Formale Spezifikation und Verifikation
Transitionssysteme

Wintersemester 2024

Lösung Übungsblatt 02

Prof. Dr. Gidon Ernst, Marian Lingsch-Rosenfeld, Simon Rossmair, Noah König

4. November 2024

Komponenten eines Transitionssystems

- Menge Σ aller Zustände
- Teilmenge $\sigma^I \subseteq \Sigma$ der Startzustände (nicht leer)
- Transitionsrelation $\rightarrow \subseteq \Sigma \times \Sigma$,
  mit $\text{post}(s) = \{s' \in \Sigma \mid s \rightarrow s'\}$ eines Zustands s

Spur

- Sequenz $\bar{s} = \langle s_0, \dots, s_n \rangle$ von Zuständen
- Zustände über Transitionsrelation verbunden:
  $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n$
  Oder: $(s_0, s_1) \in \rightarrow$, $(s_1, s_2) \in \rightarrow$, etc.
- Analogie: ausführbarer Pfad eines Programms

Algorithmus: Aufzählende Suche

(a) Laufzeit:

(b) Terminiertheit:

(c) Anwendung:

Algorithmus: Aufzählende Suche

(a) Laufzeit: linear in der Anzahl von erreichbaren Transitionen

(b) Terminiertheit:

(c) Anwendung:

Algorithmus: Aufzählende Suche

(a) Laufzeit: linear in der Anzahl von erreichbaren Transitionen

(b) Terminiertheit: Wenn Menge <u>erreichbarer</u> Zustände endlich ist

(c) Anwendung:

Algorithmus: Aufzählende Suche

(a) Laufzeit: linear in der Anzahl von erreichbaren Transitionen

(b) Terminiertheit: Wenn Menge erreichbarer Zustände endlich ist

(c) Anwendung:
  - Einfach zu implementieren
  - Liefert alle erreichbaren (konkreten) Zustände
  - Kann zur Verifikation benutzt werden
  - Für kleine, endliche Systeme ausreichend
  - Für größere Systeme existieren bessere Verfahren: symbolische Zustandsrepräsentation (aktuelle Vorlesung)
  - Generelle Struktur des Algorithmus bleibt gleich: Suche mit Warteliste τ

Verifikation durch aufzählende Suche

Algorithm 1 Aufzählende Suche in Transitionssystemen

Input : $T = (\Sigma, \sigma^I, \rightarrow)$
Output : erreichbare Zustände σ^R von T
Local : Warteliste τ noch zu untersuchender Zustände

begin
  $\sigma^R := \emptyset$ `// schon gefunden und verfolgt`
  $\tau := \sigma^I$ `// alle Initialzustände noch zu besuchen`
  while $\tau \neq \emptyset$ **do**
    wähle $s \in \tau$ `// nächster zu besuchender Zustand`
    $\tau := \tau \setminus \{s\}$ `// aus Warteliste entfernen`
    if $s \notin \sigma^R$ **then**
      `// falls neuer Zustand:`
      $\sigma^R := \sigma^R \cup \{s\}$ `// als besucht merken`
      $\tau := \tau \cup post(s)$; `// und Nachfolger in die Warteliste einfügen`
    end
  end
end

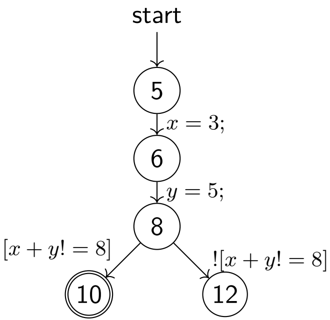
Programm (a)

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

  x = 3;
  y = 5;

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

Programm (a)

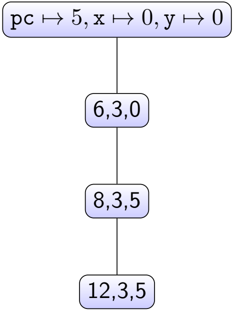
```
unsigned int x = 0;
unsigned int y = 0;
int main(void) {

    x = 3;
    y = 5;

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

Kontrollflussautomat

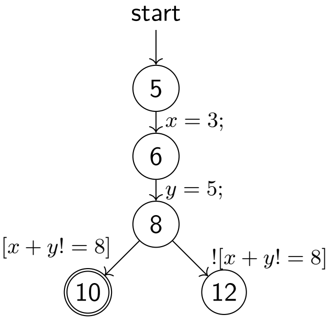


Programm (a)

Erreichbares Transitionssystem



Kontrollflussautomat



Programm (a)

$\sigma^I \subseteq \Sigma$ (Startzustände): $\quad s_0 = \{pc \mapsto 5, \quad x \mapsto 0, \quad y \mapsto 0\}$

Erreichbare Zustände σ^R:

$$\begin{aligned}
s_0 &= \{pc \mapsto 5, & x \mapsto 0, && y \mapsto 0\} \\
s_1 &= \{pc \mapsto 6, & x \mapsto 3, && y \mapsto 0\} \\
s_2 &= \{pc \mapsto 8, & x \mapsto 3, && y \mapsto 5\} \\
s_3 &= \{pc \mapsto 12, & x \mapsto 3, && y \mapsto 5\}
\end{aligned}$$

Erreichbare Transitionen: $\quad \{s_0 \to s_1, s_1 \to s_2, s_2 \to s_3\}$

$\Rightarrow$ Programm (a) erfüllt die Spezifikation
(Fehler-Label ist nicht erreichbar, da kein Zustand mit
Programzählerwert 10 erreichbar ist!)

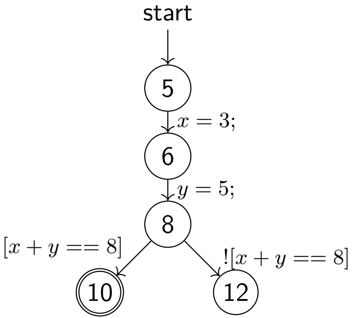
Programm (b)

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

  x = 3;
  y = 5;

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

Programm (b)

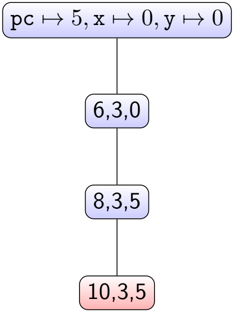
```c
unsigned int x = 0;
unsigned int y = 0;
int main(void) {

  x = 3;
  y = 5;

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

Kontrollflussautomat

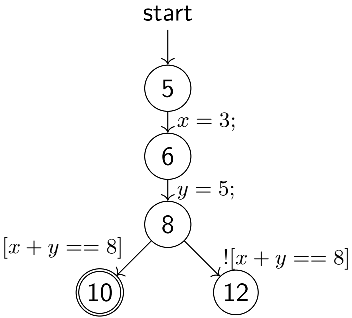


Programm (b)

Erreichbares Transitionssystem



Kontrollflussautomat



Programm (b)

$\sigma^I \subseteq \Sigma$ (Startzustände): $\quad s_0 = \{pc \mapsto 5, \quad x \mapsto 0, \quad y \mapsto 0\}$

Erreichbare Zustände σ^R:

$$\begin{aligned}
s_0 &= \{pc \mapsto 5, & x \mapsto 0, && y \mapsto 0\} \\
s_1 &= \{pc \mapsto 6, & x \mapsto 3, && y \mapsto 0\} \\
s_2 &= \{pc \mapsto 8, & x \mapsto 3, && y \mapsto 5\} \\
s_3 &= \{pc \mapsto 10, & x \mapsto 3, && y \mapsto 5\}
\end{aligned}$$

Erreichbare Transitionen: $\quad \{s_0 \to s_1, s_1 \to s_2, s_2 \to s_3\}$

$\Rightarrow$ Programm (b) erfüllt die Spezifikation <u>nicht</u>
(Fehler-Label ist erreichbar, da Zustand 10 erreichbar!)

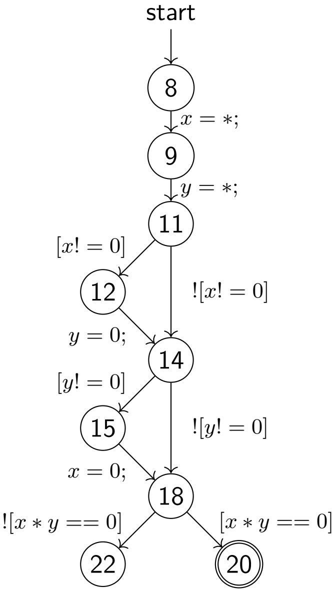
Ein Zeuge (*Witness*) dafür ist die folgende Spur: $\langle s_0, s_1, s_2, s_3 \rangle$

Programm (c)

```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 (c)

```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;
}
```



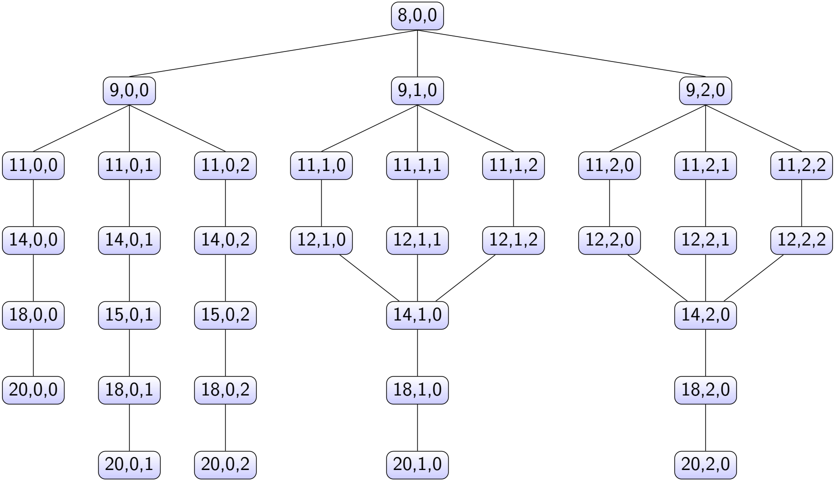
Programm (c)

- Zustandsexplosion durch nichtdeterministische Werte:

$$
\begin{aligned}
\mathit{post}(\{pc \mapsto 8, x \mapsto 0, y \mapsto 0\}) = \{ & \\
& \{pc \mapsto 9, x \mapsto 0, y \mapsto 0\}, \\
& \{pc \mapsto 9, x \mapsto 1, y \mapsto 0\}, \\
& \{pc \mapsto 9, x \mapsto 2, y \mapsto 0\}, \\
& \dots \\
& \{pc \mapsto 9, x \mapsto 4294967295, y \mapsto 0\} \\
& \}
\end{aligned}
$$

  (Unter Annahme `UINT_MAX=4294967295`)
- $\Rightarrow$ Reduziere im Folgenden auf `UINT_MAX=2`

Programm (c)



Abkürzende Schreibweise $\boxed{pc,\ x,\ y}$ für $\{\mathrm{pc} \mapsto pc, \mathrm{x} \mapsto x, \mathrm{y} \mapsto y\}$

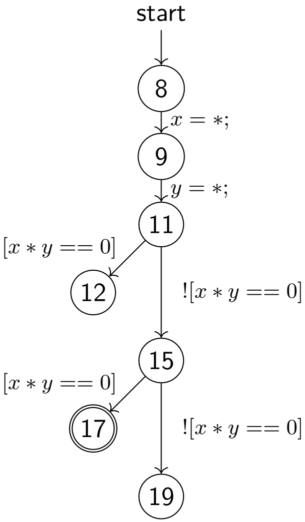
Ergebnis: Programm (c) erfüllt die Spezifikation nicht!

Programm (d)

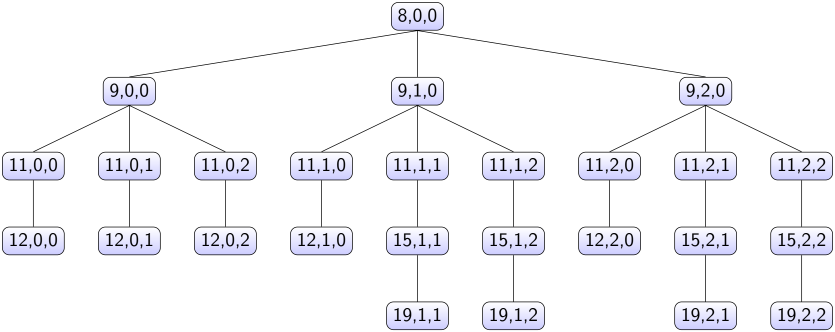
```
unsigned int x = 0;
unsigned int y = 0;

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 (d)

```c
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 (d)



Abkürzende Schreibweise $\boxed{pc,\ x,\ y}$ für $\{\mathrm{pc} \mapsto pc, \mathrm{x} \mapsto x, \mathrm{y} \mapsto y\}$

Ergebnis: Programm (d) erfüllt die Spezifikation