Combining Verifiers in Conditional Model Checking

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

LMU Munich, Germany

1https://www.sosy-lab.org/research/reducer/
Many Verification Tools Available
Facing Hard Verification Tasks

Question: Program $P \models \varphi$?

Verifier A

Verifier B
Facing Hard Verification Tasks

Question: Program $P \models \varphi$?

Verifier A + Verifier B

e.g., conditional model checking
Conditional Model Checking

Verifier A

$P \models \varphi$?

Verifier B

$P \models \varphi$?

TRUE under condition $\psi$

Problem: Often, verifiers are not conditional

Marie-Christine Jakobs LMU Munich, Germany
Conditional Model Checking

[Beyer/Henzinger/Keremoglu/Wendler FSE'12]

Problem: Often, verifiers are not conditional
Reducer-Based Conditional Verifier Construction
[Beyer/Jakobs/Lemberger/Wehrheim ICSE’18]

Verifier B

? → Conditional Verifier B
Reducer-Based Conditional Verifier Construction

[Beyer/Jakobs/Lemberger/Wehrheim ICSE'18]

Verifier B

Local Program

Diagram:

- Condition
- Input Program
- Verifier B
- Property $\varphi$

Our Solution:

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

Marie-Christine Jakobs LMU Munich, Germany
Reducer (preprocessor)

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

[Beyer/Jakobs/Lemberger/Wehrheim ICSE’18]

Verifier B ➔ Conditional Verifier B

Our Solution

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
Example Program and Condition

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

Program

0: \textbf{if} (notThursday)
1: \ textit{discount}=day\%7;
2: \textbf{else}
3: \textbf{assert} (0<=\textit{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

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

\( \text{notThursday} \)
\( \neg \text{notThursday} \)
\( \text{discount} = \text{day} \% 7; \)
\( \text{discount} = 5; \)
\( \text{assert}(0 \leq \text{discount} < 7); \)

Condition

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

\( \text{notThursday} \)
\( \neg \text{notThursday} \)
\( \text{discount} = \text{day} \% 7; \)
Reducer: Residual Program Construction

Program

notThursday
l0
¬notThursday
l1
l2
l3
discount=day%7;
¬notThursday
l0
l2
l3
l4
assert(0<=discount<7);

Condition

notThursday
q0
¬notThursday
q1
q2
q3
q4
q5
discount=day%7;

Reducer

(l0, q0)
Reducer: Residual Program Construction

Program

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

\text{notThursday}
\text{discount=day}\%7;
\text{assert(0<=discount<7)};

Condition

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

\text{notThursday}
\text{discount=day}\%7;

Residual Program

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

Program

\[ l_0 \xrightarrow{\text{notThursday}} l_1 \xrightarrow{\text{notThursday}} l_2 \xrightarrow{\text{discount}=\text{day}\%7;} l_3 \xrightarrow{\text{discount}=5;} l_4 \xrightarrow{\text{assert}(0<=\text{discount}<7);} q_0 \]

Condition

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

Residual Program

\[ (l_0, q_0) \xrightarrow{\text{notThursday}} (l_2, q_f) \]
Reducer: Residual Program Construction

Program

- $l_0$ with condition $\neg \text{notThursday}$
- $l_1$ with condition $\text{discount}=\text{day}\%7$
- $l_2$ with condition $\neg \text{notThursday}$
- $l_3$ with condition $\text{assert}(0\leq\text{discount}<7)$
- $l_4$ with condition $\text{discount}=5$

Condition

- $q_0$ with condition $\text{notThursday}$
- $q_1$ with condition $\neg \text{notThursday}$
- $q_2$ with condition $\text{discount}=\text{day}\%7$

Reducer

Residual Program

- $(l_0, q_0)$
- $(l_1, q_1)$
- $(l_2, q_f)$

Marie-Christine Jakobs
LMU Munich, Germany
Reducer: Residual Program Construction

Program

notThursday

\( l_0 \)

\( l_1 \)

discount=day\%7;

\( l_2 \)

discount=5;

\( l_3 \)

assert(0<=discount<7);

\( l_4 \)

Condition

notThursday

\( q_0 \)

\( q_1 \)

discount=day\%7;

\( q_2 \)

\( q_f \)

Reduction

Residual Program

notThursday

\( l_0, q_0 \)

\( l_1, q_1 \)

\( l_2, q_f \)

discount=day\%7;

\( l_3, q_2 \)
Reducer: Residual Program Construction

Program

\[
\begin{align*}
&l_0 \quad \text{notThursday} \\
&l_1 \quad \text{discount=day\%7;}
\end{align*}
\]

\[
\begin{align*}
&l_2 \quad \text{discount=5;}
\end{align*}
\]

\[
\begin{align*}
&l_3 \quad \text{assert(0<=discount<7);} \\
&l_4
\end{align*}
\]

Condition

\[
\begin{align*}
&q_0 \quad \text{notThursday} \\
&q_1 \quad \text{discount=day\%7;}
\end{align*}
\]

\[
\begin{align*}
&q_2
\end{align*}
\]

Residual Program

\[
\begin{align*}
&(l_0, q_0) \\
&\quad \text{notThursday} \\
&\quad \text{discount=day\%7;}
\end{align*}
\]

\[
\begin{align*}
&(l_1, q_1) \quad (l_2, q_f) \\
&\quad \text{discount=day\%7;}
\end{align*}
\]

\[
\begin{align*}
&(l_3, q_2) \\
&\quad \text{assert(0<=discount<7);} \\
&\quad \text{notThursday}
\end{align*}
\]

\[
\begin{align*}
&(l_4, q_r)
\end{align*}
\]
Reducer: C Transformation

Residual Program

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

Residual Program

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

C Translation

```c
if (notThursday)
{
    discount=d\text{ay}\%7;
    \textbf{assert}(0\leq\text{discount}<7);
}
```
Theorem

Presented reducer fulfills residual condition.

Marie-Christine Jakobs  
LMU Munich, Germany
Reducer: Soundness

Residual Condition

Program Paths

Non-verified Program Paths

Residual Program Paths

Theorem

*Presented reducer fulfills residual condition.*
Evaluation Setup
### 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 +Reducer</th>
</tr>
</thead>
<tbody>
<tr>
<td>P15l01</td>
<td>T</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>flood4</td>
<td>T</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>newt3_6</td>
<td>F</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>
Effectiveness on Hard Tasks

![Graph showing the effectiveness of different tools on hard tasks. The x-axis represents the n-th fastest correct result, and the y-axis represents CPU time in seconds. Different lines represent CPA-seq, SMACK, Ultimate Automizer, Predicate + CPA-seq, Predicate + SMACK, and Predicate + Ultimate Automizer. The graph illustrates how each tool performs across different result times.]
Conclusion

- Template-based conditional verifier construction
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