Programming Languages: Imperative Program Construction 4. Hoare Logic and Weakest Precondition: Loop

Shin-Cheng Mu

Autumn. 2021

National Taiwan University and Academia Sinica

Loop and loop invariants

- Repetition takes the form **do** $B_0 \rightarrow S_0 \mid ... \mid Bn \rightarrow Sn$ **od**.
- If none of the guards $B_0 \ldots B_n$ evaluate to true, the loop terminates. Otherwise one of the commands is chosen non-deterministically, before the next iteration.

- Repetition takes the form $\textbf{do}\ B_0 \rightarrow S_0 \mid ... \mid Bn \rightarrow Sn\ \textbf{od}$.
- If none of the guards $B_0 \ldots B_n$ evaluate to true, the loop terminates. Otherwise one of the commands is chosen non-deterministically, before the next iteration.
- To annotate a loop (for partial correctness):

$$
\begin{array}{l}
\{P\} \\
\textbf{do}\ B_0 \rightarrow \{P \wedge B_0\}\, S_0\, \{P\} \\
\mid \quad B_1 \rightarrow \{P \wedge B_1\}\, S_1\, \{P\} \\
\textbf{od} \\
\{Q, Pf\} \quad ,
\end{array}
$$

- where Pf refers to a proof of $P \wedge \neg B_0 \wedge \neg B_1 \Rightarrow Q$.

- Repetition takes the form **do** $B_0 \rightarrow S_0 \mid \ldots \mid Bn \rightarrow Sn$ **od**.
- If none of the guards $B_0 \ldots B_n$ evaluate to true, the loop terminates. Otherwise one of the commands is chosen non-deterministically, before the next iteration.
- To annotate a loop (for partial correctness):

$$
\begin{array}{l}
\{P\} \\
\textbf{do } B_0 \rightarrow \{P \wedge B_0\}\, S_0\, \{P\} \\
\mid \quad B_1 \rightarrow \{P \wedge B_1\}\, S_1\, \{P\} \\
\textbf{od} \\
\{Q, Pf\}
\end{array}
\quad ,
$$

- where Pf refers to a proof of $P \wedge \neg B_0 \wedge \neg B_1 \Rightarrow Q$.
- P is called the *loop invariant*. Every loop should be constructed with an invariant in mind!

$\textbf{con}\ N\ \{0 \leqslant N\};\ \textbf{var}\ x, n : Int$

$x, n := 1, 0$

$\textbf{do}\ n \neq N \rightarrow$

$\quad x, n := x + x, n + 1$

$\textbf{od}$

$\{x = 2^N \qquad \}$

con N $\{0 \leqslant N\}$; **var** $x, n : Int$

$x, n := 1, 0$

$\{x = 2^n\}$

do $n \neq N \rightarrow$

$\quad x, n := x + x, n + 1$

od

$\{x = 2^N\}$

con $N\ \{0 \leqslant N\}$; **var** $x, n : Int$

$x, n := 1, 0$

$\{x = 2^n\}$

do $n \neq N \rightarrow$

$\quad x, n := x + x, n + 1$

od

$\{x = 2^N, Pf2\}$

Pf2:

$$x = 2^n \wedge n \leqslant N \wedge \neg(n \neq N)$$
$$\Rightarrow x = 2^N$$

$\textbf{con}\ N\ \{0 \leqslant N\};\ \textbf{var}\ x, n : Int$

$x, n := 1, 0$
$\{x = 2^n\}$
$\textbf{do}\ n \neq N \rightarrow$

$\quad x, n := x + x, n + 1$
$\quad \{x = 2^n, Pf1\}$
$\textbf{od}$
$\{x = 2^N, Pf2\}$

Pf2:

$$x = 2^n \wedge n \leqslant N \wedge \neg(n \neq N) \Rightarrow x = 2^N$$

con N $\{0 \leqslant N\}$; **var** $x, n : Int$

$x, n := 1, 0$
$\{x = 2^n\}$
do $n \neq N \rightarrow$
  $\{x = 2^n \wedge n \neq N\}$
  $x, n := x + x, n + 1$
  $\{x = 2^n, Pf1\}$
od
$\{x = 2^N, Pf2\}$

Pf2:

$$x = 2^n \wedge n \leqslant N \wedge \neg(n \neq N) \Rightarrow x = 2^N$$

$\mathbf{con}\ N\ \{0 \leqslant N\};\ \mathbf{var}\ x, n : Int$

$x, n := 1, 0$
$\{x = 2^n\}$
$\mathbf{do}\ n \neq N \rightarrow$
$\quad \{x = 2^n \wedge n \neq N\}$
$\quad x, n := x + x, n + 1$
$\quad \{x = 2^n, Pf1\}$
$\mathbf{od}$
$\{x = 2^N, Pf2\}$

Pf1:

$$
\begin{aligned}
& (x = 2^n)[x, n \backslash x + x, n + 1] \\
\equiv\ & x + x = 2^{n+1} \\
\Leftarrow\ & x = 2^n \wedge n \neq N
\end{aligned}
$$

Pf2:

$$
\begin{aligned}
& x = 2^n \wedge n \leqslant N \wedge \neg(n \neq N) \\
& \quad \Rightarrow x = 2^N
\end{aligned}
$$

Greatest Common Divisor

- Known: $gcd(x, x) = x$; $gcd(x, y) = gcd(y, x - y)$ if $x > y$.

- Known: $gcd(x,x) = x$; $gcd(x,y) = gcd(y, x-y)$ if $x > y$.
- $$\begin{array}{l}
\textbf{con } A, B : int\ \{0 < A \wedge 0 < B\} \\
\textbf{var } x, y : int \\
\\
x, y := A, B \\
\{0 < x \wedge 0 < y \wedge gcd(x,y) = gcd(A,B)\} \\
\textbf{do } y < x \rightarrow x := x - y \\
\quad \mid\ x < y \rightarrow y := y - x \\
\textbf{od} \\
\{x = gcd(A,B) \wedge y = gcd(A,B)\}
\end{array}$$

- Known: $gcd(x,x) = x$; $gcd(x,y) = gcd(y, x-y)$ if $x > y$.
-

$$
\begin{array}{l}
\textbf{con}\ A, B : int\ \{0 < A \wedge 0 < B\} \\
\textbf{var}\ x, y : int \\
\\
x, y := A, B \\
\{0 < x \wedge 0 < y \wedge gcd(x,y) = gcd(A,B)\} \\
\textbf{do}\ y < x \rightarrow x := x - y \\
\quad \mid\ x < y \rightarrow y := y - x \\
\textbf{od} \\
\{x = gcd(A,B) \wedge y = gcd(A,B)\}
\end{array}
$$

-

$$
\begin{array}{ll}
 & (0 < x \wedge 0 < y \wedge gcd(x,y) = gcd(A,B))[x \backslash x - y] \\
\equiv & 0 < x - y \wedge 0 < y \wedge gcd(x-y, y) = gcd(A,B) \\
\Leftarrow & 0 < x \wedge 0 < y \wedge gcd(x,y) = gcd(A,B) \wedge y < x
\end{array}
$$

- Consider the following program:

$$
\begin{array}{l}
\textbf{var}\ x, y, z : int \\
\{true \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \} \\
\textbf{do}\ x < y \rightarrow x := x + 1 \\
\quad \mid\ y < z \rightarrow y := y + 1 \\
\quad \mid\ z < x \rightarrow z := z + 1 \\
\textbf{od} \\
\{x = y = z\}.
\end{array}
$$

- If it terminates at all, we do have $x = y = z$. But why does it terminate?

- Consider the following program:

$$
\begin{array}{l}
\textbf{var } x, y, z : int \\
\{true, bnd : 3 \times (x \uparrow y \uparrow z) - (x + y + z)\} \\
\textbf{do } x < y \rightarrow x := x + 1 \\
\quad | \; y < z \rightarrow y := y + 1 \\
\quad | \; z < x \rightarrow z := z + 1 \\
\textbf{od} \\
\{x = y = z\}.
\end{array}
$$

- If it terminates at all, we do have $x = y = z$. But why does it terminate?
    1. $bnd \geqslant 0$, and $bnd = 0$ implies none of the guards are true.
    2. $\{x < y \wedge bnd = t\}\, x := x + 1\, \{bnd < t\}$.

To annotate a loop for *total correctness*:

$$
\begin{array}{l}
\{P, bnd : t\} \\
\textbf{do } B_0 \rightarrow \{P \wedge B_0\}\, S_0\, \{P\} \\
\mid \quad B_1 \rightarrow \{P \wedge B_1\}\, S_1\, \{P\} \\
\textbf{od} \\
\{Q\} \quad ,
\end{array}
$$

we have got a list of things to prove:

To annotate a loop for *total correctness*:

$$
\begin{array}{l}
\{P, bnd : t\} \\
\textbf{do}\ B_0 \rightarrow \{P \wedge B_0\}\, S_0\, \{P\} \\
\mid \quad B_1 \rightarrow \{P \wedge B_1\}\, S_1\, \{P\} \\
\textbf{od} \\
\{Q\} \quad ,
\end{array}
$$

we have got a list of things to prove:

1. $P \wedge \neg B_0 \wedge \neg B_1 \Rightarrow Q$,

To annotate a loop for *total correctness*:

$$\begin{array}{l} \{P, bnd : t\} \\ \textbf{do } B_0 \rightarrow \{P \wedge B_0\}\, S_0\, \{P\} \\ \mid \quad B_1 \rightarrow \{P \wedge B_1\}\, S_1\, \{P\} \\ \textbf{od} \\ \{Q\} \quad , \end{array}$$

we have got a list of things to prove:

1. $P \wedge \neg B_0 \wedge \neg B_1 \Rightarrow Q$,
2. for all i, $\{P \wedge B_i\}\, S_i\, \{P\}$,

To annotate a loop for *total correctness*:

$$
\begin{array}{l}
\{P, bnd : t\} \\
\textbf{do } B_0 \rightarrow \{P \wedge B_0\}\, S_0\, \{P\} \\
\mid \quad B_1 \rightarrow \{P \wedge B_1\}\, S_1\, \{P\} \\
\textbf{od} \\
\{Q\} \quad ,
\end{array}
$$

we have got a list of things to prove:

1. $P \wedge \neg B_0 \wedge \neg B_1 \Rightarrow Q$,
2. for all i, $\{P \wedge B_i\}\, S_i\, \{P\}$,
3. $P \wedge (B_0 \vee B_1) \Rightarrow t \geqslant 0$,

To annotate a loop for *total correctness*:

$$
\begin{array}{l}
\{P, bnd : t\} \\
\textbf{do}\ B_0 \rightarrow \{P \wedge B_0\}\ S_0\ \{P\} \\
\ \mid\quad B_1 \rightarrow \{P \wedge B_1\}\ S_1\ \{P\} \\
\textbf{od} \\
\{Q\}\quad ,
\end{array}
$$

we have got a list of things to prove:

1. $P \wedge \neg B_0 \wedge \neg B_1 \Rightarrow Q$,
2. for all i, $\{P \wedge B_i\}\ S_i\ \{P\}$,
3. $P \wedge (B_0 \vee B_1) \Rightarrow t \geqslant 0$,
4. for all i, $\{P \wedge B_i \wedge t = C\}\ S_i\ \{t < C\}$.

- What is the bound function?

$$\textbf{con}\ N\ \{0 \leqslant N\};\ \textbf{var}\ x, n : Int$$

$$x, n := 1, 0$$
$$\{x = 2^n \wedge n \leqslant N \qquad\qquad\qquad \}$$
$$\textbf{do}\ n \neq N \rightarrow$$
$$\quad x, n := x + x, n + 1$$
$$\textbf{od}$$
$$\{x = 2^N\}$$
$$]|$$

- What is the bound function?

$$\textbf{con}\ N\ \{0 \leqslant N\};\ \textbf{var}\ x, n : Int$$

$$x, n := 1, 0$$
$$\{x = 2^n \wedge n \leqslant N, bnd : N - n\}$$
$$\textbf{do}\ n \neq N \rightarrow$$
$$\quad x, n := x + x, n + 1$$
$$\textbf{od}$$
$$\{x = 2^N\}$$
$$]|$$

- $x = 2^n \wedge n \leqslant N \wedge n \neq N \Rightarrow N - n \geqslant 0$,
- $\{\ldots \wedge N - n = t\}\, x, n := x + x, n + 1\, \{N - n < t\}$.

- What is the bound function?

$$
\begin{array}{l}
\textbf{con}\ A, B : Int\ \{0 < A \wedge 0 < B\} \\
\textbf{var}\ x, y : Int \\
\\
x, y := A, B \\
\{0 < x \wedge 0 < y \wedge gcd(x, y) = gcd(A, B) \qquad\qquad\qquad \} \\
\textbf{do}\ y < x \rightarrow x := x - y \\
\quad \mid\ x < y \rightarrow y := y - x \\
\textbf{od} \\
\{x = gcd(A, B) \wedge y = gcd(A, B)\} \\
]|
\end{array}
$$

- What is the bound function?

$$
\begin{array}{l}
\textbf{con } A, B : Int\ \{0 < A \wedge 0 < B\} \\
\textbf{var } x, y : Int \\
\\
x, y := A, B \\
\{0 < x \wedge 0 < y \wedge gcd(x, y) = gcd(A, B), bnd : x + y\} \\
\textbf{do } y < x \rightarrow x := x - y \\
\quad \mid\ x < y \rightarrow y := y - x \\
\textbf{od} \\
\{x = gcd(A, B) \wedge y = gcd(A, B)\} \\
]|
\end{array}
$$

- $\ldots \Rightarrow x + y \geqslant 0$,
- $\{\ldots 0 < y \wedge y < x \wedge x + y = t\}\ x := x - y\ \{x + y < t\}$.

Weakest Precondition

- What about the weakest precondition?
- Denote the program $\mathbf{do}\ B \rightarrow S\ \mathbf{od}$ by DO. It should behave the same as

$$\mathbf{if}\ B \rightarrow S; DO \mid \neg B \rightarrow skip\ \mathbf{fi} \quad .$$

- For any R, if $wp\ DO\ R = X$, it should satisfy

$$X = (B \Rightarrow wp\ S\ X) \wedge (\neg B \Rightarrow R) \quad ,$$

- which is equivalent to

$$X = (B \wedge wp\ S\ X) \vee (\neg B \wedge R) \quad . \text{(Why?)}$$

- We let $wp\ DO\ R$ be the *strongest* X satifying the equation above.

To be slightly more general,

- denote **do** $B_0 \rightarrow S_0 \mid B_1 \rightarrow S_1$ **od** by DO,
- denote **if** $B_0 \rightarrow S_0 \mid B_1 \rightarrow S_1$ **fi** by IF, and
- denote $B_0 \lor B_1$ by BB.
- For all R, $wp\ DO\ R$ is the strongest predicate satisfying

$$X \equiv wp\ IF\ X \lor (R \land \neg\ BB) \quad .$$

- Alternatively, let H_i denote "DO terminates, in at most i iterations, in a state satisfying R."
- $H_0 = R \wedge \neg BB$.
- $H_{n+1} = wp\ IF\ (H_n) \vee (R \wedge \neg BB)$.
- We may define

$$wp\ DO\ R = \langle \exists i : 0 \leqslant i : H_i \rangle \quad .$$

- Theory on *fixed points* shows that the two definitions are equivalent.

- However, how does $wp\ DO\ R$ relate to the way we annotate loops in the previous section?
- We had a theorem about IF which justified the way to annotate branches:

$$\begin{aligned} wp\ IF\ R = {} & (B_0 \Rightarrow wp\ S_0\ R) \\ & \wedge (B_1 \Rightarrow wp\ S_1\ R) \wedge (B_0 \vee B_1) \ . \end{aligned}$$

- Do we have a similar result about loops?

Theorem Let $(D, \leqslant)$ be a partially ordered set; let C be a subset of D such that $(C, <)$ is *well-founded*. Let t be a function on the state with value of type D. Then

$$
\begin{aligned}
&(P \wedge BB \Rightarrow t \in C) \wedge \\
&\langle \forall x :: P \wedge t = x \Rightarrow wp\ IF\ (P \wedge t < x) \rangle \\
&\quad \Rightarrow (P \Rightarrow wp\ DO\ (P \wedge \neg\, BB)) \quad .
\end{aligned}
$$

- Informally, $(C, <)$ being *well-founded* means that there is no infinite chain $c1 > c2 > c3\ldots$ in C.
- The Fundamental Invariance Theorem was proved several times. Proving this theorem motivated developments in many related fields.