# KoAT: An Automatic Complexity Analysis Tool for Integer Programs

## Workshop on Termination 2023

Nils Lommen, Eleanore Meyer, and Jürgen Giesl

# Motivation

**Goal**: Infer (upper) runtime bounds for “real-world” programs

$$
\begin{array}{l}
\texttt{while}\ (\mathtt{x}_3 > 0)\ \texttt{do} \\
\quad \begin{bmatrix}\mathtt{x}_1 \\ \mathtt{x}_2\end{bmatrix} \leftarrow \begin{bmatrix}\mathtt{x}_4 \\ \mathtt{x}_5^2\end{bmatrix} \\
\quad \texttt{while}\ (\mathtt{x}_1^2 < \mathtt{x}_2)\ \texttt{do} \\
\qquad \begin{bmatrix}\mathtt{x}_1 \\ \mathtt{x}_2\end{bmatrix} \leftarrow \begin{bmatrix}2 \cdot \mathtt{x}_1 \\ 3 \cdot \mathtt{x}_2\end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix}\mathtt{x}_3\end{bmatrix} \leftarrow \begin{bmatrix}\mathtt{x}_3 - 1\end{bmatrix} \\
\texttt{end}
\end{array}
$$

# Motivation

**Goal**: Infer (upper) runtime bounds for "real-world" programs

$$
\begin{array}{l}
\texttt{while } (x_3 > 0) \texttt{ do} \\
\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_4 \\ x_5^2 \end{bmatrix} \\
\quad \texttt{while } (x_1^2 < x_2) \texttt{ do} \\
\quad\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot x_1 \\ 3 \cdot x_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} x_3 \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- Does this program terminate?



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Motivation

**Goal**: Infer (upper) runtime bounds for “real-world” programs

$$
\begin{array}{l}
\texttt{while}\ (\mathtt{x}_3 > 0)\ \texttt{do} \\
\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_4 \\ \mathtt{x}_5^2 \end{bmatrix} \\
\quad \texttt{while}\ (\mathtt{x}_1^2 < \mathtt{x}_2)\ \texttt{do} \\
\qquad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot \mathtt{x}_1 \\ 3 \cdot \mathtt{x}_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} \mathtt{x}_3 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- Does this program terminate?
- How often do we execute the inner loop?



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Motivation

**Goal**: Infer (upper) runtime bounds for "real-world" programs

$$
\begin{array}{l}
\texttt{while } (\mathtt{x}_3 > 0) \texttt{ do} \\
\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_4 \\ \mathtt{x}_5^2 \end{bmatrix} \\
\quad \texttt{while } (\mathtt{x}_1^2 < \mathtt{x}_2) \texttt{ do} \\
\quad\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot \mathtt{x}_1 \\ 3 \cdot \mathtt{x}_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} \mathtt{x}_3 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- Does this program terminate?
- How often do we execute the inner loop?
  - Solution: Use KoAT!


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Motivation

**Goal**: Infer (upper) runtime bounds for “real-world” programs

$$
\begin{array}{l}
\texttt{while}\ (\mathtt{x}_3 > 0)\ \texttt{do} \\
\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_4 \\ \mathtt{x}_5^2 \end{bmatrix} \\
\quad \texttt{while}\ (\mathtt{x}_1^2 < \mathtt{x}_2)\ \texttt{do} \\
\qquad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot \mathtt{x}_1 \\ 3 \cdot \mathtt{x}_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} \mathtt{x}_3 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$

- Does this program terminate?
- How often do we execute the inner loop?
  - Solution: Use `KoAT`!
  - Open-source complexity analysis tool for **I**nteger **T**ransition **S**ystems

# Overview

- `KoAT` uses

# Overview

- `KoAT` uses
  - a modular approach to compute time bounds combining



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Overview

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops
      [IJCAR '22]

# Overview

- `KoAT` uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - M$\Phi$RFs [RH '22]

# Overview

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - MΦRFs [RH '22]
  - a modular approach to compute size bounds combining


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Overview

- `KoAT` uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - M$\Phi$RFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Overview

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - M$\Phi$RFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]

# Overview

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - MΦRFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]
  - local control flow-refinement by `iRankFinder` [RH '22].


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Overview

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - M$\Phi$RFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]
  - local control flow-refinement by `iRankFinder` [RH '22].

# General Architecture

- Translation to ITS

C Integer

$$
\begin{array}{l}
\texttt{while } (x_3 > 0) \texttt{ do} \\
\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_4 \\ x_5^2 \end{bmatrix} \\
\quad \texttt{while } (x_1^2 < x_2) \texttt{ do} \\
\quad\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot x_1 \\ 3 \cdot x_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} x_3 \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Translation to ITS

$$\text{C Integer} \xrightarrow{\texttt{clang}} \text{LLVM}$$

$$\begin{array}{l}
\texttt{while } (\mathtt{x}_3 > 0) \texttt{ do} \\
\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_4 \\ \mathtt{x}_5^2 \end{bmatrix} \\
\quad \texttt{while } (\mathtt{x}_1^2 < \mathtt{x}_2) \texttt{ do} \\
\qquad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot \mathtt{x}_1 \\ 3 \cdot \mathtt{x}_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} \mathtt{x}_3 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}$$


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Translation to ITS

$$\text{C Integer} \xrightarrow{\texttt{clang}} \text{LLVM} \underset{\texttt{AProVE}}{\overset{\texttt{llvm2kittel}}{\rightrightarrows}} \text{ITS}$$

$$
\begin{array}{l}
\texttt{while } (\mathtt{x}_3 > 0) \texttt{ do} \\
\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_4 \\ \mathtt{x}_5^2 \end{bmatrix} \\
\quad \texttt{while } (\mathtt{x}_1^2 < \mathtt{x}_2) \texttt{ do} \\
\quad\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot \mathtt{x}_1 \\ 3 \cdot \mathtt{x}_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} \mathtt{x}_3 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Translation to ITS

$$
\begin{tikzcd}
 & & & \text{Upper Bound} \\
\text{C Integer} \arrow[r, "\texttt{clang}"] & \text{LLVM} \arrow[r, bend left, "\texttt{llvm2kittel}"] \arrow[r, bend right, "\texttt{AProVE}"'] & \text{ITS} \arrow[ur] \arrow[dr] \arrow[r, phantom, "\texttt{KoAT}"] & \phantom{x} \\
 & & & \text{Termination}
\end{tikzcd}
$$

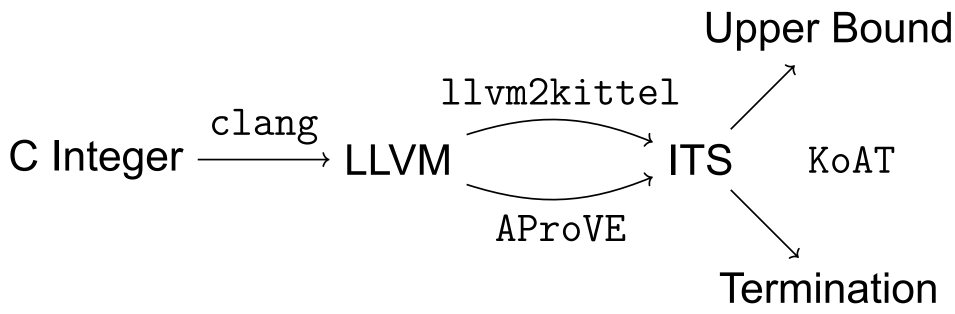
$$
\begin{array}{l}
\texttt{while } (x_3 > 0) \texttt{ do} \\
\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_4 \\ x_5^2 \end{bmatrix} \\
\quad \texttt{while } (x_1^2 < x_2) \texttt{ do} \\
\qquad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot x_1 \\ 3 \cdot x_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} x_3 \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Translation to ITS



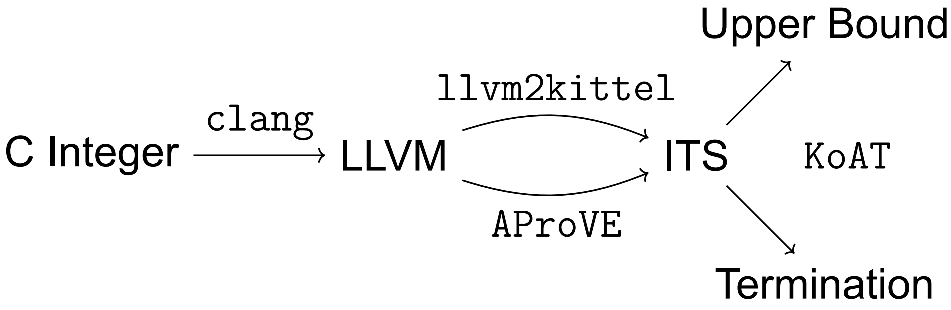

$$
\begin{array}{l}
\texttt{while}\ (x_3 > 0)\ \texttt{do} \\
\quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_4 \\ x_5^2 \end{bmatrix} \\
\quad \texttt{while}\ (x_1^2 < x_2)\ \texttt{do} \\
\qquad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot x_1 \\ 3 \cdot x_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} x_3 \end{bmatrix} \leftarrow \begin{bmatrix} x_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$



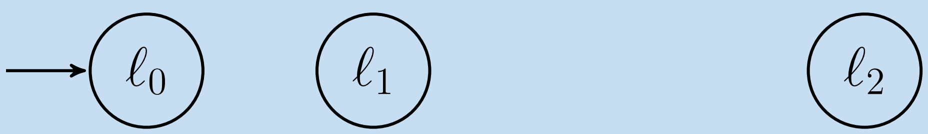
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Translation to ITS

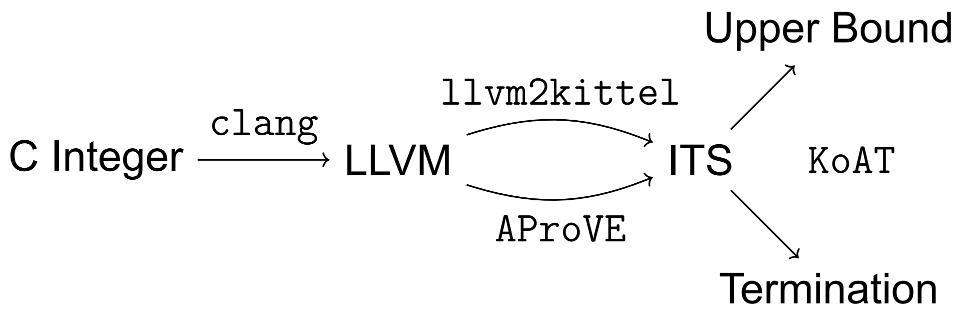



$$
\begin{array}{l}
\texttt{while } (\mathtt{x}_3 > 0) \texttt{ do} \\
\quad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_4 \\ \mathtt{x}_5^2 \end{bmatrix} \\
\quad \texttt{while } (\mathtt{x}_1^2 < \mathtt{x}_2) \texttt{ do} \\
\qquad \begin{bmatrix} \mathtt{x}_1 \\ \mathtt{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot \mathtt{x}_1 \\ 3 \cdot \mathtt{x}_2 \end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix} \mathtt{x}_3 \end{bmatrix} \leftarrow \begin{bmatrix} \mathtt{x}_3 - 1 \end{bmatrix} \\
\texttt{end}
\end{array}
$$



# General Architecture

- Translation to ITS



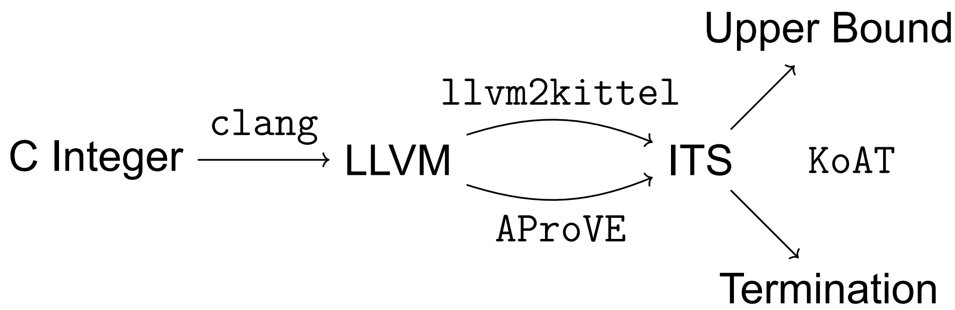

$$
\begin{array}{l}
\texttt{while}\ (\mathtt{x}_3 > 0)\ \texttt{do} \\
\quad \begin{bmatrix}\mathtt{x}_1 \\ \mathtt{x}_2\end{bmatrix} \leftarrow \begin{bmatrix}\mathtt{x}_4 \\ \mathtt{x}_5^2\end{bmatrix} \\
\quad \texttt{while}\ (\mathtt{x}_1^2 < \mathtt{x}_2)\ \texttt{do} \\
\qquad \begin{bmatrix}\mathtt{x}_1 \\ \mathtt{x}_2\end{bmatrix} \leftarrow \begin{bmatrix}2 \cdot \mathtt{x}_1 \\ 3 \cdot \mathtt{x}_2\end{bmatrix} \\
\quad \texttt{end} \\
\quad \begin{bmatrix}\mathtt{x}_3\end{bmatrix} \leftarrow \begin{bmatrix}\mathtt{x}_3 - 1\end{bmatrix} \\
\texttt{end}
\end{array}
$$




**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Translation to ITS




$\texttt{while } (x_3 > 0) \texttt{ do}$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_4 \\ x_5^2 \end{bmatrix}$$

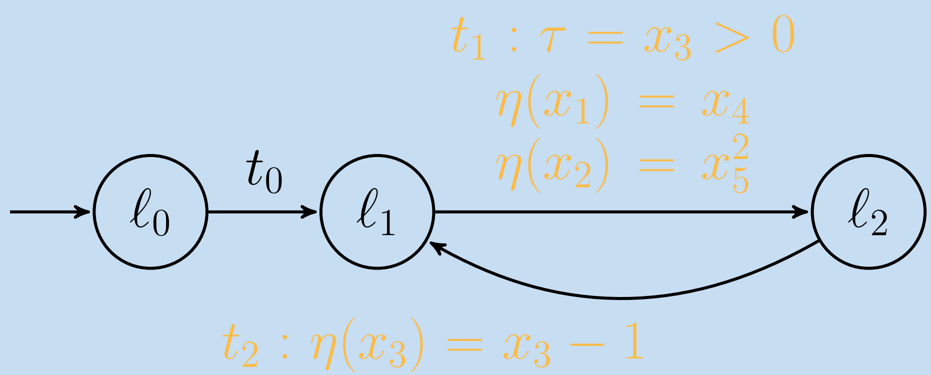
$\quad \texttt{while } (x_1^2 < x_2) \texttt{ do}$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot x_1 \\ 3 \cdot x_2 \end{bmatrix}$$

$\quad \texttt{end}$

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

$\texttt{end}$

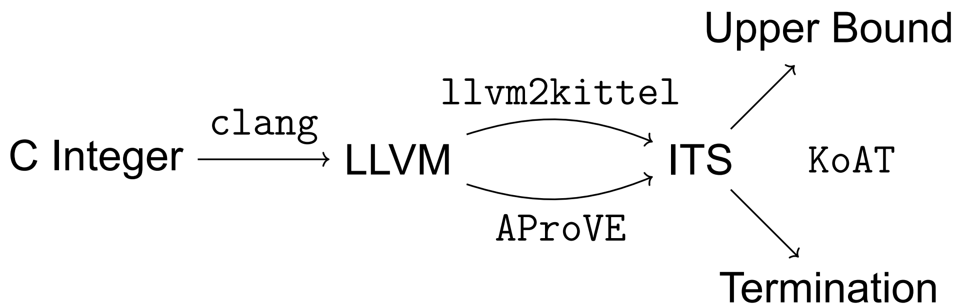




WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Translation to ITS




while $(x_3 > 0)$ do

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_4 \\ x_5^2 \end{bmatrix}$$

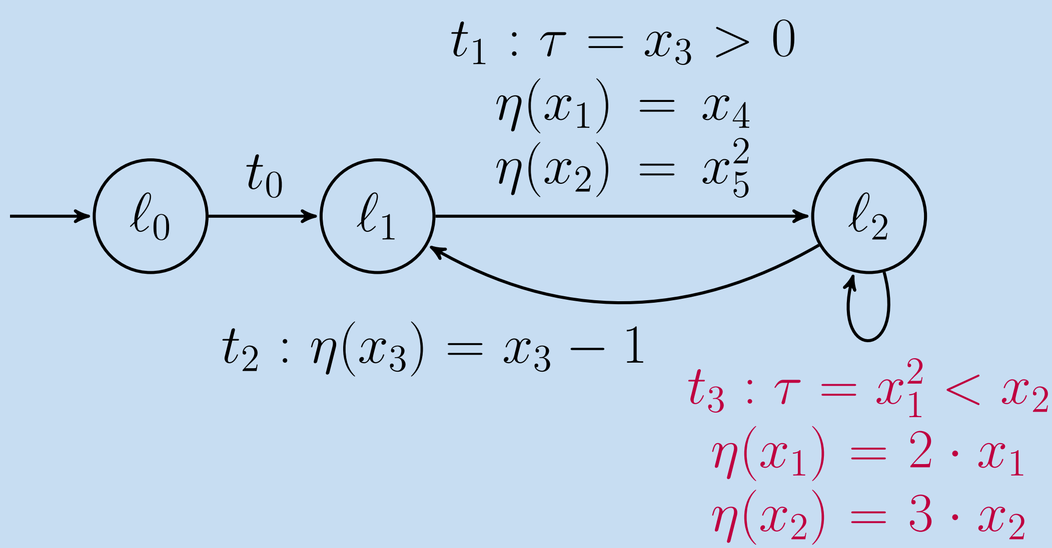
while $(x_1^2 < x_2)$ do

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot x_1 \\ 3 \cdot x_2 \end{bmatrix}$$

end

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

end



# General Architecture

- Preprocess Programs:

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?

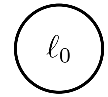

**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?

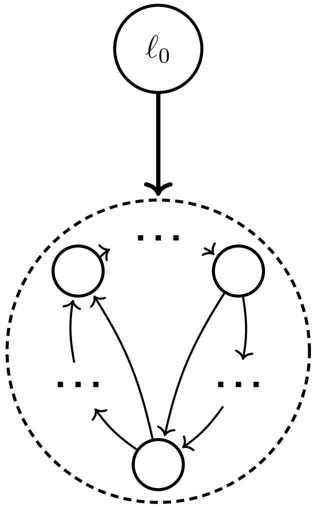
# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?



# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?

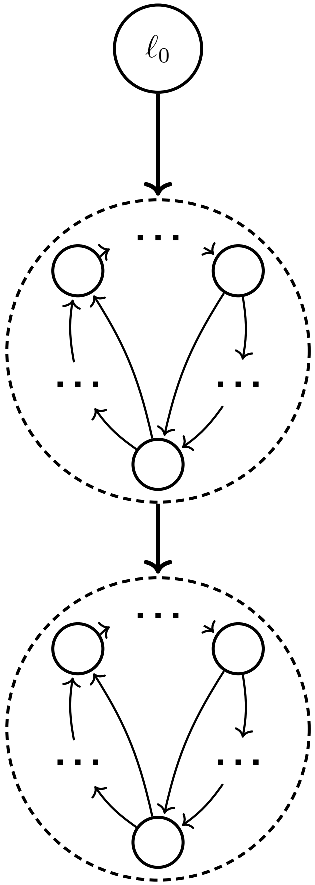




WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?

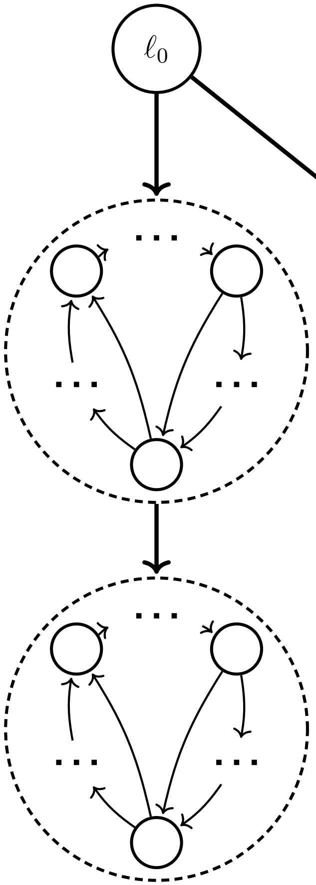




**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?

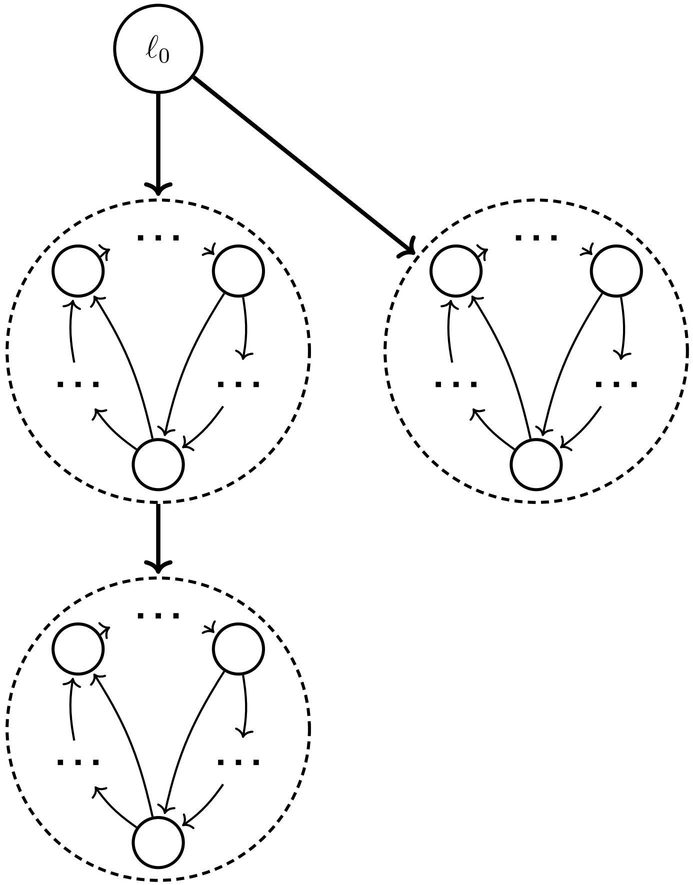





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?
- Analyze SCCs one after another

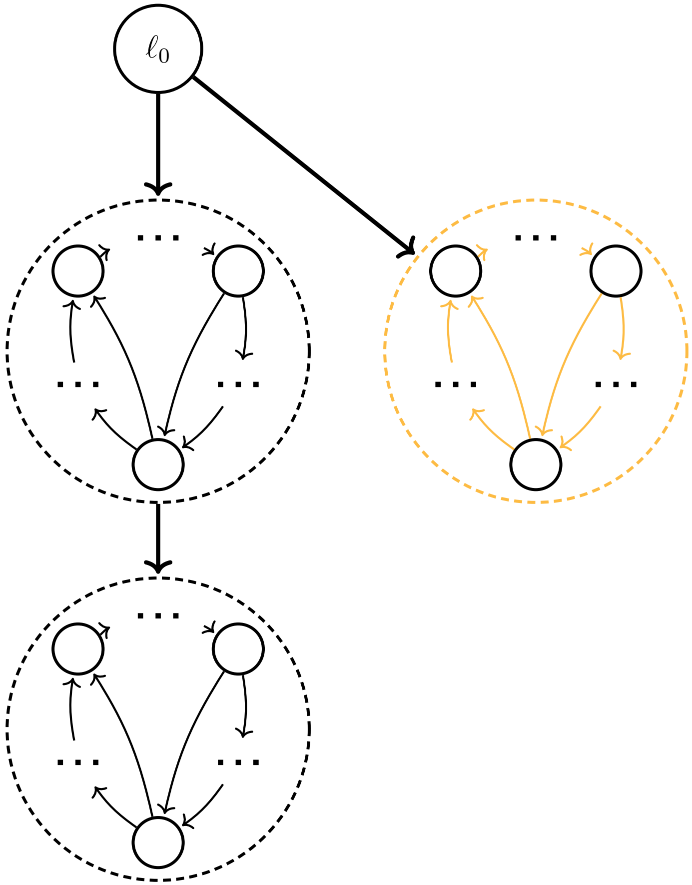





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more . . .
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?
- Analyze SCCs one after another

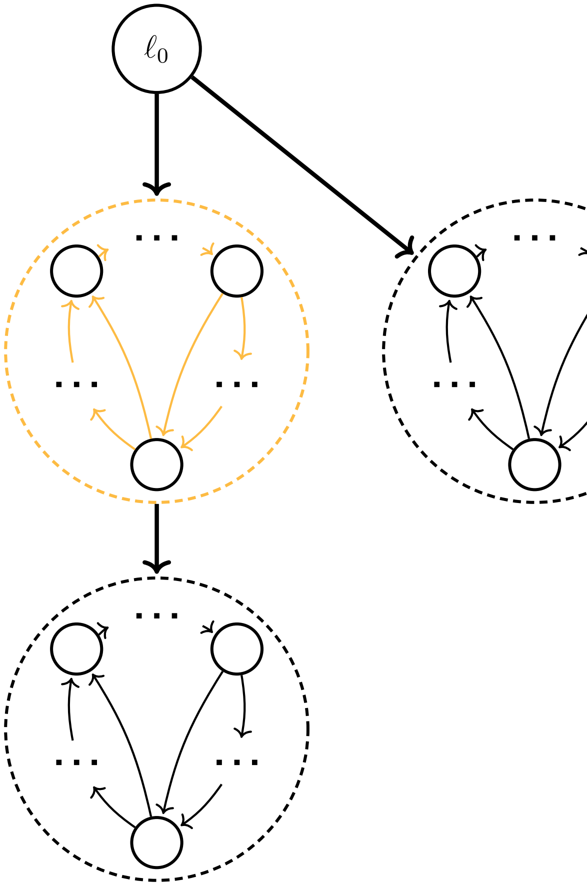




**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?
- Analyze SCCs one after another

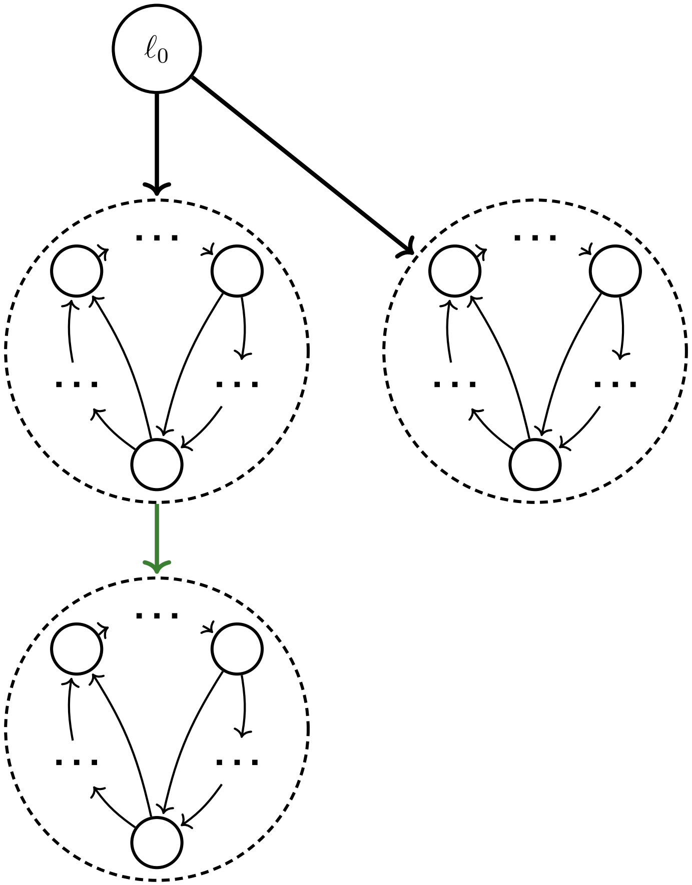




WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?
- Analyze SCCs one after another
- Propagate information from “previous” SCCs to “later” SCCs via *size bounds* [TOPLAS ’16]

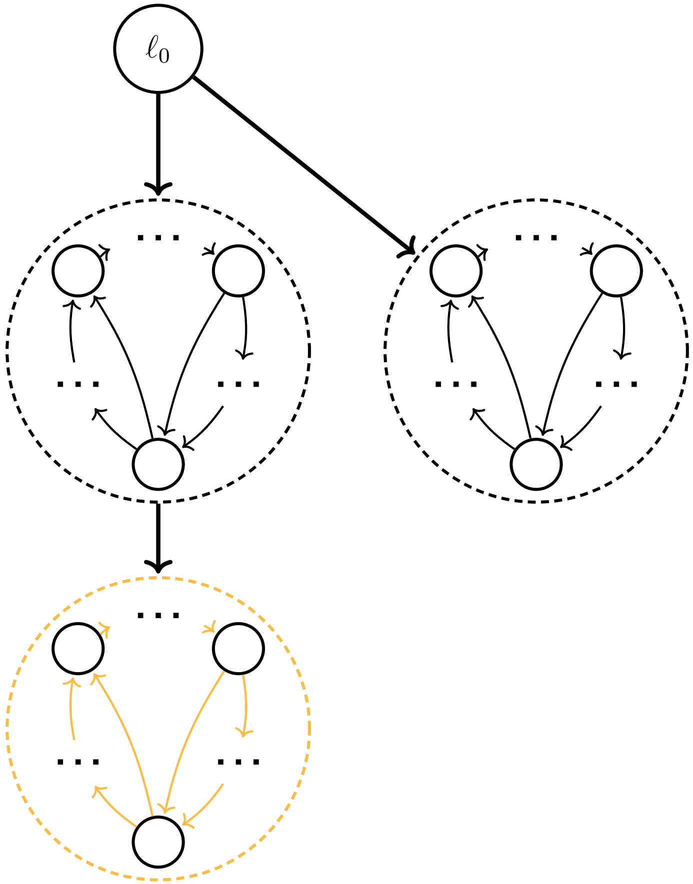





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture

- Preprocess Programs:
  - Invariants by `Apron`
  - Remove unsatisfiable Transitions
  - Remove unreachable Locations
  - many more ...
- Time bounds: How many executions?
- Size bounds: What is the value of a variable after evaluating transition?
- Analyze SCCs one after another
- Propagate information from “previous” SCCs to “later” SCCs via *size bounds* [TOPLAS ’16]

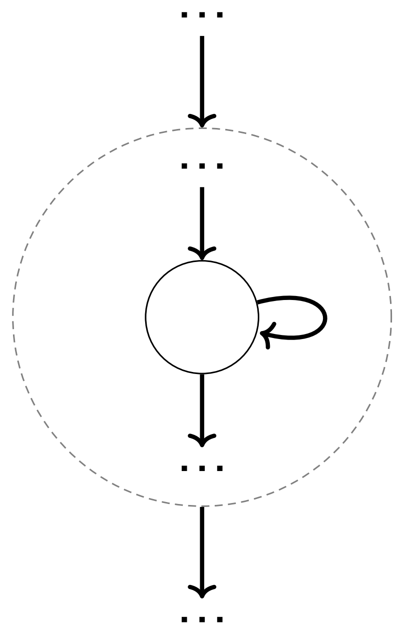





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

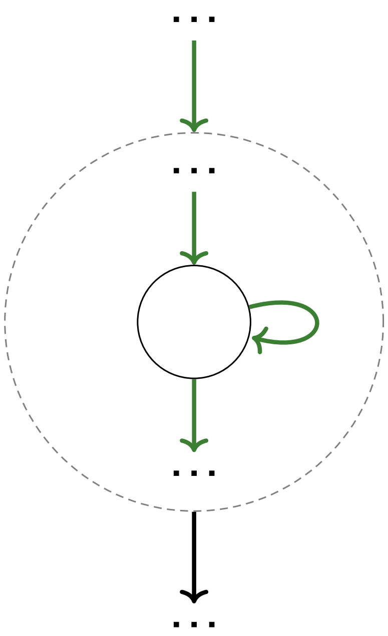
# General Architecture – Analyzing an SCC

- Alternatingly compute time and size bounds



# General Architecture – Analyzing an SCC

- Alternatingly compute time and size bounds
  - Compute initial global size bounds



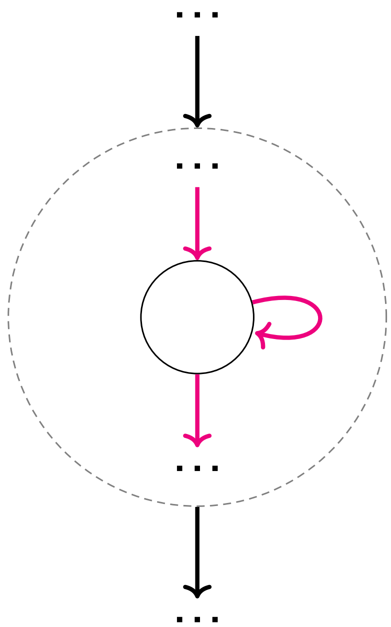





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture – Analyzing an SCC

- Alternatingly compute time and size bounds
  - Compute initial global size bounds
  - Compute as many finite global time bounds as possible

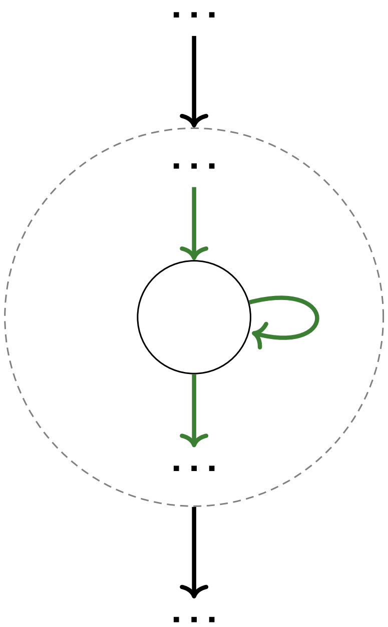




WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture – Analyzing an SCC

- Alternatingly compute time and size bounds
  - Compute initial global size bounds
  - Compute as many finite global time bounds as possible
  - Improve global size bounds



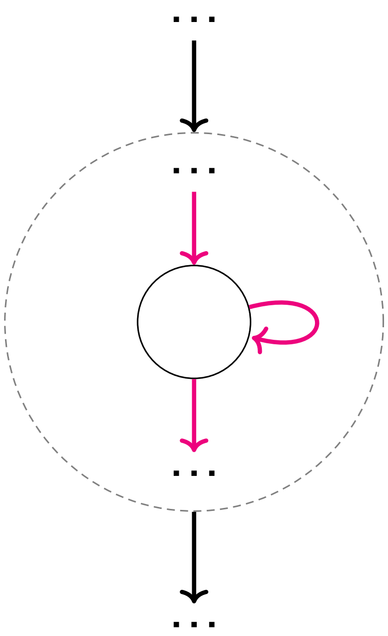





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture – Analyzing an SCC

- Alternatingly compute time and size bounds
  - Compute initial global size bounds
  - Compute as many finite global time bounds as possible
  - Improve global size bounds
  - Improve global time bounds

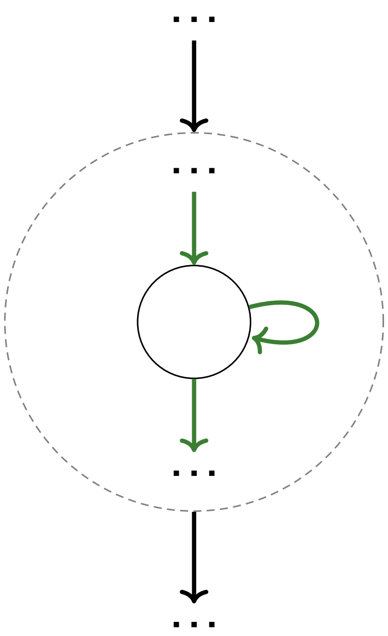




**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture – Analyzing an SCC

- Alternatingly compute time and size bounds
  - Compute initial global size bounds
  - Compute as many finite global time bounds as possible
  - Improve global size bounds
  - Improve global time bounds
  - Improve global size bounds

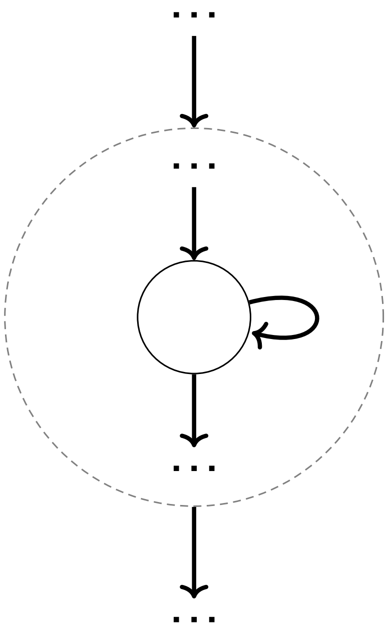




**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture – Analyzing an SCC

- Alternatingly compute time and size bounds
  - Compute initial global size bounds
  - Compute as many finite global time bounds as possible
  - Improve global size bounds
  - Improve global time bounds
  - Improve global size bounds
  - …

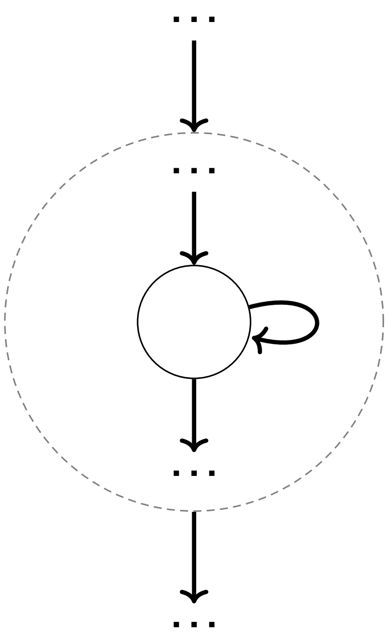





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture – Analyzing an SCC

- Alternatingly compute time and size bounds
  - Compute initial global size bounds
  - Compute as many finite global time bounds as possible
  - Improve global size bounds
  - Improve global time bounds
  - Improve global size bounds
  - …
- Possibly apply sub-SCC CFR with `iRankFinder`

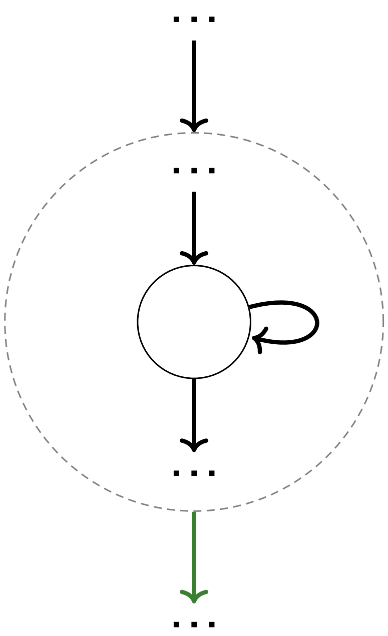




**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# General Architecture – Analyzing an SCC

- Alternatingly compute time and size bounds
  - Compute initial global size bounds
  - Compute as many finite global time bounds as possible
  - Improve global size bounds
  - Improve global time bounds
  - Improve global size bounds
  - …
- Possibly apply sub-SCC CFR with `iRankFinder`
- Analysis results are passed on via global size bounds of connecting transitions






**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Overview

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - M$\Phi$RFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]
  - local control flow-refinement by `iRankFinder` [RH '22].


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Time Bounds

- Triangular Weakly Non-Linear Loops [IJCAR ’22]


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Time Bounds

- Triangular Weakly Non-Linear Loops [IJCAR ’22]
- Multiphase-Linear Ranking Functions [Ben-Amram, Genaim] & [RH ’22]

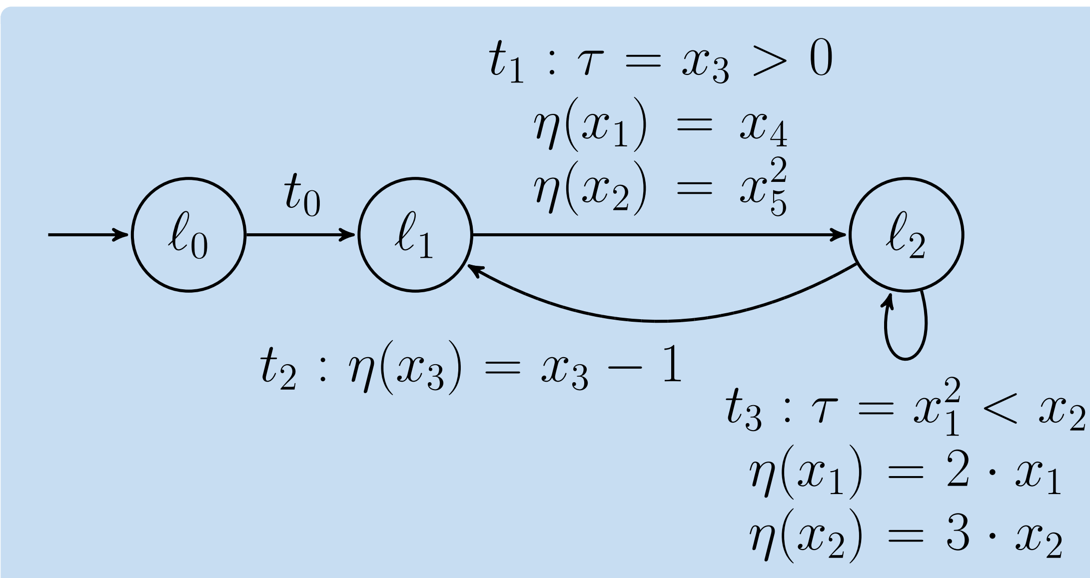
# Analyzing Programs: Time Bounds

- Triangular Weakly Non-Linear Loops [IJCAR '22]
- Multiphase-Linear Ranking Functions [Ben-Amram, Genaim] & [RH '22]
  - Try to bound $t_<$ w.r.t. subprogram $\mathcal{T}'$


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Time Bounds

- Triangular Weakly Non-Linear Loops [IJCAR '22]
- Multiphase-Linear Ranking Functions [Ben-Amram, Genaim] & [RH '22]
  - Try to bound $t_<$ w.r.t. subprogram $\mathcal{T}'$

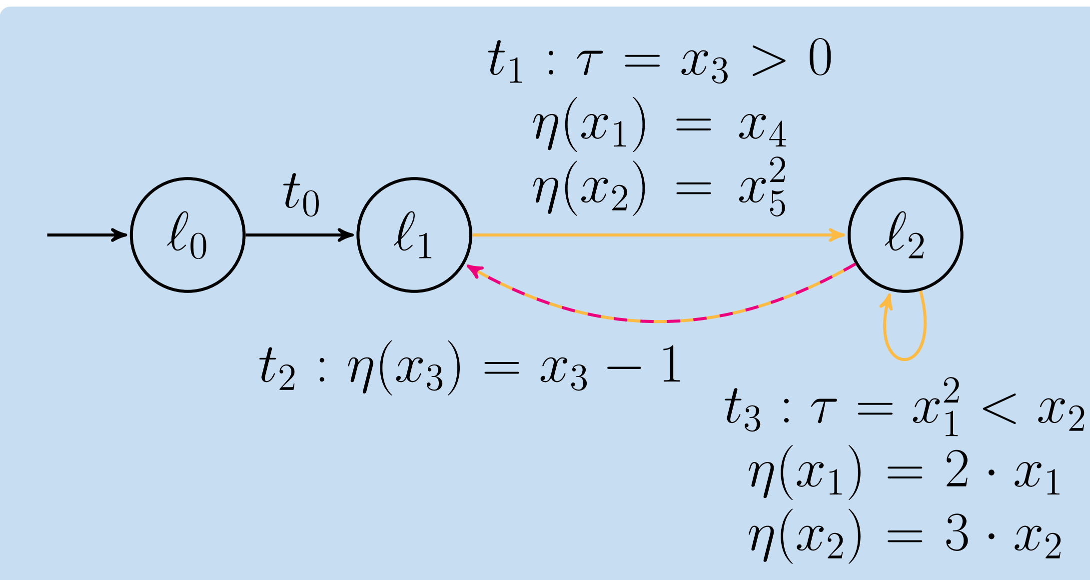





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Time Bounds

- Triangular Weakly Non-Linear Loops [IJCAR '22]
- Multiphase-Linear Ranking Functions [Ben-Amram, Genaim] & [RH '22]
  - Try to bound $t_<$ w.r.t. subprogram $\mathcal{T}'$

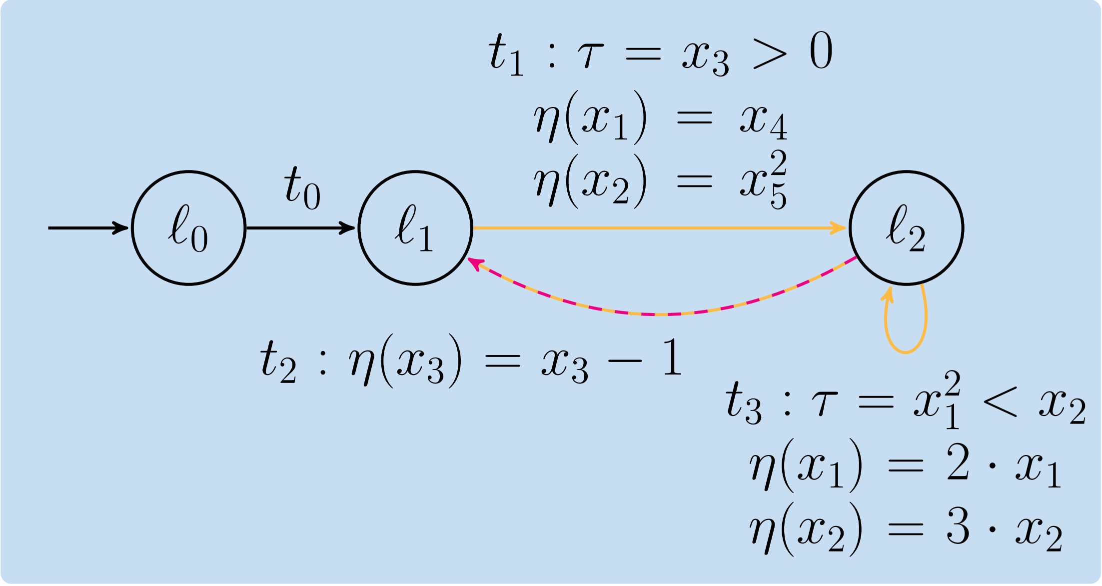



- $t_< = t_2$ and $\mathcal{T} = \{t_1, t_2, t_3\}$


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Time Bounds

- Triangular Weakly Non-Linear Loops [IJCAR '22]
- Multiphase-Linear Ranking Functions [Ben-Amram, Genaim] & [RH '22]
  - Try to bound $t_<$ w.r.t. subprogram $\mathcal{T}'$
  - $t_<$ decreases value of ranking function

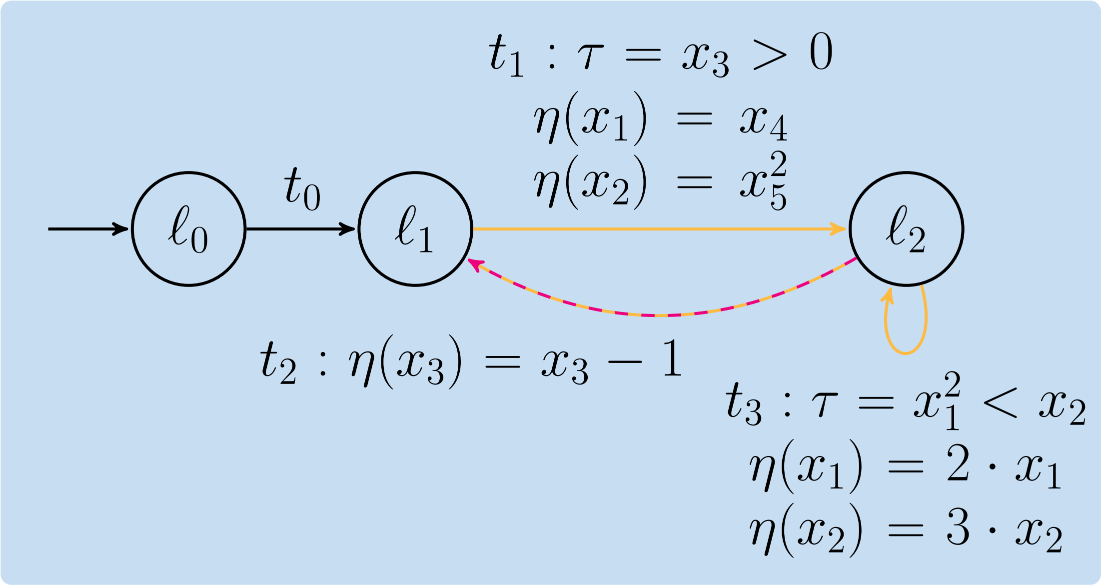



- $t_< = t_2$ and $\mathcal{T} = \{t_1, t_2, t_3\}$


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Time Bounds

- Triangular Weakly Non-Linear Loops [IJCAR '22]
- Multiphase-Linear Ranking Functions [Ben-Amram, Genaim] & [RH '22]
  - Try to bound $t_<$ w.r.t. subprogram $\mathcal{T}'$
  - $t_<$ decreases value of ranking function
  - $\mathcal{T}_{\subseteq} \setminus \{t_<\}$ does not increase value of ranking function

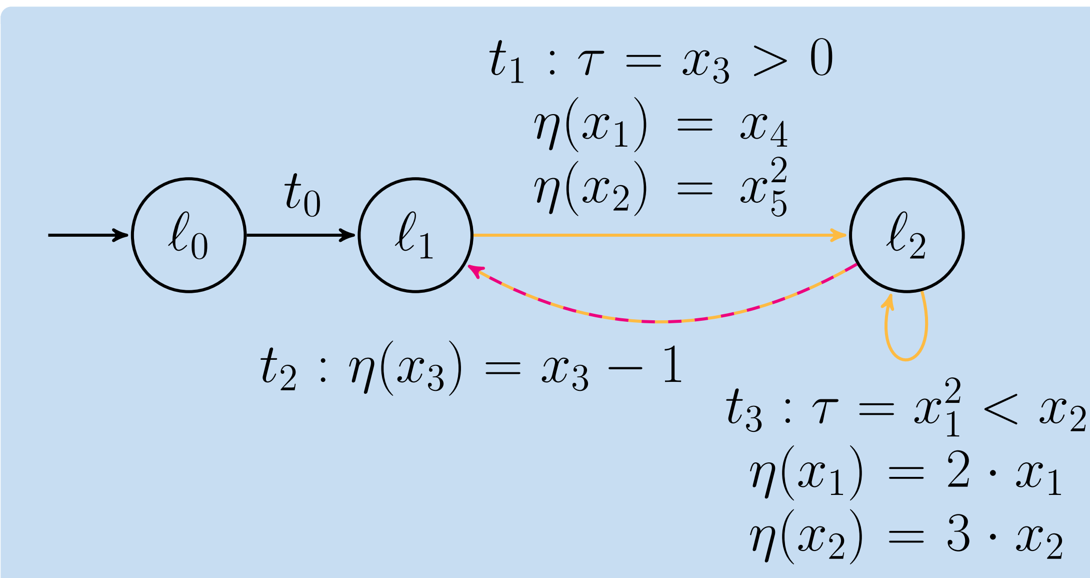



- $t_< = t_2$ and $\mathcal{T} = \{t_1, t_2, t_3\}$


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Time Bounds

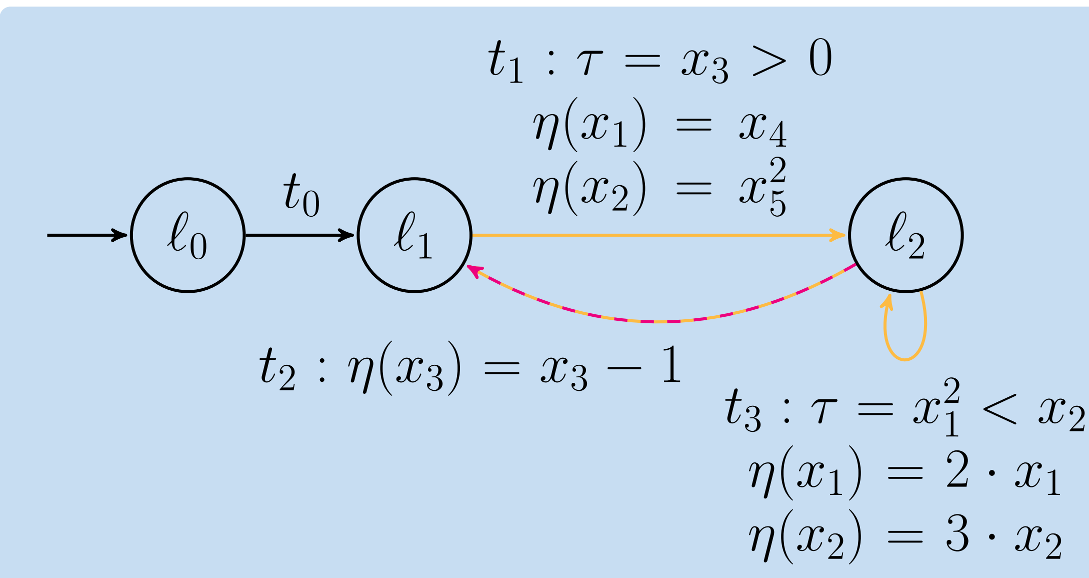
- Triangular Weakly Non-Linear Loops [IJCAR '22]
- Multiphase-Linear Ranking Functions [Ben-Amram, Genaim] & [RH '22]
  - Try to bound $t_<$ w.r.t. subprogram $\mathcal{T}'$
  - $t_<$ decreases value of ranking function
  - $\mathcal{T}_\subseteq \setminus \{t_<\}$ does not increase value of ranking function




- $t_< = t_2$ and $\mathcal{T} = \{t_1, t_2, t_3\}$
- ranking function $x_3$ yields well-founded order on $\mathbb{N}$

# Analyzing Programs: Time Bounds

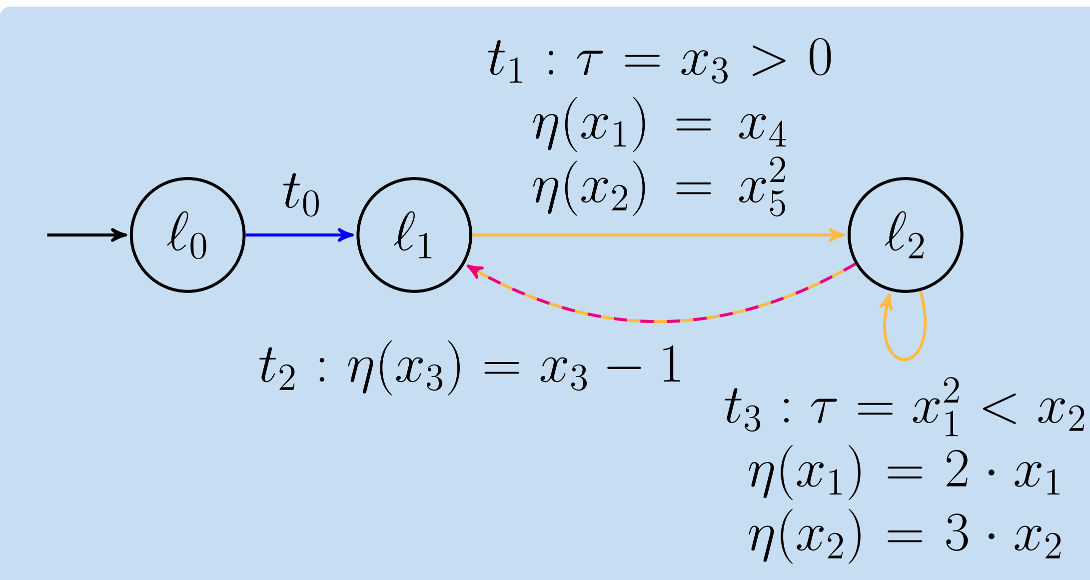
- Triangular Weakly Non-Linear Loops [IJCAR ’22]
- Multiphase-Linear Ranking Functions [Ben-Amram, Genaim] & [RH ’22]
  - Try to bound $t_<$ w.r.t. subprogram $\mathcal{T}'$
  - $t_<$ decreases value of ranking function
  - $\mathcal{T}_\subseteq \setminus \{t_<\}$ does not increase value of ranking function
  - Lift *local* time bound for subprogram to time bound for complete program




- $t_< = t_2$ and $\mathcal{T} = \{t_1, t_2, t_3\}$
- ranking function $x_3$ yields well-founded order on $\mathbb{N}$

# Analyzing Programs: Time Bounds

- Triangular Weakly Non-Linear Loops [IJCAR ’22]
- Multiphase-Linear Ranking Functions [Ben-Amram, Genaim] & [RH ’22]
  - Try to bound $t_<$ w.r.t. subprogram $\mathcal{T}'$
  - $t_<$ decreases value of ranking function
  - $\mathcal{T}_\subseteq \setminus \{t_<\}$ does not increase value of ranking function
  - Lift *local* time bound for subprogram to time bound for complete program




- $t_< = t_2$ and $\mathcal{T} = \{t_1, t_2, t_3\}$
- ranking function $x_3$ yields well-founded order on $\mathbb{N}$
- consider entry transition $t_0$


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Overview

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - M$\Phi$RFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]
  - local control flow-refinement by `iRankFinder` [RH '22].


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS ’23]



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS ’23]
  - Compute closed form for loop


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS '23]
  - Compute closed form for loop

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


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS '23]
  - Compute closed form for loop

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

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


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS '23]
  - Compute closed form for loop
  - Over-approximate closed form to non-negative, weakly monotonic increasing expression

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

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


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS '23]
  - Compute closed form for loop
  - Over-approximate closed form to non-negative, weakly monotonic increasing expression

```
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
```

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

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS '23]
  - Compute closed form for loop
  - Over-approximate closed form to non-negative, weakly monotonic increasing expression
  - Substitute $n$ by runtime bound

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

- Closed form:

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

- Over-approximation:

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

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS '23]
  - Compute closed form for loop
  - Over-approximate closed form to non-negative, weakly monotonic increasing expression
  - Substitute $n$ by runtime bound

```
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
```

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

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS ’23]
  - Compute closed form for loop
  - Over-approximate closed form to non-negative, weakly monotonic increasing expression
  - Substitute $n$ by runtime bound
- Changed accumulated size bounds [TOPLAS ’16]

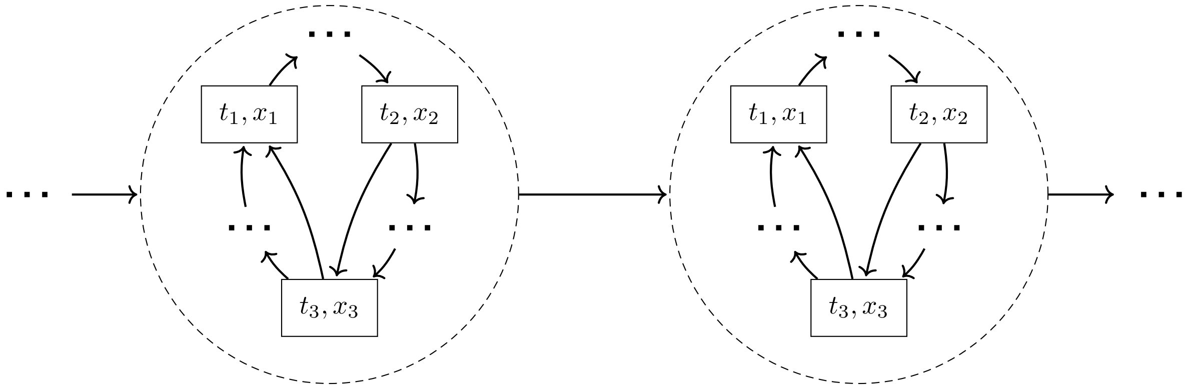
# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS '23]
  - Compute closed form for loop
  - Over-approximate closed form to non-negative, weakly monotonic increasing expression
  - Substitute $n$ by runtime bound
- Changed accumulated size bounds [TOPLAS '16]
  - Construct variable dependence graph


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS ’23]
  - Compute closed form for loop
  - Over-approximate closed form to non-negative, weakly monotonic increasing expression
  - Substitute $n$ by runtime bound
- Changed accumulated size bounds [TOPLAS ’16]
  - Construct variable dependence graph

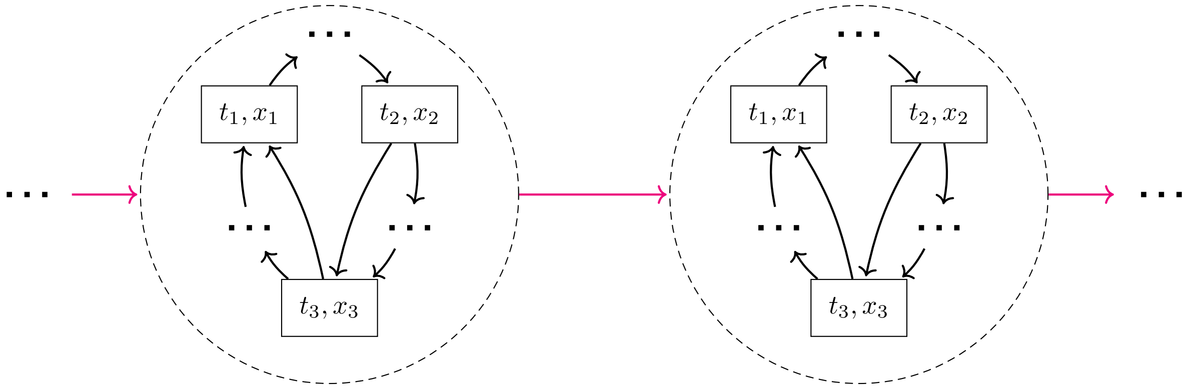





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS ’23]
  - Compute closed form for loop
  - Over-approximate closed form to non-negative, weakly monotonic increasing expression
  - Substitute $n$ by runtime bound
- Changed accumulated size bounds [TOPLAS ’16]
  - Construct variable dependence graph
  - Connecting transitions: Directly apply updates

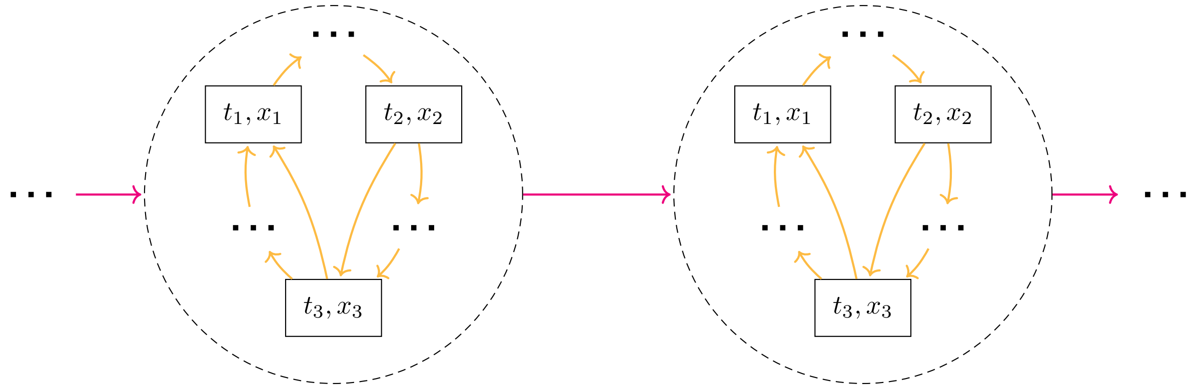





**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Analyzing Programs: Size Bounds

- Size bounds by closed forms [FroCoS '23]
  - Compute closed form for loop
  - Over-approximate closed form to non-negative, weakly monotonic increasing expression
  - Substitute $n$ by runtime bound
- Changed accumulated size bounds [TOPLAS '16]
  - Construct variable dependence graph
  - Connecting transitions: Directly apply updates
  - SCC transitions: Capture repetitions by using runtime bounds



# Overview

- `KoAT` uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - M$\Phi$RFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]
  - local control flow-refinement by `iRankFinder` [RH '22].

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops
- Loop consists of *two* phases:

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops
- Loop consists of *two* phases:
  1. **then**-case is repeated until $\mathtt{y} \leq 0$

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops
- Loop consists of *two* phases:
  1. **then**-case is repeated until $y \leq 0$
  2. **else**-case is repeated until $x \leq 0$

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops
- Loop consists of *two* phases:

  1. **then**-case is repeated until $y \leq 0$
  2. **else**-case is repeated until $x \leq 0$

  $\Rightarrow$ No run, where second phase is executed before first phase

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops
- Loop consists of *two* phases:
  1. **then**-case is repeated until $y \leq 0$
  2. **else**-case is repeated until $x \leq 0$

  $\Rightarrow$ No run, where second phase is executed before first phase

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```

**Control-Flow Refinement by Partial Evaluation (CFR)** [Doménech et al. '19]


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops
- Loop consists of *two* phases:
  1. **then**-case is repeated until $y \le 0$
  2. **else**-case is repeated until $x \le 0$

  $\Rightarrow$ No run, where second phase is executed before first phase

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```

**Control-Flow Refinement by Partial Evaluation (CFR)** [Doménech et al. ’19]

- sort out certain program paths


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops
- Loop consists of *two* phases:
  1. **then**-case is repeated until $y \leq 0$
  2. **else**-case is repeated until $x \leq 0$

  $\Rightarrow$ No run, where second phase is executed before first phase

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```

$\rightsquigarrow$

```
while (x > 0 ∧ y > 0) do
  y ← y − x
end
while (x > 0 ∧ y ≤ 0) do
  x ← x − 1
end
```

**Control-Flow Refinement by Partial Evaluation (CFR)** [Doménech et al. ’19]

- sort out certain program paths

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops
- Loop consists of *two* phases:
  1. **then**-case is repeated until $y \le 0$
  2. **else**-case is repeated until $x \le 0$

  $\Rightarrow$ No run, where second phase is executed before first phase

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```

$\rightsquigarrow$

```
while (x > 0 ∧ y > 0) do
  y ← y − x
end
while (x > 0 ∧ y ≤ 0) do
  x ← x − 1
end
```

**Control-Flow Refinement by Partial Evaluation (CFR)** [Doménech et al. ’19]

- sort out certain program paths

$\Rightarrow$ integrate CFR into our modular approach


WST 2023
**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Integrating Control-Flow Refinement

- **Problem**: complex, nested loops
- Loop consists of *two* phases:
  1. **then**-case is repeated until $y \leq 0$
  2. **else**-case is repeated until $x \leq 0$

  $\Rightarrow$ No run, where second phase is executed before first phase

```
while (x > 0) do
  if (y > 0) then
    y ← y − x
  else
    x ← x − 1
end
```

$\rightsquigarrow$

```
while (x > 0 ∧ y > 0) do
  y ← y − x
end
while (x > 0 ∧ y ≤ 0) do
  x ← x − 1
end
```

**Control-Flow Refinement by Partial Evaluation (CFR)** [Doménech et al. ’19]

- sort out certain program paths

$\Rightarrow$ integrate CFR into our modular approach

- CFR *modular* for SCCs with “problematic” transitions on-demand [RH ’22]

# Live-Demo

```
(GOAL COMPLEXITY)
(STARTTERM (FUNCTIONSYMBOLS l0))
(VAR A B C D E)
(RULES
  l0(A,B,C,D,E) -> l1(A,B,C,D,E) :|: C > 0
  l1(A,B,C,D,E) -> l2(D,E^2,C,D,E) :|: C > 0
  l2(A,B,C,D,E) -> l2(2*A,3*B,C,D,E) :|: A^2 < B && A > 0
  l2(A,B,C,D,E) -> l1(A,B,C - 1,D,E)
)
```

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


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Conclusion

- `KoAT` uses

# Conclusion

- KoAT uses
  - a modular approach to compute time bounds combining

# Conclusion

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops
      [IJCAR ’22]


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Conclusion

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR ’22]
    - MΦRFs [RH ’22]

# Conclusion

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR ’22]
    - M$\Phi$RFs [RH ’22]
  - a modular approach to compute size bounds combining


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Conclusion

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR ’22]
    - MΦRFs [RH ’22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS ’23]

# Conclusion

- `KoAT` uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - MΦRFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]


**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Conclusion

- `KoAT` uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - M$\Phi$RFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]
  - local control flow-refinement by `iRankFinder` [RH '22].

# Conclusion

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - MΦRFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]
  - local control flow-refinement by `iRankFinder` [RH '22].

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



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Conclusion

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - MΦRFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]
  - local control flow-refinement by iRankFinder [RH '22].

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



**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2

# Conclusion

- KoAT uses
  - a modular approach to compute time bounds combining
    - a procedure to handle twn-loops [IJCAR '22]
    - MΦRFs [RH '22]
  - a modular approach to compute size bounds combining
    - a procedure using closed-forms [FroCoS '23]
    - changed accumulated size bounds [TOPLAS '16]
  - local control flow-refinement by iRankFinder [RH '22].

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

Thank You!




**Nils Lommen**, Eleanore Meyer, and Jürgen Giesl
RWTH Aachen University – LuFGi2