Announcements

- Homework 4 is out, due Mon Feb 23
  - Smaller than Homework 3
  - Think about final projects
- Final project guidelines posted
  - Some ideas and suggestions coming soon

Final Project

- Options:
  - Research project (recommended, in general)
  - Literature survey
  - Implementation project
- Write a 5-page paper (conference style)
- Give a presentation
- On a topic of your choice
  - Ideal: integrate PL with your research
- Pair projects possible if large

Survey from Homework 3

- Time spent: 10.3 (mean), 11 (median)
  - Discussion could be more focused
    - Still great to interrupt and get clarification!

Questions?

Review of Axiomatic Semantics
Notation: Assertions

\( \{A\} c \{B\} \)

with the meaning that:
- if \(A\) holds in state \(\sigma\) and if \(<c, \sigma> \Downarrow \sigma'\)
- then \(B\) holds in \(\sigma'\)
• \(A\) is the **precondition**
• \(B\) 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**
  - Doesn’t imply termination (= it is valid if \(c\) diverges)
• \([A]\ c \ [B]\) is a **total correctness assertion**
  meaning that
  If \(A\) holds in state \(\sigma\)
  Then there exists \(\sigma'\) such that \(<c, \sigma> \Downarrow \sigma'\)
  and \(B\) holds in state \(\sigma'\)

Derivation Rules for Hoare Triples

• Similarly we write \(\vdash \{A\} c \{B\}\) when we can derive the triple using derivation rules
• There is one derivation rule for each command in the language
• Plus, the "evil" rule of consequence

\[
\begin{align*}
\vdash A' &\Rightarrow A \\
\vdash \{A\} c \{B\} &\Rightarrow B \Rightarrow B' \\
\vdash \{A'\} c \{B'\}
\end{align*}
\]

Derivation Rules for Hoare Logic

• One rule for each syntactic construct:

\[
\begin{align*}
\vdash \{A\} \text{skip} &\{A\} \\
\vdash \{[e/x]A\} x := e &\{A\} \\
\vdash \{A\} c_1 &\vdash \{B\} c_2 &\vdash \{c_1(c_2 \circ c)\} \{B\} \\
\vdash \{A\} \text{if } b \text{ then } c_1 &\text{ else } c_2 &\vdash \{A\} \text{ if } b \text{ then } c_1 &\text{ else } c_2 &\vdash \{A\} \text{ while } b \text{ do } c \{A \land \neg b\}
\end{align*}
\]

The Assignment Axiom

• "Assignment is undoubtedly the most characteristic feature of programming a digital computer, and one that most clearly distinguishes it from other branches of mathematics. It is surprising therefore that the axiom governing our reasoning about assignment is quite as simple as any to be found in elementary logic." — Tony Hoare

• Caveats are sometimes needed for languages with aliasing (the strong update problem):
  - If \(x\) and \(y\) are aliased then
    \((\text{true }) \ x := 5 \ (x \times y = 10)\)
is true

Example: Conditional

\[
\begin{align*}
D_1 &:: (\text{true } y \leq 03 \ x := 1 \ x > 0) \\
D_2 &:: (\text{true } y \neq 03 \ y := 4 \ x > 03)
\end{align*}
\]

\[
\begin{align*}
\vdash (\text{true} \text{ if } y \leq 0 \text{ then } x := 1 \text{ else } x := y (x > 0)) \\
\vdash \{\text{true}\} \text{ if } y \leq 0 \text{ then } x := 1 \text{ else } x := y (x > 0)
\end{align*}
\]

\[
\begin{align*}
\vdash \{\text{true}\} \text{ if } y \leq 0 \text{ then } x := 1 \text{ else } x := y (x > 0)
\end{align*}
\]
Example: Conditional

\[ D_1 :: \vdash (true \land y \leq 0) \ x := 1 \ (x > 0) \]

\[ D_2 :: \vdash (true \land y > 0) \ x := y \ (x > 0) \]

\[ \vdash (true) \text{ if } y \leq 0 \text{ then } x := 1 \text{ else } x := y \ (x > 0) \]

- \( D_1 \) and \( D_2 \) were obtained by consequence and assignment. \( D_1 \) details:

\[ \vdash (1 > 0) \ x := 1 \ (x > 0) \]

\[ \vdash \text{true} \land y \leq 0 \Rightarrow 1 > 0 \]

\[ \vdash \text{true} \land y \leq 0 \ x := 1 \ (x > 0) \]

Example: Loop

- We want to derive that

\[ \vdash \{x \leq 0\} \text{ while } x \leq 5 \text{ do } x := x + 1 \ (x = 6) \]

- Use the rule for while with invariant \( A \) ?

\[ \vdash \{A\} \text{ while } x \leq 5 \text{ do } x := x+1 \ (A \land x > 5) \]

- Then finish-off with consequence

\[ \vdash (x \leq 0) \text{ while ... } (x = 6) \]

Using Hoare Rules

- Hoare rules are mostly syntax directed

- There are three wrinkles:
  - What invariant to use for while? (fixedpoints, widening)
  - When to apply consequence? (theorem proving)
  - How do we prove the implications involved in consequence? (theorem proving)

- This is how theorem proving gets in the picture

  - This turns out to be doable!
  - The loop invariants turn out to be the hardest problem!
  (Should the programmer give them? See Dijkstra, ESC)

Where Do We Stand?

- We have a language for asserting properties of programs
- We know when such an assertion is true
- We also have a symbolic method for deriving assertions

End of Review
One-Slide Summary

• A system of axiomatic semantics is sound if everything we can prove is also true.
  \[ \vdash \{ A \} \subseteq \{ B \} \text{ then } \models \{ A \} \subseteq \{ B \} \]
• We prove this by nested induction on the structure of the operational semantics derivation and the axiomatic semantics proof.
• A system of axiomatic semantics is complete if we can prove all true things.
  \[ \vdash \{ A \} \subseteq \{ B \} \text{ then } \vdash \{ A \} \subseteq \{ B \} \]
• Our system is relatively complete (= just as complete as the underlying logic). We use weakest preconditions to reason about soundness. Verification conditions are preconditions that are easy to compute.

Soundness of Axiomatic Semantics

• Formal statement of soundness:
  \[ \vdash \{ A \} \subseteq \{ B \} \text{ then } \models \{ A \} \subseteq \{ B \} \]
  or, equivalently
For all \( \sigma \), if \( \sigma \models A \) and Op :: <c, \( \sigma \) \( \cup \) \( \sigma' \) and Pr :: \( \vdash \{ A \} \subseteq \{ B \} \)
  then \( \sigma' \models B \)
• "Op" === "Opsem Derivation"
• "Pr" === "Axiomatic Proof"

How should we prove soundness?

Not easily!

• By induction on the structure of \( c \)?
  - No, problems with while and rule of consequence
• By induction on the structure of \( \text{Op} \)?
  - No, problems with consequence
• By induction on the structure of \( \text{Pr} \)?
  - No, problems with while
• By nested induction on the structure of \( \text{Op} \) and \( \text{Pr} \)
  - Yes! New Technique!

Nested Induction

• Consider two structures \( \text{Op} \) and \( \text{Pr} \)
  - Assume that \( x \prec y \) iff \( x \) is a substructure of \( y \)
• Define the ordering
  \[ (o, p) \prec (o', p') \text{ iff } o \prec o' \text{ or } o = o' \text{ and } p < p' \]
  - Called lexicographic (dictionary) ordering
• This \( \prec \) is a well-founded order and leads to nested induction
• If \( o \prec o' \) then \( p \) can actually be larger than \( p' \)
• It can even be unrelated to \( p' \)

Soundness of the Consequence Rule

If \( \sigma \models A, \text{Op} :: <c, \sigma \cup \sigma', \text{ and Pr} :: \vdash \{ A \} \subseteq \{ B \} \) then \( \sigma' \models B \)

• Case: last rule used in \( \text{Pr} :: \vdash \{ A \} \subseteq \{ B \} \)
  is the consequence rule:
  \[ \vdash A \Rightarrow A' \] \( \vdash A' \Rightarrow \vdash \{ A \} \subseteq \{ B \} \) \( \vdash \{ B \} \Rightarrow B \] \( \vdash \{ A \} \subseteq \{ B \} \)
• From soundness of the first-order logic derivations we have \( \sigma \models A \Rightarrow A' \), hence \( \sigma \models A' \)
  - By i.h. with \( \text{Pr} \) and \( \text{Op} \) we get that \( \sigma' \models B' \)
• From soundness of the first-order logic derivations we have that \( \sigma' \models B' \Rightarrow B \), hence \( \sigma \models B \)
Soundness of the Assignment Axiom

\[
\text{If } \sigma \models A, \text{ Op:: } <e, \sigma > \Downarrow \sigma' \text{ and } \text{Pr:: } \vdash (A \land b) \text{, then } \sigma' \models B
\]

- Case: the last rule used in Pr :: \(\vdash (A \land b) \text{ c } (B)\) is the assignment rule
  \[
  \vdash \left( \left( \left( e/x \right) B \right) x := e \left( B \right) \right)
  \]
- The last rule used in Op :: \(\left< x := e, \sigma \right> \Downarrow \sigma'\) must be
  \[
  \left< x := e, \sigma \right> \Downarrow \sigma[x := n]
  \]
- We must prove the substitution lemma:
  \[
  \text{If } \sigma \models \left( e/x \right) B \text{ and } \left< e, \sigma \right> \Downarrow \sigma' \text{ then } \sigma' \models B
  \]

Soundness of the While Rule (!)

\[
\text{If } \sigma \models A, \text{ Op:: } <c, \sigma > \Downarrow \sigma' \text{ and } \text{Pr:: } \vdash (A \land \neg b) \text{, then } \sigma' \models B
\]

- Case: last rule used in Pr :: \(\vdash (A \land \neg b) \text{ c } (B)\) was the while rule:
  \[
  \vdash (A \land \neg b) \text{ while } c \text{ do } c \text{ } (A \land \neg b)
  \]
- Two possible rules for the root of Op (by inversion)
  - We’ll only do the complicated case:
    \[
    \text{Op}_1 \:: \left< b, \sigma \right> \Downarrow \text{true}
    \]
    \[
    \text{Op}_2 \:: \left< c, \sigma \right> \Downarrow \sigma'
    \]
    \[
    \text{Op}_3 \:: \left< \text{while } b \text{ do } c, \sigma \right> \Downarrow \sigma'
    \]

To show: \(\sigma' \models A \land \neg b\)

- By soundness of booleans and Op
  - We get \(\sigma \models b\)
- Hence \(\sigma \models A \land b\)
- By i.h. on Pr and Op
  - We get \(\sigma' \models A \land b\)
- By i.h. on Pr and Op
  - We get \(\sigma'' \models A \land \neg b\)

Soundness of the While Rule (II)

\[
\text{If } \sigma \models A, \text{ Op:: } <c, \sigma > \Downarrow \sigma' \text{ and } \text{Pr:: } \vdash (A \land \neg b) \text{, then } \sigma' \models B
\]

- Case: last rule used in Pr :: \(\vdash (A \land b) \text{ c } (B)\) was the while rule:
  \[
  \vdash (A \land b) \text{ while } b \text{ do } c \text{ } (A \land b)
  \]
- Two possible rules for the root of Op (by inversion)
  - We’ll only do the complicated case:
    \[
    \text{Op}_1 \:: \left< b, \sigma \right> \Downarrow \text{true}
    \]
    \[
    \text{Op}_2 \:: \left< c, \sigma \right> \Downarrow \sigma'
    \]
    \[
    \text{Op}_3 \:: \left< \text{while } b \text{ do } c, \sigma \right> \Downarrow \sigma'
    \]

To show: \(\sigma'' \models A \land b\)

- By soundness of booleans and Op
  - We get \(\sigma \models b\)
- Hence \(\sigma \models A \land b\)
- By i.h. on Pr and Op
  - We get \(\sigma' \models A \land b\)
- By i.h. on Pr and Op
  - We get \(\sigma'' \models A \land \neg b\)

Completeness of Axiomatic Semantics

- If \(\models (A) \text{ c } (B)\) can we always derive \(\vdash (A) \text{ c } (B)\)?
- If so, axiomatic semantics is \underline{complete}
- If not then there are valid properties of programs that we cannot verify with Hoare rules :-(
  - Good news: for our language the Hoare triples are \underline{complete}
  - Bad news: only if the underlying logic is \underline{complete}
    
    (whenever \(\models A\) we also have \(\vdash A\))
    - This is called \underline{relative completeness}

Examples, General Plan

- OK, so:
  \[
  \models \left( x < 5 \land z = 2 \right) y := x + 2 \{ y < 7 \}
  \]
- Can we prove it?
  \[
  \vdash \left( x < 5 \land z = 2 \right) y := x + 2 \{ y < 7 \}
  \]
- Well, we could easily prove:
  \[
  \vdash \left( x + 2 < 7 \right) y := x + 2 \{ y < 7 \}
  \]
- And we know ...
  \[
  \vdash x < 5 \land z = 2 \Rightarrow x + 2 < 7
  \]
- Shouldn’t those two proofs be enough?

Proof Idea

- Dijkstra’s idea: To verify that \(\{ A \} \text{ c } (B)\)
  a) Find all predicates \(A'\) such that \(\models (A') \text{ c } (B)\)
    - call this set \(\text{Pre}(c, B)\) (Pre = “pre-conditions”)
  b) Verify for one \(A' \in \text{Pre}(c, B)\) that \(A \Rightarrow A'\)
- Assertions can be ordered:
  \[
  \begin{array}{c}
  \text{false} \\
  \text{true} \\
  \text{weakest} \\
  \text{strong}
  \end{array}
  \]
  \[
  \text{precondition: WP}(c, B)
  \]
- Thus: compute \(\text{WP}(c, B)\) and prove \(A \Rightarrow \text{WP}(c, B)\)”
Proof Idea

- **Completeness** of axiomatic semantics:
  \[ \vdash (A) \implies (B) \]  
  \[ \vdash (A) \implies (B) \]

- Assuming that we can compute \( wp(c, B) \) with the following properties:
  - \( wp \) is a precondition (according to the Hoare rules)
    \[ \vdash (wp(c, B)) \implies (B) \]
  - \( wp \) is (truly) the weakest precondition
    \[ \vdash (A) \implies wp(c, B) \]

- We also need that whenever \( \vdash A \) then \( \vdash A ! \)

Weakest Preconditions

- Define \( wp(c, B) \) inductively on \( c \), following the Hoare rules:
  - \( wp(c_1, c_2, B) = wp(c_1, wp(c_2, B)) \)
  - \( wp(x := e, B) = \{e/x\}B \)
  - \( wp(if E then c_1 else c_2, B) = E \implies wp(c_1, B) \land \neg E \implies wp(c_2, B) \)

Weakest Preconditions for while

- We start from the unwinding equivalence
  \[ while \ b \ do \ c = \]
  \[ if \ b \ then \ c; \ while \ b \ do \ c \ else \ skip \]

- Let \( w = while \ b \ do \ c \) and \( W = wp(w, B) \)

- We have that
  \[ W = b \implies wp(c, W) \land \neg b \implies B \]

- But this is a recursive equation!
  - We know how to solve these using domain theory
  - But we need a domain for assertions