Announcements

• Homework 6 is due Thu
• Homework 3 grades posted
• Homework 4 returned very 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
Example: Loop

- We want to derive that
  \[ \{ x \leq 0 \} \text{ while } x \leq 5 \text{ do } x := x + 1 \{ x = 6 \} \]
- Use the rule for while with invariant \( x \leq 6 \)
  \[ \vdash x \leq 6 \land x \leq 5 \Rightarrow x \leq 5 \]
  \[ \vdash x = x + 1 \{ x \leq 6 \} \]
  \[ \vdash x = x + 1 \{ x \leq 6 \} \]
- Then finish off with consequence
  \[ \vdash x \leq 6 \]

Using Hoare Rules

- Hoare rules are mostly syntax directed
- There are three wrinkles:
  - What invariant to use for while? (fixpoints, widening)
  - When to apply consequence? (theorem proving)
  - How do you 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

\[ (A) \vdash (B) \]

\[ \vdash (A) \vdash (B) \]

Soundness and Completeness of Axiomatic Semantics

- A system of axiomatic semantics is sound if everything we can prove is also true.
  \[ \vdash (A) \vdash (B) \]
- 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.
  \[ \vdash (A) \vdash (B) \]
- 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.

End of Review

Questions?
Soundness of Axiomatic Semantics

- Formal statement of soundness:
  
  \[ \text{if } \{ A \} \subseteq \{ B \} \text{ then } \{ A \} \subseteq \{ B \} \]

  or, equivalently

  For all \( \sigma \), if \( \sigma \models A \)
  and Op :: \( \langle \sigma, \sigma \uplus \sigma' \rangle \downarrow \sigma' \)
  and Pr :: \( \{ A \} \subseteq \{ B \} \)
  then \( \sigma' \models B \)

  • "Op" === "Oopsem 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 Op?
  - No, problems with consequence

- By induction on the structure of Pr?
  - No, problems with while

- By nested induction on the structure of Op and Pr
  - Yes! New Technique!

Nested Induction

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

Soundness of the Consequence Rule

If \( \sigma \models A \), Op :: \( \langle \sigma, \sigma \uplus \sigma' \rangle \downarrow \sigma' \), and Pr :: \( \{ A \} \subseteq \{ B \} \), then \( \sigma' \models B \)

- Case: last rule used in Pr :: \( \{ A \} \subseteq \{ B \} \) is the consequence rule:
  
  \[ \vdash A \quad \text{Pr} :: \vdash \{ A \} \subseteq \{ B \} \quad \vdash B \implies B \]

- From soundness of the first-order logic derivations we have \( \models A \implies A' \), hence \( \models A \)

- By i.h. with Pr and Op we get that \( \sigma' \models B' \)

- From soundness of the first-order logic derivations we have that \( \models B' \implies B \), hence \( \models B \)

Soundness of the Assignment Axiom

If \( \sigma \models A \), Op :: \( \langle \sigma, \sigma \uplus \sigma' \rangle \downarrow \sigma' \), and Pr :: \( \{ A \} \subseteq \{ B \} \), then \( \sigma' \models B \)

- Case: the last rule used in Pr :: \( \{ A \} \subseteq \{ B \} \) is the assignment rule:
  
  \[ \vdash \{ e/x \} B \quad \text{Op} :: \{ e \} \models e \quad \text{Pr} :: \vdash \{ A \} \subseteq \{ B \} \quad \vdash B \]

- The last rule used in Op :: \( \langle x : e, \sigma \uplus \sigma' \rangle \downarrow \sigma' \) must be
  
  \[ \text{Op} :: \{ e \} \models e \quad \text{Pr} :: \vdash \{ A \} \subseteq \{ B \} \]

- We must prove the substitution lemma:
  
  \[ \text{If } \models [e/x] B \text{ and } e, \sigma \uplus \sigma'[x := n] \text{ then } \sigma'[x := n] \models B \]
Soundness of the While Rule (1)

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

- Case: last rule used in \( \text{Pr} \): \( \{A\} c \{B\} \)
  - Two possible rules for the root of \( \text{Op} \)
    - We'll only do the complicated case:
      
      \[
      \sigma' \vdash A \leftarrow b
      \]
      
      To show: \( \sigma' \vdash A \land \neg b \)
      - By soundness of booleans and \( \text{Op}_2 \), we get \( \sigma \vdash b \)
        - Hence \( \sigma' \vdash A \land b \)
      - By i.h. on \( \text{Pr}_1 \) and \( \text{Op}_2 \), we get \( \sigma'' \vdash A \)
        - By i.h. on \( \text{Pr} \) and \( \text{Op}_3 \), we get \( \sigma'' \vdash A \land b \) (tricky!)

Soundness of the While Rule

- Note that in the last use of i.h., the derivation \( \text{Pr} \) did not decrease
- But \( \text{Op}_3 \) was a sub-derivation of \( \text{Op} \)

Completeness of Axiomatic Semantics

- If \( \vdash \{A\} c \{B\} \) can we always derive \( \vdash \{A\} c \{B\} \)?
- If so, axiomatic semantics is 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 complete
  - Bad news: only if the underlying logic is complete
  - (whenever \( \vdash A \) we also have \( \vdash A \))
  - This is called relative completeness