Korrekte Software: Grundlagen und Methoden
Vorlesung 14 vom 23.06.16: VCG Revisited

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2016



## Motivation

- Rückwärtsrechnung: es entstehen viele indeterminierte Zwischenzustände, über die wir nichts sagen können.
- Bsp. *swap*: Validität von $*y$ in der letzten Anweisung.
- Dadurch können Beweisverpflichtungen nicht direkt bewiesen werden.
- Die den Zustand beschreibenden Ausdrücke werden immer größer.

```
void swap (int *x, int *y)
/** post \old(*x) == *y
       && \old(*y) == x; */
{
  int z;

  z= *x;
  *x= *y;
  *y= z;
}
```



## Approximative stärkste Nachbedingung (revisited)

$$\mathrm{asp}(\Gamma, \Lambda, P, c) \qquad \mathrm{svc}(\Gamma, \Lambda, P, c)$$

- $\Gamma$ ist das environment
- $\Lambda$ ist der modification set
- $P : \Sigma \to \mathbf{T}$ ist die Vorbedingung (vor $c$)
- $c$ ist ein Statement
- $\mathrm{svc}(\Gamma, \Lambda, P, c)$ sind die Verifikationsbedingungen
- $\mathrm{asp}(\Gamma, \Lambda, P, c) : \Sigma \to \mathbf{T}$ gilt nach $c$, wenn:
  (i) vorher $P$ gilt,
  (ii) $c$ terminiert, und
  (iii) die Verifikationsbedingungen $\mathrm{svc}(\Gamma, \Lambda, P, c)$ gelten:
  $$\mathrm{svc}(\Gamma, \Lambda, P, c) \longrightarrow \models \{P\}\, c\, \{\mathrm{asp}(\Gamma, \Lambda, P, c)\}$$



## Approximative stärkste Nachbedingung

$$\begin{aligned}
\mathrm{asp}(\Gamma, \Lambda, P, \{\,\}) &\stackrel{def}{=} P \\
\mathrm{asp}(\Gamma, \Lambda, P, \{c\ c_s\}) &\stackrel{def}{=} \mathrm{asp}(\Gamma, \Lambda, \mathrm{asp}(\Gamma, \Lambda, P, c), \{c_s\}) \\
\mathrm{asp}(\Gamma, \Lambda, P, l = e) &\stackrel{def}{=} \lambda S.\, \exists S_0.\, S = upd(S_0, [\![l]\!]^\Gamma_{S_0}, [\![e]\!]^\Gamma_{S_0}) \wedge P(S_0) \\
\mathrm{asp}(\Gamma, \Lambda, P, l = f(e_1, \ldots, e_n)) &\stackrel{def}{=} \\
&\lambda S.\, \exists S_0.\, S = upd(S_0, [\![l]\!]^\Gamma_{S_0}, F([\![e_1]\!]^\Gamma_{S_0}, \ldots, [\![e_n]\!]^\Gamma_{S_0})) \wedge P(S_0) \\
&\text{mit } post(\Gamma!f) \equiv (\forall v_1, \ldots, v_n.\, \mathbf{result} = F(v_1, \ldots, v_n)) \\
\mathrm{asp}(\Gamma, \Lambda, P, f(e_1, \ldots, e_n)) &\stackrel{def}{=} \lambda S.\, \exists S_0.\, Q([\![e_1]\!]^\Gamma_{S_0}, \ldots, [\![e_n]\!]^\Gamma_{S_0})(S_0, S) \\
&\text{mit } post(\Gamma!f) \equiv (\forall v_1, \ldots, v_n. Q(v_1, \ldots, v_n)) \\
\mathrm{asp}(\Gamma, \Lambda, P, \mathbf{if}\ (b)\ c_0\ \mathbf{else}\ c_1) &\stackrel{def}{=} [\![b]\!]^\Gamma \wedge \mathrm{asp}(\Gamma, \Lambda, P, c_0) \\
&\quad \vee (\neg [\![b]\!]^\Gamma \wedge \mathrm{asp}(\Gamma, \Lambda, c_1, P)) \\
\mathrm{asp}(\Gamma, \Lambda, P, /\!*\!*\ \{q\}\ *\!/, P) &\stackrel{def}{=} [\![q]\!]^\Gamma \\
\mathrm{asp}(\Gamma, \Lambda, P, \mathbf{while}\ (b)\ /\!*\!*\ \mathbf{inv}\ i\ *\!/\ c, P) &\stackrel{def}{=} [\![i]\!]^\Gamma \wedge \neg([\![b]\!]^\Gamma) \\
\mathrm{asp}(\Gamma, \Lambda, P, \mathbf{return}\ e) &\stackrel{def}{=} \lambda S.\, post(\Gamma)[[\![e]\!]^\Gamma_S / \mathbf{result}]\ S \\
\mathrm{asp}(\Gamma, \Lambda, P, \mathbf{return}) &\stackrel{def}{=} post(\Gamma)
\end{aligned}$$



## ASP: Sonderregeln

$$\begin{aligned}
&\mathrm{asp}(\Gamma, \Lambda, \lambda S.\, \exists S_0.\, S = f(S_0) \wedge P(S_0), l = e) \stackrel{def}{=} \\
&\qquad \lambda S.\, \exists S_0.\, S = upd(f(S_0), [\![l]\!]^\Gamma_{f(S_0)}, [\![e]\!]^\Gamma_{f(S_0)}) \wedge P(S_0)
\end{aligned}$$

$$\begin{aligned}
&\mathrm{asp}(\Gamma, \Lambda, \lambda S.\, \exists S_0.\, S = f(S_0) \wedge P(S_0), l = g(e_1, \ldots, e_n)) \stackrel{def}{=} \\
&\quad \lambda S.\, \exists S_0.\, S = upd(f(S_0), [\![l]\!]^\Gamma_{f(S_0)}, G([\![e_1]\!]^\Gamma_{f(S_0)}, \ldots, [\![e_n]\!]^\Gamma_{f(S_0)})) \wedge P(S_0) \\
&\quad \text{mit } post(\Gamma, \Lambda!g) \equiv (\forall v_1, \ldots, v_n.\, \mathbf{result} = G(v_1, \ldots, v_n))
\end{aligned}$$

$$\begin{aligned}
&\mathrm{asp}(\Gamma, \Lambda, \lambda S.\, \exists S_0.\, S = f(S_0) \wedge P(S_0), p(e_1, \ldots, e_n)) \stackrel{def}{=} \\
&\qquad \lambda S.\, \exists S_0.\, Q([\![e_1]\!]^\Gamma_{f(S_0)}, \ldots, [\![e_n]\!]^\Gamma_{f(S_0)})(f(S_0), S) \\
&\qquad \text{mit } post(\Gamma, \Lambda!f) \equiv (\forall v_1, \ldots, v_n. Q(v_1, \ldots, v_n)
\end{aligned}$$



## ASP: Weitere Anmerkungen

- $\mathrm{asp}(\Gamma, \Lambda, P, c)$ ist vom Typ $\Sigma \to \mathbf{T}$.
- Boolsche Operatoren sind geliftet:
  $$[\![i]\!]^\Gamma \wedge \neg([\![b]\!]^\Gamma) \equiv \lambda S.\, [\![i]\!]^\Gamma_S \wedge (\neg([\![b]\!]^\Gamma_S))$$
- Zusatzbedingungen:
  1. Für alle Zuweisungsregeln:
     $[\![l]\!]^\Gamma_S$ ist im modification set und eine gültige Lokation
  2. Für die Zuweisungsregel mit Funktionsaufruf:
     modification set von $f$ ist leer ($mod(\Gamma!f) = \emptyset$)



## Verifikationsbedingungen

$$\begin{aligned}
\mathrm{svc}(\Gamma, \Lambda, P, \{\,\}) &\stackrel{def}{=} \emptyset \\
\mathrm{svc}(\Gamma, \Lambda, P, \{c\ c_s\}) &\stackrel{def}{=} \mathrm{svc}(\Gamma, \Lambda, P, c) \\
&\quad \cup \mathrm{svc}(\Gamma, \Lambda, \mathrm{asp}(\Gamma, \Lambda, P, c), \{c_s\}) \\
\mathrm{svc}(\Gamma, P, l = e) &\stackrel{def}{=} \{\ \forall S.\, P(S) \longrightarrow valid(S, [\![l]\!]^\Gamma_S), \\
&\qquad \forall S.\, P(S) \longrightarrow [\![l]\!]^\Gamma_S \in \Lambda\,\} \\
\mathrm{svc}(\Gamma, \Lambda, P, l = f(e_1, \ldots, e_n)) &\stackrel{def}{=} P \longrightarrow pre(\Gamma!f)([\![e_1]\!]^\Gamma_S, \ldots, [\![e_n]\!]^\Gamma) \\
&\quad \cup \{\forall S.\, P(S) \longrightarrow valid(S, [\![l]\!]^\Gamma_S), \\
&\qquad \forall S.\, P(S) \longrightarrow [\![l]\!]^\Gamma_S \in \Lambda\,\} \\
\mathrm{svc}(\Gamma, \Lambda, P, \mathbf{if}\ (b)\ c_0\ \mathbf{else}\ c_1) &\stackrel{def}{=} \mathrm{svc}(\Gamma, \Lambda, [\![b]\!]^\Gamma \wedge P, c_0) \\
&\quad \cup \mathrm{svc}(\Gamma, \Lambda, \neg [\![b]\!]^\Gamma \wedge P, c_1) \\
\mathrm{svc}(\Gamma, \Lambda, P, /\!*\!*\ \{q\}\ *\!/) &\stackrel{def}{=} \{\forall S.\, P(S) \longrightarrow [\![q]\!]^\Gamma_S\} \\
\mathrm{svc}(\Gamma, \Lambda, P, \mathbf{while}\ (b)\ /\!*\!*\ \mathbf{inv}\ i\ *\!/\ c) &\stackrel{def}{=} \\
&\{\forall S.\, \mathrm{asp}(\Gamma, \Lambda, [\![b]\!]^\Gamma \wedge [\![i]\!]^\Gamma, c)(S) \longrightarrow [\![i]\!]^\Gamma_S(S)\} \\
&\cup \{\forall S.\, P(S) \longrightarrow [\![i]\!]^\Gamma_S\} \cup \mathrm{svc}(\Gamma, \Lambda, [\![b]\!]^\Gamma \wedge [\![i]\!]^\Gamma, c) \\
\mathrm{svc}(\Gamma, \Lambda, P, \mathbf{return}\ e) &\stackrel{def}{=} \{\forall S.\, P(S) \longrightarrow post(\Gamma)[[\![e]\!]^\Gamma_S / \mathbf{return}](S)\}
\end{aligned}$$



## Beispiel

```
void zero(int a[], int a_len)
/** pre \array(a, a_len);
    post forall int i; 0 <= i && i < a_len --> a[i] == 0;
*/
{
  int x;

  x= 0;
  while (x < a_len)
/** inv x <= a_len &&
        forall int j; 0 <= j && j < x --> a[j] == 0; */ {
     a[x]= 0;
     x= x+1;
     }
  return;
}
```

## Beispiel

```
int max(int a[], int a_len)
/** pre \array(a, a_len);
    post forall int i; 0 <= i && i < a_len—> a[i]<= \result;
 */
{
 int x;
 int r;

 x= 0;
 r= a[0];
 while(x< a_len)
/** inv x <= a_len &&
        forall int j; 0 <= j && j< x —> a[j] <= r; */ {
   if (a[x] > r) r= a[x];
   x= x+1;
   }
 return r;
}
```