191220154 AsHL

1. $[\, y+2*(x+1)-1=(x+1)^2 \,]\ x:=x+1;\ [\, y+2*x-1=x^2 \,]\ _{AS}$

   $y=x^2 \ \Rightarrow\ y+2*x+1=x^2+2*x+1$

   $[\, y=x^2 \,]\ x:=x+1;\ [\, y+2*x-1=x^2 \,]\ _{SP}$

   $[\, y+2*x-1=x^2 \,]\ y:=y+2*x-1;\ [\, y=x^2 \,]\ _{AS}$

   $[\, y=x^2 \,]\ x:=x+1;\ y:=y+2*x-1;\ [\, y=x^2 \,]\ _{SC}$

   The weakest assertion is $y=x^2$.

   $[\, y=z \wedge (x+y)-y=w \,]\ x=x+y\ [\, y=z \wedge x-y=w \,]\ _{AS}$

   $[\, x-(x-y)=z \wedge x-y=w \,]\ y:=x-y;\ [\, x-y=z \wedge y=w \,]\ _{AS}$

   $[\, x-y=z \wedge y=w \,]\ x:=x-y;\ [\, x=z \wedge y=w \,]\ _{AS}$

   The weakest assertion is $y=z \wedge x=w$.

   $[\, i \wedge (b-(a+1)<x_0) \,]\ a:=a+1;\ [\, i \wedge (b-a<x_0) \,]\ _{AS}$

   $i \wedge (a<b) \wedge (b-a=x_0) \ \Rightarrow\ i \wedge (b-a-1<x_0)$

   $[\, i \wedge (a<b) \wedge (b-a=x_0) \,]\ a:=a+1;\ [\, i \wedge (b-a<x_0) \,]\ _{SP}$

   $[\, i \wedge (b-a<x_0) \,]\ y:=x+y;\ [\, i \wedge (b-a<x_0) \,]\ _{AS}$

   $[\, i \wedge (a<b) \wedge (b-a=x_0) \,]\ a:=a+1;\ y:=x+y;\ [\, i \wedge (b-a<x_0) \,]\ _{SC}$

   $i \wedge (a<b) \ \Rightarrow\ b-a \geq 0$

   $[\, i \,]\ \textbf{\textit{while}}\ a<b\ \textbf{\textit{do}}\ (a:=a+1;\ y:=x+y;)\ [\, i \wedge \neg(a<b) \,]\ _{WHT}$

   $i \wedge \neg(a<b) \ \Rightarrow\ y=x*b$

   Let the loop invariant i be $y=x*a \wedge a \leq b$ to satisfy the above derivation

   Also consider the case where while does not execute at all, then $y=x*b$

   The weakest assertion is $(y=x*a \wedge a \leq b) \vee y=x*b$.

   $\textbf{\textit{while true do skip}}$ changes nothing in the state, it also does not terminate,

   so it is impossible to find a non-negative metric that decreases on each iteration

   We simply put down $\textbf{\textit{false}}$ for the weakest assertion.

   $\{\, i \wedge \textbf{\textit{true}} \,\}\ \textbf{\textit{skip}}\ \{\, i \,\}$

   $i \wedge \textbf{\textit{false}} \ \Rightarrow\ \textbf{\textit{false}}$

   $\{\, i \,\}\ \textbf{\textit{while true do skip}}\ \{\, i \wedge \neg \textbf{\textit{true}} \,\}$

   We simply put down $\textbf{\textit{true}}$ for the weakest assertion.

2. The loop invariant is $(x = y \wedge x \leq 100) \vee x = 0$.

Actually $x = y \wedge x \leq 100$ is the “real" i, however $x = 0$ does not satisfy it.

$\{\, x+1 \leq 100 \,\}\ x := x+1;\ \{\, x \leq 100 \,\}\ _{AS}$

$((x = y \wedge x \leq 100) \vee x = 0) \wedge x < 100 \ \Rightarrow\ x+1 \leq 100$

$\{\, ((x = y \wedge x \leq 100) \vee x = 0) \wedge x < 100 \,\}\ x := x+1;\ \{\, x \leq 100 \,\}\ _{SP}$

$\{\, x = x \wedge x \leq 100 \,\}\ y = x;\ \{\, x = y \wedge x \leq 100 \,\}\ _{AS}$

$x \leq 100 \ \Rightarrow\ x = x \wedge x \leq 100$

$\{ x \leq 100 \,\}\ y = x;\ \{\, x = y \wedge x \leq 100 \,\}\ _{SP}$

$x = y \wedge x \leq 100 \ \Rightarrow\ (x = y \wedge x \leq 100) \vee x = 0$

$\{ x \leq 100 \,\}\ y = x;\ \{\, (x = y \wedge x \leq 100) \vee x = 0 \,\}\ _{WC}$

$\{\, ((x = y \wedge x \leq 100) \vee x = 0) \wedge x < 100 \,\}\ x := x+1;\ y := x;$

$\quad \{\, (x = y \wedge x \leq 100) \vee x = 0 \,\}\ _{SC}$

$\{\, (x = y \wedge x \leq 100) \vee x = 0 \,\}\ \boldsymbol{while}\ x < 100\ \boldsymbol{do}\ (x := x+1;\ y := x;)$

$\quad \{\, ((x = y \wedge x \leq 100) \vee x = 0) \wedge \neg(x < 100) \,\}\ _{WHT}$

$x = 0 \ \Rightarrow\ (x = y \wedge x \leq 100) \vee x = 0$

$\{\, x = 0 \,\}\ \boldsymbol{while}\ x < 100\ \boldsymbol{do}\ (x := x+1;\ y := x;)$

$\quad \{\, ((x = y \wedge x \leq 100) \vee x = 0) \wedge \neg(x < 100) \,\}\ _{SP}$

$((x = y \wedge x \leq 100) \vee x = 0) \wedge \neg(x < 100) \ \Rightarrow\ x = 100 \wedge y = 100$

$\{\, x = 0 \,\}\ \boldsymbol{while}\ x < 100\ \boldsymbol{do}\ (x := x+1;\ y := x;)\ \{\, x = 100 \wedge y = 100 \,\}\ _{WC}$

3. (a) i. $\{\, \boldsymbol{true} \,\}\ x := x+1;\ \{\, x = x+1 \,\}$

the precondition of the triple "$\boldsymbol{true}$" always holds,

the program as a single assignment command always terminates,

however the postcondition "$x = x+1$" makes no sense logically.

If we try to apply the assignment rule to this triple we may get:

$\{\, x+1 = x+1+1 \,\}\ x := x+1;\ \{\, x = x+1 \,\}$

but there is no such thing as $\boldsymbol{true} \ \Rightarrow\ x+1 = x+2$.

ii. The condition that supports $\{\, \boldsymbol{true} \,\}\ x := e;\ \{\, x = e \,\}$ is $e[e/x] = e$,

that is to say, either x does not occur in e at all, or $e = x$.

We then have $\{\, e = e \,\}\ x := e;\ \{\, x = e \,\}\ _{AS}$

$\boldsymbol{true} \ \Rightarrow\ e = e$

$\{\, \boldsymbol{true} \,\}\ x := e;\ \{\, x = e \,\}\ _{SP}$

(b) $[\, \boldsymbol{true} \,]\ \boldsymbol{while\ true\ do\ skip}\ [\, \boldsymbol{true} \,]$

The triple cannot be proved simply because the program does not terminate.

4. The partial correctness Hoare logic rule for $\boldsymbol{repeat}\ c\ \boldsymbol{until}\ b$ is

$$\frac{\{p\}\,c\,\{i\} \quad \{i \wedge \neg b\}\,c\,\{i\}}{\{p\}\ \boldsymbol{repeat}\ c\ \boldsymbol{until}\ b\ \{i \wedge b\}}$$