Rigorous Software Development
CSCI-GA 3033-009

Instructor: Thomas Wies

Spring 2013

Lecture 11

Semantics of Programming Languages

- Denotational Semantics
  - Meaning of a program is defined as the mathematical object it computes (e.g., partial functions).
  - Example: Abstract Interpretation
- Axiomatic Semantics
  - Meaning of a program is defined in terms of its effect on the truth of logical assertions.
  - Example: Hoare Logic
- (Structural) Operational Semantics
  - Meaning of a program is defined by formalizing the individual computation steps of the program.
  - Example: Labeled Transition Systems

IMP: A Simple Imperative Language

An IMP program:

```
p := 0;
x := 0;
while x < n do
    x := x + 1;
    p := p + m;
```

Syntax of IMP Commands

- Commands (*Com*)
  c ::= `skip`
  | $x := e$
  | $c_1 \,;\, c_2$
  | `if` b `then` c_1 `else` c_2
  | `while` b `do` c
- Notes:
  - The typing rules have been embedded in the syntax definition.
  - Other parts are not context-free and need to be checked separately (e.g., all variables are declared).
  - Commands contain all the side-effects in the language.
  - Missing: references, function calls, …

Labeled Transition Systems

A labeled transition system (LTS) is a structure $LTS = (Q, Act, \rightarrow)$ where

- Q is a set of states,
- Act is a set of actions,
- $\rightarrow \subseteq Q \times Act \times Q$ is a transition relation.

We write $q \xrightarrow{a} q'$ for $(q, a, q') \in \rightarrow$.

Operational Semantics of IMP

$$q \xrightarrow{\texttt{skip}} q$$

$$\frac{\langle e, q\rangle \Downarrow n}{q \xrightarrow{x\ :=\ e} q \mathbin{++} \{x \longmapsto n\}}$$

$$\frac{q \xrightarrow{c_1} q' \qquad q' \xrightarrow{c_2} q''}{q \xrightarrow{c_1\,;\,c_2} q''}$$

$$\frac{\langle b, q\rangle \Downarrow \text{true} \qquad q \xrightarrow{c_1} q'}{q \xrightarrow{\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2} q'}$$

$$\frac{\langle b, q\rangle \Downarrow \text{false} \qquad q \xrightarrow{c_2} q'}{q \xrightarrow{\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2} q'}$$

$$\frac{\langle b, q\rangle \Downarrow \text{false}}{q \xrightarrow{\texttt{while}\ b\ \texttt{do}\ c} q}$$

$$\frac{\langle b, q\rangle \Downarrow \text{true} \qquad q \xrightarrow{c} q' \qquad q' \xrightarrow{\texttt{while}\ b\ \texttt{do}\ c} q''}{q \xrightarrow{\texttt{while}\ b\ \texttt{do}\ c} q''}$$

Axiomatic Semantics

- An axiomatic semantics consists of:
  - a language for stating assertions about programs;
  - rules for establishing the truth of assertions.
- Some typical kinds of assertions:
  - This program terminates.
  - If this program terminates, the variables x and y have the same value throughout the execution of the program.
  - The array accesses are within the array bounds.
- Some typical languages of assertions
  - First-order logic
  - Other logics (temporal, linear)
  - Special-purpose specification languages (Z, Larch, JML)

Assertions for IMP

- The assertions we make about IMP programs are of the form:

  $\{A\}\ c\ \{B\}$

  with the meaning that:
  - If A holds in state q and $q \xrightarrow{c} q'$
  - then B holds in q'
- A is the precondition and B is the postcondition
- For example:

  $\{\ y \leq x\ \}\ z := x;\ z := z + 1\ \{\ y < z\ \}$

  is a valid assertion
- These are called Hoare triples or Hoare assertions

Assertions for IMP

- {A} c {B} is a partial correctness assertion. It does not imply termination of c.
- [A] c [B] is a total correctness assertion meaning that
  - If A holds in state q
  - then there exists q' such that $q \xrightarrow{c} q'$ and B holds in state q'
- Now let's be more formal
  - Formalize the language of assertions, A and B
  - Say when an assertion holds in a state
  - Give rules for deriving valid Hoare triples

The Assertion Language

- We use first-order predicate logic with IMP expressions

$$A ::= \text{true} \mid \text{false} \mid e_1 = e_2 \mid e_1 \geq e_2 \mid A_1 \wedge A_2 \mid A_1 \vee A_2 \mid A_1 \Rightarrow A_2 \mid \forall x.A \mid \exists x.A$$

- Note that we are somewhat sloppy and mix the logical variables and the program variables.
- Implicitly, all IMP variables range over integers.
- All IMP Boolean expressions are also assertions.

Semantics of Assertions

- We introduced a language of assertions, we need to assign meanings to assertions.
- Notation $q \vDash A$ says that assertion A holds in a given state q.
  - This is well-defined when q is defined on all variables occurring in A.
- The $\vDash$ judgment is defined inductively on the structure of assertions.
- It relies on the semantics of arithmetic expressions from IMP.

Semantics of Assertions

- $q \models \text{true}$ always
- $q \models e_1 = e_2$ iff $<e_1,q>\Downarrow = <e_2,q>\Downarrow$
- $q \models e_1 \geq e_2$ iff $<e_1,q>\Downarrow \geq <e_2,q>\Downarrow$
- $q \models A_1 \wedge A_2$ iff $q \models A_1$ and $q \models A_2$
- $q \models A_1 \vee A_2$ iff $q \models A_1$ or $q \models A_2$
- $q \models A_1 \Rightarrow A_2$ iff $q \models A_1$ implies $q \models A_2$
- $q \models \forall x.A$ iff $\forall n \in \mathbb{Z}.\ q[x:=n] \models A$
- $q \models \exists x.A$ iff $\exists n \in \mathbb{Z}.\ q[x:=n] \models A$

Semantics of Hoare Triples

- Now we can define formally the meaning of a partial correctness assertion:

  $\models \{A\}\ c\ \{B\}$ iff

  $\forall q \in Q.\ \forall q' \in Q.\ q \models A \wedge q \xrightarrow{c} q' \Rightarrow q' \models B$
- and the meaning of a total correctness assertion:

  $\models [A]\ c\ [B]$ iff

  $\forall q \in Q.\ q \models A \Rightarrow \exists q' \in Q.\ q \xrightarrow{c} q' \wedge q' \models B$

  or even better:

  $$\begin{array}{l} \quad \forall q \in Q.\ \forall q' \in Q.\ q \models A \wedge q \xrightarrow{c} q' \Rightarrow q' \models B \\ \wedge \\ \quad \forall q \in Q.\ q \models A \Rightarrow \exists q' \in Q.\ q \xrightarrow{c} q' \wedge q' \models B \end{array}$$

Inferring Validity of Assertions

- Now we have the formal mechanism to decide when $\{A\}\ c\ \{B\}$
  - But it is not satisfactory,
  - because $\models \{A\}\ c\ \{B\}$ is defined in terms of the operational semantics.
  - We practically have to run the program to verify an assertion.
  - Also it is impossible to effectively verify the truth of a $\forall x.\ A$ assertion (by using the definition of validity)
- So we define a symbolic technique for deriving valid assertions from others that are known to be valid
  - We start with validity of first-order formulas

Inference Rules

- We write $\vdash A$ when A can be inferred from basic axioms.
- The inference rules for $\vdash A$ are the usual ones from first-order logic with arithmetic.
- Natural deduction style rules:

$$\frac{\vdash A \qquad \vdash B}{\vdash A \wedge B} \qquad \frac{\vdash A[a/x]}{\vdash \forall x.\, A} \ \text{where } a \text{ is fresh} \qquad \frac{\vdash \forall x.\, A}{\vdash A[e/x]}$$

$$\frac{\vdash A}{\vdash A \vee B} \qquad \frac{\vdash B}{\vdash A \vee B}$$

$$\frac{\vdash A[e/x]}{\vdash \exists x.\, A} \qquad \frac{\vdash \exists x.\, A \qquad \begin{array}{c}\vdash A[a/x] \\ \cdots \\ \vdash B\end{array}}{\vdash B} \ \text{where } a \text{ is fresh} \qquad \frac{\vdash A \Rightarrow B \qquad \vdash A}{\vdash B} \qquad \frac{\begin{array}{c}\vdash A \\ \cdots \\ \vdash B\end{array}}{\vdash A \Rightarrow B}$$

Inference Rules for Hoare Triples

- Similarly we write $\vdash \{A\}\ c\ \{B\}$ when we can derive the triple using inference rules
- There is one inference rule for each command in the language.
- Plus, the rule of consequence

$$\frac{\vdash A' \Rightarrow A \qquad \vdash \{A\}\ c\ \{B\} \qquad \vdash B \Rightarrow B'}{\vdash \{A'\}\ c\ \{B'\}}$$

Inference Rules for Hoare Logic

- One rule for each syntactic construct:

$\vdash \{A\}\ \texttt{skip}\ \{A\}$

$\vdash \{A[e/x]\}\ x{:=}e\ \{A\}$

$$\frac{\vdash \{A\}\ c_1\ \{B\} \quad \vdash \{B\}\ c_2\ \{C\}}{\vdash \{A\}\ c_1;\ c_2\ \{C\}}$$

Exercise: Hoare Rules

- Is the following alternative rule for assignment still correct?

$$\vdash \{\text{true}\}\ x{:=}e\ \{x = e\}$$

Hoare Rules

- For some constructs, multiple rules are possible

  alternative "forward axiom" for assignment:

  $$\vdash \{A\}\ x{:=}e\ \{\exists x_0.\ x = e[x_0/x] \wedge A[x_0/x]\}$$

  alternative rule for `while` loops:

  $$\frac{\vdash \mathtt{I} \wedge b \Rightarrow \mathtt{C} \quad \vdash \{\mathtt{C}\}\ c\ \{\mathtt{I}\} \quad \vdash \mathtt{I} \wedge \neg b \Rightarrow \mathtt{B}}{\vdash \{\mathtt{I}\}\ \mathtt{while}\ b\ \mathtt{do}\ c\ \{\mathtt{B}\}}$$

- These alternative rules are derivable from the previous rules, plus the rule of consequence.

Example: Conditional

$$\dfrac{}{\vdash \{\text{true}\}\ \texttt{if}\ y \leq 0\ \texttt{then}\ x := 1\ \texttt{else}\ x := y\ \{x > 0\}}$$

Example: a simple loop

- We want to infer that
  $\vdash \{x \leq 0\}\ \texttt{while}\ x \leq 5\ \text{do}\ x := x + 1\ \{x = 6\}$
- Use the rule for while with invariant $\mathtt{I} \equiv x \leq 6$

$$\frac{\dfrac{\vdash x \leq 6 \wedge x \leq 5 \Rightarrow x + 1 \leq 6 \qquad \vdash \{x + 1 \leq 6\}\ x := x + 1\ \{x \leq 6\}}{\vdash \{x \leq 6 \wedge x \leq 5\}\ x := x + 1\ \{x \leq 6\}}}{\vdash \{\mathrm{x} \leq 6\}\ \texttt{while}\ \mathrm{x} \leq 5\ \text{do}\ \mathrm{x} := \mathrm{x} + 1\ \{\ \mathrm{x} \leq 6 \wedge \mathrm{x} > 5\}}$$

Example: a more interesting program

- We want to derive that

$\{n \geq 0\}$

$p := 0;$

$x := 0;$

`while` $x < n$ do

    $x := x + 1;$

    $p := p + m$

$\{p = n * m\}$

Example: a more interesting program

Only applicable rule (except for rule of consequence):

$$\frac{\vdash \{A\}\, c_1 \{C\} \quad \vdash \{C\}\, c_2\, \{B\}}{\vdash \{A\}\, c_1;\, c_2\, \{B\}}$$

$$\frac{\vdash \{n \geq 0\}\ p{:=}0;\ x{:=}0\ \{C\} \quad \vdash \{C\}\ \text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)\ \{p = n * m\}}{\vdash \underbrace{\{n \geq 0\}}_{A}\ \underbrace{p{:=}0;\ x{:=}0;}_{c_1}\ \underbrace{\text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)}_{c_2}\ \underbrace{\{p = n * m\}}_{B}}$$

Example: a more interesting program

What is C? Look at the next possible matching rules for c_2!

Only applicable rule (except for rule of consequence):

$$\frac{\vdash \{I \wedge b\}\ c\ \{I\}}{\vdash \{I\}\ \texttt{while}\ b\ \texttt{do}\ c\ \{I \wedge \neg b\}}$$

We can match {I} with {C} but we cannot match $\{I \wedge \neg b\}$ and {p = n * m} directly. Need to apply the rule of consequence first!

$$\frac{\vdash \{n \geq 0\}\ p{:=}0;\ x{:=}0\ \{C\} \qquad \vdash \{C\}\ \text{while } x < n \text{ do } (x{:=}x{+}1;\ p{:=}p{+}m)\ \{p = n * m\}}{\vdash \underbrace{\{n \geq 0\}}_{A}\ \underbrace{p{:=}0;\ x{:=}0;}_{c_1}\ \underbrace{\text{while } x < n \text{ do } (x{:=}x{+}1;\ p{:=}p{+}m)}_{c_2}\ \underbrace{\{p = n * m\}}_{B}}$$

Example: a more interesting program

What is C? Look at the next possible matching rules for c_2!

Only applicable rule (except for rule of consequence):

$$\frac{\vdash \{\mathtt{I} \wedge b\}\ c\ \{\mathtt{I}\}}{\vdash \underbrace{\{\mathtt{I}\}}_{\mathrm{A}}\ \underbrace{\mathtt{while}\ b\ \mathrm{do}\ c}_{c'}\ \underbrace{\{\mathtt{I} \wedge \neg b\}}_{\mathrm{B}}}$$

Rule of consequence:

$$\frac{\vdash \mathrm{A}' \Rightarrow \mathrm{A} \qquad \vdash \{\mathrm{A}\}\ c'\ \{\mathrm{B}\} \qquad \vdash \mathrm{B} \Rightarrow \mathrm{B}'}{\vdash \{\mathrm{A}'\}\ c'\ \{\mathrm{B}'\}}$$

$\mathtt{I} = \mathrm{A} = \mathrm{A}' = \mathrm{C}$

$$\frac{\vdash \{n \geq 0\}\ \text{p:=0; x:=0}\ \{\mathrm{C}\} \qquad \vdash \overbrace{\{\mathrm{C}\}}^{\mathrm{A}'}\ \overbrace{\text{while x < n do (x:=x+1; p:=p+m)}}^{c'}\ \overbrace{\{p = n * m\}}^{\mathrm{B}'}}{\vdash \{n \geq 0\}\ \text{p:=0; x:=0; while x < n do (x:=x+1; p:=p+m)}\ \{p = n * m\}}$$

Example: a more interesting program

What is I? Let's keep it as a placeholder for now!

Next applicable rule:

$$\frac{\vdash \{A\}\ c_1 \{C\} \qquad \vdash \{C\}\ c_2\ \{B\}}{\vdash \{A\}\ c_1;\ c_2\ \{B\}}$$

$$\frac{\vdash \{n \geq 0\}\ p:=0;\ x:=0\ \{I\} \qquad \dfrac{\dfrac{\vdash \overbrace{\{I \wedge x<n\}}^{A}\ \overbrace{x := x+1;}^{c_1}\ \overbrace{p:=p+m}^{c_2}\ \overbrace{\{I\}}^{B}}{\vdash \{I\}\ \text{while } x < n \text{ do } (x:=x+1;\ p:=p+m)\ \{I \wedge x \geq n\}} \qquad \vdash I \wedge x \geq n \Rightarrow p = n * m}{\vdash \{I\}\ \text{while } x < n \text{ do } (x:=x+1;\ p:=p+m)\ \{p = n * m\}}}{\vdash \{n \geq 0\}\ p:=0;\ x:=0;\ \text{while } x < n \text{ do } (x:=x+1;\ p:=p+m)\ \{p = n * m\}}$$

Example: a more interesting program

What is C? Look at the next possible matching rules for c_2!

Only applicable rule (except for rule of consequence):

$\vdash \{A[e/x]\}\ x{:=}e\ \{A\}$

$$\dfrac{\vdash \{n \geq 0\}\ \text{p:=0; x:=0}\ \{I\} \qquad \dfrac{\begin{array}{c}\dfrac{\dfrac{\vdash \overbrace{\{I \wedge x<n\}}^{A}\ \overbrace{\text{x := x+1}}^{c_1}\ \{C\} \qquad \vdash \{C\}\ \overbrace{\text{p:=p+m}}^{c_2}\ \overbrace{\{I\}}^{B}}{\vdash \{I \wedge x<n\}\ \text{x := x+1; p:=p+m}\ \{I\}}}{\vdash \{I\}\ \text{while x < n do (x:=x+1; p:=p+m)}\ \{I \wedge x \geq n\}} \\ \vdash I \wedge x \geq n \Rightarrow p = n * m\end{array}}{\vdash \{I\}\ \text{while x < n do (x:=x+1; p:=p+m)}\ \{p = n * m\}}}{\vdash \{n \geq 0\}\ \text{p:=0; x:=0; while x < n do (x:=x+1; p:=p+m)}\ \{p = n * m\}}$$

Example: a more interesting program

What is C? Look at the next possible matching rules for c_2!

Only applicable rule (except for rule of consequence):

$\vdash \{A[e/x]\}\ x{:=}e\ \{A\}$

$$\dfrac{\dfrac{\vdash \{n \geq 0\}\ p{:=}0;\ x{:=}0\ \{I\} \qquad \dfrac{\dfrac{\dfrac{\vdash \{I \land x<n\}\ x{:=}x+1\ \{I[p+m/p]\} \quad \vdash \{I[p+m/p]\}\ p{:=}p+m\ \{I\}}{\vdash \{I \land x<n\}\ x{:=}x+1;\ p{:=}p+m\ \{I\}}}{\vdash \{I\}\ \text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)\ \{I \land x \geq n\}} \quad \vdash I \land x \geq n \Rightarrow p = n * m}{\vdash \{I\}\ \text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)\ \{p = n * m\}}}{\vdash \{n \geq 0\}\ p{:=}0;\ x{:=}0;\ \text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)\ \{p = n * m\}}}{}$$

Example: a more interesting program

Only applicable rule (except for rule of consequence):

$\vdash \{A[e/x]\}\ x{:=}e\ \{A\}$

Need rule of consequence to match $\{I \wedge x<n\}$ and $\{I[x+1/x, p+m/p]\}$

$$\dfrac{\vdash \{n \geq 0\}\ p{:=}0;\ x{:=}0\ \{I\} \qquad \dfrac{\dfrac{\dfrac{\vdash\{I \wedge x<n\}\ x{:=}x+1\ \{I[p+m/p]\} \quad \vdash\{I[p+m/p]\}\ p{:=}p+m\ \{I\}}{\vdash\{I \wedge x<n\}\ x{:=}x+1;\ p{:=}p+m\ \{I\}}}{\vdash\{I\}\ \text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)\ \{I \wedge x \geq n\}} \qquad \vdash I \wedge x \geq n \Rightarrow p = n * m}{\vdash\{I\}\ \text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)\ \{p = n * m\}}}{\vdash \{n \geq 0\}\ p{:=}0;\ x{:=}0;\ \text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)\ \{p = n * m\}}$$

Example: a more interesting program

Let’s just remember the open proof obligations!

$$
\dfrac{
\vdash \{n \geq 0\}\ p:=0;\ x:=0\ \{I\}
\qquad
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\begin{array}{c}
\vdash \{I[x+1/x,\ p+m/p]\}\ x:=x+1\ \{I[p+m/p]\} \\
\vdash I \wedge x < n \Rightarrow I[x+1/x,\ p+m/p]
\end{array}
}{
\vdash \{I \wedge x<n\}\ x:=x+1\ \{I[p+m/p]\}
}
\quad
\vdash \{I[p+m/p\}\ p:=p+m\ \{I\}
}{
\vdash \{I \wedge x<n\}\ x:=x+1;\ p:=p+m\ \{I\}
}
}{
\vdash \{I\}\ \text{while}\ x < n\ \text{do}\ (x:=x+1;\ p:=p+m)\ \{I \wedge x \geq n\}
}
\quad
\begin{array}{c}
\vdots \\
\vdash I \wedge x \geq n \Rightarrow p = n * m
\end{array}
}{
\vdash \{I\}\ \text{while}\ x < n\ \text{do}\ (x:=x+1;\ p:=p+m)\ \{p = n * m\}
}
}{
\vdash \{n \geq 0\}\ p:=0;\ x:=0;\ \text{while}\ x < n\ \text{do}\ (x:=x+1;\ p:=p+m)\ \{p = n * m\}
}
$$

Example: a more interesting program

Let's just remember the open proof obligations!

$$\vdash I \land x < n \Rightarrow I[x+1/x,\ p+m/p]$$

$$\vdash I \land x \geq n \Rightarrow p = n * m$$

Continue with the remaining part of the proof tree, as before.

$$\dfrac{\dfrac{\dfrac{\vdash n \geq 0 \Rightarrow I[0/p,\ 0/x] \qquad \vdash \{I[0/p,\ 0/x]\}\ p{:=}0\ \{I[0/x]\}}{\vdash \{n \geq 0\}\ p{:=}0\ \{I[0/x]\}} \qquad \vdash \{I[0/x]\}\ x{:=}0\ \{I\}}{\vdash \{n \geq 0\}\ p{:=}0;\ x{:=}0\ \{I\}} \qquad \dfrac{\vdots}{\vdash \{I\}\ \text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)\ \{p = n * m\}}}{\vdash \{n \geq 0\}\ p{:=}0;\ x{:=}0;\ \text{while } x < n \text{ do } (x{:=}x+1;\ p{:=}p+m)\ \{p = n * m\}}$$

Now we only need to solve the remaining constraints!

Example: a more interesting program

Find $\mathtt{I}$ such that all constraints are simultaneously valid:

$\vdash n \geq 0 \Rightarrow \mathtt{I}[0/p, 0/x]$

$\vdash \mathtt{I} \wedge x < n \Rightarrow \mathtt{I}[x+1/x, p+m/p]$

$\vdash \mathtt{I} \wedge x \geq n \Rightarrow p = n * m$

$\mathtt{I} \equiv p = x * m \wedge x \leq n$

$\vdash n \geq 0 \Rightarrow 0 = 0 * m \wedge 0 \leq n$

$\vdash p = x * m \wedge x \leq n \wedge x < n \Rightarrow p+m = (x+1) * m \wedge x+1 \leq n$

$\vdash p = x * n \wedge x \leq n \wedge x \geq n \Rightarrow p = n * m$

All constraints are valid!