1 Predicate Logic & Correctness

- Propositional operators
  - not ($\neg$), and ($\wedge$), or ($\vee$)
  - implication ($\Rightarrow$) – $P \Rightarrow Q \equiv \neg P \vee Q$
  - equivalence ($\Leftrightarrow$) – $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$
- Operator precedence (tightest-binding first): $\neg$, $\wedge$, $\vee$, $\Rightarrow$
- Quantifiers
  - for all ($\forall$)
  - there exists ($\exists$)
  - Example: $\forall\, x \in \mathbb{N} \cdot x \geq 0$
  - The type of the bound variable may be implicit, e.g. $\forall\, x \cdot x \geq 0$
- Entailment
  - If $P \Rightarrow Q$ is a tautology (always true) then P is stronger than Q
  - Equivalently: P entails Q ($P \Rrightarrow Q$)
- Substitution
  - $P[x \backslash a]$ – Substitute all occurrences of x by a in P
  - $P[x, y \backslash a, b]$ – Substitute a and b for x and y **simultaneously**
- Hoare triples – $\{P\}\, S\, \{Q\}$
  - P is the precondition, S is the program and Q is the postcondition
  - If P is true before S executes, then S will terminate and Q will be true when it does
  - The program must terminate if started in States_P (total correctness)
- Weakest preconditions
  - For a program S and postcondition Q, $wp(S, Q)$ is the unique weakest possible precondition such that the triple $\{P\}\, S\, \{Q\}$ will be true
  - $\forall\, P \cdot (\{P\}\, S\, \{Q\}) \Rightarrow (P \Rrightarrow wp(S, Q))$

2 Guarded Command Language

- **skip** – Empty Command
  - $wp(\textbf{skip}, Q) \equiv Q$
  - Hence $\{Q\}\,\textbf{skip}\,\{Q\} \equiv$ true for any Q
  - $\{P\}\,\textbf{skip}\,\{Q\}$ is false if and only if P is strictly weaker than Q
- **abort** – Chaotic Command
  - $wp(\textbf{abort}, Q) \equiv$ false
  - No precondition can guarantee a postcondition
  - Represents 'chaotic/undefined behaviour'
- $:=$ – Assignment
  - $wp(x := E, Q) \equiv Q[x\backslash E]$
  - So $\{Q[x\backslash E]\}\, x := E\, \{Q\}$ is true
  - Generalises to multiple assignment: $wp((x, y := E, F), Q) \equiv Q[x, y\backslash E, F]$
- ; – Composition/Concatenation
  - $wp(S_1;\ S_2, Q) \equiv wp(S_1, wp(S_2, Q))$
  - There exists some 'middle' predicate true after S_1 and before S_2
  - If $\{P\}\, S_1\, \{M\} \wedge \{M\}\, S_2\, \{Q\}$ then $\{P\}\, S_1;\ S_2\, \{Q\}$
- **if** – Selection
  - **if** $G_1 \rightarrow S_1$
$[]$ $G_2 \rightarrow S_2$
...
$[]$ $G_n \rightarrow S_n$
fi
  - Evaluate all guards $G_1 \dots G_n$, choose a true guard G_i nondeterministically, and execute S_i
  - if all guards evaluate to false then **abort** is executed
  - $wp(\textbf{if}, Q) \equiv \bigvee_{i=1}^{n} G_i \wedge \bigwedge_{i=1}^{n} (G_i \Rightarrow wp(S_i), Q)$
  - The disjunction of the guards *must* be true
- **do** – Repetition
  - **do** $G_1 \rightarrow S_1$
$[]$ $G_2 \rightarrow S_2$
...
$[]$ $G_n \rightarrow S_n$
od
  - Evaluate all guards, choose a true guard nondeterministically, execute S_i and repeat
  - If all G_i are false, the loop ends and the program continues
  - Weakest precondition rule is complex, so *loop invariants* are used instead
  - Predicate I is a loop invariant if $\{I \wedge G_i\}\, S_i\, \{I\}$ for all $1 \leq i \leq n$
  - There are usually many possible invariants for a loop

3 Refinement & Verification

4 Derivation

5 Procedures

6 Recursion

7 Modules