Announcements

- Homework 3 due Thu at 11:55pm

Questions?

Review of Induction

Induction on the Structure of Derivations

- To prove that for all derivations $D$ of a judgment, property $P$ holds

  - For each derivation rule of the form
    \[ \frac{H_1, H_2}{C} \]

  - Assume $P$ holds for derivations of $H_i$ ($i = 1..n$)

- Prove the property holds for the derivation obtained from the derivations of $H_i$ using the given rule
Notation: Naming Derivations

- Write $D :: \text{Judgment}$ to mean "$D$ is the derivation that proves \text{Judgment}"
- Example: $D :: \langle e_1 + e_2, \sigma \vdash n_1 + n_2 \rangle$

Proving Com Evaluation is Deterministic

If $D :: \langle c, \sigma \cup \alpha' \rangle$ and $D :: \langle c, \sigma \cup \alpha'' \rangle$, then $\alpha' = \alpha''$.

- Case $D :: \langle \text{skip}, \sigma \cup \alpha \rangle$

  - This means that $c = \text{skip}$ and $\alpha' = \alpha$.
  - By inversion, $D :: \langle c, \sigma \cup \alpha' \rangle$ uses the rule for \text{skip}.
  - Thus, $\alpha' = \alpha$.

This is a base case in the induction.

Proof: By induction on the structure of derivation $D$.

Proving Com Evaluation is Deterministic

If $D :: \langle c, \sigma \cup \alpha' \rangle$ and $D :: \langle c, \sigma \cup \alpha'' \rangle$, then $\alpha' = \alpha''$.

- Case $D :: \langle \text{while} b \text{ do } c, \sigma \cup \alpha \rangle$

  - By inversion, $D :: \langle \text{while} b \text{ do } c, \sigma \cup \alpha' \rangle$ uses the rule for \text{while}.

This is a simple inductive case.
Proving Com Evaluation is Deterministic

If \( D :: c, a \Downarrow a' \) and \( D' :: c, a \Downarrow a'' \), then \( a' = a'' \).

- Case
  - \( D :: b, \sigma \Downarrow \text{true} \)
  - \( D_1 :: c, \sigma \Downarrow \sigma' \)
  - \( D_2 :: c, \sigma \Downarrow \sigma'' \)
  - \( \sigma' = \sigma'' \)

Try to do this on a piece of paper. In a moment, we'll have some lucky winners come on down!

Summary: Induction on Derivations

- If you must prove \( \forall x \in A. P(x) \Rightarrow Q(x) \)
  - with \( A \) inductively defined and \( P(x) \) rule-defined
  - we pick arbitrary \( x \in A \) and \( D :: P(x) \)
  - we could do induction on both facts
    - \( x \in A \) leads to induction on the structure of \( x \)
    - \( D :: P(x) \) leads to induction on the structure of \( D \)
  - Generally, the induction on the structure of the derivation is more powerful and a safer bet
  - Sometimes there are many choices for induction
    - choosing the right one is a trial-and-error process
    - a bit of practice can help a lot

Summary: Operational Semantics

- Precise specification of dynamic semantics
  - order of evaluation (or that it doesn't matter)
  - error conditions (sometimes implicitly by rule applicability: "no applicable rule" = "get stuck")
- Simple and abstract (vs. implementations)
  - no low-level details such as stack and memory management, data layout, etc.
- Often not compositional (see while)
- Basis for many proofs about a language
  - Especially when combined with type systems!
- Basis for much reasoning about programs
- Point of reference for other semantics

Survey

- What is the most important thing you learned in today's class?
- What question about today's material is foremost in your mind?
- What feedback do you have about the course so far? What is going/not going well?