Vorlesung vom 23.05.06: Generierung von Verifikationsbedingungen: Beispiele

Wo sind wir?

- *Vorletzte Wochen*: Korrektheit, Vollständigkeit des Hoare-Kalküls
- *Letzte Woche*: Verifikationsbedingungen
- *Diese Woche*: Beispiele für Generierung von Verifikationsbedingungen

Beispiel 1: `swap`

$$\begin{aligned}
\texttt{swap} &\equiv t := x; x := y; y := t \\
pre &\equiv x \texttt{ = } X \texttt{ and } y \texttt{ = } Y \\
post &\equiv x \texttt{ = } Y \texttt{ and } y \texttt{ = } X
\end{aligned}$$

Beispiel 2: `max`

$$\begin{aligned}
\texttt{max} &\equiv \texttt{if } x \texttt{ < } y \texttt{ then } m := y \texttt{ else } m := x \texttt{ fi} \\
pre &\equiv \texttt{True} \\
post &\equiv x \texttt{ <= } m \texttt{ and } y \texttt{ <= } m
\end{aligned}$$

Beispiel 3: `sum`

$$\begin{aligned}
\texttt{sum} &\equiv i := 0; s := 0; \\
&\quad \{J\}\texttt{while not } (i \texttt{ = } n) \texttt{ do } \{I\} i := i \texttt{ + } 1; s := s \texttt{ + } i \texttt{ od} \\
pre &\equiv 0 \texttt{ <= } n \\
post &\equiv s = sum(1, n) \\
I &\equiv s = sum(1, i) \\
J &\equiv i \texttt{ = } 0 \texttt{ and } s \texttt{ = } 0
\end{aligned}$$

Beispiel 3a: `sum`

$$\begin{aligned}
\texttt{sum} &\equiv i := 0; s := 0; \\
&\quad \{J\}\texttt{while } i \texttt{ < } n \texttt{ do } \{I\} s := s \texttt{ + } i; i := i \texttt{ + } 1 \texttt{ od} \\
pre &\equiv 0 \texttt{ <= } n \\
post &\equiv s = sum(1, n) \\
I &\equiv s = sum(1, (i + 1)) \texttt{ and } i \texttt{ <= } n \\
J &\equiv i \texttt{ = } 0 \texttt{ and } s \texttt{ = } 0
\end{aligned}$$

Beispiel 4: `times`

$$\begin{aligned}
\texttt{times} &\equiv i := 0; p := 0; \\
&\quad \{J\}\texttt{while } i \texttt{ < } n \texttt{ do } \{I\} p := p \texttt{ + } m; i := i \texttt{ + } 1 \texttt{ od} \\
pre &\equiv 0 \texttt{ <= } n \\
post &\equiv p = m \texttt{ * } n \\
I &\equiv p = m * i \texttt{ and } i \texttt{ <= } n \\
J &\equiv i \texttt{ = } 0 \texttt{ and } p \texttt{ = } 0
\end{aligned}$$

Beispiel 5: `div`

$$\begin{aligned}
\texttt{div} &\equiv q := 0; r := x; \\
&\quad \{J\}\texttt{while not } (r \texttt{ < } y) \texttt{ do } \{I\} r := r \texttt{ - } y; q := q \texttt{ + } 1 \texttt{ od} \\
pre &\equiv 0 \texttt{ <= } x \texttt{ and } 0 \texttt{ <= } y \\
post &\equiv x \texttt{ = } q \texttt{ * } y \texttt{ + } r \texttt{ and } 0 \texttt{ <= } r \texttt{ and } r \texttt{ < } y \\
I &\equiv x \texttt{ = } q \texttt{ * } y \texttt{ + } r \texttt{ and } 0 \texttt{ <= } r \\
J &\equiv q \texttt{ = } 0 \texttt{ and } r \texttt{ = } x
\end{aligned}$$

Beispiel 6: `euclid`

$$\begin{array}{rcl}
\texttt{euclid} & \equiv & a := x; b := y; \{J\} \\
& & \texttt{while not } (a = b) \texttt{ do} \\
& & \quad \{I\} \texttt{ if } a \texttt{ <= } b \texttt{ then } b := b - a \\
& & \quad\quad\quad\quad\quad \texttt{else } a := a - b \\
& & \quad\quad\quad\quad\quad \texttt{fi od} \\
pre & \equiv & \texttt{True} \\
post & \equiv & a = \mathrm{gcd}(x, y) \\
I & \equiv & \mathrm{gcd}(x, y) = \mathrm{gcd}(a, b) \\
J & \equiv & x \texttt{ = } a \texttt{ and } b \texttt{ = } y
\end{array}$$

Beispiel 7: `pow`

$$\begin{array}{rcl}
\texttt{pow} & \equiv & \texttt{while not } (y = 0) \texttt{ do} \\
& & \quad \{I_1\} \texttt{ while } even(y)) \texttt{ do} \\
& & \quad\quad \{I_2\}\, x := x \texttt{ * } x; y := y/2; \\
& & \quad\quad \texttt{od;} \\
& & \quad z := z \texttt{ * } x; y := y - 1 \\
& & \quad \texttt{od} \\
pre & \equiv & x = m \texttt{ and } y = n \texttt{ and } z = 1 \\
post & \equiv & z = m^n \\
I_1 & \equiv & m^n = z * x^y \\
I_2 & \equiv & m^n = z * x^y
\end{array}$$

Beispiel 8: `isqr`

$$\begin{array}{rcl}
\texttt{isqr} & \equiv & t := 1; s := t; i := 0; \{J\} \\
& & \texttt{while } s \texttt{ <= } a \texttt{ do } \{I\} \\
& & \quad t := t + 2; \\
& & \quad s := s + t; \\
& & \quad i := i + 1 \texttt{fi od} \\
pre & \equiv & 0 \texttt{ <= } a \\
post & \equiv & i \texttt{ * } i \texttt{ <= } a \texttt{ and } a \texttt{ < } (i+1) \texttt{ * } (i+1) \\
I & \equiv & ? \\
J & \equiv & t \texttt{ = } 1 \texttt{ and } s \texttt{ = } 1 \texttt{ and } i \texttt{ = } 0 \\
Hinweis: & & 1 + 3 + 5 + \ldots + (2n - 1) = n^2
\end{array}$$

Zusammenfassung

- Auch *einfache* Programme generieren *umfangreiche* Verifikationsbedingungen
- Finden der *Invariante* nicht immer einfach
- Rest *mechanisch*
- *Deduktion* (zur *Vereinfachung*) vonnöten