\mathrm{i}$ ✗

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

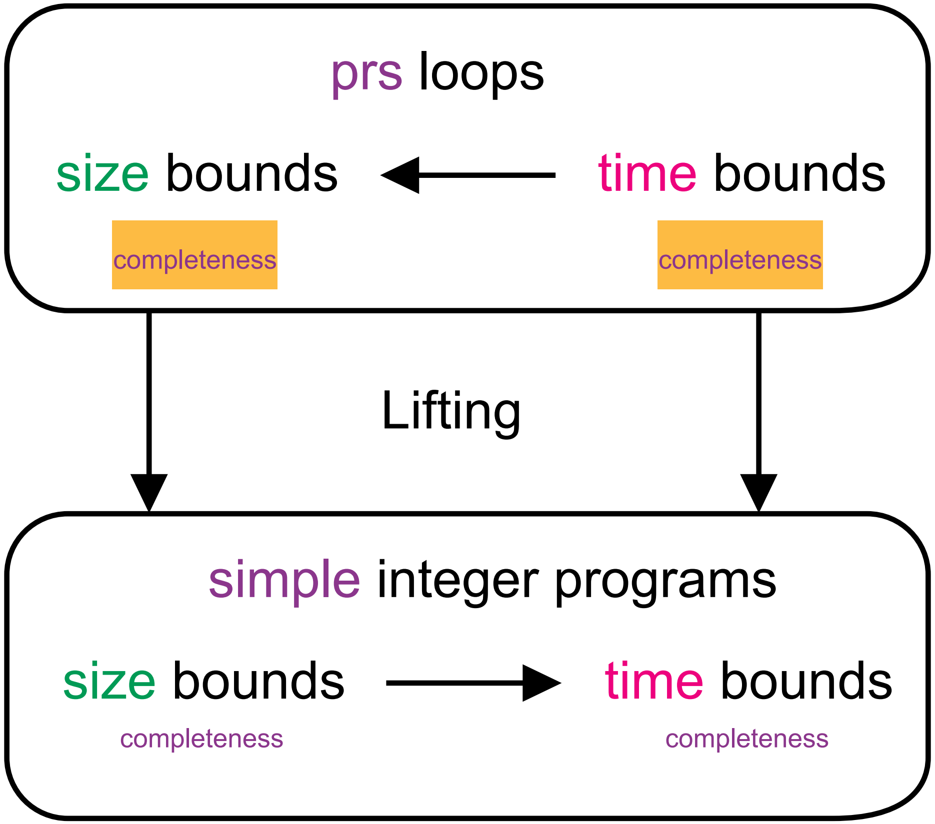
Overview

Goal: Infer (upper) size and time bounds for “real-world” programs



Overview

Goal: Infer (upper) size and time bounds for "real-world" programs



Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

```
while (x_1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```


Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

$$
\texttt{while } (x_1 > 0) \texttt{ do}
$$

$$
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}
$$

$$
\texttt{end}
$$

- 1 has period 1

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

```
while (x₁ > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- 1 has period 1
- i has period 2 as $\mathrm{i}^2 = -1 \in \mathbb{Q}$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$
$$\texttt{end}$$

- 1 has period 1
- i has period 2 as $\mathrm{i}^2 = -1 \in \mathbb{Q}$
- $-\mathrm{i}$ has period 2 as $(-\mathrm{i})^2 = -1 \in \mathbb{Q}$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

```
while (x₁ > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- 1 has period 1
- i has period 2 as $\mathrm{i}^2 = -1 \in \mathbb{Q}$
- $-\mathrm{i}$ has period 2 as $(-\mathrm{i})^2 = -1 \in \mathbb{Q}$

  $\Rightarrow$ chain loop once

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

```
while (x₁ > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- 1 has period 1
- i has period 2 as $\mathrm{i}^2 = -1 \in \mathbb{Q}$
- $-\mathrm{i}$ has period 2 as $(-\mathrm{i})^2 = -1 \in \mathbb{Q}$

$\Rightarrow$ chain loop once

```
while (x₁ > 0) do
```

$$\begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} \leftarrow \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$$

```
end
```


Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$
$$\texttt{end}$$

- 1 has period 1
- i has period 2 as $i^2 = -1 \in \mathbb{Q}$
- $-i$ has period 2 as $(-i)^2 = -1 \in \mathbb{Q}$

  $\Rightarrow$ chain loop once

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ \\ \\ \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} x_1 \\ \\ \\ \end{bmatrix} + \begin{bmatrix} -2 \\ \\ \\ \end{bmatrix}$$
$$\texttt{end}$$

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$
$$\texttt{end}$$

- 1 has period 1
- i has period 2 as $\mathrm{i}^2 = -1 \in \mathbb{Q}$
- $-\mathrm{i}$ has period 2 as $(-\mathrm{i})^2 = -1 \in \mathbb{Q}$

$\Rightarrow$ chain loop once

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ x_2 \\ \\ \\ \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ \\ \end{bmatrix} + \begin{bmatrix} -2 \\ x_1^2 + (x_1 - 1)^2 \\ \\ \\ \end{bmatrix}$$
$$\texttt{end}$$

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

$$\texttt{while}\ (x_1 > 0)\ \texttt{do}\quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}\quad \texttt{end}$$

- 1 has period 1
- i has period 2 as $i^2 = -1 \in \mathbb{Q}$
- $-i$ has period 2 as $(-i)^2 = -1 \in \mathbb{Q}$

  $\Rightarrow$ chain loop once

$$\texttt{while}\ (x_1 > 0)\ \texttt{do}\quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -2 \\ x_1^2 + (x_1 - 1)^2 \\ 0 \\ 0 \end{bmatrix}\quad \texttt{end}$$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

```
while (x_1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- 1 has period 1
- i has period 2 as $\mathrm{i}^2 = -1 \in \mathbb{Q}$
- $-\mathrm{i}$ has period 2 as $(-\mathrm{i})^2 = -1 \in \mathbb{Q}$
  $\Rightarrow$ chain loop once

```
while (x_1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -2 \\ x_1^2 + (x_1 - 1)^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- Prove termination for chained loops [SAS '20]
  - co-NP-complete for linear arithmetic

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

```
while (x_1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- 1 has period 1
- i has period 2 as $\mathrm{i}^2 = -1 \in \mathbb{Q}$
- $-\mathrm{i}$ has period 2 as $(-\mathrm{i})^2 = -1 \in \mathbb{Q}$
  $\Rightarrow$ chain loop once

```
while (x_1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -2 \\ x_1^2 + (x_1 - 1)^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- Prove termination for chained loops [SAS '20]
  - co-NP-complete for linear arithmetic
- Find time bounds for terminating chained loops [LPAR '20]

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- (Polynomial) time bounds are computable for all terminating prs loops.
  - chain (unroll) loops accordingly to their period $\rightsquigarrow$ integer eigenvalues

```
while (x1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- 1 has period 1
- i has period 2 as $\mathrm{i}^2 = -1 \in \mathbb{Q}$
- $-\mathrm{i}$ has period 2 as $(-\mathrm{i})^2 = -1 \in \mathbb{Q}$

  $\Rightarrow$ chain loop once

```
while (x1 > 0) do
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -2 \\ x_1^2 + (x_1 - 1)^2 \\ 0 \\ 0 \end{bmatrix}$$

```
end
```

- Prove termination for chained loops [SAS '20]
  - co-NP-complete for linear arithmetic
- Find time bounds for terminating chained loops [LPAR '20]
- Derive time bound for original loops

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- Closed forms are computable for all prs loops.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- Closed forms are computable for all prs loops.
- Polynomial time bounds are computable for all terminating prs loops. [LPAR '20]

Completeness: PRS Loops

- Closed forms are computable for all prs loops.
- Polynomial time bounds are computable for all terminating prs loops. [LPAR '20]
- Size bounds are computable for all terminating prs loops.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: PRS Loops

- Closed forms are computable for all prs loops.
- Polynomial time bounds are computable for all terminating prs loops. [LPAR '20]
- Size bounds are computable for all terminating prs loops.
- Polynomial size bounds are computable for all *unit* prs loops.

Completeness: PRS Loops

- Closed forms are computable for all prs loops.
- Polynomial time bounds are computable for all terminating prs loops. [LPAR '20]
- Size bounds are computable for all terminating prs loops.
- Polynomial size bounds are computable for all *unit* prs loops.
  - *unit*: for all eigenvalues $\lambda \in \overline{\mathbb{Q}}$ we have $|\lambda| \leq 1$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

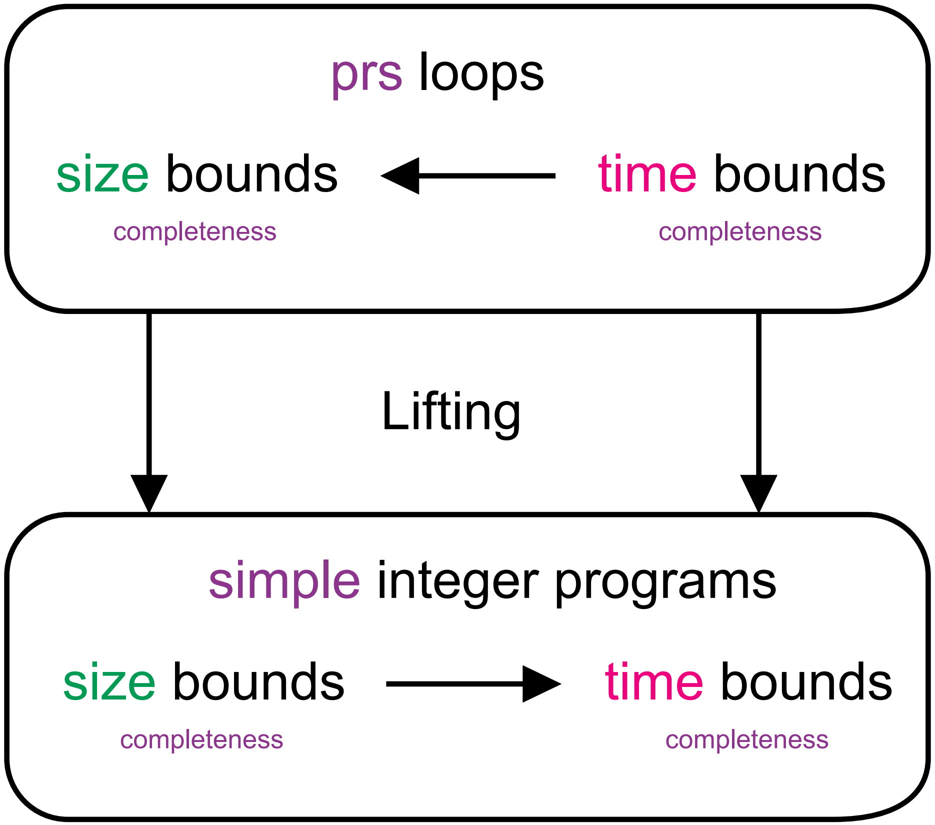
Completeness: PRS Loops

- Closed forms are computable for all prs loops.
- Polynomial time bounds are computable for all terminating prs loops. [LPAR '20]
- Size bounds are computable for all terminating prs loops.
- Polynomial size bounds are computable for all *unit* prs loops.
  - *unit*: for all eigenvalues $\lambda \in \overline{\mathbb{Q}}$ we have $|\lambda| \leq 1$

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & -5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -1 \\ x_1^2 \\ 0 \\ 0 \end{bmatrix}$$
$$\texttt{end}$$

Overview

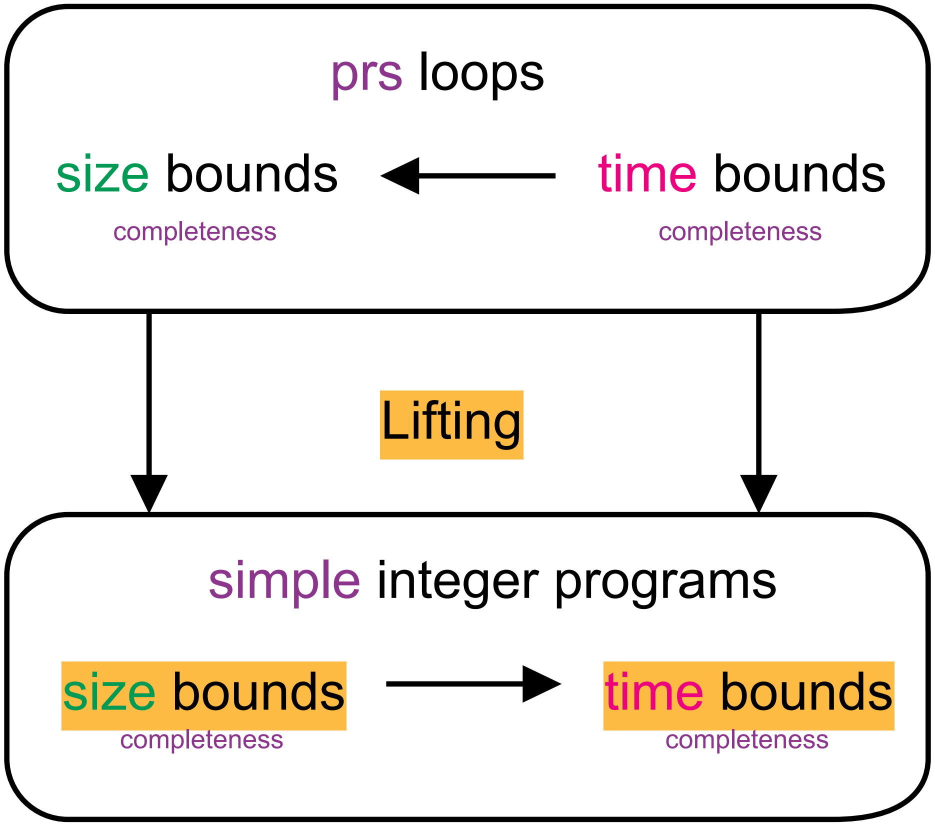
Goal: Infer (upper) size and time bounds for "real-world" programs



FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

Goal: Infer (upper) size and time bounds for "real-world" programs



Size Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

$$
\texttt{while } (x_1 > 0) \texttt{ do}
$$

$$
\begin{bmatrix} x_1 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ 3x_3 + 2x_4 \\ -5x_3 + -3x_4 \end{bmatrix}
$$

$$
\texttt{end}
$$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ 3x_3 + 2x_4 \\ -5x_3 + -3x_4 \end{bmatrix}$$
$$\texttt{end}$$
$$\texttt{while } (x_3 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_3 \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \\ y + 1 \end{bmatrix}$$
$$\texttt{end}$$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Complexity of Integer Programs

Goal: Infer size and time bounds for "real-world" programs

$$\texttt{while } (x_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_1 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ 3x_3 + 2x_4 \\ -5x_3 + -3x_4 \end{bmatrix}$$
$$\texttt{end}$$
$$\texttt{while } (x_3 > 0) \texttt{ do}$$
$$\begin{bmatrix} x_3 \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \\ y + 1 \end{bmatrix}$$
$$\texttt{end}$$

- Size of y after second loop:

Size Complexity of Integer Programs

Goal: Infer size and time bounds for "real-world" programs

```
while (x₁ > 0) do
```
$$\begin{bmatrix} x_1 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ 3x_3 + 2x_4 \\ -5x_3 + -3x_4 \end{bmatrix}$$
```
end
while (x₃ > 0) do
```
$$\begin{bmatrix} x_3 \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \\ y + 1 \end{bmatrix}$$
```
end
```

- Size of y after second loop:
- Idea: Analyze different subprograms and combine results

Size Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
while (x_3 > 0) do
```
$$\begin{bmatrix} x_3 \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \\ y + 1 \end{bmatrix}$$
```
end
```

- Size of y after second loop:
- Idea: Analyze different subprograms and combine results

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

$$
\begin{array}{l}
\texttt{while } (\mathtt{x}_3 > 0) \texttt{ do} \\
\quad \begin{bmatrix} \mathtt{x}_3 \\ \mathtt{y} \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_3 - 1 \\ \mathtt{y} + 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- Size of $\mathtt{y}$ after second loop:
- Idea: Analyze different subprograms and combine results
  - $\mathtt{y}$ “locally” has size $y + x_3$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Complexity of Integer Programs

Goal: Infer size and time bounds for "real-world" programs

$$\texttt{while } (\mathtt{x}_3 > 0) \texttt{ do} \quad \begin{bmatrix} \mathtt{x}_3 \\ \mathtt{y} \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_3 - 1 \\ \mathtt{y} + 1 \end{bmatrix} \quad \texttt{end}$$

- Size of $\mathtt{y}$ after second loop:
- Idea: Analyze different subprograms and combine results
  - $\mathtt{y}$ "locally" has size $y + x_3$

Size of $\mathtt{y}$: $y + x_3$

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
while (x₁ > 0) do
```

$$\begin{bmatrix} x_1 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 - 1 \\ 3x_3 + 2x_4 \\ -5x_3 + -3x_4 \end{bmatrix}$$

```
end
while (x₃ > 0) do
```

$$\begin{bmatrix} x_3 \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \\ y + 1 \end{bmatrix}$$

```
end
```

- Size of y after second loop:
- Idea: Analyze different subprograms and combine results
  - y “locally” has size $y + x_3$
- Respect size of variables:

Size of y: $y + x_3$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
while (x1 > 0) do
    [x1]     [    x1 - 1    ]
    [x3]  ←  [  3x3 + 2x4   ]
    [x4]     [ -5x3 + -3x4  ]
end
while (x3 > 0) do
    [x3]  ←  [x3 - 1]
    [y ]     [y + 1 ]
end
```

- ▶ Size of y after second loop:
- ▶ Idea: Analyze different subprograms and combine results
  - y “locally” has size $y + x_3$
- ▶ Respect size of variables:
  - x_3 is size bounded by $4 \cdot x_3 + 2 \cdot x_4$.

Size of y: $y + x_3$

FroCoS ’23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
while (x₁ > 0) do
    [x₁]    [    x₁ − 1      ]
    [x₃] ← [  3x₃ + 2x₄     ]
    [x₄]    [−5x₃ + −3x₄    ]
end
while (x₃ > 0) do
    [x₃] ← [x₃ − 1]
    [y ]    [y + 1 ]
end
```

- Size of $\mathtt{y}$ after second loop:
- Idea: Analyze different subprograms and combine results
  - $\mathtt{y}$ “locally” has size $y + x_3$
- Respect size of variables:
  - $\mathtt{x}_3$ is size bounded by $4 \cdot x_3 + 2 \cdot x_4$.

Size of $\mathtt{y}$: $y + x_3\ [x_3/size(x_3)]$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Size Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

$$\texttt{while } (\mathtt{x}_1 > 0) \texttt{ do}$$
$$\begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_3 \\ \mathtt{x}_4 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_1 - 1 \\ 3\mathtt{x}_3 + 2\mathtt{x}_4 \\ -5\mathtt{x}_3 + -3\mathtt{x}_4 \end{bmatrix}$$
$$\texttt{end}$$
$$\texttt{while } (\mathtt{x}_3 > 0) \texttt{ do}$$
$$\begin{bmatrix} \mathtt{x}_3 \\ \mathtt{y} \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_3 - 1 \\ \mathtt{y} + 1 \end{bmatrix}$$
$$\texttt{end}$$

- Size of $\mathtt{y}$ after second loop:
- Idea: Analyze different subprograms and combine results
  - $\mathtt{y}$ “locally” has size $y + x_3$
- Respect size of variables:
  - $\mathtt{x}_3$ is size bounded by $4 \cdot x_3 + 2 \cdot x_4$.

Size of $\mathtt{y}$: $y + 4 \cdot x_3 + 2 \cdot x_4$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

$L_1\,;$
$L_2\,;$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
L₁;
L₂;
// y has size y + 4·x₃ + 2·x₄
```


Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for "real-world" programs

```
L_1;
L_2;
// y has size y + 4·x_3 + 2·x_4
 while (y > 0) do
    [y] ← [y − 1]
 end
```


Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

L_1;
L_2;
// y has size $y + 4 \cdot x_3 + 2 \cdot x_4$

while (y $>$ 0) do
  $[\mathtt{y}] \leftarrow [\mathtt{y} - 1]$
end

- How often do we execute the loop?

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

L_1;
L_2;
// y has size $y + 4 \cdot x_3 + 2 \cdot x_4$

```
while (y > 0) do
    [y] ← [y − 1]
end
```

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
while (y > 0) do
   [y] ← [y − 1]
end
```

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
while (y > 0) do
   [y] ← [y − 1]
end
```

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results
  - loop is “locally” executed y times

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for "real-world" programs

```
while (y > 0) do
   [y] ← [y − 1]
end
```

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results
  - loop is "locally" executed y times

Number of loop executions: y

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

L_1;
L_2;
// y has size $y + 4 \cdot x_3 + 2 \cdot x_4$

while (y $>$ 0) do
  $[\mathtt{y}] \leftarrow [\mathtt{y} - 1]$
end

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results
  - loop is “locally” executed y times
- Respect size of variables:

Number of loop executions: y

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
L1;
L2;
// y has size y + 4·x3 + 2·x4
while (y > 0) do
   [y] ← [y − 1]
end
```

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results
  - loop is “locally” executed y times
- Respect size of variables:
  - y is size bounded by $y + 4 \cdot x_3 + 2 \cdot x_4$

Number of loop executions: y

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Goal: Infer size and time bounds for “real-world“ programs

```
L₁;
L₂;
// y has size y + 4·x₃ + 2·x₄
 while (y > 0) do
    [y] ← [y − 1]
 end
```

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results
  - loop is “locally” executed y times
- Respect size of variables:
  - y is size bounded by $y + 4 \cdot x_3 + 2 \cdot x_4$

Number of loop executions: $y\ [y/size(y)]$

FroCoS ’23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
L1;
L2;
// y has size y + 4·x3 + 2·x4
while (y > 0) do
    [y] ← [y − 1]
end
```

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results
  - loop is “locally” executed y times
- Respect size of variables:
  - y is size bounded by $y + 4 \cdot x_3 + 2 \cdot x_4$

Number of loop executions: $y + 4 \cdot x_3 + 2 \cdot x_4$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
L1;
L2;
// y has size y + 4·x3 + 2·x4

while (y > 0) do
    [y] ← [y − 1]
end
```

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results
  - loop is “locally” executed y times
- Respect size of variables:
  - y is size bounded by $y + 4 \cdot x_3 + 2 \cdot x_4$
- How many times do we *start* to evaluate the loop?

Number of loop executions: $y + 4 \cdot x_3 + 2 \cdot x_4$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
L₁;
L₂;
// y has size y + 4·x₃ + 2·x₄
while (y > 0) do
    [y] ← [y − 1]
end
```

- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results
  - loop is “locally” executed y times
- Respect size of variables:
  - y is size bounded by $y + 4 \cdot x_3 + 2 \cdot x_4$
- How many times do we *start* to evaluate the loop?

Number of loop executions: $1 \cdot (y + 4 \cdot x_3 + 2 \cdot x_4)$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Time Complexity of Integer Programs

Goal: Infer size and time bounds for “real-world“ programs

```
L1;
L2;
// y has size y + 4·x3 + 2·x4
 while (y > 0) do
    [y] ← [y − 1]
 end
```

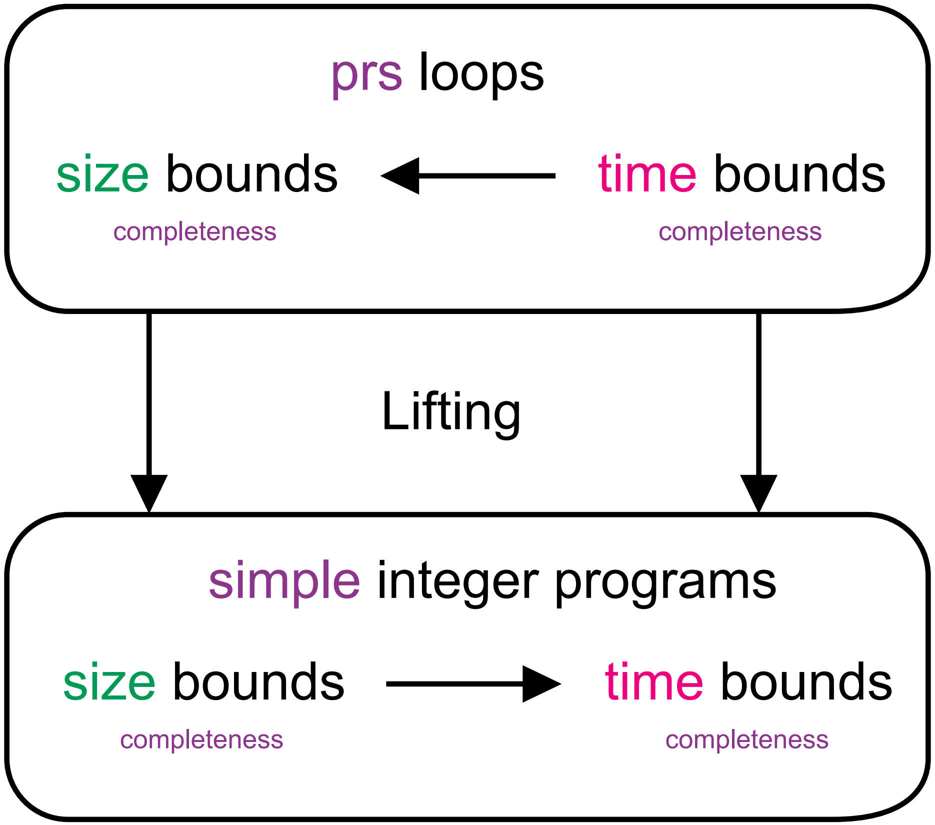
- How often do we execute the loop?
- Idea: Analyze different subprograms and combine results
  - loop is “locally” executed y times
- Respect size of variables:
  - y is size bounded by $y + 4 \cdot x_3 + 2 \cdot x_4$
- How many times do we *start* to evaluate the loop?

Number of loop executions: $y + 4 \cdot x_3 + 2 \cdot x_4$

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

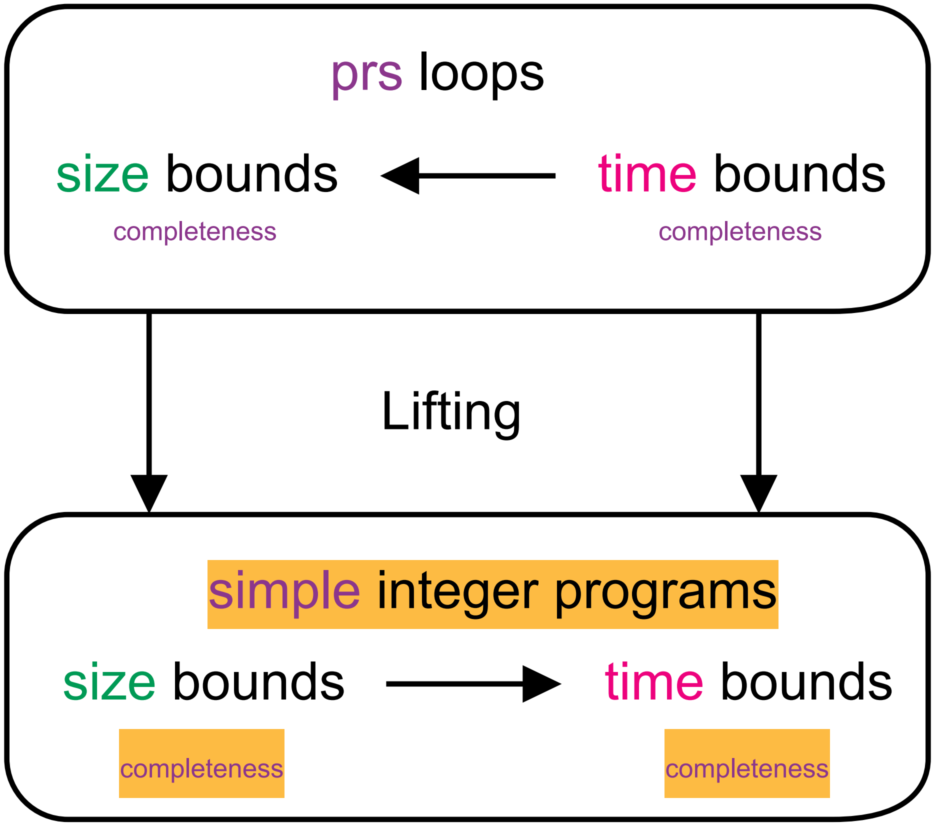
Overview

Goal: Infer (upper) size and time bounds for "real-world" programs



Overview

Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

- Simple Integer Program:

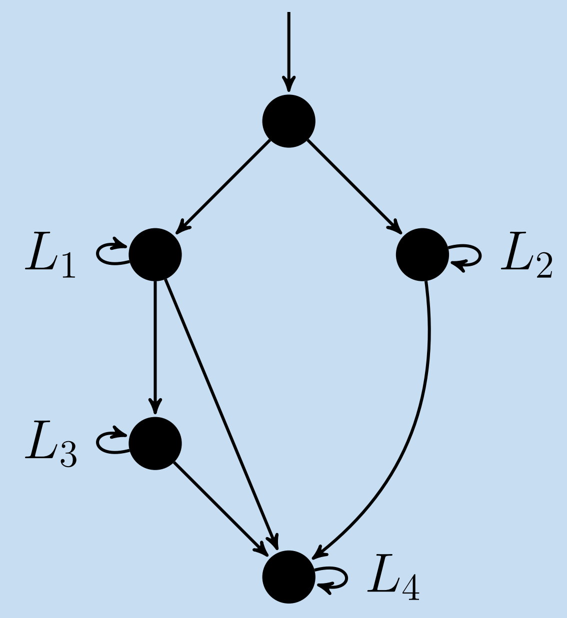
Completeness: Simple Integer Programs

- Simple Integer Program:
  - No nested loops

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

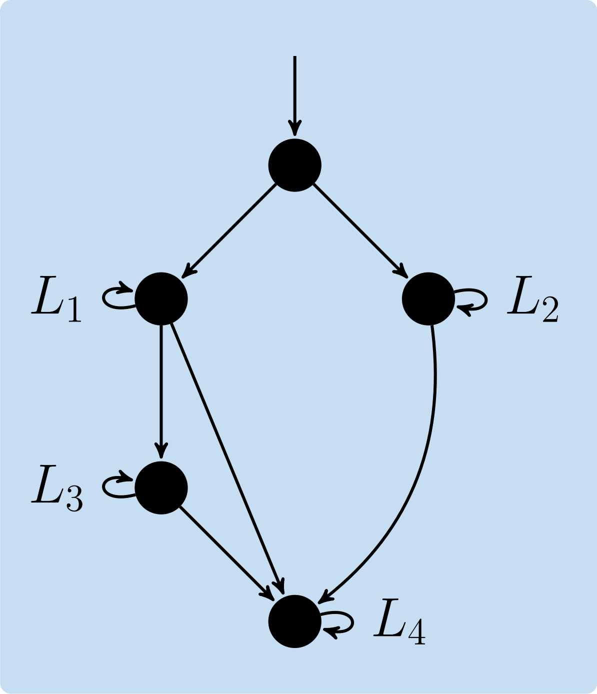
Completeness: Simple Integer Programs

- Simple Integer Program:
  - No nested loops



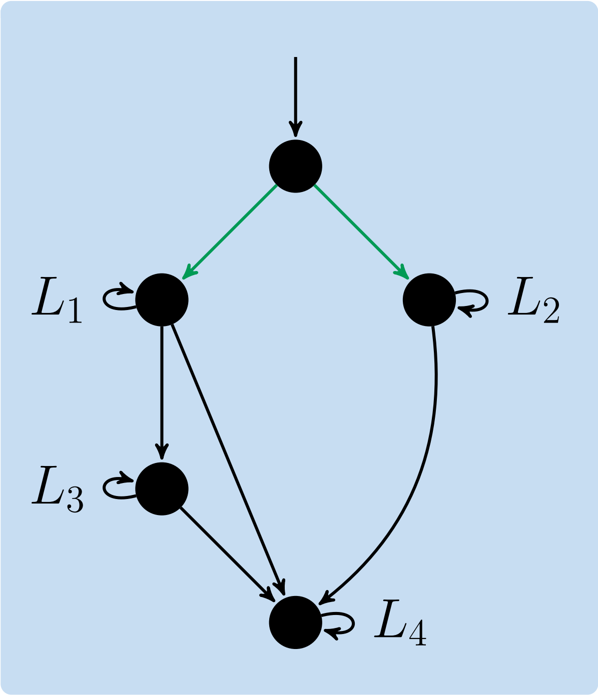
Completeness: Simple Integer Programs

- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:



Completeness: Simple Integer Programs

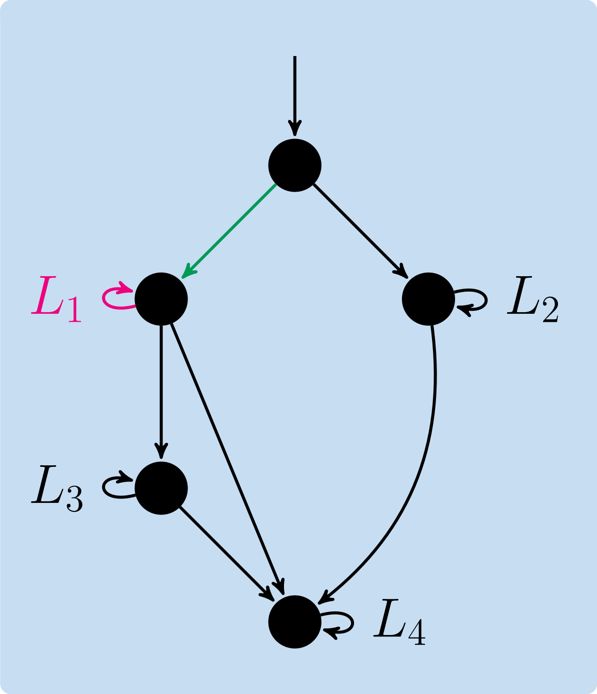
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

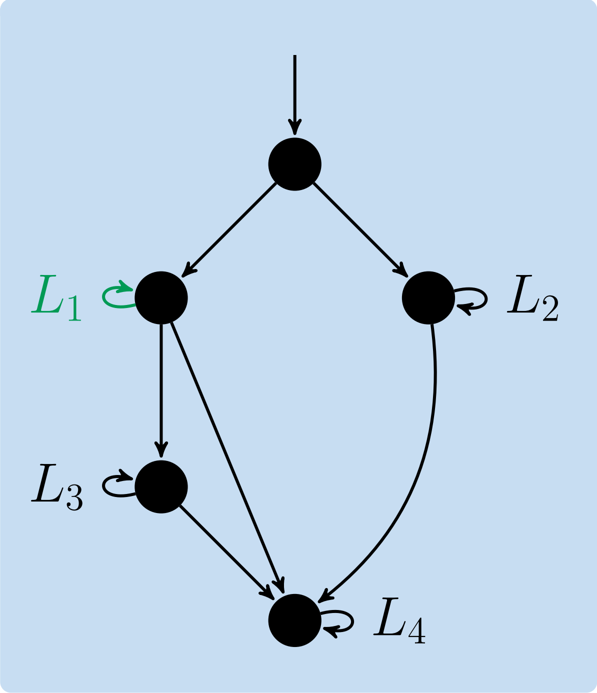
Completeness: Simple Integer Programs

- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.



Completeness: Simple Integer Programs

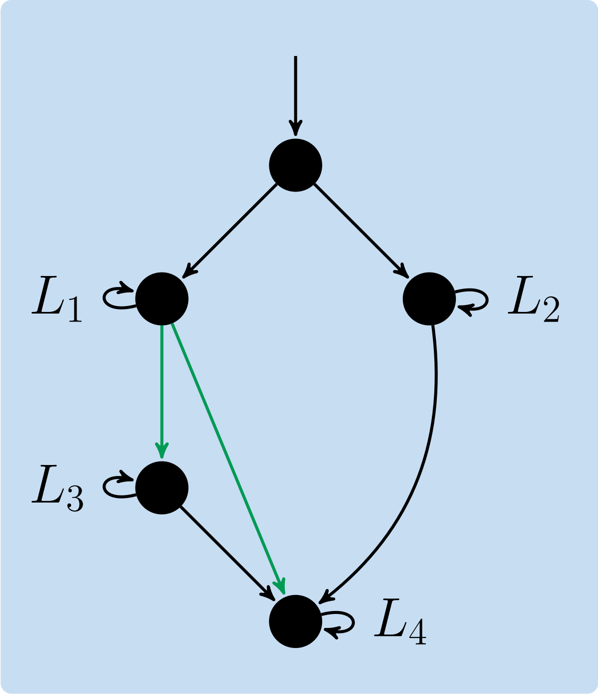
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

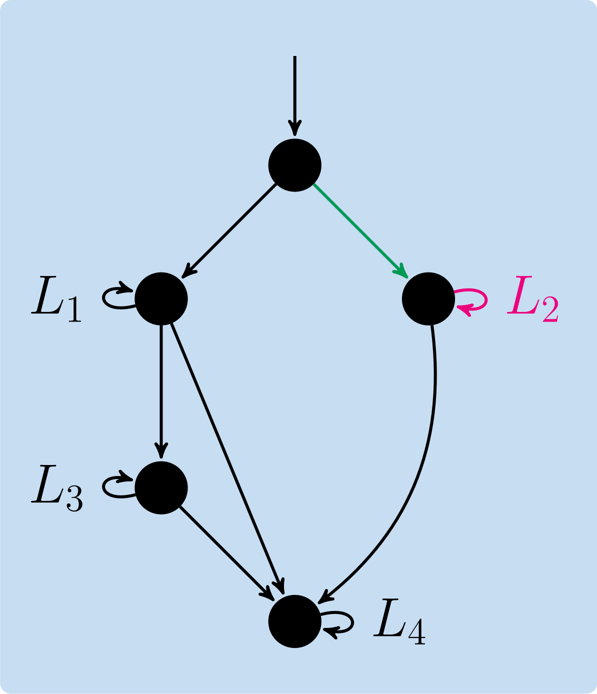
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

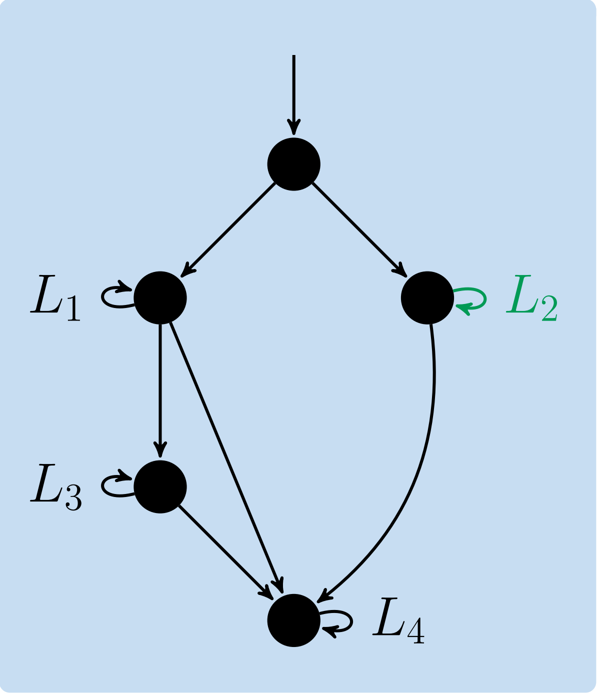
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

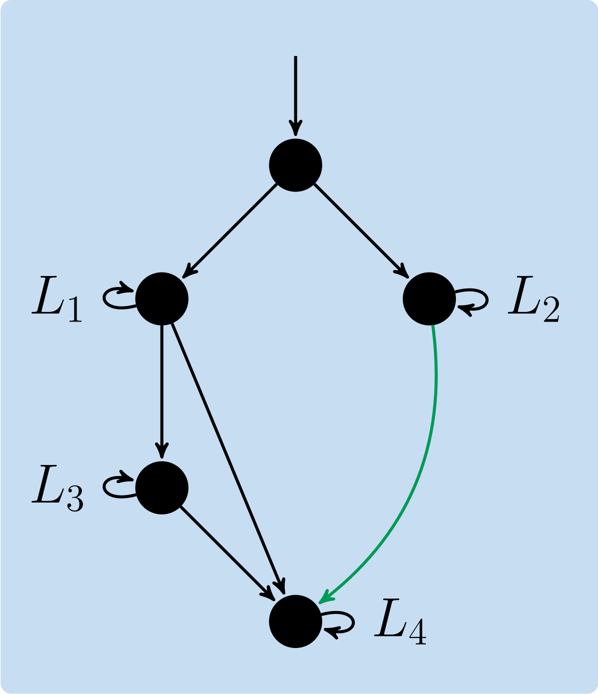
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

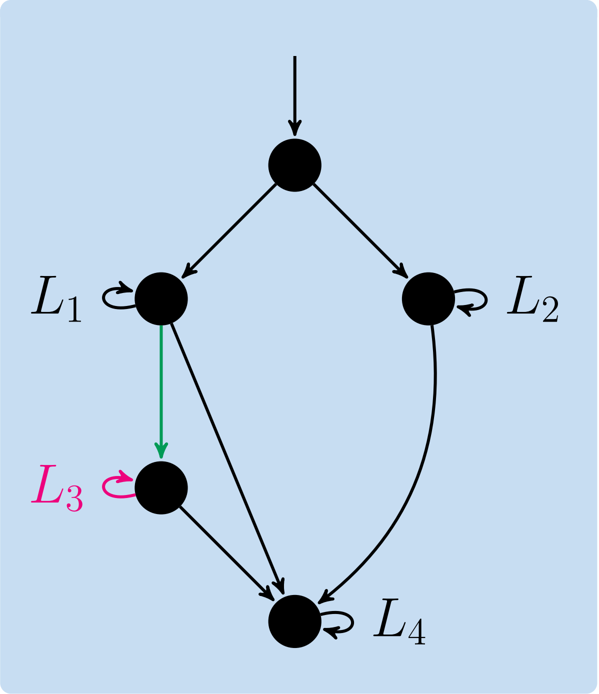
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

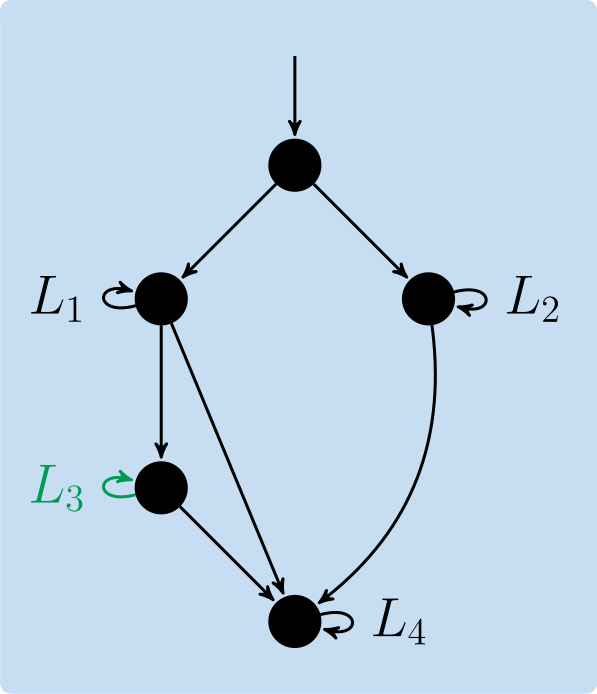
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

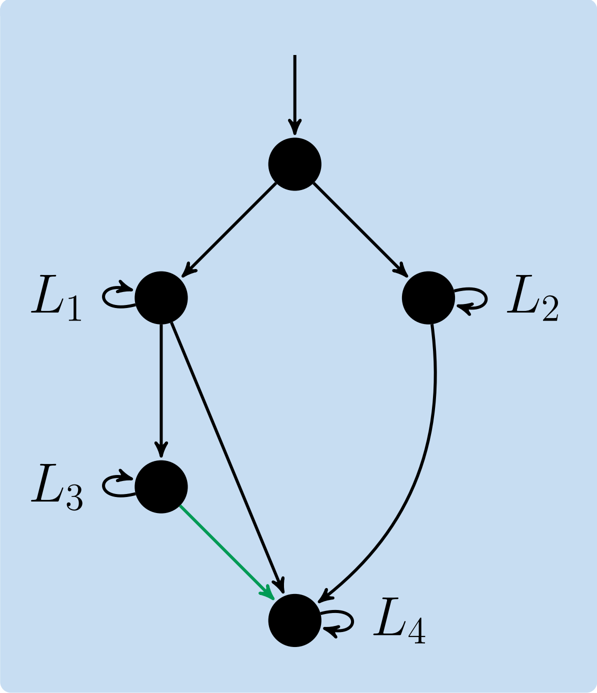
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

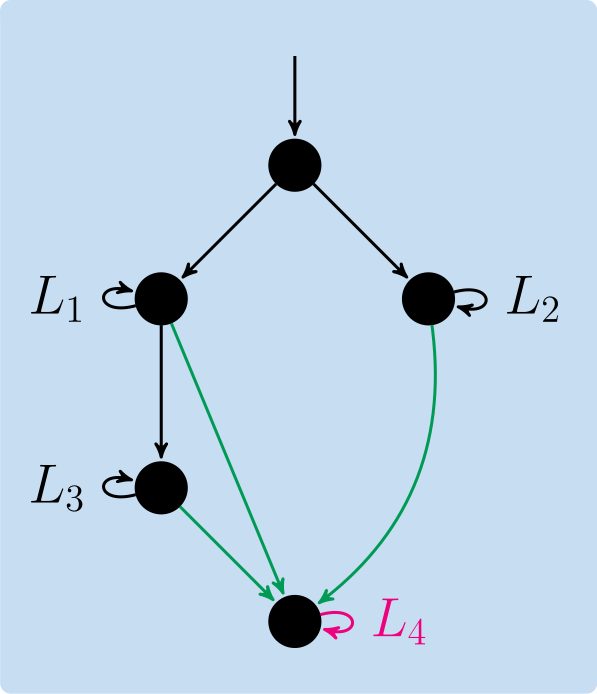
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

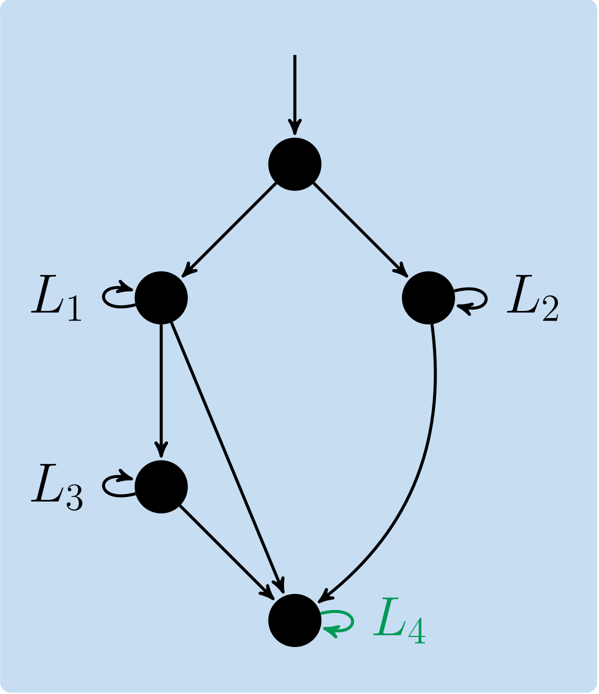
- ▶ Simple Integer Program:
  - No nested loops
- ▶ Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

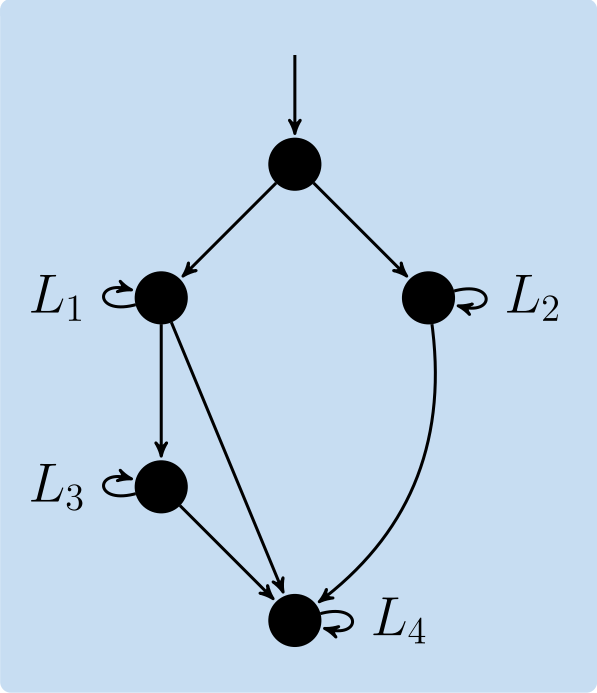
- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Completeness: Simple Integer Programs

- Simple Integer Program:
  - No nested loops
- Solve loops in topological order:
  - Infer time bound by considering previous size bounds.
  - Compute size bounds for loops.
  - Propagate size bounds to subsequent loops.

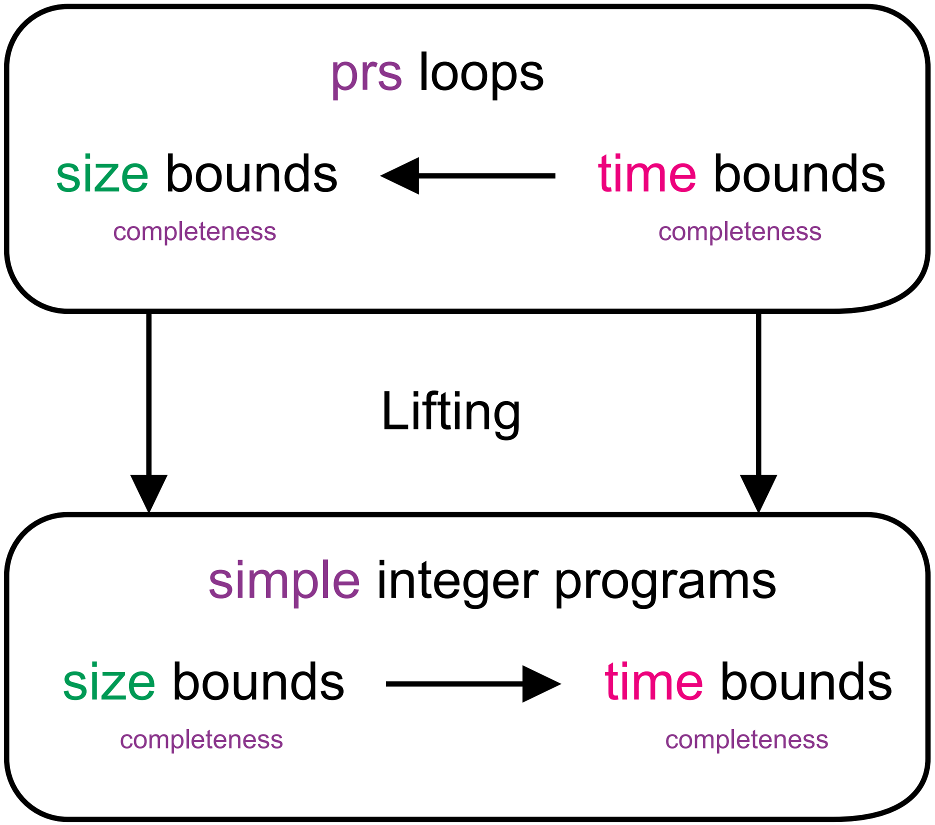


- Polynomial size and time bounds are computable if all loops are terminating unit prs loops.

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

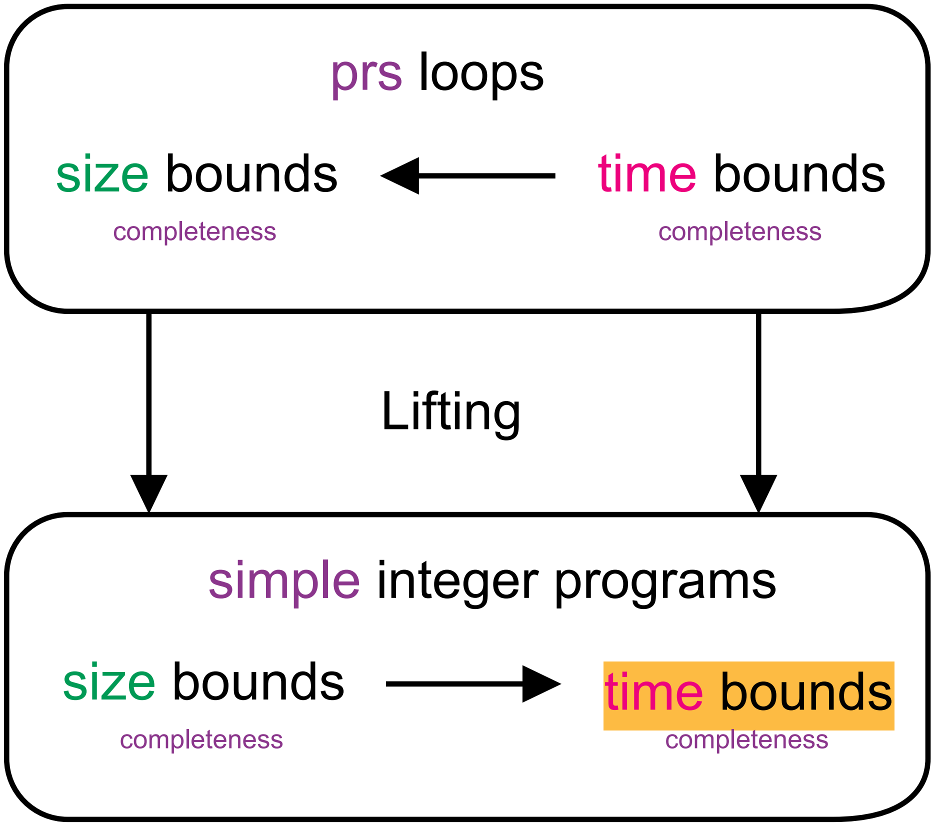
Goal: Infer (upper) size and time bounds for “real-world” programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

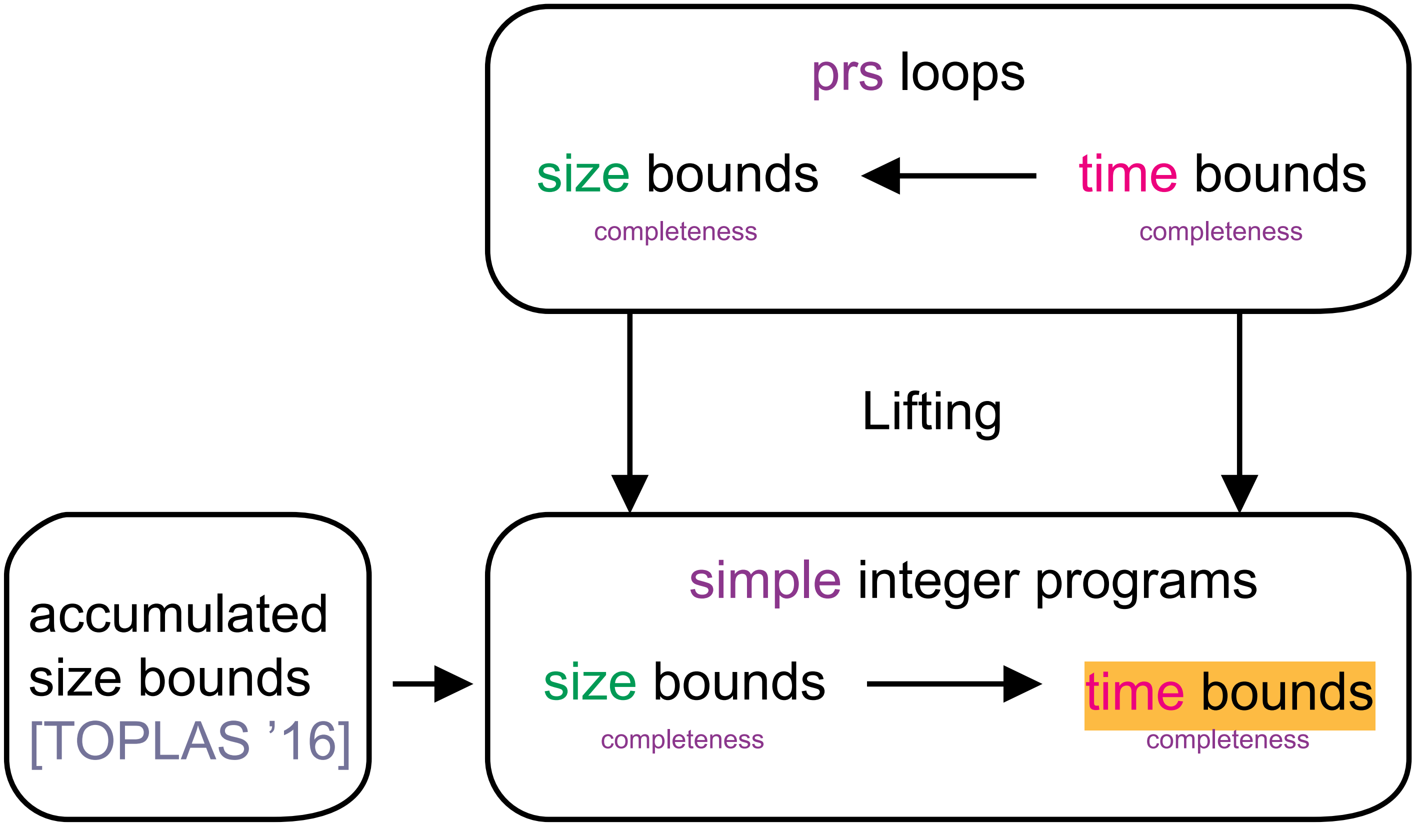
Overview

Goal: Infer (upper) size and time bounds for "real-world" programs



Overview

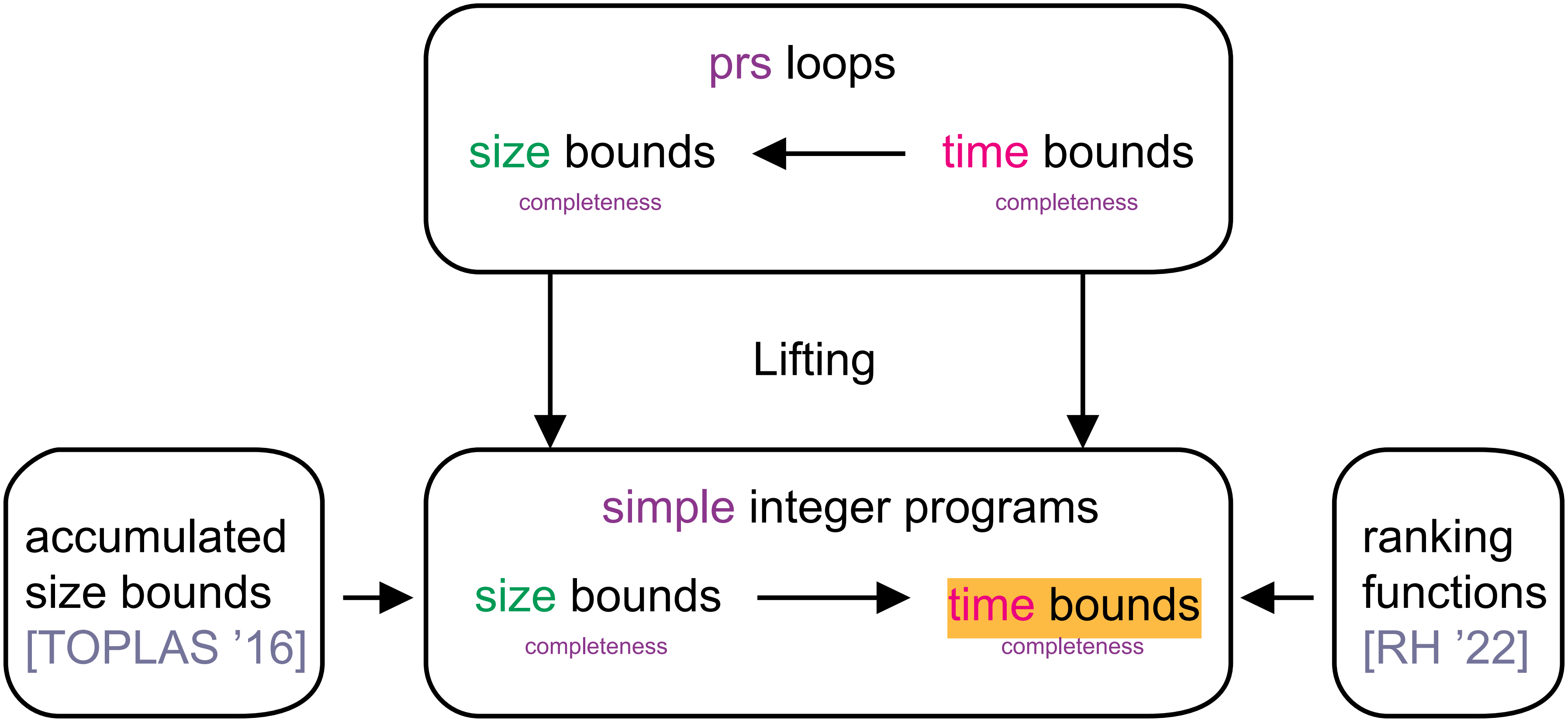
Goal: Infer (upper) size and time bounds for "real-world" programs



Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Overview

Goal: Infer (upper) size and time bounds for “real-world” programs



Evaluation of our Implementation in KoAT2

- C_Complexity consisting of 519 (mainly linear) benchmarks from TPDB

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	AVG(s)
Loopus	17	171	50	6	0	244	0.40
KoAT1	25	170	74	12	8	289	0.96
CoFloCo	22	197	66	5	0	290	0.59
MaxCore	23	220	67	7	0	317	1.96

- KoAT1: original KoAT implementation [TOPLAS' 16]

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Evaluation of our Implementation in KoAT2

- C_Complexity consisting of 519 (mainly linear) benchmarks from TPDB

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	AVG(s)
Loopus	17	171	50	6	0	244	0.40
KoAT1	25	170	74	12	8	289	0.96
CoFloCo	22	197	66	5	0	290	0.59
MaxCore	23	220	67	7	0	317	1.96
KoAT2	26	232	70	15	5	348	8.29

- KoAT1: original KoAT implementation [TOPLAS' 16]
- KoAT2: reimplementation of KoAT1 [RH '22] + [IJCAR '22]

Evaluation of our Implementation in KoAT2

- C_Complexity consisting of 519 (mainly linear) benchmarks from TPDB

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	AVG(s)
Loopus	17	171	50	6	0	244	0.40
KoAT1	25	170	74	12	8	289	0.96
CoFloCo	22	197	66	5	0	290	0.59
MaxCore	23	220	67	7	0	317	1.96
KoAT2	26	232	70	15	5	348	8.29
KoAT2 + SIZE	26	233	71	25	3	358	9.97

- KoAT1: original KoAT implementation [TOPLAS' 16]
- KoAT2: reimplementation of KoAT1 [RH '22] + [IJCAR '22]

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Evaluation of our Implementation in KoAT2

- C_Complexity consisting of 519 (mainly linear) benchmarks from TPDB

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	AVG(s)
Loopus	17	171	50	6	0	244	0.40
KoAT1	25	170	74	12	8	289	0.96
CoFloCo	22	197	66	5	0	290	0.59
MaxCore	23	220	67	7	0	317	1.96
KoAT2	26	232	70	15	5	348	8.29
KoAT2 + SIZE	26	233	71	25	3	358	9.97

- KoAT1: original KoAT implementation [TOPLAS' 16]
- KoAT2: reimplementation of KoAT1 [RH '22] + [IJCAR '22]
- At most 386 benchmarks might terminate

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Evaluation of our Implementation in KoAT2

- C_Complexity consisting of 519 (mainly linear) benchmarks from TPDB

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	AVG(s)	succ. rate
Loopus	17	171	50	6	0	244	0.40	62%
KoAT1	25	170	74	12	8	289	0.96	74%
CoFloCo	22	197	66	5	0	290	0.59	75%
MaxCore	23	220	67	7	0	317	1.96	80%
KoAT2	26	232	70	15	5	348	8.29	85%
KoAT2 + SIZE	26	233	71	25	3	358	9.97	89%

- KoAT1: original KoAT implementation [TOPLAS' 16]
- KoAT2: reimplementation of KoAT1 [RH '22] + [IJCAR '22]
- At most 386 benchmarks might terminate

Evaluation of our Implementation in KoAT2

- C_Complexity consisting of 519 (mainly linear) benchmarks from TPDB

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	AVG(s)	succ. rate
Loopus	17	171	50	6	0	244	0.40	62%
KoAT1	25	170	74	12	8	289	0.96	74%
CoFloCo	22	197	66	5	0	290	0.59	75%
MaxCore	23	220	67	7	0	317	1.96	80%
KoAT2	26	232	70	15	5	348	8.29	85%
KoAT2 + SIZE	26	233	71	25	3	358	9.97	89%

- KoAT1: original KoAT implementation [TOPLAS' 16]
- KoAT2: reimplementation of KoAT1 [RH '22] + [IJCAR '22]
- At most 386 benchmarks might terminate
- KoAT2 + SIZE solves 89% of benchmarks which might terminate.

Conclusion

- Conclusion

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Conclusion

- Conclusion
  - Introduced modular approach for complexity analysis combining

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Conclusion

- Conclusion
  - Introduced modular approach for complexity analysis combining
    - Procedure to infer size bounds by closed forms

Conclusion

- Conclusion
  - Introduced modular approach for complexity analysis combining
    - Procedure to infer size bounds by closed forms
    - time bound computations

Conclusion

- Conclusion
  - Introduced modular approach for complexity analysis combining
    - Procedure to infer size bounds by closed forms
    - time bound computations
  - Handle loops with non-linear arithmetic

Conclusion

- Conclusion
  - Introduced modular approach for complexity analysis combining
    - Procedure to infer size bounds by closed forms
    - time bound computations
  - Handle loops with non-linear arithmetic
  - Complete for a large class of integer programs

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Conclusion

- Conclusion
  - Introduced modular approach for complexity analysis combining
    - Procedure to infer size bounds by closed forms
    - time bound computations
  - Handle loops with non-linear arithmetic
  - Complete for a large class of integer programs
  - `KoAT2` outperforms other state-of-the-art tools

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Conclusion

- Conclusion
  - Introduced modular approach for complexity analysis combining
    - Procedure to infer size bounds by closed forms
    - time bound computations
  - Handle loops with non-linear arithmetic
  - Complete for a large class of integer programs
  - `KoAT2` outperforms other state-of-the-art tools

`https://koat.verify.rwth-aachen.de/size`

Conclusion

- Conclusion
  - Introduced modular approach for complexity analysis combining
    - Procedure to infer size bounds by closed forms
    - time bound computations
  - Handle loops with non-linear arithmetic
  - Complete for a large class of integer programs
  - KoAT2 outperforms other state-of-the-art tools

https://koat.verify.rwth-aachen.de/size

Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2

Conclusion

- Conclusion
  - Introduced modular approach for complexity analysis combining
    - Procedure to infer size bounds by closed forms
    - time bound computations
  - Handle loops with non-linear arithmetic
  - Complete for a large class of integer programs
  - `KoAT2` outperforms other state-of-the-art tools

`https://koat.verify.rwth-aachen.de/size`

Thank You!

FroCoS '23
Nils Lommen and Jürgen Giesl
RWTH Aachen University – LuFGi2