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

Shin-Cheng Mu

Autumn. 2021

1 Loop and loop invariants

Loops

- Repetition takes the form **do** $B_0 \to S_0$ | ... | $Bn \to Sn$ **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):

$$\begin{array}{l} \{P\} \\ \mathbf{do}\ B_0 \to \{P \land B_0\}\ S_0\ \{P\} \\ \mid\ \ B_1 \to \{P \land B_1\}\ S_1\ \{P\} \\ \mathbf{od} \\ \{Q, \mathit{Pf}\}\ \ , \end{array}$$

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

Linear-Time Exponentiation

$$\begin{array}{l} \mathbf{con}\ N\ \{0 \leqslant N\};\ \mathbf{var}\ x, n : \mathit{Int} \\ \\ x, n := 1, 0 \\ \{x = 2^n\} \\ \mathbf{do}\ n \neq N \to \\ \quad \{x = 2^n \land n \neq N\} \\ \quad x, n := x + x, n + 1 \\ \quad \{x = 2^n, \mathit{Pf1}\} \\ \mathbf{od} \\ \{x = 2^N, \mathit{Pf2}\} \end{array}$$

Pf1:

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

Pf2:

$$\begin{array}{l} x = 2^n \land n \leqslant N \land \neg(n \neq N) \\ \quad \Rightarrow x = 2^N \end{array}$$

Greatest Common Divisor

- Known: $gcd(x, x) = x$; $gcd(x, y) = gcd(y, x - y)$ if $x > y$.
- $$\begin{array}{l} \mathbf{con}\ A, B : \mathit{int}\ \{0 < A \land 0 < B\} \\ \mathbf{var}\ x, y : \mathit{int} \\ \\ x, y := A, B \\ \{0 < x \land 0 < y \land gcd(x, y) = gcd(A, B)\} \\ \mathbf{do}\ y < x \to x := x - y \\ \ \ \mid\ x < y \to y := y - x \\ \mathbf{od} \\ \{x = gcd(A, B) \land y = gcd(A, B)\} \end{array}$$
- $$\begin{array}{ll} & (0 < x \land 0 < y \land gcd(x, y) = gcd(A, B))[x \backslash x - y] \\ \equiv & 0 < x - y \land 0 < y \land gcd(x - y, y) = gcd(A, B) \\ \Leftarrow & 0 < x \land 0 < y \land gcd(x, y) = gcd(A, B) \land y < x \end{array}$$

A Weird Equilibrium

- Consider the following program:

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

- 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. $\{x < y \wedge bnd = t\}\ x := x + 1\ \{bnd < t\}$.

Repetition

To annotate a loop for *total correctness*:

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

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, $\{P \wedge B_i \wedge t = C\}\ S_i\ \{t < C\}$.

E.g. Linear-Time Exponentiation

- What is the bound function?

$$\begin{array}{l} \mathbf{con}\ N\ \{0 \leqslant N\};\ \mathbf{var}\ x, n : Int \\ \\ x, n := 1, 0 \\ \{x = 2^n \wedge n \leqslant N, bnd : N - n\} \\ \mathbf{do}\ n \neq N \rightarrow \\ \qquad x, n := x + x, n + 1 \\ \mathbf{od} \\ \{x = 2^N\} \\ ]| \end{array}$$

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

E.g. Greatest Common Divisor

- What is the bound function?

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

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

2 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

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

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

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

- which is equivalent to

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

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

Weakest Precondition for Loop

To be slightly more general,

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

$$X \equiv wp\ IF\ X \vee (R \wedge \neg\, BB)\ \ .$$

A Bottom-Up Formulation

- 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

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

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

Relationship to Hoare Logic

- 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?

Fundamental Invariance Theorem

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

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

- 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 [Dij76, Bac81, Boo82, DvG86, Mor89]. Proving this theorem motivated developments in many related fields.

References

[Bac81] R. J. R. Back. Proving total correctness or non-deterministic programs in infinitary logic. *Acta Informatica*, 15:223–249, 1981.

[Boo82] H. J. Boom. A weaker precondition for loops. *ACM Transactions on Programming Languages and Systems*, 4(4):668–677, 1982.

[Dij76] E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall, 1976.

[DvG86] E. W. Dijkstra and A. J. M. van Gasteren. A simple fixpoint argument without the restriction to continuity. *Acta Informatica*, 23(1):1–7, 1986. EWD 901.

[Mor89] J. M. Morris. Well-founded induction and the invariance theorem for loops. *Information Processing Letters*, 32(3):155–158, 1989.