Correctness Witness Validation using Predicate Analysis

Maximilian Wiesholler

June 5, 2019
Outline

▶ Motivation
▶ Background
▶ Concepts
▶ Evaluation
▶ Outlook
Motivation

Validation of correctness witnesses can increase trust in the verification result.

Only two validators for correctness witnesses exist:

- $k$-Induction-based validator in CPAchecker [1]
- Automata-based validator in Ultimate Automizer [1]

Goal: Use predicate analysis in CPAchecker as new validator.
Motivation

Approaches for predicate analysis as new validator:

- Reuse witness invariants for the initial precision
- Define witness invariants as additional verification goal
Outline

▶ Motivation
▶ Background
▶ Concepts
▶ Evaluation
▶ Outlook
Correctness witness

- is a partial correctness proof when a program fulfills a specification
- can contain invariants which have been found during the verification of the program
- Syntactic level: Witness is stored in an exchangeable format
- Semantic level: Witness is represented by an observer automaton
Example program and its corresponding correctness witness automaton

```c
int main(void) {
    unsigned int x = __VERIFIER_nondet_uint();
    unsigned int y = x;
    while (x < 1024) {
        x++;
        y++;
    }
    if (x != y){
        ERROR: return 1;
    }
    return 0;
}
```

```
\begin{center}
\begin{tikzpicture}
  \node[initial,state] (n1) {
    \text{\textbf{n$_1$}}
  };
  \node[state] (n2) [right of=n1] {
    \text{\textbf{n$_2$}}
  };
  \node[state] (n3) [below of=n2] {
    \text{\textbf{n$_3$}}
  };
  \node[state] (n4) [right of=n3] {
    \text{\textbf{n$_4$}}
  };
  \path[->,auto,thick]
    (n1) edge [loop above, bend left] node {	ext{\textbf{o/w}}}(n1)
    (n1) edge [below] node {	ext{\textbf{PL=3}}}(n2)
    (n2) edge [above] node {	ext{\textbf{PL=4}}}(n3)
    (n3) edge [loop below] node {	ext{\textbf{o/w}}}(n3)
    (n3) edge [below] node {	ext{\textbf{PL=4}}}(n4)
    (n4) edge [loop below] node {	ext{\textbf{o/w}}}(n4)
    (n4) edge [above] node {	ext{\textbf{PL=6}}}(n1)
    (n1) edge [above] node {	ext{\textbf{true}}}(n2)
    (n2) edge [above] node {	ext{\textbf{x = y}}}(n3)
    (n3) edge [loop left] node {	ext{\textbf{true}}}(n3)
    (n2) edge [loop left] node {	ext{\textbf{true}}}(n2);
\end{tikzpicture}
\end{center}
```
Background: Witnesses on the Syntactic Level

Witness file

- includes general information in the header:
  - witness type (violation | correctness)
  - producer
  - proved specification
  - program hash
  - ...

- contains the witness automaton
Background: Predicate Analysis

**Predicate analysis**

Using *predicate abstraction* in a *reachability analysis* with CEGAR.

**Predicate abstraction**

Computation of predicates (boolean expressions) over program variables.

Motivation: Abstract concrete states and their assigned variables.

Predicates can be solved by a SMT solver.
Reachability analysis

A set of reachable abstract states is computed to create an abstract model of the program.

The abstract model has the form of an abstract reachability graph (ARG) [3].

The computation of abstract states is guided by the current precision [3].

Precision

The precision describes the current stored information.
Countercexample-Guided Abstraction Refinement (CEGAR)

Motivation: Find a suitable precision during the verification that is neither too coarse nor too accurate.

CEGAR iteratively calls the reachability analysis with a refined precision

CEGAR terminates when a counterexample is satisfiable or no abstract state violates the specification.
Predicate precision

A predicate precision $\pi$ is a mapping from program locations to sets of predicates over the program variables [2].

$\pi(l)$ describes the predicate precision at program location $l$. 
Outline

▶ Motivation
▶ Background
▶ Concepts
▶ Evaluation
▶ Outlook
Location invariant

A location invariant is a tuple \((l, \theta)\) where \(l\) denotes the CFA location and \(\theta\) denotes the invariant.*

\(I\) denotes the set of location invariants.

\(I\) can be derived from the correctness witness.

*An implementation of a location invariant already exists for \(k\)-induction-based validator in \(CPAchecker\)
Concepts: Location Invariants

$I$ can be received in CPAchecker by using the following steps:

- Parsing the correctness witness file into a witness automaton*
- Performing a reachability analysis on the witness automaton*
- Extracting the invariants and their corresponding CFA locations from set reached

To guarantee soundness: If $(l, \theta)$ is received so that $l = l'$ holds for $(l', \theta') \in I$, $(l, \theta \land \theta')$ is created and $(l', \theta')$ removed.

*Implementations already exist for $k$-induction-based validator in CPAchecker
Motivation: Witness invariants as precision facts might decrease number of refinements and CPU time.

Approach:
- get $I$ from the correctness witness
- for each $i \in I$: convert $\theta$ from $i$ into a predicate $\rho$
- add $\rho$ to the location, function or global predicate precision
  - CFA location is known because of $l$

Atomic predicate: Splitting the predicate would create components that are no boolean formulas anymore.

Getting atomic predicates: Split predicates into predicate components until all components have an atomic form.
Motivation: Validate the invariants in the correctness witness.

### Invariants specification automaton (ISA)

- Is a control automaton which is constructed by using the invariants from the correctness witness.
- Is used as an additional verification goal.

Three ISA concepts are presented.
ISA$^{2S}$ has **two states**: an initial state and an error state.

Example:

$I = \{(l_4, x = y)\}$

![Diagram](attachment:image.png)
Concepts: CFA based Invariants Specification Automaton (ISA$^{CFA}$)

ISA$^{CFA}$ structure refers to structure of the CFA of the program. Motivation: better performance expected compared to ISA$^{2S}$.

Example:

$I = \{ (l_4, x = y) \}$

\[
\begin{array}{c}
\text{start} \\
\text{unsigned int } x = \text{nondet_uint();} \\
\text{unsigned y = x;} \\
[(x < 1024)] \\
[l_4] \\
[(x \neq y)] \\
[l_9] \\
[x \neq y] \\
[l_8] \\
[l_11, \text{return 0;} ] \\
[l_12] \\
[l_6] \\
[l_5, \text{ERROR: return -1;}] \\
[l_4] \\
[l_12] \\
[l_8] \\
[l_5, x++] \\
[l_4] \\
[l_12] \\
[l_8] \\
[l_6] \\
\end{array}
\]

CFA

\[
\begin{array}{c}
\text{start} \\
\text{s_1} \\
\text{l_3} \\
l_3 \land (x \neq y) \\
\text{s_2} \\
\text{l_4} \\
l_4 \land (x = y) \\
\text{s_3} \\
\text{l_8} \\
\text{s_6} \\
l_8 \\
\text{s_7} \\
l_11 \\
\text{s_6} \\
\text{l_9} \\
\text{s_7} \\
\text{l_12} \\
\text{s_4} \\
\text{l_5} \\
\text{l_4} \land (x = y) \\
\text{s_3} \\
\text{l_6} \\
\text{s_5} \\
\text{l_4} \land (x \neq y) \\
\text{s_5} \\
\text{E} \\
\end{array}
\]

ISA$^{CFA}$
Concepts: Witness Invariants Specification Automaton ($ISA^{WI}$)

$ISA^{WI}$ extends the original witness with invariant-based assumptions.

Computation of $I$ not required.

Example:
Three types of correctness witnesses are distinguished:

- **non-trivial-witness**: Witness has states with *non-trivial*-Invariants and at least one of these states is reachable in the analysis.
- **true-witness**: Witness has no states labeled with *non-trivial*-Invariants.
- **hidden-true-witness**: Witness has states with *non-trivial*-Invariants but each of these states is unreachable in the analysis.
Consequences of correctness witness types:

A *true*-witness or *hidden-true*-witness leads to an empty set of location invariants.

<table>
<thead>
<tr>
<th></th>
<th>non-trivial-witness</th>
<th>true-witness</th>
<th>hidden-true-witness</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_0$</td>
<td>$\exists l \in L.\pi_0(l) \neq \emptyset$</td>
<td>$\forall l \in L.\pi_0(l) = \emptyset$</td>
<td>$\forall l \in L.\pi_0(l) = \emptyset$</td>
</tr>
<tr>
<td>ISA$^{2S}$</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ISA$^{CFA}$</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>ISA$^{WI}$</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

1: transitions into error states exist
2: transitions into error states do not exist
3: transitions into error states exist but are not reachable
Outline

▶ Motivation
▶ Background
▶ Concepts
▶ Evaluation
▶ Outlook
Evaluation: Specification and Benchmarks

**Specification**

Unreachability of error function `__VERIFIER_error()`.

**Benchmarks**

- Tasks taken from SV-COMP 2019 from categories `ReachSafety*` and `SoftwareSystems**`
- Excluding all tasks that violate the specification (focus is on correctness witnesses)
- In total **4668** tasks

*without subcategory `ReachSafety-Recursive`
**only with subcategory `Systems_DeviceDriversLinux64_ReachSafety`
**Evaluation: System Settings**

**Benchmark environment**

- **Machines:** 8 core CPUs with 3.40 GHz (Intel Xeon E3-1230 v5) and 33 GB of RAM memory
- **Operating system:** Ubuntu 18.04 (64 Bit)
- **Time limit:** 900 s
- **Memory limit:** 15 GB
- **Requirement of 8 CPU cores for a task**
Evaluation: Interesting Questions

▶ Does reuse of invariants-based predicates lead to less CPU time and fewer CEGAR refinements?

▶ Is predicate analysis able to validate the original specification and invariants of
  ▶ its own correctness witnesses?
  ▶ correctness witnesses from \( k \)-induction in \textsc{CPAchecker} and from \textsc{Ultimate Automizer}?
The evaluation can be divided into three parts:

1. Producing correctness witnesses
2. Initialize predicate precision with invariant-based predicates
3. Validate correctness witnesses by using an ISA
## Evaluation: Producing Correctness Witnesses

<table>
<thead>
<tr>
<th>status</th>
<th>all</th>
<th>correct-true</th>
<th>correct-false</th>
<th>incorrect-true</th>
<th>incorrect-false</th>
<th>timeout</th>
<th>error</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>$pA$-Verification</td>
<td>4668</td>
<td><strong>2396</strong></td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1533</td>
<td>696</td>
<td>38</td>
</tr>
<tr>
<td>$kl$-Verification</td>
<td>4668</td>
<td><strong>2599</strong></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1784</td>
<td>134</td>
<td>150</td>
</tr>
<tr>
<td>$uA$-Verification</td>
<td>4668</td>
<td><strong>2688</strong></td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1684</td>
<td>90</td>
<td>204</td>
</tr>
</tbody>
</table>
Evaluation: Producing Correctness Witnesses

\[ \text{Witnesses}^{\text{no-true}} \]
Correctness witnesses are no \textit{true}-witnesses.

\[ \text{Witnesses}^{\text{no-true}^+} \]
Correctness witnesses are no \textit{true}-witnesses and no \textit{hidden-true}-witnesses.
Evaluation: Outline

- Producing correctness witnesses
- **Initialize predicate precision with invariant-based predicates**
- Validate correctness witnesses by using an ISA
### Evaluation: Witness Invariants for Precision Reuse

Six different configurations:

<table>
<thead>
<tr>
<th>Scope</th>
<th>atomic predicates</th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>$pA$-Validation-$PR^{lo}$</td>
<td>$pA$-Validation-$PR^{lo+a}$</td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>$pA$-Validation-$PR^{fu}$</td>
<td>$pA$-Validation-$PR^{fu+a}$</td>
<td></td>
</tr>
<tr>
<td>global</td>
<td>$pA$-Validation-$PR^{gl}$</td>
<td>$pA$-Validation-$PR^{gl+a}$</td>
<td></td>
</tr>
</tbody>
</table>
Evaluation: $pA$-Witnesses$^{no\text{-}true^+}$ for Precision Reuse

*pA-Validation-PR initialized with invariants from 223* $pA$-Witnesses$^{no\text{-}true^+}$.*

<table>
<thead>
<tr>
<th>$pA$-Validation-PR</th>
<th>$pA$-Validation-$PR_0$</th>
<th>$pA$-Validation-$PR_0^{+a}$</th>
<th>$pA$-Validation-$PR_l$</th>
<th>$pA$-Validation-$PR_{l+a}$</th>
<th>$pA$-Validation-$PR_{gl}$</th>
<th>$pA$-Validation-$PR_{gl+a}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>accepted</td>
<td>223</td>
<td>219</td>
<td>223</td>
<td>219</td>
<td>222</td>
<td>217</td>
</tr>
<tr>
<td>rejected</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>error</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>timeout</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>other</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*filtered from 2396 $pA$-Witnesses*
Evaluation: Using Invariants from $pA$-Witnesses\textsuperscript{no-true$^{+h}$}

Predicate precision in $pA$-Validation-$PR^{lo}$ and $pA$-Validation-$PR^{gl+a}$ initialized with invariants from $pA$-Witnesses\textsuperscript{no-true$^{+h}$}.

→ For $pA$-Validation-$PR^{lo}$ no changes.
→ For $pA$-Validation-$PR^{gl+a}$ no speedup.
Evaluation: \(pA\)-Witnesses\(^{\text{no-true}^+}\) for Precision Reuse

<table>
<thead>
<tr>
<th>CEGAR refinements</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6-10</th>
<th>11-20</th>
<th>21-30</th>
<th>30&lt;</th>
<th>Number of tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pA)-Verification</td>
<td>0</td>
<td>180</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>12</td>
<td>223</td>
</tr>
<tr>
<td>(pA)-Validation-PR(^{\text{lo}})</td>
<td>0</td>
<td>182</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>6</td>
<td>1</td>
<td>12</td>
<td>223</td>
</tr>
<tr>
<td>(pA)-Verification</td>
<td>0</td>
<td>179</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>9</td>
<td>219</td>
</tr>
<tr>
<td>(pA)-Validation-PR(^{\text{lo+a}})</td>
<td>0</td>
<td>181</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>219</td>
</tr>
<tr>
<td>(pA)-Verification</td>
<td>0</td>
<td>180</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>12</td>
<td>223</td>
</tr>
<tr>
<td>(pA)-Validation-PR(^{\text{fu}})</td>
<td>2</td>
<td>181</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>8</td>
<td>0</td>
<td>11</td>
<td>223</td>
</tr>
<tr>
<td>(pA)-Verification</td>
<td>0</td>
<td>179</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>9</td>
<td>219</td>
</tr>
<tr>
<td>(pA)-Validation-PR(^{\text{fu+a}})</td>
<td>2</td>
<td>180</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>9</td>
<td>8</td>
<td>2</td>
<td>7</td>
<td>219</td>
</tr>
<tr>
<td>(pA)-Verification</td>
<td>0</td>
<td>180</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>11</td>
<td>222</td>
</tr>
<tr>
<td>(pA)-Validation-PR(^{\text{gl}})</td>
<td>115</td>
<td>69</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>9</td>
<td>7</td>
<td>0</td>
<td>9</td>
<td>222</td>
</tr>
<tr>
<td>(pA)-Verification</td>
<td>0</td>
<td>179</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>7</td>
<td>217</td>
</tr>
<tr>
<td>(pA)-Validation-PR(^{\text{gl+a}})</td>
<td>114</td>
<td>69</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>217</td>
</tr>
</tbody>
</table>

CEGAR refinements only reduced for \(pA\)-Validation-PR\(^{\text{gl}}\) and \(pA\)-Validation-PR\(^{\text{gl+a}}\)
Evaluation: $pA$-Witnesses$^{no\text{-}true}$ for Precision Reuse - Conclusion

CPU time almost never decreased for any $pA$-Validation-PR when compared with $pA$-Verification.

Why no changes in CPU time and CEGAR refinements in particular for $pA$-Validation-PR$^{lo}$?

Possible reasons:

- Information from witness invariants not sufficient for an efficient precision reuse
- Bug in the implementation in CPAchecker
Evaluation: Outline

- Producing correctness witnesses
- Initialize predicate precision with invariant-based predicates
  
  "Validation of correctness witnesses by using an ISA"
Evaluation: Validating $pA$-Witnesses

2396 Tasks

- accepted
- violated
- timeout
- error
- other

Maximilian Wiesholler
Evaluation: Validating \( pA\)-Witnesses\(^{no\text{-true}} \)

222 Tasks

- accepted
- violated
- timeout
- error
- other

Maximilian Wiesholler
**Evaluation: Inspecting pA-Validation-ISA with pA-Witnesses as Input**

<table>
<thead>
<tr>
<th>configuration</th>
<th>pA-Validation-ISA^{2S}</th>
<th>pA-Validation-ISA^{CFA}</th>
<th>pA-Validation-ISA^{WI}</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIV</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Why does each pA-Validation-ISA approach produces for the same two pA-Witnesses a witness invariant violation (WIV)?

- **Witness is imprecise:** Invariant is validated at a location where the invariant variables are not yet assigned.

- **Witness invariant contains pointer values from the SMT solver which can not be validated**
Evaluation: Validating $pA$-Witnesses$^{\text{no-true}}$

Why do the $pA$-Validation-ISA approaches sometimes exceed the time limit?

Likely reason: Additional computation effort to validate the invariants.
Evaluation: Validating $kI$-Witnesses

2599 Tasks

- accepted
- violated
- timeout
- error
- other

Maximilian Wiesholler
Evaluation: Validating $kI$-Witnesses$^{-}\text{no-true}$

1748 Tasks

- **accepted**
- **violated**
- **timeout**
- **error**
- **other**

Tasks accepted, violated, timeout, error, other.
**Evaluation: Inspecting $pA$-Validation-ISA with $kI$-Witnesses as Input**

<table>
<thead>
<tr>
<th>configuration</th>
<th>$pA$-Validation-ISA$^{2S}$</th>
<th>$pA$-Validation-ISA$^{CFA}$</th>
<th>$pA$-Validation-ISA$^{WI}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIV</td>
<td>533</td>
<td>533</td>
<td>523</td>
</tr>
</tbody>
</table>

Why are so many WIVs detected?

Found reasons:

▶ $kI$-Witnesses$^{no$-true} are sometimes "imprecise" for the ISA approaches

▶ Loop invariants in $kI$-Witnesses$^{no$-true} do not hold for all loop iterations
Evaluation: Validating \textit{uA-Witnesses}

2688 Tasks

- **accepted**
- violated
- timeout
- **error**
- **other**
Evaluation: Validating $uA$-Witnesses$^{\text{no-true}}$

248 Tasks

- accepted
- violated
- timeout
- error
- other
Evaluation: Inspecting $pA$-Validation-ISA for $uA$-Witnesses as Input

<table>
<thead>
<tr>
<th>configuration</th>
<th>$pA$-Validation-ISA$^{2S}$</th>
<th>$pA$-Validation-ISA$^{CFA}$</th>
<th>$pA$-Validation-ISA$^{WI}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIV</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

For ISA$^{2S}$ or ISA$^{CFA}$ three WIVs are produced more compared to ISA$^{WI}$.

Reason: Error states in ISA$^{2S}$ and ISA$^{CFA}$ are overapproximated when analyzing the three $uA$-Witnesses.

→ Only the ISA$^{WI}$ is precise. It can guarantee to reflect the semantics of the original witness.
Evaluation: Detecting *hidden-true-*witnesses

No *hidden-true-*witnesses in \( pA \)-Witnesses and \( kI \)-Witnesses. 147 \( uA \)-Witnesses are *hidden-true-*witnesses in the context of a validation analysis in CPAchecker.

Found reasons:

- Calling \texttt{__VERIFIER_nondet_uint();} leads to two states and transitions in \( uA \)-Witnesses. CPAchecker does not expect this. (e.g. witness for task \#2)

- \( uA \)-Witnesses can have an invariant 0 to label explicitly unreachable witness states

#2 https://github.com/sosy-lab/sv-benchmarks/blob/svcomp19/c/loop-invariants/eq1_true-unreach-call_true-valid-memsafety_true-no-overflow_false-termination.c
Evaluation: Comparing the \( pA \)-Validation–ISA Approaches for \( pA \)-Witnesses$^{\text{no-true}}$
Evaluation: Comparing the \( pA \)-Validation-ISA Approaches for \( kl \)-Witnesses\(^{\text{no-true}} \)
Evaluation: Comparing the $pA$-Validation-ISA Approaches for $uA$-Witnesses$^{\text{no-true}}$
Evaluation: Validating Witnesses - Conclusion

Predicate analysis is able to validate the majority of its own witnesses. It understands other correctness witnesses but cannot always validate them.

Verifying invariants in an ISA that is based on an imprecise correctness witness leads in general to a WIV.

Proposing that correctness witnesses should be precise because:

- Transitions in witness file correspond to a certain program operation. If a transition enters a state labeled with an invariant the invariant should indeed hold at the program location that follows the program operation.

- Future validators might have problems with imprecise witnesses as well.
Validation of negated witness invariants → Checking if predicate analysis can reject the intentionally wrong correctness witnesses.

An ISA is independent from the applied abstraction-technique. Hence, it can theoretically be used with other analyzes.

The ISA\textsuperscript{WI} can be used for the program generation concept shown in [4][5][6].

- Concept: Transform program into a behaviorally equal program that is more efficiently verifiable. Use the ARG to create this program.
- Invariant-based error states affect the ARG → Verifying the transformed program will also verify invariants.


Appendix
Appendix: CEGAR

Scetch from Lecture "Software Analysis and Verification" (Lecturer: M.-C. Jakobs)
Appendix: Using Invariants from $kI$-Witnesses $^{\text{no-true}^+}$

Predicate precision in $pA$-Validation-$\text{PR}^{\text{lo}}$ is initialized with invariants from $kI$-Witnesses $^{\text{no-true}^+}$.

$\rightarrow$ No speedup.
Appendix: Using Invariants from $uA$-Witnesses$^{\text{no-true}^h}$

Predicate precision in $pA$-Validation-$\text{PR}^{\text{lo}}$ initialized with invariants from $uA$-Witnesses$^{\text{no-true}^h}$.

→ No statement possible because too many witnesses are detected as $\textit{hidden-true}$-witnesses or $\textit{true}$-witnesses.
### Appendix: \( pA \)-Witnesses for Precision Reuse

<table>
<thead>
<tr>
<th>( pA )-Validation-PR</th>
<th>( pA )-Validation-( Pr^l )</th>
<th>( pA )-Validation-( Pr^{l+a} )</th>
<th>( pA )-Validation-( Pr_u )</th>
<th>( pA )-Validation-( Pr^{u+a} )</th>
<th>( pA )-Validation-( Pr_l )</th>
<th>( pA )-Validation-( Pr^{l+a} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>accepted</td>
<td>2396</td>
<td>2351</td>
<td>2393</td>
<td>2352</td>
<td>2392</td>
<td>2349</td>
</tr>
<tr>
<td>rejected</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>error</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>timeout</td>
<td>1</td>
<td>43</td>
<td>1</td>
<td>42</td>
<td>2</td>
<td>45</td>
</tr>
<tr>
<td>other</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2396 ( pA )-Witnesses</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>accepted</th>
<th>rejected</th>
<th>error</th>
<th>timeout</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>2393</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2351</td>
<td>0</td>
<td>1</td>
<td>43</td>
<td>1</td>
</tr>
<tr>
<td>2393</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2352</td>
<td>0</td>
<td>1</td>
<td>42</td>
<td>1</td>
</tr>
<tr>
<td>2392</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2349</td>
<td>0</td>
<td>1</td>
<td>45</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>223 ( pA )-Witnesses(^{no-true}^h)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>accepted</th>
<th>rejected</th>
<th>error</th>
<th>timeout</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>223</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>219</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>223</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>219</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>222</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>217</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Appendix: Inspecting $pA$-Validation-ISA for $uA$-Witnesses as Input

Correctness witness produced by **Ultimate Automizer** for task #1:

- $n_9$ corresponds to a loop head in the program and is labeled with invariant $\theta$
- Original witness semantics: $\theta$ needs only to be **validated once**
- $ISA^{2S}$ and $ISA^{CFA}$ semantics: $\theta$ must be **validated every time** when entering the loop head
- But: $\theta$ only valid when entering the loop head the first time $\rightarrow$ $ISA^{2S}$ and $ISA^{CFA}$ trigger a false alarm

$\rightarrow$ Only the $ISA^{WI}$ is precise. It can guarantee to reflect the original witness.

## Appendix: Comparing Accepted $pA$-Witnesses for $pA$-Validation-ISA and $kI$-Validation

<table>
<thead>
<tr>
<th>Witness accepted</th>
<th>$pA$-Validation-ISA(^{2S})</th>
<th>$\neg$ $pA$-Validation-ISA(^{2S})</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kI$-Validation</td>
<td>83</td>
<td>12</td>
<td>95</td>
</tr>
<tr>
<td>$\neg$ $kI$-Validation</td>
<td>74</td>
<td>53</td>
<td>127</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>157</td>
<td>65</td>
<td>222</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Witness accepted</th>
<th>$pA$-Validation-ISA(^{CFA})</th>
<th>$\neg$ $pA$-Validation-ISA(^{CFA})</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kI$-Validation</td>
<td>84</td>
<td>11</td>
<td>95</td>
</tr>
<tr>
<td>$\neg$ $kI$-Validation</td>
<td>75</td>
<td>52</td>
<td>127</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>159</td>
<td>63</td>
<td>222</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Witness accepted</th>
<th>$pA$-Validation-ISA(^{WI})</th>
<th>$\neg$ $pA$-Validation-ISA(^{WI})</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kI$-Validation</td>
<td>84</td>
<td>11</td>
<td>95</td>
</tr>
<tr>
<td>$\neg$ $kI$-Validation</td>
<td>71</td>
<td>56</td>
<td>127</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>155</td>
<td>67</td>
<td>222</td>
</tr>
</tbody>
</table>
Appendix: Comparing Accepted $kI$-Witnesses\textsuperscript{no-true} for $pA$-Validation-ISA and $kI$-Validation

<table>
<thead>
<tr>
<th>Witness accepted</th>
<th>$pA$-Validation-ISA\textsuperscript{$2S$}</th>
<th>$\neg pA$-Validation-ISA\textsuperscript{$2S$}</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kI$-Validation</td>
<td>807</td>
<td>862</td>
<td>1669</td>
</tr>
<tr>
<td>$\neg kI$-Validation</td>
<td>7</td>
<td>72</td>
<td>79</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>814</td>
<td>934</td>
<td>1748</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Witness accepted</th>
<th>$pA$-Validation-ISA\textsuperscript{CFA}</th>
<th>$\neg pA$-Validation-ISA\textsuperscript{CFA}</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kI$-Validation</td>
<td>807</td>
<td>862</td>
<td>1669</td>
</tr>
<tr>
<td>$\neg kI$-Validation</td>
<td>7</td>
<td>72</td>
<td>79</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>814</td>
<td>934</td>
<td>1748</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Witness accepted</th>
<th>$pA$-Validation-ISA\textsuperscript{WI}</th>
<th>$\neg pA$-Validation-ISA\textsuperscript{WI}</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kI$-Validation</td>
<td>791</td>
<td>878</td>
<td>1669</td>
</tr>
<tr>
<td>$\neg kI$-Validation</td>
<td>7</td>
<td>72</td>
<td>79</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>798</td>
<td>950</td>
<td>1748</td>
</tr>
</tbody>
</table>
Appendix: Comparing Accepted $uA$-Witnesses \textsuperscript{no-true} for $pA$-Validation-ISA and $kI$-Validation

<table>
<thead>
<tr>
<th>Witness accepted</th>
<th>$pA$-Validation-ISA\textsuperscript{2S}</th>
<th>$\neg pA$-Validation-ISA\textsuperscript{2S}</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kI$-Validation</td>
<td>104</td>
<td>63</td>
<td>167</td>
</tr>
<tr>
<td>$\neg kI$-Validation</td>
<td>19</td>
<td>62</td>
<td>81</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>123</td>
<td>125</td>
<td>248</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Witness accepted</th>
<th>$pA$-Validation-ISA\textsuperscript{CFA}</th>
<th>$\neg pA$-Validation-ISA\textsuperscript{CFA}</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kI$-Validation</td>
<td>104</td>
<td>63</td>
<td>167</td>
</tr>
<tr>
<td>$\neg kI$-Validation</td>
<td>19</td>
<td>62</td>
<td>81</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>123</td>
<td>125</td>
<td>248</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Witness accepted</th>
<th>$pA$-Validation-ISA\textsuperscript{WI}</th>
<th>$\neg pA$-Validation-ISA\textsuperscript{WI}</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kI$-Validation</td>
<td>107</td>
<td>60</td>
<td>167</td>
</tr>
<tr>
<td>$\neg kI$-Validation</td>
<td>17</td>
<td>64</td>
<td>81</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>124</td>
<td>124</td>
<td>248</td>
</tr>
</tbody>
</table>
Appendix: Violation Reason of Correctness Witnesses for \( pA\)-Validation-ISA

**\( pA\)-Witnesses:**

<table>
<thead>
<tr>
<th>configuration</th>
<th>( pA)-Validation-ISA(^{2S} )</th>
<th>( pA)-Validation-ISA(^{CFA} )</th>
<th>( pA)-Validation-ISA(^{WI} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>all violations</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>original specification</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WIV</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**\( kI\)-Witnesses:**

<table>
<thead>
<tr>
<th>configuration</th>
<th>( pA)-Validation-ISA(^{2S} )</th>
<th>( pA)-Validation-ISA(^{CFA} )</th>
<th>( pA)-Validation-ISA(^{WI} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>all violations</td>
<td>534</td>
<td>534</td>
<td>524</td>
</tr>
<tr>
<td>original specification</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>WIV</td>
<td>533</td>
<td>533</td>
<td>523</td>
</tr>
</tbody>
</table>

**\( uA\)-Witnesses:**

<table>
<thead>
<tr>
<th>configuration</th>
<th>( pA)-Validation-ISA(^{2S} )</th>
<th>( pA)-Validation-ISA(^{CFA} )</th>
<th>( pA)-Validation-ISA(^{WI} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>all violations</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>original specification</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>WIV</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Evaluation: Detecting true-witnesses and hidden-true-witnesses

Detection:

If the witness is a true-witness or hidden-true-witness for ISA$^{CFA}$ or ISA$^{2S}$ and the witness is not a true-witness for ISA$^{WI}$ then the witness is a hidden-true-witness.
## Appendix: Detecting *true*-witnesses and *hidden-true*-witnesses

<table>
<thead>
<tr>
<th>approach</th>
<th>non-trivial-witnesses</th>
<th>hidden-true-witnesses or true-witnesses</th>
<th>true-witnesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>pA-Validation-ISA$^{2S}$</td>
<td>220</td>
<td>2169</td>
<td>-</td>
</tr>
<tr>
<td>pA-Validation-ISA$^{CFA}$</td>
<td>220</td>
<td>2169</td>
<td>-</td>
</tr>
<tr>
<td>pA-Validation-ISA$^{WI}$</td>
<td>220</td>
<td>-</td>
<td>2169</td>
</tr>
<tr>
<td>pA-Validation-ISA$^{2S}$</td>
<td>1696</td>
<td>559</td>
<td>-</td>
</tr>
<tr>
<td>pA-Validation-ISA$^{CFA}$</td>
<td>1696</td>
<td>559</td>
<td>-</td>
</tr>
<tr>
<td>pA-Validation-ISA$^{WI}$</td>
<td>1696</td>
<td>-</td>
<td>559</td>
</tr>
<tr>
<td>pA-Validation-ISA$^{2S}$</td>
<td>40</td>
<td>2337</td>
<td>-</td>
</tr>
<tr>
<td>pA-Validation-ISA$^{CFA}$</td>
<td>40</td>
<td>2337</td>
<td>-</td>
</tr>
<tr>
<td>pA-Validation-ISA$^{WI}$</td>
<td>187</td>
<td>-</td>
<td>2190</td>
</tr>
</tbody>
</table>

→ 147 *uA*-Witnesses are *hidden-true*-witnesses for in the context of an analysis in CPACHECKER.