COMP2111 Week 7
Term 1, 2019
Week 6 recap
Week 6 recap

Hoare Logic:
- Soundness proof
- Finding a derivation
  - Weakest precondition
  - Invariants
- Total correctness (termination) and variants
- Operational semantics
- $L^+$: $L$ with non-determinism
- Refinement calculus
Soundness and (relative) completeness

**Theorem (Soundness)**

Every derivable Hoare triple is valid: If $\vdash \{ \varphi \} P \{ \psi \}$ then $\models \{ \varphi \} P \{ \psi \}$.

**Theorem (Relative completeness)**

Given an oracle that can determine the truth of predicates, every valid Hoare triple is derivable: If $\models \{ \varphi \} P \{ \psi \}$ then $\vdash \{ \varphi \} P \{ \psi \}$. 
Finding a proof: weakest precondition

Given a program $P$ and a postcondition $\psi$, the weakest precondition $wp(P, \psi)$ is the predicate $\varphi$ such that

- $\{\varphi\} P \{\psi\}$ is valid
- If $\{\varphi'\} P \{\psi\}$ is valid then $\varphi' \rightarrow \varphi$.

Computable based on the structure of $P$. Difficulty with loops...
Finding a proof: Invariants

In order to establish the validity of

\[ \{ \varphi \} \textbf{ while } b \textbf{ do } P \textbf{ od } \{ \psi \} \]

we find an \textbf{Invariant}, Inv, such that:

- \( \varphi \rightarrow \text{Inv} \) \hspace{1cm} (establish)
- \( \{ b \land \text{Inv} \} P \{ \text{Inv} \} \) \hspace{1cm} (maintain)
- \( \neg b \land \text{Inv} \rightarrow \psi \) \hspace{1cm} (conclude)
Total correctness (termination)

\[ [\varphi] \ P \ [\psi] \]

Represents the statement that with precondition \( \varphi \), program \( P \) will terminate at a state that satisfies \( \psi \).

Can derive validity of \([\varphi] \ P \ [\psi]\) using Hoare logic with modified loop command:

\[
[\varphi \land g \land (v = N)] \ P [\varphi \land (v < N)] \quad (\varphi \land g) \rightarrow (v > 0)
\]

\[ [\varphi] \ \text{while} \ g \ \text{do} \ P \ \text{od} [\varphi \land \neg g] \]
Finding a (total correctness) proof: Variants

In order to establish the validity of

$$[\varphi] \textbf{while} \ b \ \textbf{do} \ P \ \textbf{od} \ [\psi]$$

we find an \textbf{Invariant}, Inv, such that:

- \(\varphi \rightarrow \text{Inv}\) *(establish)*
- \([b \land \text{Inv}] \ P \ [\text{Inv}]\) *(maintain)*
- \(\neg b \land \text{Inv} \rightarrow \psi\) *(conclude)*

and a \textbf{variant}, Var \(\in \text{Exp}\), such that:

- \((b \land \text{Inv}) \rightarrow (\text{Var} > 0)\) *(positivity)*
- \([\text{Inv} \land b \land (\text{Var} = N)] \ P \ [\text{Inv} \land (v < N)]\) *(progress)*
Operational semantics

- **Denotational semantics**: Assign a mathematical object (relation between states) to Programs

- **Operational semantics**: Construct (inductively) a relation, \( \Downarrow \), between Programs and pairs of states

Example rule:

\[
[P, \eta] \Downarrow \eta' \quad [Q, \eta'] \Downarrow \eta'' \\
\hline
[P; Q, \eta] \Downarrow \eta''
\]
Non-determinism

Non-determinism = unspecified program branching

- More powerful: Encompasses deterministic behaviour
- More abstract: Mathematically nicer

$L^+$: Non-deterministic extension of $L$

- $P + Q$ – non-deterministic choice between $P$ and $Q$
- $P^*$ – loop for a non-deterministic number of times
Refinement calculus

Process for transforming abstract specifications into concrete code.

- Start with the most abstract program relating pre- and post-conditions
- Use refinement rules (based on rules of Hoare Logic) to refine the program – i.e. make it less abstract
- The end result will be some program in $\mathcal{L}$ (or similar)
Nothing will be assessed in great detail, however, a good understanding of:

- Weak precondition
- Invariants
- Termination and variants
- Non-determinism

will help a lot.