Optimization of Model Checking by Large Block Encoding

Thomas Mertens

24.09.2014
Outline

1. Motivation

2. Background
   - Program and Control-Flow Automaton
   - Model Checker

3. Large-Block Encoding
   - Summarization
   - Post-processing

4. Evaluation

5. Conclusion
Outline

1. Motivation

2. Background
   - Program and Control-Flow Automaton
   - Model Checker

3. Large-Block Encoding
   - Summarization
   - Post-processing

4. Evaluation

5. Conclusion
Why do we do Model Checking?

Reduce runtime and memory of model checking in particular CEGAR through minimizing the abstract reachability tree (ART).
Motivation

Why do we do Model Checking?

- Reduce runtime and memory of model checking in particular CEGAR through minimizing the abstract reachability tree (ART)
Outline

1 Motivation

2 Background
   - Program and Control-Flow Automaton
   - Model Checker

3 Large-Block Encoding
   - Summarization
   - Post-processing

4 Evaluation

5 Conclusion
Definition (Program)
A sequential program \( P \) with the operations \( Ops \) is defined as
\[
P = op_0, op_1, \ldots, op_n \text{ with } op_i \in Ops.
\]
Definition (Program)
A sequential program $P$ with the operations $Ops$ is defined as $P = op_0, op_1, ..., op_n$ with $op_i \in Ops$.

Definition (Control-Flow Automaton)
A Control Flow Automaton is defined as $A = (L, G, l_0, L_E)$, where $L$ is a finite set of locations, $G \subseteq L \times Ops \times L$, $l_0 \in L$ is a unique initial location and $L_E \subseteq L$ is a set of error locations.
CEGAR

- Initial Abstraction
- Model Checker
  - Assertion is not violated
  - Assertion is violated
  - Counterexample Analysis
    - real
    - spurious
  - Unsafe
- Abstraction Refinement
  - P
  - P'
  - P''
- Safe
- Optimization of Model Checking by Large Block Encoding
Outline

1. Motivation

2. Background
   - Program and Control-Flow Automaton
   - Model Checker

3. Large-Block Encoding
   - Summarization
   - Post-processing

4. Evaluation

5. Conclusion
Definition

Let $l_e \in L_E$ with $\text{outdegree}(l_e) \geq 1$ and $G_{out} = \{(l_e, op_x, l_x) \in G\}$. The error sink rule removes all edges $g \in G_{out}$. 
Error sink rule (Rule 1)

Definition

Let $l_e \in L_E$ with $\text{outdegree}(l_e) \geq 1$ and $G_{out} = \{(l_e, op_x, l_x) \in G\}$. The error sink rule removes all edges $g \in G_{out}$. 

- Complexity $O(k)$, where $k$ is the outdegree of $l_e$. 

![Diagram of the error sink rule](image-url)
Definition

Let $l_e \in L_E$ with $\text{outdegree}(l_e) \geq 1$ and $G_{out} = \{(l_e, op_x, l_x) \in G\}$. The error sink rule removes all edges $g \in G_{out}$. 

Error sink rule (Rule 1)
Error sink rule (Rule 1)

**Definition**

Let $l_e \in L_E$ with $\text{outdegree}(l_e) \geq 1$ and $G_{\text{out}} = \{(l_e, op_x, l_x) \in G\}$. The error sink rule removes all edges $g \in G_{\text{out}}$.

**Complexity**

$O(k)$, where $k$ is the outdegree of $l_e$.
**Sequence rule (Rule 2)**

**Definition**

Let \( l_x \in L \setminus L_E \) with \( \text{indegree}(l_x) = 1 \), \( \text{outdegree}(l_x) \geq 1 \), \( l_x \notin \text{succ}(l_x) \) and the edge \((l_y, op_y, l_x) \in G\), \( G_{out} = \{(l_x, op_x, l_z) \in G\} \). The sequence rule adds new edges with \((l_y, op_y; op_x, l_z) \forall g \in G_{out}\) to the CFA and removes all edges \( g \in \{(l_y, op_y, l_x) \in G\} \cup G_{out}\).
**Sequence rule (Rule 2)**

**Definition**

Let $l_x \in L \setminus L_E$ with $\text{indegree}(l_x) = 1$, $\text{outdegree}(l_x) \geq 1$, $l_x \notin \text{succ}(l_x)$ and the edge $(l_y, op_y, l_x) \in G$, $G_{out} = \{(l_x, op_x, l_z) \in G\}$. The sequence rule adds new edges with $(l_y, op_y; op_x, l_z) \forall g \in G_{out}$ to the CFA and removes all edges $g \in \{(l_y, op_y, l_x) \in G\} \cup G_{out}$. 

![Diagram](image-url)
Sequence rule (Rule 2)

Definition

Let \( l_x \in L \setminus L_E \) with \( \text{indegree}(l_x) = 1 \), \( \text{outdegree}(l_x) \geq 1 \), \( l_x \not\in \text{succ}(l_x) \) and the edge \( (l_y, op_y, l_x) \in G \),
\[
G_{out} = \{(l_x, op_x, l_z) \in G\}.
\]
The sequence rule adds new edges with \( (l_y, op_y; op_x, l_z) \ \forall g \in G_{out} \) to the CFA and removes all edges \( g \in \{(l_y, op_y, l_x) \in G\} \cup G_{out} \).
Sequence rule (Rule 2)

Definition
Let $l_x \in L \setminus L_E$ with $\text{indegree}(l_x) = 1$, $\text{outdegree}(l_x) \geq 1$, $l_x \notin \text{succ}(l_x)$ and the edge $(l_y, op_y, l_x) \in G$, $G_{out} = \{(l_x, op_x, l_z) \in G\}$. The sequence rule adds new edges with $(l_y, op_y; op_x, l_z) \forall g \in G_{out}$ to the CFA and removes all edges $g \in \{(l_y, op_y, l_x) \in G\} \cup G_{out}$.

Complexity
$O(k)$, where $k$ is the outdegree of $l_y$
Choice rule (Rule 3)

Definition

Let \( l_x \in L \setminus L_E \) with \( l_y \in \text{succ}(l_x) \),
\( G_{\text{con}} = \text{connecting\_edges}(l_x, l_y) \) and
\( |G_{\text{con}}| = n > 1 \).
The choice rule adds a new edge
\( (l_x, op, l_y) \) with
\( op = op_1 \| op_2 \| \ldots \| op_n \) to the
CFA and removes all edges \( g \in G_{\text{con}} \).
Choice rule (Rule 3)

Definition

Let $l_x \in L \setminus L_E$ with $l_y \in \text{succ}(l_x)$, $G_{\text{con}} = \text{connecting \_ edges}(l_x, l_y)$ and $|G_{\text{con}}| = n > 1$.

The choice rule adds a new edge $(l_x, \text{op}, l_y)$ with $\text{op} = \text{op}_1 \parallel \text{op}_2 \parallel \ldots \parallel \text{op}_n$ to the CFA and removes all edges $g \in G_{\text{con}}$. 

\[ \text{ Complexity } \mathcal{O}(k), \text{ where } k \text{ is the outdegree of } l_x. \]
Choice rule (Rule 3)

Definition

Let $l_x \in L \setminus L_E$ with $l_y \in \text{succ}(l_x)$, $G_{con} = \text{connecting}\_\text{edges}(l_x, l_y)$ and $|G_{con}| = n > 1$.

The choice rule adds a new edge $(l_x, op, l_y)$ with $op = op_1 \parallel op_2 \parallel \ldots \parallel op_n$ to the CFA and removes all edges $g \in G_{con}$.
Choice rule (Rule 3)

Definition
Let $l_x \in L \setminus L_E$ with $l_y \in \text{succ}(l_x)$, $G_{\text{con}} = \text{connecting edges}(l_x, l_y)$ and $|G_{\text{con}}| = n > 1$.

The choice rule adds a new edge $(l_x, op, l_y)$ with $op = op_1 \parallel op_2 \parallel \ldots \parallel op_n$ to the CFA and removes all edges $g \in G_{\text{con}}$.

Complexity
$O(k)$, where $k$ is the outdegree of $l_x$
Advanced sequence rule (Rule 4)

Definition

Let $l_x \in L \setminus L_E$ with $\text{indegree}(l_x) > 1$, $\text{outdegree}(l_x) > 1$, $l_x \not\in \text{succ}(l_x)$ and $G_{\text{in}} = \{(l_y, op_y, l_x) \in G\}$, $G_{\text{out}} = \{(l_x, op_x, l_z) \in G\}$.

The sequence rule adds new edges with $(l_y, op_y; op_x, l_z)$ for all possible connections of edges in $G_{\text{in}}$ and $G_{\text{out}}$ to the CFA and removes all edges $g \in G_{\text{in}} \cup G_{\text{out}}$. 
Advanced sequence rule (Rule 4)

**Definition**

Let \( l_x \in L \setminus L_E \) with \( \text{indegree}(l_x) > 1 \), \( \text{outdegree}(l_x) > 1 \), \( l_x \notin \text{succ}(l_x) \) and

\[
G_{in} = \{ (l_y, op_y, l_x) \in G \},
\]

\[
G_{out} = \{ (l_x, op_x, l_z) \in G \}.
\]

The sequence rule adds new edges with \((l_y, op_y; op_x, l_z)\) for all possible connections of edges in \( G_{in} \) and \( G_{out} \) to the CFA and removes all edges \( g \in G_{in} \cup G_{out} \).
Advanced sequence rule (Rule 4)

Definition

Let $l_x \in L \setminus L_E$ with $\text{indegree}(l_x) > 1$, $\text{outdegree}(l_x) > 1$, $l_x \notin \text{succ}(l_x)$ and $G_{in} = \{(l_y, op_y, l_x) \in G\}$, $G_{out} = \{(l_x, op_x, l_z) \in G\}$.

The sequence rule adds new edges with $(l_y, op_y; op_x, l_z)$ for all possible connections of edges in $G_{in}$ and $G_{out}$ to the CFA and removes all edges $g \in G_{in} \cup G_{out}$. 

Complexity $O(k \cdot m)$, where $k$ is the outdegree of $l_x \in L$ and $m$ is the indegree of $l_x \in L$. 
Advanced sequence rule (Rule 4)

**Definition**

Let $l_x \in L \setminus L_E$ with $\text{indegree}(l_x) > 1$, $\text{outdegree}(l_x) > 1$, $l_x \notin \text{succ}(l_x)$ and $G_{in} = \{(l_y, op_y, l_x) \in G\}$, $G_{out} = \{(l_x, op_x, l_z) \in G\}$. The sequence rule adds new edges with $(l_y, op_y; op_x, l_z)$ for all possible connections of edges in $G_{in}$ and $G_{out}$ to the CFA and removes all edges $g \in G_{in} \cup G_{out}$.

**Complexity**

$O(k \cdot m)$, where $k$ is the outdegree of $l_x \in L$ and $m$ is the indegree of $l_x \in L$.
Rule 1: $\mathcal{O}(k)$, where $k$ is the outdegree of $l_e$

Rule 2: $\mathcal{O}(k)$, where $k$ is the outdegree of $l_y \in L$

Rule 3: $\mathcal{O}(k)$, where $k$ is the outdegree of $l_y$

Rule 4: $\mathcal{O}(k \cdot m)$, where $k$ is the outdegree of $l_x \in L$ and $m$ is the indegree of $l_x \in L$
Complexity

- Rule 1: $O(k)$, where $k$ is the outdegree of $l_e$
- Rule 2: $O(k)$, where $k$ is the outdegree of $l_y \in L$
- Rule 3: $O(k)$, where $k$ is the outdegree of $l_y$
- Rule 4: $O(k \cdot m)$, where $k$ is the outdegree of $l_x \in L$ and $m$ is the indegree of $l_x \in L$

Complexity in total

$O(|G| \cdot |L| \cdot k \cdot m)$
Example

The diagram illustrates a state transition graph with the following transitions and conditions:

- From state I3 to I2: \( z' = 6 \) with condition \( x \neq 0 \)
- From state I2 to I1: \( y' = 6 \) with condition \( x = 12 \)
- From state I1 to I0: \( x' = 12 \) with condition \( y = 0 \)
- From state I4 to I9: \( y = 0 \)
- From state I5 to I4: \( y \neq 0 \)
- From state I7 to I8: \( x' = x + 0 \)
- From state I8 to I7: \( x \leq y \) with condition \( y' = x - y \)
- From state I8 to I10: \( x' = x - y \)
- From state I10 to I11: \( x = z \)
- From state I11 to I12: \( y = z \)
- From state I11 to I10: \( y \neq z \)
- From state I12 to I11: \( y \neq z \)

The states and transitions are labeled with conditions and guard conditions, indicating the flow and constraints of the system.
Example

\[x = 0\]
\[y = 0\]
\[y = z\]
\[x = z\]
\[x' = x + 0\]
\[x' = x - y\]
\[y' = x - y\]
\[x' = 12\]
\[y' = 6\]
\[z' = 6\]
Example

Graph representation of example with transitions:
- From state 13: $z' = 6$ (to state 12), $x' = 12; y' = 6$ (to state 10), $x \neq 0$ (to state 14), $y \neq 0$ (to state 15), $x' = x + 0$ (to state 17).
- From state 12: $x' = 12; y' = 6$ (to state 10), $x \neq 0$ (to state 14).
- From state 10: $x = 0$ (to state 19).
- From state 14: $y' = 0$ (to state 15), $y 
eq 0$ (to state 19).
- From state 15: $x \geq y$ (to state 16), $x = z$ (to state 11).
- From state 16: $x' = x - y$ (to state 18), $x' = x - y$ (to state 12), $x 
eq z$ (to state 17).
- From state 17: $y' = x - y$ (to state 18), $x 
eq z$ (to state 12).
- From state 18: $x' = x - y$ (to state 12), $x 
eq z$ (to state 17).
- From state 19: $y = z$ (to state 11).
- From state 11: $y 
eq z$ (to state 12).
- From state 12: $x = z$ (to state 11).
- From state 17: $z' = 6$ (to state 13), $x' = x + 0$ (to state 17), $x 
eq z$ (to state 17).
- From state 18: $x' = x - y$ (to state 12), $x 
eq z$ (to state 17).
- From state 19: $y = z$ (to state 11).
- From state 27: $y = z$ (to state 27).
- From state 28: $x' = x - y$ (to state 27), $x' = x - y$ (to state 27).
Example

\[ x' = x + 0 \]
\[ y' = x - y \]
\[ x \neq 0 \]
\[ y \neq 0 \]
\[ x \leq y \]
\[ x > y \]
\[ x = 0 \]
\[ y = 0 \]
\[ z' = 6 \]
\[ x' = 12; y' = 6 \]
\[ x' = 12; y' = 6; z' = 6 \]
\[ x' = x - y \]
\[ y = z \]
\[ y \neq z \]
\[ x = z \]
\[ x \neq z \]
\[ x \neq 0; y \neq 0 \]
\[ x \neq 0; y = 0; x \neq z \]
\[ x = 0; y = z \]
\[ x 
\neq 0; y = 0; x \neq z \]
\[ x = 0; y = z \]

Thomas Mertens  
Optimization of Model Checking by Large Block Encoding  
14 / 23
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x \neq 0 \]

\[ y \neq 0 \]

\[ y = 0 \]

\[ x = 0 \]

\[ y = z \]

\[ x = z \]

\[ y \neq z \]

\[ x' = x + 0 \]

\[ y' = x - y \]

\[ x' = x - y \]

\[ x' = x - y \]

\[ x' = x - y \]

\[ x' = x - y \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x' = x + 0 \]

\[ x \neq 0 \]

\[ y \neq 0 \]

\[ y = 0 \]

\[ y = z \]

\[ y' = x - y \]

\[ x' = x - y \]

\[ x' = x - y \]

\[ x \leq y \]

\[ x > y \]

\[ x = z \]

\[ x \neq z \]

\[ x = 0 \]

\[ x' = x + 0 \]

\[ x = 0 \]

\[ y = 0 \]

\[ y \neq z \]

\[ x \neq 0 \]

\[ y' = x - y \]

\[ x = 0 \]

\[ y = 0 \]

\[ x = 0 \]

\[ y = 0 \]

\[ x = 0 \]

\[ y = 0 \]

\[ x = 0 \]

\[ y = 0 \]

\[ x = 0 \]

\[ y = 0 \]

\[ x = 0 \]

\[ y = 0 \]

\[ x = 0 \]

\[ y = 0 \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x \neq 0 \]

\[ y \neq 0 \]

\[ y = 0 \]

\[ x = 0 \]

\[ x' = x + 0 \]

\[ y' = x - y \]

\[ x' = x - y \]

\[ x = z \]

\[ y = z \]

\[ y \neq z \]

\[ x' = x - y \]

\[ x' = x + 0 \]

\[ x \leq y \]

\[ x > y \]

\[ x = 0 \]

\[ y = 0 \]

\[ x \neq 0 \]

\[ y \neq 0 \]

\[ x' = 12; y' = 6; z' = 6 \]

\[ x \neq 0; y \neq 0 \]

\[ x = 0; y = 0 \]

\[ x \neq 0; y \neq 0; (x > y; x' = x - y) \]

\[ x = 0; y \neq z \]

\[ x 
eq 0 \]

\[ y = 0 \]

\[ x \neq z \]

\[ y = z \]

\[ y \neq z \]

\[ x = 0 \]

\[ y = 0 \]

\[ x \neq 0 \]

\[ y \neq 0 \]

\[ x' = x - y \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0 \]

\[ x \neq 0; y \neq 0 \]

\[ x = z \]

\[ y = z \]

\[ y \neq z \]
Example

\begin{align*}
&x' = 12; y' = 6; z' = 6 \\
&x = 0 \\
&x \neq 0; y \neq 0 \\
&x \neq 0; y = 0 \\
&y = z \\
&y \neq z \\
&x' = x + 0 \\
&x' = x - y \\
&x \leq y \\
&y' = x - y \\
&x' = x - y \\
\end{align*}
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0 \]

\[ y = z \]

\[ y \neq z \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x \neq 0; y \neq 0 \]

\[ x' = x + 0 \]

\[ x \leq y \]

\[ y' = x - y \]

\[ x' = x - y \]

\[ x > y \]

\[ x \neq z \]

\[ x=0; y=z \]

\[ x=0; y \neq z \]

\[ x=0; y=0 \]

\[ x' = x - y \]

\[ x'=x+0 \]

\[ x=0; y \neq z \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0; y = z \]

\[ x = 0; y \neq z \]

\[ x \neq 0; y \neq 0 \]

\[ x = z \]

\[ x \neq z \]

\[ x' = x + 0 \]

\[ y' = x - y \]

\[ x' = x - y \]

\[ x' = x + 0 \]

\[ x \leq y \]

\[ x > y \]

\[ y' = x - y \]
Example

\begin{align*}
&x' = 12; y' = 6; z' = 6 \\
&x = 0; y = z \\
&x = 0; y \neq z \\
&x \neq 0; y = 0 \\
&x \neq 0; y \neq 0 \\
&x = z \\
&x \neq z \\
&x' = x + 0 \\
&x' = x - y \\
&y' = x - y \\
x \neq 0; y \neq 0; (x \neq 0; y = 0; x \neq z) \\
(x = 0; y \neq z) \\
(x \neq 0; y = 0; x = z) \\
(x = 0; y \neq z) \\
(x = 0; y = z) \\
(x' = x - y) \\
\end{align*}
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x \neq 0; y \neq 0 \]

\[ x = 0; y = z \]

\[ x' = x + 0 \]

\[ x > y; x' = x - y \]

\[ y' = x - y \]

\[ x = z \]

\[ x \neq z \]

\[ x = 0; y = z \]

\[ x = 0; y \neq z \]
Example

\( x' = 12; y' = 6; z' = 6 \)

\( x = 0; y = z \)

\( x = 0; y \neq z \)

\( x \neq 0; y = 0 \)

\( x \neq z \)

\( x > y; x' = x - y \)

\( x = 0; y \neq z \)

\( x \neq 0; y = 0; x = z \)

\( x \neq z \)

\( x \neq 0; y \neq 0 \)

\( x = 0; y = z \)

\( x \neq 0; y = 0; x = z \)

\( x \neq z \)

\( x' = x + 0 \)

\( x \neq 0; y \neq 0 \)

\( x = 0; y = z \)

\( x = 0; y \neq z \)

\( x \neq 0; y = 0; x = z \)

\( x \neq z \)

\( x' = x + 0 \)

\( x \neq 0; y \neq 0 \)

\( x = 0; y = z \)

\( x = 0; y \neq z \)

\( x \neq 0; y = 0; x = z \)

\( x \neq z \)
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x 
eq 0; y 
eq 0 \]

\[ x' = x + 0 \]

\[ x \leq y; y' = x - y \]

\[ x > y; x' = x - y \]

\[ x = z \]

\[ x' = x + 0 \]

\[ x = 0; y = z \]

\[ x = 0; y = 0 \]

\[ x' = x + 0 \]

\[ x' = x - y \]

\[ x = 0; y = 0 \]

\[ x \neq z \]

\[ x' = x - y \]

\[ x = 0; y \neq z \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x \neq 0; y \neq 0 \]

\[ x = 0; y = z \]

\[ x = 0; y \neq z \]

\[ x' = x + 0 \]

\[ x \leq y; y' = x - y \]

\[ x > y; x' = x - y \]

\[ x \neq z \]

\[ x \neq 0; y = 0 \]

\[ x \neq z \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x \neq 0; y \neq 0 \]

\[ x = 0; y = z \]

\[ x = z \]

\[ x > y; x' = x - y \| x \leq y; y' = x - y \]

\[ x = 0; y \neq z \]

\[ x \neq z \]

\[ x \neq 0; y = 0 \]

\[ x \neq 0; y \neq 0 \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x' = x + 0 \]

\[ x \neq 0; y \neq 0 \]

\[ x > y; x' = x - y \] \[ \lor \] \[ x \leq y; y' = x - y \]

\[ x = 0; y = z \]

\[ x = 0; y \neq z \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x \neq 0; y \neq 0 \]

\[ x' = x + 0 \]

\[ x > y; x' = x - y \parallel x \leq y; y' = x - y \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0; y = z \]

\[ x \neq 0; y = 0 \]

\[ x = z \]

\[ x \neq z \]

\[ x \neq 0; y \neq 0; (x > y; x' = x - y \lor x \leq y; y' = x - y) \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0; y = z \]

\[ x \neq 0; y = 0 \]

\[ x = z \]

\[ x \neq z \]

\[ x \neq 0; y \neq 0; (x > y; x' = x - y \| x \leq y; y' = x - y) \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0; y = z \]

\[ x \neq 0; y = 0 \]

\[ x = z \]

\[ x \neq z \]

\[ x' = x + 0 \]

\[ x \neq 0; y \neq 0; (x > y; x' = x - y \parallel x \leq y; y' = x - y) \]
Example

\begin{align*}
&x \neq 0; y \neq 0; (x > y; x' = x-y) \lor (x \leq y; y' = x-y); x' = x+0 \\
x' = 12; y' = 6; z' = 6
\end{align*}
Example

\[ x \neq 0; y \neq 0; (x > y; x' = x - y) \lor (x \leq y; y' = x - y); x' = x + 0 \]
\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0; y = z \]
\[ x \neq 0; y = 0 \]
\[ x \neq 0; y = 0 \]

\[ x = z \]
\[ x \neq z \]

\[ x = 0; y \neq z \]
Example

\[ x \neq 0; y \neq 0; (x > y; x' = x - y \parallel x \leq y; y' = x - y); x' = x + 0 \]
\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0; y = z \]

\[ x \neq 0; y = 0 \]

\[ x = z \]

\[ x \neq z \]

\[ x = 0; y \neq z \]
Example

\begin{align*}
\text{x} \neq 0; \text{y} \neq 0; (\text{x} \geq \text{y}; \text{x}' = \text{x} - \text{y}) \mid \text{x} \leq \text{y}; \text{y}' = \text{x} - \text{y}); \\
\text{x}' = 12; \text{y}' = 6; \text{z}' = 6
\end{align*}
Example

\[ x \neq 0; y \neq 0; (x > y; x' = x - y) \lor (x \leq y; y' = x - y); x' = x + 0 \]
\[ x' = 12; y' = 6; z' = 6 \]
Example

\[ x \neq 0; y \neq 0; (x > y; x' = x - y | x \leq y; y' = x - y); x' = x + 0 \]
\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0; y = z \]

\[ x = 0; y \neq z \]

\[ x \neq 0; y = 0; x = z \]

\[ x \neq 0; y = 0; x \neq z \]
Example

\(x \neq 0; y \neq 0; (x>y;x'=x-y) \parallel (x\leq y;y'=x-y); x'=x+0 \parallel (x=0;y=0;x'=x-y)\)

\(x'=12; y'=6; z'=6\)

\(x=0; y=z\)

\(x \neq 0; y=0; x=z\)

\((x \neq 0; y=0; x \neq z) \parallel (x=0; y \neq z)\)
Example

\[
x \neq 0; y \neq 0; (x > y; x' = x - y | x \leq y; y' = x - y); x' = x + 0
\]
\[
x' = 12; y' = 6; z' = 6
\]

\[
x = 0; y = z
\]

\[
x \neq 0; y = 0; x = z
\]

\[
(x \neq 0; y = 0; x \neq z) \lor (x = 0; y \neq z)
\]
Example

\[
x \neq 0; y \neq 0; (x > y; x' = x - y | x \leq y; y' = x - y); x' = x + 0
\]

\[
x' = 12; y' = 6; z' = 6
\]

\[
x = 0; y = z
\]

\[
x \neq 0; y = 0; x = z
\]

\[
(x \neq 0; y = 0; x \neq z) \lor (x = 0; y \neq z)
\]
Example

\[ x \neq 0; y \neq 0; (x > y; x' = x - y) | x \leq y; y' = x - y); x' = x + 0 \]
\[ x' = 12; y' = 6; z' = 6 \]

\[ (x \neq 0; y = 0; x = z) || (x = 0; y = z) \]

\[ (x \neq 0; y = 0; x \neq z) || (x = 0; y \neq z) \]
Example

\[ x \neq 0; y \neq 0; (x > y; x' = x - y) \mid (x \leq y; y' = x - y); x' = x + 0 \]
\[ x' = 12; y' = 6; z' = 6 \]

\( (x \neq 0; y = 0; x = z) \mid (x = 0; y = z) \)

\( (x \neq 0; y = 0; x \neq z) \mid (x = 0; y \neq z) \)
Variable Substitution

Model Checker interprets an assignment as $f : \text{VAR} \rightarrow \text{VAR}'$. A sequential execution of assignments looks as follows:

$$x' = x + 1; \quad x' = x + 2; \quad x > 2$$
Model Checker interprets an assignment as $f : \text{VAR} \rightarrow \text{VAR}'$. 
Variable Substitution

Model Checker interprets an assignment as \( f : \text{VAR} \rightarrow \text{VAR}' \).

A sequential execution of assignments looks as follows:

\[
\begin{align*}
  x' &= x + 1; \\
  x' &= x_1' + 2; \\
  x' &> 2
\end{align*}
\]
Variable Substitution

Model Checker interprets an assignment as $f : \text{VAR} \rightarrow \text{VAR}'$.

A sequential execution of assignments looks as follow: $f_2(f_1(x))$
Variable Propagation

\[ x = 3; \ w' = 7 \]

\[ x = 1; \ z' = 3 \]
Variable Propagation

Ensure that model checker does not skip a variable assignment
Variable Propagation

Ensure that model checker does not skip a variable assignment

\[ x = 1; z' = 3 \] \[ || \] \[ x = 3; w' = 7 \]
Variable Propagation

Ensure that model checker does not skip a variable assignment
Example

\[ x' := 12; y' := 6; z' := 6 \]

\[ x = 0; y = z \]

\[ x = 0; y \neq z \]

\[ x \neq 0; y \neq 0 \]

\[ x' = x + 0 \]

\[ x \leq y; y' = x - y \]

\[ x > y; x' = x - y \]

\[ x \neq z \]
Example

\[ x' := 12; y' := 6; z' := 6 \]

\[ x = 0; y = z \]

\[ x = 0; y \neq z \]

\[ x \neq 0; y \neq 0 \]

\[ x' = x + 0 \]

\[ x \leq y; y' = x - y \]

\[ x > y; x' = x - y \]

\[ x' = x + 0 \]

\[ x \neq 0; y \neq 0; (x > y; x' = x - y) \]

\[ x \neq z \]
Example

\[ x' := 12; y' := 6; z' := 6 \]

\[ x = 0; y = z \]

\[ x \neq 0; y = 0 \]

\[ x = z \]

\[ x \neq z \]

\[ (x > y; x' = x - y; y' = y) \parallel (x \leq y; y' = x - y; x' = x) \]
Example

\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0; y = z \]

\[ x = 0; y = 0 \]

\[ (x > y; x' = x - y; y' = y) \parallel (x \leq y; y' = x - y; x = x) \]

\[ x' = x + 0 \]

\[ x \neq 0; y \neq 0 \]

\[ x' = x + 0 \]

\[ x = z \]

\[ x = 0; y \neq z \]

\[ x = 0; y \neq z \]

\[ x \neq z \]
Example

$x' = 12; y' = 6; z' = 6$

$x = 0; y = z$

$x \neq 0; y = 0$

$x = z$

$x \neq z$

$x \neq 0; y \neq 0; (x > y; x' = x - y; y' = y) \| (x \leq y; y' = x - y; x' = x)$
Example

\[ x \neq 0; y \neq 0; (x > y; x_1' = x-y; y' = y || x \leq y; y' = x-y; x_1' = x); x' = x_1' + 0 \\
\]
\[ x' = 12; y' = 6; z' = 6 \]

\[ x = 0; y = z \]

\[ x \neq 0; y = 0 \]

\[ x = 0; y \neq z \]

\[ x = z \]

\[ x \neq z \]
\( x \neq 0; y \neq 0; (x > y; x_1' = x - y; y' = y || x \leq y; y' = x - y; x_1' = x); x' = x_1' + 0 \)

\( x' = 12; y' = 6; z' = 6 \)

\( x = 0; y = z \)

\( x \neq 0; y = 0 \)

\( x \neq 0; y \neq z \)

\( x = z \)

\( x \neq z \)
Outline

1 Motivation

2 Background
   - Program and Control-Flow Automaton
   - Model Checker

3 Large-Block Encoding
   - Summarization
   - Post-processing

4 Evaluation

5 Conclusion
CFA Reduction

![Graph showing CFA and CFA\textsubscript{LBE} comparison]

Thomas Mertens
Optimization of Model Checking by Large Block Encoding
Runtime Analysis (CEGAR)

- LBE
- CEGAR LBE
- CEGAR SBE
Outline

1. Motivation

2. Background
   - Program and Control-Flow Automaton
   - Model Checker

3. Large-Block Encoding
   - Summarization
   - Post-processing

4. Evaluation

5. Conclusion
Conclusion

- Factor of minimization up to exponentially
- CFA can be summarized in polynomial time
Questions?