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 $G_1 \ldots G_n$, 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

- Specification statement: $w : [P, Q]$
- P is the precondition, Q is the postcondition, w is the 'frame' of variables that may be modified
- A program C satisfies $w : [P, Q]$ if and only if
  - $\{P\}\, C\, \{Q\}$
  - C only changes variables in w
- If P is not true when C is executed it may do anything, and it need not terminate
- Mixing specification statements with GCL forms a 'wide-spectrum language'
- Refinement ($\sqsubseteq$) – a partial ordering on programs (similar to $\leq$ for reals)
  - $S \sqsubseteq S'$ means a user expecting program S would be satisfied with S'
  - $S \sqsubseteq S' \Leftrightarrow \forall\, Q \cdot wp(S, Q) \Rrightarrow wp(S', Q)$
  - For a specification: $wp(x : [P, Q], Q') \mathrel{\widehat{=}} P \wedge (\forall\, x \cdot Q \Rightarrow Q')[v_0 \backslash v]$
- General approach to refining a program
  - Start with a specification $S = w : [P, Q]$
  - Use rules to replace S with S' mixing specifications with GCL
  - Each rule must preserve correctness – i.e. every program C that satisfies S' must satisfy S
  - Eventually arrive at a pure GCL program C such that $\{P\}\, C\, \{Q\} \equiv$ true

3.1 Refinement Rules

- Rule 1: **Strengthen Postcondition**
  - If $P[w\backslash w_0] \wedge Q' \Rrightarrow Q$ then $w : [P, Q] \sqsubseteq w : [P, Q']$ ($P[w\backslash w_0]$ usually not needed)
- Rule 2: **Weaken Precondition**
  - If $P \Rrightarrow P'$ then $w : [P, Q] \sqsubseteq w : [P', Q]$
- Rule 3: **Skip**
  - If $P \Rrightarrow Q$ then $w : [P, Q] \sqsubseteq$ **skip**
- Rule 4: **Assignment**
  - If $P \Rrightarrow Q[x\backslash E]$ then $x : [P, Q] \sqsubseteq x := E$
- Rule 5: **Composition**
  - $w : [P, Q] \sqsubseteq w : [P, M];\ w : [M, Q]$ (no side condition)
- Rule 6: **Following Assignment** (combined assignment and composition)
  - $w, x : [P, Q] \sqsubseteq w, x : [P, Q[x\backslash E]];\ x := E$
- Rule 7: **Selection**
  - If $P \Rrightarrow \bigvee_{i=1}^{n} G_i$ then $w : [P, Q] \sqsubseteq$
if $G_1 \rightarrow w : [G_1 \wedge P, Q]$
$\dots$
$[\!]\ G_n \rightarrow w : [G_n \wedge P, Q]$
fi
- Rule 8: **Repetition**
  - For repetition, a loop invariant I and loop variant (an integer expression) V are required
    - Let V_0 be the value of V at the start of each iteration
    - Then $0 \leq V < V_0$ is true at the end of each iteration
    - (i.e. V is *strictly decreasing* on every iteration, and won't be negative before loop termination)
  - To apply the repetition rule
    - Strengthen postcondition to $I \wedge \neg G$ (side condition: $I \wedge \neg G \Rrightarrow Q$)
    - Use composition to perform $w : [P, I \wedge \neg G] \sqsubseteq w : [P, I];\ w : [I, I \wedge \neg G]$
    - Refine the first half into initialisation (e.g. an assignment)
    - Refine the second half using the repetition rule (no side conditions!)
  - Rule: Let $G \mathrel{\widehat{=}} \bigvee_{i=1}^{n} G_i$, then $w : [I, I \wedge \neg G] \sqsubseteq$
do $G_1 \rightarrow w : [I \wedge G_1, I \wedge (0 \leq V < V_0)]$
$\dots$
$[\!]\ G_n \rightarrow w : [I \wedge G_n, I \wedge (0 \leq V < V_0)]$
od
- Rule 9: **Contract frame**
  - $w, x : [P, Q] \sqsubseteq w : [P, Q[x_0\backslash x]]$
- Rule 10: **Remove invariant**
  - If w does not occur in I then $w : [P \wedge I, Q \wedge I] \sqsubseteq w : [P, Q]$

4 Derivation

5 Procedures

6 Recursion

7 Modules