Reducer-Based Construction of Conditional Verifiers

Marie-Christine Jakobs
Joint work with Dirk Beyer, Thomas Lemberger, and Heike Wehrheim

LMU Munich, Germany
Facing Hard Verification Tasks

Given: Program $P \models \varphi$?

Verifier A

Verifier B

Program Paths

$P \models \varphi$?
UNKNOWN

Program Paths

$P \models \varphi$?
UNKNOWN
Facing Hard Verification Tasks

Given: Program $P \models \varphi$?

Verifier A

Verifier B

Verifier A + Verifier B

e.g., conditional model checking

Program Paths

$P \models \varphi$?

UNKNOWN

Program Paths

$P \models \varphi$?

UNKNOWN

Program Paths

$P \models \varphi$ ✔
Conditional Model Checking

[Beyer/Henzinger/Keremoglu/Wendler FSE'12]
Reducer-Based Conditional Verifier Construction

Verifier B → Conditional Verifier B

Reducer (preprocessor)
▶ Builds standard input (C program)
▶ Representing a subset of paths
▶ Contains at least all non-verified paths

+ Verifier-unspecific approach
+ Many conditional verifiers possible
Reducer-Based Conditional Verifier Construction

Verifier B → Conditional Verifier B

Our Solution

Condition

Verifier B

Input Program
Reducer-Based Conditional Verifier Construction

Verifier B ➔ Conditional Verifier B

Our Solution

Reducer

Residual Program

Verifier B

Condition

Input Program

Reducer (preprocessor)

- Builds standard input (C program)
- Representing a subset of paths
- Contains at least all non-verified paths
Reducer-Based Conditional Verifier Construction

Reducer (preprocessor)
- Builds standard input (C program)
- Representing a subset of paths
- Contains at least all non-verified paths

+ Verifier-unspecific approach
+ Many conditional verifiers possible

Marie-Christine Jakobs
LMU Munich, Germany
Example Program and Condition

0: if(notThursday)
1: discount=day%7;
else
2: discount=5;
3: assert(0<=discount<7);
4: 

Verifies that the discount is valid given the day of the week.

Marie-Christine Jakobs, LMU Munich, Germany
Example Program and Condition

0: if (notThursday)  
1:    discount = day % 7;  
else  
2:    discount = 5;  
3: assert (0 <= discount < 7);  
4:  

Verifier A only proofs else branch
Example Program and Condition

0: if (notThursday)
1:   discount = day % 7;
else
2:   discount = 5;
3: assert (0 <= discount < 7);
4: 

Verifier A only proofs else branch
Reducer: Residual Program Construction

Program

\begin{center}
\begin{tikzpicture}[node distance=1cm,auto]
  \tikzstyle{every state}=[minimum size=1cm]

  \node[initial,accepting](l0) {$l_0$};
  \node[state](l1)[below of=l0] {$l_1$};
  \node[state](l2)[right of=l1] {$l_2$};
  \node[state](l3)[below of=l1] {$l_3$};
  \node[state](l4)[right of=l3] {$l_4$};

  \path[->]
  (l0) edge node{\texttt{notThursday}} (l1)
  (l1) edge node{\texttt{discount=day\%7;}} (l2)
  (l1) edge node{\texttt{discount=5;}} (l3)
  (l2) edge node{\texttt{notThursday}} (l4)
  (l3) edge[loop below] node{\texttt{assert(0<=\texttt{discount}<7);}} (l3)

\end{tikzpicture}
\end{center}

Residual Program

\begin{center}
\begin{tikzpicture}[node distance=1cm,auto]
  \tikzstyle{every state}=[minimum size=1cm]

  \node[initial,accepting](q0) {$q_0$};
  \node[state](q1)[below of=q0] {$q_1$};
  \node[state](qf)[right of=q1] {$q_f$};
  \node[state](q2)[below of=q1] {$q_2$};

  \path[->]
  (q0) edge node{\texttt{notThursday}} (q1)
  (q0) edge node{\texttt{\neg \texttt{notThursday}}} (qf)
  (q1) edge node{\texttt{discount=\texttt{day\%7;}}} (q2)

\end{tikzpicture}
\end{center}
Reducer: Residual Program Construction

Program

notThursday

¬notThursday

discount = day % 7;

discount = 5;

assert (0 <= discount < 7);

Condition

notThursday

¬notThursday

discount = day % 7;

Reducer

(l₀, q₀)
Reducer: Residual Program Construction

Program

\( \text{notThursday} \)

\( \text{discount} = \text{day} \% 7; \)

\( \text{assert(0} \leq \text{discount} < 7); \)

Condition

\( \text{notThursday} \)

\( \text{discount} = \text{day} \% 7; \)

Residual Program

\( (l_0, q_0) \)

\( (l_2, q_f) \)
Reducer: Residual Program Construction

Program

\[ l_0 \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4 \]

\begin{align*}
&\text{notThursday} \\
&\text{discount=day}\%7; \\
&\text{assert}(0\leq\text{discount}<7); \\
&\text{discount}=5; \\
&\neg\text{notThursday}
\end{align*}

Condition

\[ q_0 \rightarrow q_1 \rightarrow q_f \rightarrow q_2 \]

\begin{align*}
&\text{notThursday} \\
&\neg\text{notThursday} \\
&\text{discount=day}\%7; \\
&\text{assert}(0\leq\text{discount}<7)
\end{align*}

Reducer

\[ (l_0, q_0) \rightarrow (l_2, q_f) \]
Reducer: Residual Program Construction

Program

\begin{align*}
&l_0 \xrightarrow{\text{notThursday}} l_1 \\
&l_1 \xrightarrow{\text{discount}=\text{day}\%7} l_2 \\
&l_2 \xrightarrow{\text{discount}=5} l_3 \\
&l_3 \xleftarrow{\text{assert}(0\leq \text{discount}<7)} l_4 \\
&l_4 \xrightarrow{\text{notThursday}} l_0
\end{align*}

Condition

\begin{align*}
&q_0 \\
&q_1 \xleftarrow{\text{notThursday}} q_f \\
&q_2 \xrightarrow{\text{discount}=\text{day}\%7} q_0
\end{align*}

Residual Program

\begin{align*}
&(l_0, q_0) \\
&(l_1, q_1) \\
&(l_2, q_f)
\end{align*}
Reducer: Residual Program Construction

Program

\[
\begin{align*}
(l_0, q_0) & \quad (l_1, q_1) & \quad (l_2, q_f) & \quad (l_3, q_2) \\
\text{notThursday} & \quad \neg\text{notThursday} & \quad \text{discount}=\text{day}\%7; & \quad \text{assert}(0<=\text{discount}<7); \\
\text{discount}=5; & \quad & & \\
\end{align*}
\]

Condition

\[
\begin{align*}
(l_0, q_0) & \quad (l_1, q_1) & \quad (l_2, q_f) & \quad (l_3, q_2) \\
\text{notThursday} & \quad \neg\text{notThursday} & \quad \text{discount}=\text{day}\%7; & \quad \text{assert}(0<=\text{discount}<7); \\
\end{align*}
\]
Reducer: Residual Program Construction

Program

\[ l_0 \xrightarrow{\text{notThursday}} l_1 \xrightarrow{\text{discount=day\%7}} l_3 \xrightarrow{\text{assert(0<=discount<7)}} l_4 \xrightarrow{\text{notThursday}} l_2 \xrightarrow{\text{discount=5}} l_3 \]

Condition

\[ q_0 \xrightarrow{\text{notThursday}} q_1 \xrightarrow{\text{discount=day\%7}} q_2 \]

Residual Program

\[ (l_0, q_0) \xrightarrow{\text{notThursday}} (l_1, q_1) \xrightarrow{\text{discount=day\%7}} (l_3, q_2) \xrightarrow{\text{assert(0<=discount<7)}} (l_4, q_r) \]
Reducer: C Transformation

Residual Program

\[(l_0, q_0)\]
\[\rightarrow \text{notThursday}\]
\[\rightarrow \neg\text{notThursday}\]
\[(l_1, q_1) \quad (l_2, q_f)\]
\[\rightarrow \text{discount=} \text{day}\%7;\]
\[(l_3, q_2)\]
\[\rightarrow \text{assert}(0\leq\text{discount}<7);\]
\[(l_4, q_r)\]
Reducer: C Transformation

Residual Program

\[(l_0, q_0) \quad \frac{\text{notThursday}}{\rightarrow \neg \text{notThursday}} \quad (l_1, q_1) \quad (l_2, q_f)\]

\[\text{discount} = \text{day} \% 7;\]
\[\text{assert}(0 \leq \text{discount} < 7);\]

\[\text{if} (\neg \text{notThursday})\]
\[
\{
\quad \text{discount} = \text{day} \% 7;
\quad \text{assert}(0 \leq \text{discount} < 7);
\}
\]
Reducer: Soundness

Residual Condition

Program Paths

Non-verified Program Paths

Residual Program Paths
Reducer: Soundness

Residual Condition

Theorem

Presented reducer fulfills residual condition.
Evaluation Setup

SV-COMP

CPAchecker

Predicate Analysis

100 s

Reducer

Conditional Verifiers

CPA-Seq

Smack

UAutomizer

AFL-fuzz

Crest

Klee

Condition

Residual Program

Test Generation

Model Checking
## Small Extract of Results

<table>
<thead>
<tr>
<th>Task</th>
<th>R</th>
<th>CPA-Seq</th>
<th>UAutomizer</th>
<th>Predicate +Reducer</th>
<th>Predicate +CPA-Seq</th>
<th>Predicate +UAutomizer</th>
</tr>
</thead>
<tbody>
<tr>
<td>P15l01</td>
<td>T</td>
<td>x 910</td>
<td>x 900</td>
<td>✓</td>
<td>120</td>
<td>✓ 130</td>
</tr>
<tr>
<td>flood4</td>
<td>T</td>
<td>x 910</td>
<td>x 910</td>
<td>✓</td>
<td>450</td>
<td>x 1100</td>
</tr>
<tr>
<td>newt3_6</td>
<td>F</td>
<td>x 950</td>
<td>x 490</td>
<td>x</td>
<td>910</td>
<td>✓ 260</td>
</tr>
<tr>
<td>P07l38</td>
<td>T</td>
<td>x 950</td>
<td>x 910</td>
<td>x</td>
<td>1100</td>
<td>✓ 470</td>
</tr>
</tbody>
</table>
Effectiveness on Hard Tasks

![Graph showing the effectiveness on hard tasks with various tools and CPU times.](graph.png)
Conclusion

- Template-based conditional verifier construction

Diagram:
- Input Program
- Condition
- Reducer
- Residual Program
- Verifier B
Conclusion

- Template-based conditional verifier construction

- One Reducer
  - Proven sound
  - Used in many conditional verifiers
Conclusion

- Template-based conditional verifier construction

- One Reducer
  - Proven sound
  - Used in many conditional verifiers

- Effective on hard tasks for verifiers and test tools
Conclusion

- Template-based conditional verifier construction

- One Reducer
  - Proven sound
  - Used in many conditional verifiers

- Effective on hard tasks for verifiers and test tools

- Future Work
  - More reducers
  - Using conditions from other tools
Comparison Setup

Predicate Analysis

100s

Identity

Original Program

Reducer

Residual Program

Value Analysis

Conditional Value Analysis

Marie-Christine Jakobs LMU Munich, Germany
Comparison Results

(a) Identity vs. reducer

(b) Native vs. reducer-based