Formale Spezifikation und Verifikation
Symbolische Suche

Wintersemester 2023

Übungsblatt 03

14. November 2023

Verifikation durch Suche im Transitionssystem

Algorithmus: Suche im Transitionssystem mit Mengenoperationen
Eingabe: $T = (\Sigma, \sigma^I, \rightarrow)$
Ausgabe: σ^R erreichbare Zustände von T
Lokal: τ noch zu untersuchende Menge von Zuständen

$\sigma^R := \varnothing$ schon erreicht

$\tau := \sigma^I$ noch zu besuchen

while not $\tau \subseteq \sigma^R$ **do**

  $\sigma^R := \sigma^R \cup \tau$

  $\tau := Post(\tau)$

end while

Verifikation durch Suche im Transitionssystem

Algorithmus: Suche im Transitionssystem mit Mengenoperationen
Eingabe: $T = (\Sigma, \sigma^I, \rightarrow)$
Ausgabe: σ^R erreichbare Zustände von T
Lokal: τ noch zu untersuchende Menge von Zuständen

$\sigma^R := \emptyset$ schon erreicht

$\tau := \sigma^I$ noch zu besuchen

while not $\tau \subseteq \sigma^R$ **do**

  $\sigma^R := \sigma^R \cup \tau$

  $\tau := Post(\tau)$

end while

Symbolische Suche: Repräsentiere Mengen σ^I, σ^R, τ mit Formeln

Programm (a)

```c
unsigned int x = 0;
unsigned int y = 0;

int main(void) {

  x = __VERIFIER_nondet_uint();
  y = __VERIFIER_nondet_uint();

  if (x != 0) {
    y = 0;
  }
  if (y != 0) {
    x = 0;
  }

  if (x * y == 0) {
    ERROR:
    return 1;
  }
  return 0;
}
```

Programm (a)

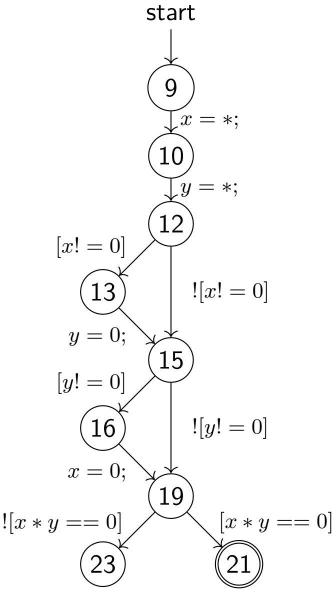
```c
unsigned int x = 0;
unsigned int y = 0;

int main(void) {

  x = __VERIFIER_nondet_uint();
  y = __VERIFIER_nondet_uint();

  if (x != 0) {
    y = 0;
  }
  if (y != 0) {
    x = 0;
  }

  if (x * y == 0) {
    ERROR:
    return 1;
  }
  return 0;
}
```



Symbolische Verifikation für Programm (a)

Spezifikation: Zeile 21 soll nicht erreicht werden.
Als Formel: $pc = 21$

Initiale Zustände sind alle unmittelbar vor Ausführung der Zeile 9 mit $x = 0$ und $y = 0$.
Als Formel: $\sigma^I := pc = 9 \wedge x = 0 \wedge y = 0$

σ^R für Programm (a)

$$\sigma^R := (pc = 9 \land x = 0 \land y = 0)$$

σ^R für Programm (a)

$$\begin{aligned}\sigma^R := {} & (pc = 9 \land x = 0 \land y = 0) \\ & \lor (pc = 10 \land x \in [\![\texttt{uint}]\!] \land y = 0)\end{aligned}$$

σ^R für Programm (a)

$$\begin{aligned}\sigma^R := {} & (pc = 9 \land x = 0 \land y = 0) \\ & \lor (pc = 10 \land x \in [\![\texttt{uint}]\!] \land y = 0) \\ & \lor (pc = 12 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!])\end{aligned}$$

σ^R für Programm (a)

$$
\begin{aligned}
\sigma^R := {} & (pc = 9 \land x = 0 \land y = 0) \\
& \lor (pc = 10 \land x \in [\![\texttt{uint}]\!] \land y = 0) \\
& \lor (pc = 12 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!]) \\
& \lor (pc = 13 \land x \in [\![\texttt{uint}]\!] \setminus \{0\} \land y \in [\![\texttt{uint}]\!]) \\
& \lor (pc = 15 \land {\color{blue} x \in [\![\texttt{uint}]\!]} \land y \in [\![\texttt{uint}]\!] \land x = 0)
\end{aligned}
$$

σ^R für Programm (a)

$$
\begin{aligned}
\sigma^R := {} & (pc = 9 \land x = 0 \land y = 0) \\
& \lor (pc = 10 \land x \in [\![\texttt{uint}]\!] \land y = 0) \\
& \lor (pc = 12 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!]) \\
& \lor (pc = 13 \land x \in [\![\texttt{uint}]\!] \setminus \{0\} \land y \in [\![\texttt{uint}]\!]) \\
& \lor (pc = 15 \land {\color{blue} x \in [\![\texttt{uint}]\!]} \land y \in [\![\texttt{uint}]\!] \land x = 0) \\
& \lor (pc = 15 \land x \in [\![\texttt{uint}]\!] \setminus \{0\} \land y = 0) \\
& \lor (pc = 16 \land x = 0 \land y \in [\![\texttt{uint}]\!] \setminus \{0\}) \\
& \lor (pc = 19 \land x = 0 \land y = 0)
\end{aligned}
$$

σ^R für Programm (a)

$$\begin{aligned}
\sigma^R := (&pc = 9 \land x = 0 \land y = 0)\\
&\lor (pc = 10 \land x \in [\![\mathtt{uint}]\!] \land y = 0)\\
&\lor (pc = 12 \land x \in [\![\mathtt{uint}]\!] \land y \in [\![\mathtt{uint}]\!])\\
&\lor (pc = 13 \land x \in [\![\mathtt{uint}]\!] \setminus \{0\} \land y \in [\![\mathtt{uint}]\!])\\
&\lor (pc = 15 \land x \in [\![\mathtt{uint}]\!] \land y \in [\![\mathtt{uint}]\!] \land x = 0)\\
&\lor (pc = 15 \land x \in [\![\mathtt{uint}]\!] \setminus \{0\} \land y = 0)\\
&\lor (pc = 16 \land x = 0 \land y \in [\![\mathtt{uint}]\!] \setminus \{0\})\\
&\lor (pc = 19 \land x = 0 \land y = 0)\\
&\lor (pc = 16 \land x \in [\![\mathtt{uint}]\!] \setminus \{0\} \land y = 0 \land y \neq 0)\\
&\lor (pc = 19 \land x \in [\![\mathtt{uint}]\!] \setminus \{0\} \land y = 0 \land y = 0)\\
&\lor (pc = 19 \land x = 0 \land y \in [\![\mathtt{uint}]\!] \setminus \{0\})\\
&\lor (pc = 21 \land x = 0 \land y = 0 \land x \cdot y = 0)\\
&\lor (pc = 23 \land x = 0 \land y = 0 \land x \cdot y \neq 0)
\end{aligned}$$

Entferne redundante Terme und unerfüllbare Formeln

σ^R für Programm (a)

$$
\begin{aligned}
\sigma^R := {} & (pc = 9 \wedge x = 0 \wedge y = 0) \\
& \vee (pc = 10 \wedge x \in [\![\texttt{uint}]\!] \wedge y = 0) \\
& \vee (pc = 12 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!]) \\
& \vee (pc = 13 \wedge x \in [\![\texttt{uint}]\!] \setminus \{0\} \wedge y \in [\![\texttt{uint}]\!]) \\
& \vee (pc = 15 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!] \wedge x = 0) \\
& \vee (pc = 15 \wedge x \in [\![\texttt{uint}]\!] \setminus \{0\} \wedge y = 0) \\
& \vee (pc = 16 \wedge x = 0 \wedge y \in [\![\texttt{uint}]\!] \setminus \{0\}) \\
& \vee (pc = 19 \wedge x = 0 \wedge y = 0) \\
& \vee (pc = 16 \wedge x \in [\![\texttt{uint}]\!] \setminus \{0\} \wedge y = 0 \wedge y \neq 0) \\
& \vee (pc = 19 \wedge x \in [\![\texttt{uint}]\!] \setminus \{0\} \wedge y = 0 \wedge y = 0) \\
& \vee (pc = 19 \wedge x = 0 \wedge y \in [\![\texttt{uint}]\!] \setminus \{0\}) \\
& \vee (pc = 21 \wedge x = 0 \wedge y = 0 \wedge x \cdot y = 0) \\
& \vee (pc = 23 \wedge x = 0 \wedge y = 0 \wedge x \cdot y \neq 0)
\end{aligned}
$$

Nun gilt $\sigma^R \wedge pc = 21$ ist erfüllbar $\rightarrow$ Fehlerzustand ist erreichbar

Programm (b)

```
unsigned int x = 0;
unsigned int y = 0;

int main(void) {

  x = __VERIFIER_nondet_uint();
  y = __VERIFIER_nondet_uint();

  if (x * y == 0) {
    return 0;
  }

  if (x * y == 0) {
    ERROR:
    return 1;
  }
  return 0;
}
```

Programm (b)

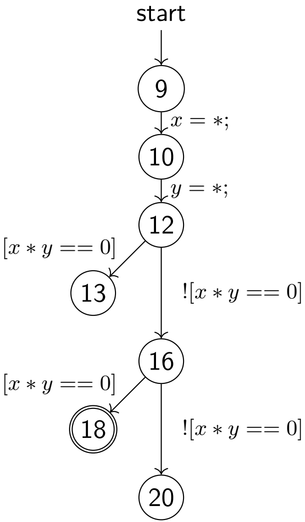
```
unsigned int x = 0;
unsigned int y = 0;

int main(void) {

  x = __VERIFIER_nondet_uint();
  y = __VERIFIER_nondet_uint();

  if (x * y == 0) {
    return 0;
  }

  if (x * y == 0) {
    ERROR:
    return 1;
  }
  return 0;
}
```



Symbolische Verifikation für Programm (b)

Spezifikation: Zeile 18 soll nicht erreicht werden.
Als Formel: $:= pc = 18$

Initiale Zustände sind alle unmittelbar vor Ausführung der Zeile 9 mit $x = 0$ und $y = 0$:
Als Formel: $\sigma^I := pc = 9 \wedge x = 0 \wedge y = 0$

σ^R für Programm (b)

$$\sigma^R := (pc = 9 \land x = 0 \land y = 0)$$

σ^R für Programm (b)

$$\sigma^R := (pc = 9 \land x = 0 \land y = 0)$$
$$\lor (pc = 10 \land x \in [\![\texttt{uint}]\!] \land y = 0)$$

σ^R für Programm (b)

$$\begin{aligned}\sigma^R := {} & (pc = 9 \land x = 0 \land y = 0)\\ & \lor (pc = 10 \land x \in [\![\texttt{uint}]\!] \land y = 0)\\ & \lor (pc = 12 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!])\end{aligned}$$

σ^R für Programm (b)

$$\begin{aligned}
\sigma^R := {} & (pc = 9 \land x = 0 \land y = 0) \\
& \lor (pc = 10 \land x \in [\![\texttt{uint}]\!] \land y = 0) \\
& \lor (pc = 12 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!]) \\
& \lor (pc = 13 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!] \land x \cdot y = 0) \\
& \lor (pc = 16 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!] \land x \cdot y \neq 0)
\end{aligned}$$

σ^R für Programm (b)

$$
\begin{aligned}
\sigma^R := {} & (pc = 9 \wedge x = 0 \wedge y = 0) \\
& \vee (pc = 10 \wedge x \in [\![\texttt{uint}]\!] \wedge y = 0) \\
& \vee (pc = 12 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!]) \\
& \vee (pc = 13 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!] \wedge x \cdot y = 0) \\
& \vee (pc = 16 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!] \wedge x \cdot y \neq 0) \\
& \vee (pc = 18 \wedge x, y \in [\![\texttt{uint}]\!] \wedge x \cdot y \neq 0 \wedge x \cdot y = 0) \\
& \vee (pc = 20 \wedge x = 0 \wedge y \in [\![\texttt{uint}]\!] \setminus \{0\})
\end{aligned}
$$

σ^R für Programm (b)

$$
\begin{aligned}
\sigma^R := {} & (pc = 9 \wedge x = 0 \wedge y = 0) \\
& \vee (pc = 10 \wedge x \in [\![\texttt{uint}]\!] \wedge y = 0) \\
& \vee (pc = 12 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!]) \\
& \vee (pc = 13 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!] \wedge x \cdot y = 0) \\
& \vee (pc = 16 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!] \wedge x \cdot y \neq 0) \\
& \vee (pc = 18 \wedge x, y \in [\![\texttt{uint}]\!] \wedge x \cdot y \neq 0 \wedge x \cdot y = 0) \\
& \vee (pc = 20 \wedge x, y \in R \wedge x \cdot y \neq 0)
\end{aligned}
$$

Vereinfachung: Entferne widersprüchliche Terme aus Disjunktion

σ^R für Programm (b)

$$
\begin{aligned}
\sigma^R := \; & (pc = 9 \land x = 0 \land y = 0) \\
& \lor (pc = 10 \land x \in [\![\texttt{uint}]\!] \land y = 0) \\
& \lor (pc = 12 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!]) \\
& \lor (pc = 13 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!] \land x \cdot y = 0) \\
& \lor (pc = 16 \land x \in [\![\texttt{uint}]\!] \land y \in [\![\texttt{uint}]\!] \land x \cdot y \neq 0) \\
& \lor (pc = 18 \land x, y \in [\![\texttt{uint}]\!] \land x \cdot y \neq 0 \land x \cdot y = 0) \\
& \lor (pc = 20 \land x, y \in R \land x \cdot y \neq 0)
\end{aligned}
$$

Nun gilt $Post(\sigma^R) \subseteq \sigma^R$ weshalb der Algorithmus keine neuen Zustände mehr finden wird und dann terminiert.

σ^R für Programm (b)

$$
\begin{aligned}
\sigma^R := {} & (pc = 9 \wedge x = 0 \wedge y = 0) \\
& \vee (pc = 10 \wedge x \in [\![\texttt{uint}]\!] \wedge y = 0) \\
& \vee (pc = 12 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!]) \\
& \vee (pc = 13 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!] \wedge x \cdot y = 0) \\
& \vee (pc = 16 \wedge x \in [\![\texttt{uint}]\!] \wedge y \in [\![\texttt{uint}]\!] \wedge x \cdot y \neq 0) \\
& \vee (pc = 18 \wedge x, y \in [\![\texttt{uint}]\!] \wedge x \cdot y \neq 0 \wedge x \cdot y = 0) \\
& \vee (pc = 20 \wedge x, y \in R \wedge x \cdot y \neq 0)
\end{aligned}
$$

Nun gilt $Post(\sigma^R) \subseteq \sigma^R$ weshalb der Algorithmus keine neuen Zustände mehr finden wird und dann terminiert.
Fehlerzustand nicht erreicht: $\sigma^R \wedge pc = 18$ ist unerfüllbar.