Exercise 2021/02/19


1 Hands-on

1.1 (Bounded) Model Checking

Jupyter Notebook: 03_BMC.ipynb

1.2 Configurable Program Analysis

Jupyter Notebook: 07_Verifier-Design-part-1.ipynb

2 Theory

2.1 Observer Automata

```cpp
1 int a = 1;
2 int b = 0;
3 while (b <= a) {
4     b = b + 1;
5     a = a * b;
6 }
7 ERR:;
```

Program

CFA $P = (L, E, l_0)$

1. Define an observer automaton for the given program $P = (L, E, l_0)$ and program variables $X$, for each of the following specifications:

   a) $\square (l' \neq l_{ERR})$ (for $g = (l, op, l')$)
   
   b) $\forall x \in X. \forall z \in X. \square(op = [x \leq z] \implies \diamond \forall y \in X \cup \mathbb{Z}.\ op \neq [x \leq y] W op = x := x + 1))$

   c) $\forall x \in X. \forall y \in X. \square(op = x := x * y \implies y > 0)$
2. Consider the following observer automaton $A$:

\[ (G, a \geq b) \]

\[ \text{start} \xrightarrow{} q_{\text{init}} \xrightarrow{} q_{\text{err}} \]

(a) State the LTL-formula equivalent of $A$.

(b) Consider observer analysis $O$ for observer automaton $A$ and precision $\pi$:

\[ \pi = \{ x = n \mid x \in X, n \in \mathbb{Z} \} \cup \{ x \geq n \mid x \in X, n \in \mathbb{N} \} \]
\[ \cup \{ b \leq a, b \leq a + 1 \} \cup \{ \text{false} \} \]

Apply the CPA algorithm with composite analysis $L \times P \times O$ and initial state $e_0 = (l_0, \emptyset, (q_{\text{init}}, \text{true}))$ to program $P$. State the final reached set and whether the specification is violated, according to the algorithm.

### 2.2 Verification-Result Witnesses (45 minutes)

A detailed definition of the GraphML Witness-Exchange Format is available here:

https://github.com/sosy-lab/sv-witnesses

#### 2.2.1 Violation Witnesses

```
int i, j; // defined, but arbitrary value
int result;
int k = 0;
if (i <= j)
  k = k + 1;
else
  k = 0;
if (k == 1)
  result = i - j;
else
  result = i - j;
if (result < 0)
  ERR:;
```

Program $P_e$

Above you see faulty program $P_e$ and its CFA $A_e$. Each CFA edge lists the file location it was created from (e.g., $4_{\text{if}}$).
1. For each violation witness below, list all (syntactic) program paths described by that witness:

**Witness a:**

![Diagram of witness a]

**Witness b:**

![Diagram of witness b]

**Witness c:**

```xml
<graphml
<!-- snip metadata -->
<graph>
  <node id="A0">
    <data key="entry">true</data>
  </node>
  <node id="A2">
    <data key="violation">true</data>
  </node>
  <edge source="A0" target="A2">
    <data key="startline">1</data>
    <data key="assumption">i == 1 ; j == 2 ;</data>
    <data key="assumption_scope">main</data>
  </edge>
  <node id="sink">
    <data key="sink">true</data>
  </node>
  <edge source="A2" target="sink">
    <data key="startline">10</data>
    <data key="control">condition=false</data>
  </edge>
</graph>
</graphml>
```

2. For some witnesses A and B, we say $A <_{testified} B$ iff witness A describes a subset of the state-space that is described by witness B. Check all correct statements:

- [ ] Witness a $<_{testified}$ Witness b
- [ ] Witness b $<_{testified}$ Witness a
- [ ] Witness b $<_{testified}$ Witness c
- [ ] Witness c $<_{testified}$ Witness a

2.2.2 Correctness Witnesses

```c
int x; // defined, but arbitrary value
int y = x;
while (x < 1024) {
  x = x + 1;
  y = y + 1;
}
if (x != y)
  goto ERR;
if (x < 0)
  ERR:
```

Program $P_c$

```
```
For each correctness witness below, list all candidate invariants \((l, i)\) described by that witness. Each candidate invariant \((l, i)\) consists of a program location \(l \in A_c\) where the invariant is supposed to hold, and the invariant \(i\).

Witness \(d\):

Witness \(e\):

Witness \(f\):

```
<graphml>
<!-- .. snip metadata .. -->
<graph>
  <node id="q0">true</node>
  <node id="q1">x == y</node>
  <node id="q2">true</node>
  <node id="q3">true</node>
  <node id="q4">true</node>
  <node id="q5">true</node>
  <edge source="q0" target="q1">true</edge>
  <edge source="q0" target="q4">true</edge>
  <edge source="q0" target="q5">true</edge>
  <edge source="q1" target="q2">true</edge>
  <edge source="q1" target="q5">true</edge>
  <edge source="q3" target="q2">true</edge>
  <edge source="q3" target="q5">true</edge>
  <edge source="q4" target="q2">true</edge>
  <edge source="q4" target="q5">true</edge>
  <edge source="q5" target="q2">true</edge>
  <edge source="q5" target="q5">true</edge>
</graph>
<graphml>
```