Lehrstuhl für Software and Computational Systems
Prof. Dr. Gidon Ernst, Marian Lingsch-Rosenfeld

Übungen zur Vorlesung
Formale Spezifikation und Verifikation

Wintersemester 2024/25

Übungsblatt 02

Bekanntgabe am 04.11.2024

Für dieses Übungsblatt wird Aufgabe 1 in der Übung vorgerechnet, bitte bereiten Sie Aufgabe 2 selbständig vor.

1 Transitionssysteme

Auf der nächsten Seite sind vier C-Programme gegeben. Verifizieren Sie die Programme 1a und 1b mit Hilfe der expliziten Erreichbarkeitsanalyse, d.h. dem Algorithmus für die aufzählende Suche (Folie 37). Die Zustände sind hierbei *konkret*, d.h., die Variablen sind mit Zahlen oder Wahrheitswerten belegt.

- Verwenden Sie die Notation $\{\texttt{pc} \mapsto \langle pc\rangle, \texttt{x} \mapsto \langle x\rangle, \texttt{y} \mapsto \langle y\rangle\}$.
  Das Attribut $\langle pc\rangle$ steht für *program counter* und ist die Nummer eines Knotens im Kontrollflussautomaten, welche der jeweiligen Zeilennummer im Programm-Code entspricht. Für den Startzustand gilt beispielsweise $\texttt{pc} \mapsto 5$. Die Notation $\langle x\rangle$ und $\langle y\rangle$ steht jeweils für konkrete Zahlen (sind also keine Variablen!)
- Alternativ können Sie auch eine abkürzende Schreibweise wie in den Folien verwenden: $(\langle pc\rangle, \langle x\rangle, \langle y\rangle)$. Geben Sie in diesem Fall die Legende mit an.
- Hinweis: Ordnen Sie die Zustände in einem Baum mit allen möglichen Ausführungen an (Folie 34)

Beantworten Sie außerdem, ob der Zustand am Label `ERROR` erreichbar ist. Geben Sie in dem Fall eine Spur an, welche dort hin führt (bzw. markieren Sie diese in Ihrer Analyse).

2 Explizite Erreichbarkeitsanalyse mit Nichtdeterminismus

Versuchen Sie ebenfalls, die Programme 1c und 1d mit dem obigen Ansatz zu verifizieren. Für die externe Funktion `__VERIFIER_nondet_uint()` können Sie annehmen, dass bei jedem Aufruf ein nichtdeterministischer Wert aus dem Wertebereich des Typs `unsigned int` zurückgegeben wird (d.h. ein nicht-negativer Wert aus dem Bereich 0 und $2^{32} - 1$).

- Was stellen Sie fest? Wie groß ist das zugehörige Transitionssystem?
- Führen Sie die Analyse für einzelne Fälle beispielhaft durch

Die Aufgabe dient als Vorgriff auf die Vorlesung der nächsten Woche—Sie können an dem Beispiel gut erkennen, welche Limitierungen die explizite Erreichbarkeitsanalyse in der Praxis haben kann.

```c
unsigned int x = 0;
unsigned int y = 0;
int main(void) {

  x = 3;
  y = 5;

  if (x + y != 8) {
    ERROR:
    return 1;
  }
  return 0;
}
```

(a)

```c
unsigned int x = 0;
unsigned int y = 0;
int main(void) {

  x = 3;
  y = 5;

  if (x + y == 8) {
    ERROR:
    return 1;
  }
  return 0;
}
```

(b)

```c
extern unsigned int
  __VERIFIER_nondet_uint();
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;
}
```

(c)

```c
extern unsigned int
  __VERIFIER_nondet_uint();
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;
}
```

(d)