Formale Spezifikation und Verifikation
Sommersemester 2017

Übungsblatt 04
Wiederholung

Philipp Wendler

7. Juni 2017

Definitionen

- Was ist ein Zeuge?
- Was ist Interpretation?
- Was ist eine symbolische Invariante?
- Welche beiden Klassen von Eigenschaften kann man verifizieren?

Verifikation

- Was ist σ^I und für was wird es benötigt?
- Was ist σ^R und für was wird es benötigt?
- Was ist der Unterschied zwischen aufzählender und symbolischer Suche?
- Wie fängt man an bei der aufzählenden und symbolischen Suche?
- Wie zeichnet man Graphen von aufzählender und symbolischer Suche?

Verifikation durch aufzählende Suche

Algorithm 1: Aufzählende Suche in Transitionssystemen

Input : $T = (\Sigma, \sigma^I, \rightarrow)$
Output: erreichbare Region σ^R von T // bereits untersucht
Local : Menge T von Zuständen von Σ // zu untersuchen

```
1  begin
2      σ^R := ∅;
3      τ := σ^I;
4      while τ ≠ ∅ do
5          wähle s ∈ τ; τ := τ \ {s};
6          if s ∉ σ^R then
7              σ^R := σ^R ∪ {s};
8              τ := τ ∪ post(s);
9          end
10     end
11 end
```

Verifikation durch symbolische Suche

Algorithm 2: Symbolische Suche

Input : Eingabe: T, σ^T

Output: Antwort zum Erreichbarkeitsproblem (T, σ^T)

```
1 begin
2 |   σ^R := InitReg(T) % Symbolische Repräsentation von σ^I;
3 |   repeat
4 |   |   if σ^R ∩ σ^T ≠ ∅ then return YES;
5 |   |   if PostReg(σ^R, T) ⊆ σ^R then return NO;
6 |   |   σ^R = σ^R ∪ PostReg(σ^R, T);
7 |   until forever;
8 end
```

Programm (a)

```
1  int main(void) {
2
3      unsigned int x = 3;
4      unsigned int y = 5;
5
6      if (x + y != 8) {
7        ERROR:
8        return 1;
9      }
10     return 0;
11 }
```

Programm (b)

```
1  int main(void) {
2
3      unsigned int x = 3;
4      unsigned int y = 5;
5
6      if (x + y == 8) {
7          ERROR:
8          return 1;
9      }
10     return 0;
11 }
```

Programm (c)

```
1  int main(void) {
2
3    unsigned int x = nondet();
4    unsigned int y = nondet();
5
6    if (x != 0) {
7      y = 0;
8    }
9    if (y != 0) {
10     x = 0;
11   }
12
13   if (x * y == 0) {
14     ERROR:
15     return 1;
16   }
17   return 0;
18 }
```

Symbolische Verifikation für Programm (c)

Spezifikation: Zeile 14 soll nicht erreicht werden.
$\sigma^T := \{c \mid c(pc) = 14\}$
Kurznotation: $\sigma^T := (pc = 14)$

Initiale Zustände sind alle vor Zeile 3:
InitReg(T) $:= \{c \mid c(pc) = 3\}$
Kurznotation: InitReg(T) $:= (pc = 3)$

σ^R für Programm (c)

$$\sigma^R := (pc = 3)$$

σ^R für Programm (c)

$$\sigma^R := (pc = 3)$$
$$\vee\, (pc = 4 \wedge x \in R)$$

σ^R für Programm (c)

$$\begin{aligned}\sigma^R := {} & (pc = 3) \\ & \vee (pc = 4 \wedge x \in R) \\ & \vee (pc = 6 \wedge x \in R \wedge y \in R)\end{aligned}$$

σ^R für Programm (c)

$$\begin{aligned}\sigma^R := {} & (pc = 3) \\ & \vee (pc = 4 \wedge x \in R) \\ & \vee (pc = 6 \wedge x \in R \wedge y \in R) \\ & \vee (pc = 7 \wedge x \in R \setminus \{0\} \wedge y \in R) \\ & \vee (pc = 9 \wedge \cancel{x \in R} \wedge y \in R \wedge x = 0)\end{aligned}$$

σ^R für Programm (c)

$$
\begin{aligned}
\sigma^R := \; & (pc = 3) \\
& \vee (pc = 4 \wedge x \in R) \\
& \vee (pc = 6 \wedge x \in R \wedge y \in R) \\
& \vee (pc = 7 \wedge x \in R \setminus \{0\} \wedge y \in R) \\
& \vee (pc = 9 \wedge \cancel{x \in R} \wedge y \in R \wedge x = 0) \\
& \vee (pc = 9 \wedge x \in R \setminus \{0\} \wedge y = 0) \\
& \vee (pc = 10 \wedge x = 0 \wedge y \in R \setminus \{0\}) \\
& \vee (pc = 13 \wedge x = 0 \wedge y = 0)
\end{aligned}
$$

σ^R für Programm (c)

$$
\begin{aligned}
\sigma^R := \; & (pc = 3) \\
& \vee (pc = 4 \wedge x \in R) \\
& \vee (pc = 6 \wedge x \in R \wedge y \in R) \\
& \vee (pc = 7 \wedge x \in R \setminus \{0\} \wedge y \in R) \\
& \vee (pc = 9 \wedge \cancel{x \in R} \wedge y \in R \wedge x = 0) \\
& \vee (pc = 9 \wedge x \in R \setminus \{0\} \wedge y = 0) \\
& \vee (pc = 10 \wedge x = 0 \wedge y \in R \setminus \{0\}) \\
& \vee (pc = 13 \wedge x = 0 \wedge y = 0) \\
& \vee (pc = 10 \wedge x \in R \setminus \{0\} \wedge y = 0 \wedge y \neq 0) \\
& \vee (pc = 13 \wedge x \in R \setminus \{0\} \wedge y = 0 \wedge \cancel{y = 0}) \\
& \vee (pc = 13 \wedge x = 0 \wedge y \in R \setminus \{0\}) \\
& \vee (pc = 14 \wedge x = 0 \wedge y = 0 \wedge \cancel{x \cdot y = 0}) \\
& \vee (pc = 17 \wedge x = 0 \wedge y = 0 \wedge x \cdot y \neq 0)
\end{aligned}
$$

σ^R für Programm (c)

$$
\begin{aligned}
\sigma^R := \; & (pc = 3) \\
& \vee (pc = 4 \wedge x \in R) \\
& \vee (pc = 6 \wedge x \in R \wedge y \in R) \\
& \vee (pc = 7 \wedge x \in R \setminus \{0\} \wedge y \in R) \\
& \vee (pc = 9 \wedge \cancel{x \in R} \wedge y \in R \wedge x = 0) \\
& \vee (pc = 9 \wedge x \in R \setminus \{0\} \wedge y = 0) \\
& \vee (pc = 10 \wedge x = 0 \wedge y \in R \setminus \{0\}) \\
& \vee (pc = 13 \wedge x = 0 \wedge y = 0) \\
\\
& \vee (pc = 13 \wedge x \in R \setminus \{0\} \wedge y = 0 \wedge \cancel{y = 0}) \\
& \vee (pc = 13 \wedge x = 0 \wedge y \in R \setminus \{0\}) \\
& \vee (pc = 14 \wedge x = 0 \wedge y = 0 \wedge \cancel{x \cdot y = 0})
\end{aligned}
$$

Simplification: remove contradicting terms from disjunction

σ^R für Programm (c)

$$
\begin{aligned}
\sigma^R := \; & (pc = 3) \\
& \vee (pc = 4 \wedge x \in R) \\
& \vee (pc = 6 \wedge x \in R \wedge y \in R) \\
& \vee (pc = 7 \wedge x \in R \setminus \{0\} \wedge y \in R) \\
& \vee (pc = 9 \wedge \cancel{x \in R} \wedge y \in R \wedge x = 0) \\
& \vee (pc = 9 \wedge x \in R \setminus \{0\} \wedge y = 0) \\
& \vee (pc = 10 \wedge x = 0 \wedge y \in R \setminus \{0\}) \\
& \vee (pc = 13 \wedge x = 0 \wedge y = 0) \\
\\
& \vee (pc = 13 \wedge x \in R \setminus \{0\} \wedge y = 0 \wedge \cancel{y = 0}) \\
& \vee (pc = 13 \wedge x = 0 \wedge y \in R \setminus \{0\}) \\
& \vee (pc = 14 \wedge x = 0 \wedge y = 0 \wedge \cancel{x \cdot y = 0})
\end{aligned}
$$

Now $\sigma^R \cap \sigma^T = \{\{pc \mapsto 14, x \mapsto 0, y \mapsto 0\}\} \neq \emptyset$ and thus the algorithm returns YES.

Programm (d)

```
1  int main(void) {
2
3    unsigned int x = nondet();
4    unsigned int y = nondet();
5
6    if (x * y == 0) {
7      return 0;
8    }
9
10   if (x * y == 0) {
11     ERROR:
12     return 1;
13   }
14   return 0;
15 }
```