COMP2111 Week 6
Term 1, 2019
Week 5 recap
Week 5 recap

Hoare Logic:

- Simple imperative language $\mathcal{L}$
- Hoare triple $\{ \varphi \} P \{ \psi \}$ (SYNTAX)
- Derivation rules (PROOFS)
- Semantics for Hoare logic (SEMANTICS)
The language $\mathcal{L}$ is a simple imperative programming language made up of four statements:

**Assignment:** $x := e$

where $x$ is a variable and $e$ is an arithmetic expression.

**Sequencing:** $P;Q$

**Conditional:** if $b$ then $P$ else $Q$ fi

where $b$ is a boolean expression.

**While:** while $b$ do $P$ od
Hoare triple (Syntax)

\{\varphi\} \; P \; \{\psi\}

Intuition:
If $\varphi$ holds in a state of some computational model
then $\psi$ holds in the state reached after a successful execution of $P$.

$\vdash \{\varphi\} \; P \; \{\psi\}$

$\{\varphi\} \; P \; \{\psi\}$ is derivable using the proof rules of Hoare Logic

$\models \{\varphi\} \; P \; \{\psi\}$

$\{\varphi\} \; P \; \{\psi\}$ is valid according to the semantic interpretation.
Hoare triple (Syntax)

\[ \{ \varphi \} \ P \ {\psi} \]

Intuition:
If \( \varphi \) holds in a state of some computational model
then \( \psi \) holds in the state reached after a successful execution of \( P \).

\( \vdash \{ \varphi \} \ P \ {\psi} \)

\( \{ \varphi \} \ P \ {\psi} \) is **derivable** using the proof rules of Hoare Logic

\( \vdash \{ \varphi \} \ P \ {\psi} \)

\( \{ \varphi \} \ P \ {\psi} \) is **valid** according to the semantic interpretation.
Hoare triple (Syntax)

\[
\{\varphi\} \quad P \quad \{\psi\}
\]

Intuition:
If \(\varphi\) holds in a state of some computational model then \(\psi\) holds in the state reached after a successful execution of \(P\).

\[\vdash \{\varphi\} \quad P \quad \{\psi\}\]

\(\{\varphi\} \quad P \quad \{\psi\}\) is **derivable** using the proof rules of Hoare Logic

\[\models \{\varphi\} \quad P \quad \{\psi\}\]

\(\{\varphi\} \quad P \quad \{\psi\}\) is **valid** according to the semantic interpretation.
Hoare logic rules

\[ \{ \varphi[e/x] \} x := e \{ \varphi \} \]  

(ass)

\[ \{ \varphi \} P \{ \psi \} \quad \{ \psi \} Q \{ \rho \} \]  

(seq)

\[ \{ \varphi \} \text{if } g \text{ then } P \text{ else } Q \text{ fi } \{ \psi \} \]  

(if)
Hoare logic rules

(ass)

\[
\{ \varphi(e) \} x := e \{ \varphi(x) \}
\]

(seq)

\[
\begin{array}{c}
\{ \varphi \} P \{ \psi \} \\
\{ \psi \} Q \{ \rho \}
\end{array}
\Rightarrow
\{ \varphi \} P; Q \{ \rho \}
\]

(if)

\[
\begin{array}{c}
\{ \varphi \land g \} P \{ \psi \} \\
\{ \varphi \land \neg g \} Q \{ \psi \}
\end{array}
\Rightarrow
\{ \varphi \} \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{ \psi \}
\]
Hoare logic rules

\[
\frac{\{\varphi \land g\} \; P \; \{\varphi\}}{\{\varphi\} \\text{while } g \text{ do } P \text{ od } \{\varphi \land \neg g\}} \quad \text{(loop)}
\]

\[
\frac{\varphi' \rightarrow \varphi \quad \{\varphi\} \; P \; \{\psi\} \quad \psi \rightarrow \psi'}{\{\varphi'\} \; P \; \{\psi'\}} \quad \text{(cons)}
\]
**Hoare logic semantics**

\textbf{Env}: set of environments (functions that map variables to numeric values)

\[ \langle \cdot \rangle : \text{Predicates} \rightarrow \text{Pow(Env)}, \text{given by:} \]

\[ \langle \varphi \rangle := \{ \eta \in \text{Env} : [\varphi]^\eta = \text{true} \}. \]

\[ [\cdot] : \text{Programs} \cup \text{Predicates} \rightarrow \text{Pow(Env} \times \text{Env}) \]
**Hoare logic semantics**

\( \text{Env} \): set of environments (functions that map variables to numeric values)

\( \langle \cdot \rangle : \text{Predicates} \rightarrow \text{Pow}(\text{Env}) \), given by:

\[
\langle \varphi \rangle := \{ \eta \in \text{Env} : [\varphi]^{\eta} = \text{true} \}.
\]

\( [\cdot] : \text{Programs} \cup \text{Predicates} \rightarrow \text{Pow}(\text{Env} \times \text{Env}) \)
Hoare logic semantics

\( \text{Env} \): set of environments (functions that map variables to numeric values)

\( \langle \cdot \rangle : \text{Predicates} \rightarrow \text{Pow}(\text{Env}) \), given by:

\[ \langle \varphi \rangle := \{ \eta \in \text{Env} : [\varphi]^\eta = \text{true} \} \].

\( [\cdot] : \text{Programs} \cup \text{Predicates} \rightarrow \text{Pow}(\text{Env} \times \text{Env}) \)
Hoare logic semantics

\[\boxed{\cdot} : \text{Programs} \cup \text{Predicates} \rightarrow \text{Pow}(\text{Env} \times \text{Env})\]

For predicates: 
\[\boxed{\varphi} = \{(\eta, \eta) : \eta \in \langle \varphi \rangle\}\]

For programs: Inductively:

- \[\boxed{P; Q} = \boxed{P}; \boxed{Q}\]
- \[\boxed{\text{if } b \text{ then } P \text{ else } Q \text{ fi}} = \boxed{b; P} \cup \boxed{\neg b; Q}\]
- \[\boxed{\text{while } b \text{ do } P \text{ od}} = \boxed{b; P}^*; \boxed{\neg b}\]

where \(R \circ S\) is the relational composition of \(R\) and \(S\), and \(R^*\) is the transitive closure of \(R\).
Hoare logic semantics

\[ \llbracket \cdot \rrbracket : \text{Programs} \cup \text{Predicates} \to \text{Pow}(\text{Env} \times \text{Env}) \]

For predicates: \( \llbracket \varphi \rrbracket = \{ (\eta, \eta) : \eta \in \langle \varphi \rangle \} \)

For programs: Inductively:

- \( \llbracket P; Q \rrbracket = \llbracket P \rrbracket ; \llbracket Q \rrbracket \)
- \( \llbracket \text{if } b \text{ then } P \text{ else } Q \text{ fi} \rrbracket = \llbracket b \rrbracket ; \llbracket P \rrbracket \cup \llbracket \neg b \rrbracket ; Q \rrbracket \)
- \( \llbracket \text{while } b \text{ do } P \text{ od} \rrbracket = \llbracket b \rrbracket ; \llbracket P \rrbracket^* ; \llbracket \neg b \rrbracket \)

where \( R; S \) is the relational composition of \( R \) and \( S \), and \( R^* \) is the transitive closure of \( R \)...
Hoare logic semantics

$[\cdot]: \text{Programs} \cup \text{Predicates} \rightarrow \text{Pow(Env} \times \text{Env})$

For predicates: $[\varphi] = \{(\eta, \eta): \eta \in \langle \varphi \rangle\}$

For programs: Inductively:

- $[P; Q] = [P]; [Q]$
- $[\text{if } b \text{ then } P \text{ else } Q \text{ fi}] = [b; P] \cup [\neg b; Q]$
- $[\text{while } b \text{ do } P \text{ od}] = [b; P]^*; [\neg b]$

where $R; S$ is the relational composition of $R$ and $S$, and $R^*$ is the transitive closure of $R$...
Hoare logic semantics

\[ [\cdot] : \text{Programs} \cup \text{Predicates} \to \text{Pow}(\text{Env} \times \text{Env}) \]

For predicates: \[ [\varphi] = \{ (\eta, \eta) : \eta \in \langle \varphi \rangle \} \]

For programs: Inductively:

- \[ [P; Q] = [P]; [Q] \]
- \[ [\text{if } b \text{ then } P \text{ else } Q \text{ fi}] = [b; P] \cup [\neg b; Q] \]
- \[ [\text{while } b \text{ do } P \text{ od}] = [b; P]^*; [\neg b] \]

where \( R; S \) is the relational composition of \( R \) and \( S \), and \( R^* \) is the transitive closure of \( R \)...
Hoare logic semantics

\[ [\cdot] : \text{Programs} \cup \text{Predicates} \rightarrow \text{Pow}(\text{Env} \times \text{Env}) \]

For predicates: \( [\varphi] = \{(\eta, \eta) : \eta \in \langle \varphi \rangle \} \)

For programs: Inductively:

- \( [P; Q] = [P]; [Q] \)
- \( [\text{if } b \text{ then } P \text{ else } Q \text{ fi}] = [b; P] \cup [\neg b; Q] \)
- \( [\text{while } b \text{ do } P \text{ od}] = [b; P]^*; [\neg b] \)

where \( R; S \) is the relational composition of \( R \) and \( S \), and \( R^* \) is the transitive closure of \( R \)...
Transitive closure

Given a binary relation $R \subseteq A \times A$ we define $R^n$ inductively:

1. $R^0 = \Delta$ the diagonal relation
2. $R^{i+1} = R^i \cup R$ for $i \geq 0$.

The **transitive closure**, $R^*$ is then defined to be:

$$R^* := \bigcup_{i=0}^{\infty} R^i$$

$$= \{(x, y) : (x, y) \in R^i \text{ for some } i \in \mathbb{N}\}.$$
Need to know for this course

- Write programs in $\mathcal{L}$.
- Give proofs using the Hoare logic rules (full and outline)
- Definition of $[\cdot]$ 
- Definition of composition and transitive closure