CS571
- Notes 10
- Hoare’s axioms for an imperative language

Axioms and program state
- Only commands alter program state
- Only commands need axioms
- General scheme:

\[
\{p\}C\{q\}
\]

Partial correctness
- If \( p \) is true before \( C \) and if \( C \) terminates, then \( q \) is true after \( C \) finishes
- For a whole program: \( \{p\}P\{q\} \), \([p,q]\) is the specification for \( P \)
- Can we prove that \([p,q]\) is the correct specification for \( P \)? Yes, if it is true.

Weak and strong
- Pre-conditions should be as weak (general) as possible – hopefully even just true, i.e. no constraints at all on program state
- Post-conditions should be as strong (specific) as possible – we would like to prove real results for programs – even though we could prove more general results
The axioms

- \{p\}nop\{p\} nop does nothing
- \{p[E\|]\} := E\{p\} for assignment
  - This is counter-intuitive. Adding an assertion after the assignment will not work. e.g. \{x>4\}x := 2\{x>4 \land x=2\} is false from true
- However:
  - \{x>0[2\mid x]\} x := 2\{x>0\} is correct since 2>0

Control structures: sequence

- Sequence: \{p\}C_1\{r\} \{r\}C_2\{q\} \{p\}C_1;C_2\{q\}
- e.g. x := 2; y := x plus 1
  - Post-condition is \{y > x\}
  - 'push this back' through the second command:
    - \{x+1>y\} y := x plus 1 \{y>x\}
  - And again through the first one:
    - \{2+1>x\} x := 2 \{x+1>x\}
  - This is true, so \{true\} x := 2; y := x plus 1 \{y>x\} using the sequence axiom

Control structures: conditional

- Axiom: \{p \land b\} C_1\{q\} \{p \land \lnot b\} C_2\{q\} \{p\} if B then C_1 else C_2 end\{q\}
- e.g. if x less 0 then x := 1 else x := 2 end
- We can prove x>0 by proving each branch separately:
  - \{1>0\} x := 1\{x>0\} and \{2>0\} x := 2\{x>0\}
  - The test b is irrelevant since the pre-condition is already as weak as possible, so:
  - \{true\} if x less 0 then x := 1 else x := 2 end \{x>0\}

Conditional : example

- More interesting:
  - \{true\} if x less 0 then y := x plus 1 else y := neg x end\{y <= 0\}
    - \{x+1<=0\} y := x plus 1 \{y<=0\} and
    - \{-x<=0\} y := neg x \{y<=0\}
  - Need to show that \{x+1<=0\} is the same as \{x<0\} and that \{-x<=0\} is the same as \{-x<0\}
  - x+1<=0 is x<=-1 which is x<0 and
  - -x<=0 is x>=0 which is -(x<0)
  - So the goal is true
Axiom: the while loop

- Need to express the repetition of the body, and the exit condition:
- There must be an invariant assertion that is true for each repetition — the exit test will be false after the loop finishes:

\[
\{ p \land b \} C \{ p \} \\
\{ p \} \text{while} B \text{do} C \text{end} \{ p \land \neg b \}
\]

Loop example

- while x greater 0 do x := x minus 1 end
- Invariant is x>=0
- \{x-1>=0\} x := x minus 1 \{x>=0\}
- x-1>=0 is x>=1 which is x>0, so we need to show that x>=0 \land x>0 is the same as x>0
- They are identical in every case (for every value of x) so the loop is proved:
- \{x>=0\} while x greater 0 do x := x minus 1 end \{x>=0 \land \neg(x>0)\}
- The post-condition is x>=0 \land x<=0 which clearly means that x=0

Loop second example

- while y greater x do y := y minus 1; x := x plus 1 end
- Post-condition: y=x \lor y=x-1
- Invariant: y>x-2
- Loop body:
  \{(y-1)>(x+1)-2\} y := y minus 1; x := x plus 1 \{y>x-2\}
- Pre-condition is y>x
- But this is implied by y>x-2 \land y>x, so the invariant is proved
- So:
  \{y > x-2\} while y greater x do y := y minus 1; x := x plus 1 \{y > x-2 \land \neg(y > x)\}
- Then show that post-condition is same is y=x \lor y=x-1 (hard)

Strengthening and weakening

- Strengthen the pre-condition: \( p \Rightarrow r \) \{r\} C \{q\}
  \[
  \{ p \} C \{ q \}
  \]
- Weaken the post-condition: \( p \} C \{ r \} \Rightarrow q \)
  \[
  \{ p \} C \{ q \}
  \]
- Instead of proving e.g. \( x \geq 0 \land x \leq 0 \Rightarrow (x = 0) \)
- Only need to prove the implication (might be easier):
  \( x \geq 0 \land x \leq 0 \Rightarrow (x = 0) \)