A *very* brief introduction to program verification

Robert L. Baber

for 2F04

2001 November 30

Overview

1. Precondition, postcondition
2. Selected lemmata ("proof rules")
    2.1 assignment statement
    2.2 while loop
3. Correctness proof for a simple program (example)
4. Conclusions

Precondition, postcondition

If the truth of a condition V

before executing a program statement S

guarantees that a condition P is true afterward,

then we say that

V is a *precondition* of the *postcondition* P with respect to the program statement S,

written {V} S {P}.

(ambiguous but can be reformulated mathematically)

Proof rules: assignment statement

A1: $\{P^x_{(E)}\}$ x := E $\{P\}$

(the value of x after = the value of E before,

so P after = $P^x_{(E)}$ before)

A2: If $V \Rightarrow P^x_{(E)}$

then $\{V\}$ x := E $\{P\}$

Use A1 to derive a precondition,

A2, to verify a precondition.

Proof rule: while loop

If

$\{V\}$ init $\{I\}$ and

$\{I \wedge B\}$ S $\{I\}$ and

$I \wedge \neg B \Rightarrow P$

then

$\{V\}$ init; while B do S endwhile $\{P\}$

“I” is the *loop invariant*.

Example of a correctness proposition

$\{n \in Z \wedge 0 \leq n\}$ [V]

i := 1

while i≤n and A(i)≠key do [B]

    i := i+1

invariant $\{n \in Z \wedge i \in Z \wedge 1 \leq i \leq n+1 \wedge_{j=1}^{i-1} A(j) \neq key\}$

endwhile [I]

$\{n \in Z \wedge i \in Z \wedge 1 \leq i \leq n+1 \wedge_{j=1}^{i-1} A(j) \neq key$

$\wedge (i = n+1 \vee A(i) = key)\}$ [P]

Example of a correctness proposition

By the applicable proof rules, the program will be correct (satisfy its specification V, P) if:

1. {V} i:=1 {I}, i.e. if $V \Rightarrow I^{i}{}_{1}$
2. {$I \wedge B$} i:=i+1 {I}, i.e. if $I \wedge B \Rightarrow I^{i}{}_{(i+1)}$
3. $I \wedge \neg B \Rightarrow P$

$1.\ V \Rightarrow I^i_1$?

I^i_1

$=$

$n \in Z \wedge 1 \in Z \wedge 1 \leq 1 \leq n+1 \wedge_{j=1}^{1-1} A(j) \neq key$

$=$

$n \in Z \wedge 0 \leq n$

$=$

V

2. $I \wedge B \Rightarrow I^i_{(i+1)}$?

$I^i_{(i+1)}$

$=$

$n \in Z \wedge i+1 \in Z \wedge 1 \leq i+1 \leq n+1 \wedge_{j=1}^{i+1-1} A(j) \neq key$

$=$

$n \in Z \wedge i \in Z \wedge 0 \leq i \leq n \wedge_{j=1}^{i} A(j) \neq key$

$\Leftarrow$

$n \in Z \wedge i \in Z \wedge 1 \leq i \leq n \wedge_{j=1}^{i-1} A(j) \neq key \wedge A(i) \neq key$

$=$

$I \wedge B$

3. $I \wedge \neg B \Rightarrow P$?

$I \wedge \neg B$

$=$

$n \in Z \wedge i \in Z \wedge 1 \leq i \leq n+1 \wedge_{j=1}^{i-1} A(j) \neq key$

$\wedge\ (i > n \vee A(i) = key)$

$=$

$n \in Z \wedge i \in Z \wedge 1 \leq i \leq n+1 \wedge_{j=1}^{i-1} A(j) \neq key$

$\wedge\ (i = n+1 \vee A(i) = key)$

$=$

P

Conclusions

- Mathematical verification eliminates guesswork (+1?, 0?, -1?, n+1?, n?, n-1?)
- Use proof rules to decompose correctness proposition to be proved (to Boolean $\Rightarrow$s).
- Proving is a mechanistic process; creativity is in design, not verification.

- But most importantly: Use proof rules as design guidelines $\rightarrow$ **program correct by design**

Further information

- 4/6L03 (MRSD) lecture notes at http://www.cas.mcmaster.ca/~baber/Courses/46L03/MRSDLect.pdf
- other literature in 4/6L03 course outline http://www.cas.mcmaster.ca/~baber/Courses/46L03/COut46L03.html