Semantics of commands

- **Assignments**

  $\texttt{Csem}\ (V\,\texttt{:=}E)\ s_1\ s_2\ \ =\ \ (s_2 = s_1\texttt{[}(\texttt{Esem}\ E\ s_1)\texttt{/}V\texttt{]})$

- **Sequences**

  $\texttt{Csem}\ (C_1\texttt{;}C_2)\ s_1\ s_2\ \ =\ \ \exists s.\ \texttt{Csem}\ C_1\ s_1\ s\ \wedge\ \texttt{Csem}\ C_2\ s\ s_2$

- **Conditional**

  $\texttt{Csem}\ (\texttt{IF}\,S\,\texttt{THEN}\,C_1\,\texttt{ELSE}\,C_2)\ s_1\ s_2$
  $= (\texttt{Ssem}\ S\ s_1\ \wedge\ \texttt{Csem}\ C_1\ s_1\ s_2)\ \vee\ (\neg\texttt{Ssem}\ S\ s_1\ \wedge\ \texttt{Csem}\ C_2\ s_1\ s_2)$
  $= \textbf{\textit{if}}\ \texttt{Ssem}\ S\ s_1\ \textbf{\textit{then}}\ \texttt{Csem}\ C_1\ s_1\ s_2\ \textbf{\textit{else}}\ \texttt{Csem}\ C_2\ s_1\ s_2$

- **While-commands**

  $\texttt{Csem}\ (\texttt{WHILE}\,S\,\texttt{DO}\,C)\ s_1\ s_2\ \ =\ \ \exists n.\ \texttt{Iter}\ n\ (\texttt{Ssem}\ S)\ (\texttt{Csem}\ C)\ s_1\ s_2$

  where the function `Iter` is defined by recursion on n as follows:

  $\texttt{Iter}\ 0\ p\ c\ s_1\ s_2 \qquad\quad = \neg(p\ s_1) \wedge (s_1{=}s_2)$
  $\texttt{Iter}\ (n{+}1)\ p\ c\ s_1\ s_2 = p\ s_1 \wedge \exists s.\ c\ s_1\ s \wedge \texttt{Iter}\ n\ p\ c\ s\ s_2$

  - **argument n of `Iter` is the number of iterations**
  - **argument p is a predicate on states (e.g. `Ssem` S)**
  - **argument c is a semantic function (e.g. `Csem` C)**
  - **arguments s_1 and s_2 are the initial and final states, respectively**

Soundness of Hoare Logic: summary

- **Assignment axiom:**
  $\forall s_1\ s_2.\ \texttt{Ssem}\ (Q\texttt{[}E/V\texttt{]})\ s_1 \wedge \texttt{Csem}\ (V\texttt{:=}E)\ s_1\ s_2 \Rightarrow \texttt{Ssem}\ Q\ s_2$
  $\models \{Q\texttt{[}E/V\texttt{]}\}V\texttt{:=}E\{Q\}$
- **Precondition strengthening:**
  $(\forall s.\ \texttt{Ssem}\ P\ s \Rightarrow \texttt{Ssem}\ P'\ s) \wedge \texttt{Hsem}\ P'\ C\ Q \Rightarrow \texttt{Hsem}\ P\ C\ Q$
  $(\models P \Rightarrow P')\ \wedge\ \models \{P'\}C\{Q\}\ \Rightarrow\ \models \{P\}C\{Q\}$
- **Postcondition weakening:**
  $\texttt{Hsem}\ P\ C\ Q' \wedge (\forall s.\ \texttt{Ssem}\ Q'\ s \Rightarrow \texttt{Ssem}\ Q\ s) \Rightarrow \texttt{Hsem}\ P\ C\ Q$
  $\models \{P\}C\{Q'\}\ \wedge (\models Q' \Rightarrow Q)\ \Rightarrow\ \models \{P\}C\{Q\}$
- **Sequencing rule:**
  $\texttt{Hsem}\ P\ C_1\ Q \wedge \texttt{Hsem}\ Q\ C_2\ R \Rightarrow \texttt{Hsem}\ P\ (C_1\texttt{;}C_2)\ R$
  $\models \{P\}C_1\{Q\}\ \wedge\ \models \{Q\}C_2\{R\} \Rightarrow\ \models\ \{P\}C_1\texttt{;}C_2\{R\}$
- **Conditional rule:**
  $\texttt{Hsem}\ (P \wedge S)\ C_1\ Q \wedge \texttt{Hsem}\ (P \wedge \neg Q)\ C_2\ Q \Rightarrow \texttt{Hsem}\ P\ (\texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2)\ Q$
  $\models \{P \wedge S\}C_1\{Q\}\ \wedge\ \models \{P \wedge \neg S\}C_2\{Q\}\ \Rightarrow\ \models \{P\}\texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2\{Q\}$
- **WHILE rule:**
  $\texttt{Hsem}\ (P \wedge S)\ C\ P \Rightarrow \texttt{Hsem}\ P\ (\texttt{WHILE}\ S\ \texttt{DO}\ C)\ (P \wedge \neg S))$
  $\models \{P \wedge S\}C\{P\}\ \Rightarrow\ \models \{P\}\texttt{WHILE}\ S\ \texttt{DO}\ C$

Completeness and decidability of Hoare Logic

- **Soundness:** $\vdash \{P\}C\{Q\} \;\Rightarrow\; \models \{P\}C\{Q\}$

- **Decidability:** $\{\mathtt{T}\}C\{\mathtt{F}\} \;\Leftrightarrow\; C$ <u>**doesn't**</u> **halt**
  - **the Halting Problem is undecidable**

- **Completeness: really want** $\models_{\mathcal{I}_{\mathbf{PA}}} \{P\}C\{Q\} \;\Rightarrow\; \mathbf{PA} \vdash \{P\}C\{Q\}$
  - **to show this not possible, first observe that for any** P
  - $\models_{\mathcal{I}_{\mathbf{PA}}} \{\mathtt{T}\}\mathtt{X:=X}\{P\}$ **if and only if** $\models_{\mathcal{I}_{\mathbf{PA}}} P$
  - $\mathbf{PA} \vdash \{\mathtt{T}\}\mathtt{X:=X}\{P\}$ **if and only if** $\mathbf{PA} \vdash P$

- **If Hoare logic were complete, then taking** P **above to be** G_T**:**
  $\models_{\mathcal{I}_{\mathbf{PA}}} G_T \;\Rightarrow\; \models_{\mathcal{I}_{\mathbf{PA}}} \{\mathtt{T}\}\mathtt{X:=X}\{G_T\} \;\Rightarrow\; \mathbf{PA} \vdash \{\mathtt{T}\}\mathtt{X:=X}\{G_T\} \;\Rightarrow\; \mathbf{PA} \vdash G_T$
  contradicting Gödel's theorem

- **Must separate completeness of programming and specification logics**

Relative completeness (Cook 1978) – basic idea ✓

- $\models_{\mathcal{I}_{\mathbf{PA}}} \{P\}C\{Q\}$ **entails** $\Gamma_{\mathbf{PA}} \vdash \{P\}C\{Q\}$**, where** $\Gamma_{\mathbf{PA}} = \{S \mid \models_{\mathcal{I}_{\mathbf{PA}}} S\}$

- **Proof outline:**

  - **define** $\texttt{wlp}(C, Q)$ **in** $\mathcal{L}_{\mathbf{PA}}$

  - **straight line code easy - see earlier slides**

  - $\texttt{wlp}((\texttt{WHILE}\,S\,\texttt{DO}\,C), Q)$ **needs tricky encoding using Gödel's** β **function**
    (see Winskel's book *The formal semantics of programming languages: an introduction***)**

  - $\models_{\mathcal{I}_{\mathbf{PA}}} \{P\}C\{Q\}$ **implies** $\models_{\mathcal{I}_{\mathbf{PA}}} P \Rightarrow \texttt{wlp}(C, Q)$ **by induction on** C **and semantics**

  - $\Gamma_{\mathbf{PA}} \vdash \{\texttt{wlp}(C, Q)\}C\{Q\}$ **by induction on** C **and Hoare logic**

  - **hence** $\models_{\mathcal{I}_{\mathbf{PA}}} \{P\}C\{Q\}$ **implies** $\Gamma_{\mathbf{PA}} \vdash \{P\}C\{Q\}$ **by precondition strengthening**

- **Cook's theorem is for any** *expressive* **assertion language**

  - **i.e. any language in which** $\texttt{wlp}(C, Q)$ **is definable**

Summary: soundness, decidability, completeness ✓

- **Hoare logic is sound**
- **Hoare logic is undecidable**
  - **deciding $\{\mathtt{T}\}C\{\mathtt{F}\}$ is halting problem**
- **Hoare logic for our simple language is complete relative to an oracle**
  - **oracle must be able to prove $P \Rightarrow \mathtt{wlp}(C, Q)$**
  - **relative completeness**
  - **requires expressibility: $\mathtt{wlp}(C, Q)$ expressible in assertion language**

The incompleteness of the proof system for simple Hoare logic stems from the weakness of the proof system of the assertion language logic, not any weakness of the Hoare logic proof system.

- **Clarke showed relative completeness fails for complex languages**

Additional topics ✓

Note: only a fragment of these additional topics will be covered!

- **Blocks and local variables**
- `FOR`**-commands**
- **Arrays**
- **Correct-by-Construction (program refinement)**
- **Separation Logic**

Blocks and local variables ✓

- **Syntax:** `BEGIN VAR` V_1`;` $\cdots$ `VAR` V_n`;` C `END`

- **Semantics: command C is executed, then the values of $V_1, \cdots, V_n$ are restored to the values they had before the block was entered**

  - **the initial values of $V_1, \cdots, V_n$ inside the block are unspecified**

- **Example:** `BEGIN VAR R; R:=X; X:=Y; Y:=R END`

  - **the values of `X` and `Y` are swapped using `R` as a temporary variable**

  - **this command does *not* have a side effect on the variable `R`**

The Block Rule

- **The block rule takes care of local variables**

The block rule

$$\frac{\vdash\ \{P\}\ C\ \{Q\}}{\vdash\ \{P\}\ \texttt{BEGIN VAR}\ V_1\texttt{;}\ \ldots;\ \texttt{VAR}\ V_n\texttt{;}\ C\ \texttt{END}\ \{Q\}}$$

where none of the variables $V_1, \ldots, V_n$ occur in P or Q.

- **Note that the block rule is regarded as including the case when there are no local variables (the '$n = 0$' case)**

The Side Condition

- **The syntactic condition that none of the variables $V_1, \ldots, V_n$ occur in P or Q is an example of a *side condition***

- **From**

  $\vdash$ {`X=x` $\wedge$ `Y=y`} `R:=X; X:=Y; Y:=R` {`Y=x` $\wedge$ `X=y`}

  it follows by the block rule that

  $\vdash$ {`X=x` $\wedge$ `Y=y`} `BEGIN VAR R; R:=X; X:=Y; Y:=R END` {`Y=x` $\wedge$ `X=y`}

  since `R` **does not occur in** `X=x` $\wedge$ `Y=y` **or** `X=y` $\wedge$ `Y=x`

- **However from**

  $\vdash$ {`X=x` $\wedge$ `Y=y`} `R:=X; X:=Y` {`R=x` $\wedge$ `X=y`}

  one *cannot* deduce

  $\vdash$ {`X=x` $\wedge$ `Y=y`} `BEGIN VAR R; R:=X; X:=Y END` {`R=x` $\wedge$ `X=y`}

  since `R` **occurs in** `R=x` $\wedge$ `X=y`

`FOR`-commands ✓

- **Syntax:** `FOR` V`:=`E_1 `UNTIL` E_2 `DO` C

  - **restriction:** **V must not occur in** E_1 **or** E_2**,**
    or be the left hand side of an assignment in C
    (explained later)

- **Semantics:**

  - **if the values of terms** E_1 **and** E_2 **are positive numbers** e_1 **and** e_2
  - **and if** $e_1 \leq e_2$
  - **then** C **is executed** $(e_2{-}e_1){+}1$ **times with the variable** V **taking on the sequence of values** e_1**,** $e_1{+}1$**,** $\ldots$ **,** e_2 **in succession**
  - **for any other values, the** `FOR`**-command has no effect**

- **Example:** `FOR N:=1 UNTIL M DO X:=X+N`

  - **if the value of the variable** `M` **is** m **and** $m \geq 1$**, then the command** `X:=X+N` **is repeatedly executed with** `N` **taking the sequence of values** `1`**,** $\ldots$ **,** m
  - **if** $m < 1$ **then the** `FOR`**-command does nothing**

Subtleties of `FOR`-commands ✓

- **There are many subtly different versions of `FOR`-commands**

- **For example**

  - **the expressions E_1 and E_2 could be evaluated at each iteration**

  - **and the controlled variable V could be treated as global rather than local**

- **Early languages like Algol 60 failed to notice such subtleties**

- **Note that with the semantics presented here**
  `FOR`-commands cannot *generate* non termination

More on the semantics of `FOR`-commands

- **The semantics of**

$$\texttt{FOR}\ V\texttt{:=}E_1\ \texttt{UNTIL}\ E_2\ \texttt{DO}\ C$$

  is as follows

(i) E_1 **and** E_2 **are evaluated once to get values** e_1 **and** e_2**, respectively.**

(ii) If either e_1 **or** e_2 **is not a number, or if** $e_1 > e_2$**, then nothing is done.**

(iii) If $e_1 \leq e_2$ **the `FOR`-command is equivalent to:**

$$\texttt{BEGIN VAR}\ V\texttt{;}\ V\texttt{:=}e_1\texttt{;}\ C\texttt{;}\ V\texttt{:=}e_1{+}1\texttt{;}\ C\ \texttt{;}\ \ldots\ \texttt{;}\ V\texttt{:=}e_2\texttt{;}\ C\ \texttt{END}$$

i.e. C **is executed** $(e_2{-}e_1){+}1$ **times with** V **taking on the sequence of values** e_1, $e_1{+}1$, $\ldots$, e_2

- **If** C **doesn't modify** V **then `FOR`-command equivalent to:**

$$\texttt{BEGIN VAR}\ V\texttt{;}\ V\texttt{:=}e_1\texttt{;}\ \ldots\ \underbrace{C\ \texttt{;}\ V\texttt{:=}V{+}1}_{\textbf{repeated}}\texttt{;}\ \ldots\ V\texttt{:=}e_2\texttt{;}\ C\ \texttt{END}$$