Proof Techniques for Operational Semantics
Meeting 10, CSCI 5535, Spring 2010

Announcements
• Homework 1 feedback/grades posted
• Homework 2 due tonight at 11:55pm

Questions?

Plan
• Why Bother?
• Mathematical Induction
• Well-Founded Induction
• Structural Induction
  - "By induction on the structure of the derivation D"

Mathematical Induction
• Goal: prove $\forall n \in \mathbb{N}. P(n)$
• Base Case: prove $P(0)$
• Inductive Step:
  - Prove "For all n, if $P(n)$, then $P(n+1)$"
  - "Pick arbitrary n, assume $P(n)$, prove $P(n+1)$"
• Why does induction work?

Example (With IMP Eval. Semantics)
• Prove that if $\sigma(x) \leq 6$ then
  $<\text{while } x \leq 5 \text{ do } x := x + 1, \sigma> \Downarrow \sigma[x := 6]$
• Reformulate the claim:
  - Let $W = \text{while } x \leq 5 \text{ do } x := x + 1$
  - Let $\sigma_0 = \sigma[x := 6 - i]$
  - Claim: $\forall i \in \mathbb{N}. <W, \sigma_0> \Downarrow \sigma_0$
• Now looks provable by mathematical induction on $i$
Evaluation Example (Base Case)

- Base case: \( i = 0 \) or \( \langle W, \sigma_0 \rangle \Downarrow \sigma_0 \)
  
  \[ \sigma_0(x) = 6 \]
  
  \[ <x, \sigma_0> \Downarrow \]
  
  \[ 6 \leq 5, \sigma_0 \Downarrow \]
  
  \[ \text{false} \]
  
  \[ <x \leq 5, \sigma_0> \Downarrow \]
  
  \[ \text{false} \]
  
  \[ <\text{while } x \leq 5 \text{ do } x := x + 1, \sigma_0 \rangle \Downarrow \sigma_0 \]

Evaluation Example (Inductive Case)

- Must prove for all \( i \in \mathbb{N} \).
  
  \[ \forall i \in \mathbb{N}, \langle W, \sigma_i \rangle \Downarrow \sigma_0 \]
  
  \( \Rightarrow \)
  
  \[ \langle W, \sigma_{i+1} \rangle \Downarrow \sigma_0 \]

Well-Founded Induction

- A relation \( \prec \subseteq A \times A \) is well-founded if there are no infinite descending chains in \( A \)
  
  - Example: \( \prec = \{ (x, x + 1) \mid x \in \mathbb{N} \} \)
    
    - aka the predecessor relation
  
  - Example: \( \prec = \{ (x, y) \mid x, y \in \mathbb{N} \text{ and } x < y \} \)

- Well-founded induction:
  
  - To prove \( \forall x \in A \). \( P(x) \) if \( \forall x \in A \). \( \forall y < x \). \( P(y) \) then prove \( \forall x \in A \). \( P(x) \)
  
  - If \( \prec \) is \( \prec_1 \) then we obtain mathematical induction as a special case

Well-Founded Induction: Examples

- Consider \( \prec \subseteq \mathbb{Z} \times \mathbb{Z} \) with \( x \prec y \) if \( y < x \).
  
  Says what?

  - Induction principle:
    
    \[ P(0) \wedge \forall x < 0, P(x) \Rightarrow P(x + 1) \wedge \forall x > 0, P(x) \Rightarrow P(x - 1) \]

- Consider \( \prec \subseteq (\mathbb{N} \times \mathbb{N}) \times (\mathbb{N} \times \mathbb{N}) \) and \( (x_1, y_1) \prec (x_2, y_2) \) if
  
  \[ x_2 > x_1 + 1 \vee (x_2 = x_1 + 1 \wedge y_2 > y_1 + 1) \]

  Says what?

  - Induction principle:
    
    \[ P(0,0) \wedge \forall x, y, P(x, y) \Rightarrow P(x + 1, y) \wedge P(x, y + 1) \]

This has a common name. Anyone see?
Structural Induction (on Expressions)

- For $e ::= n \mid x \mid e_1 + e_2 \mid e_1 \cdot e_2$
- Define $\rho \subseteq \text{Aexp} \times \text{Aexp}$ such that
  $e_1 \cdot e_2 < e_1 + e_2$
  no other elements of $\text{Aexp} \times \text{Aexp}$ are related by $<$
- To prove $\forall e \in \text{Aexp}. P(e)$
  - prove $\forall n \in \mathbb{Z}. P(n)$
  - prove $\forall x \in L. P(x)$
  - prove $\forall e_1, e_2 \in \text{Aexp}. P(e_1) \land P(e_2) \Rightarrow P(e_1 + e_2)$
  - prove $\forall e_1, e_2 \in \text{Aexp}. P(e_1) \land P(e_2) \Rightarrow P(e_1 \cdot e_2)$

Notes on Structural Induction

- Called structural induction because the proof is guided by the structure of the expression
- One proof case per form of expression
  - Atomic expressions (with no subexpressions) are all base cases
  - Composite expressions are the inductive cases
- Structural induction is the most useful form of induction in the study of PL

Example Proof Using Induction on the Structure of Expressions

- Let $L(e)$ be the # of literals and variable occurrences in $e$
  $O(e)$ be the # of operators in $e$
- Prove that $\forall e \in \text{Aexp}. L(e) = O(e) + 1$
- Proof: By induction on the structure of $e$.
  - Case $e = n$: $L(e) = 1$ and $O(e) = 0$
  - Case $e = x$: $L(e) = 1$ and $O(e) = 0$
  - Case $e = e_1 + e_2$:
    - $L(e) = L(e_1) + L(e_2)$ and $O(e) = O(e_1) + O(e_2) + 1$
    - By the induction hypothesis,
      $L(e_1) = O(e_1) + 1$ and $L(e_2) = O(e_2) + 1$
    - Thus, $L(e) = L(e_1) + L(e_2) + 2 = O(e) + 1$
  - Case $e = e_1 \cdot e_2$: Same as the case for $+$

“Try it at home!”

- Most proofs for the Aexp sublanguage of IMP can work by structural induction
- Small-step and big-step semantics obtain equivalent results:
  - How do we state this formally?
“Try it at home!”

- Most proofs for the Aexp sublanguage of IMP can work by structural induction
- Small-step and big-step semantics obtain equivalent results:
  \[ \text{For all } e \in \text{Aexp, for all } n \in \mathbb{Z}, \text{ for all } \sigma \in \Sigma, \]
  \[ \langle e, \sigma \rangle \rightarrow^* \langle n, \sigma \rangle \iff \langle e, \sigma \rangle \Downarrow n \]
- Structural induction on Aexp works here because all of the semantics are syntax-directed.

“Obvious, right?”

- You are given a concrete state \( \sigma \).
- You have \( \langle x + 1, \sigma \rangle \Downarrow 5 \)
- You also have \( \langle x + 1, \sigma \rangle \Downarrow 88 \)
- Is this possible?

Let’s make sure

- Prove that IMP is deterministic
  \[ \text{For all } e \in \text{Aexp, for all } \sigma \in \Sigma, \text{ for all } n, n' \in \mathbb{Z}, \]
  \[ \text{if } \langle e, \sigma \rangle \Downarrow n \text{ and } \langle e, \sigma \rangle \Downarrow n' \text{ then } n = n' \]
- For all \( b \in \text{Bexp}, \text{ for all } \sigma \in \Sigma, \text{ for all } t, t' \in \mathbb{B}, \]
  \[ \text{if } \langle b, \sigma \rangle \Downarrow t \text{ and } \langle b, \sigma \rangle \Downarrow t' \text{ then } t = t' \]
- For all \( c \in \text{Com}, \text{ for all } \sigma, \sigma', \sigma'' \in \Sigma, \]
  \[ \text{if } \langle c, \sigma \rangle \Downarrow \sigma' \text{ and } \langle c, \sigma \rangle \Downarrow \sigma'' \text{ then } \sigma' = \sigma'' \]

How do we prove it?

- Prove that IMP is deterministic
  - If \( e, \sigma \Downarrow n \text{ and } e, \sigma \Downarrow n' \text{ then } n = n' \)
  - If \( b, \sigma \Downarrow t \text{ and } b, \sigma \Downarrow t' \text{ then } t = t' \)
- No immediate way to use mathematical induction
- For commands we cannot use induction on the structure of the command
  - Can you imagine why?

Let’s make sure

- Prove that IMP is deterministic
  \[ \text{For all } e \in \text{Aexp, for all } \sigma \in \Sigma, \text{ for all } n, n' \in \mathbb{Z}, \]
  \[ \text{if } e, \sigma \Downarrow n \text{ and } e, \sigma \Downarrow n' \text{ then } n = n' \]
  \[ \text{For all } b \in \text{Bexp, for all } \sigma \in \Sigma, \text{ for all } t, t' \in \mathbb{B}, \]
  \[ \text{if } b, \sigma \Downarrow t \text{ and } b, \sigma \Downarrow t' \text{ then } t = t' \]
  \[ \text{For all } c \in \text{Com, for all } \sigma, \sigma', \sigma'' \in \Sigma, \]
  \[ \text{if } c, \sigma \Downarrow \sigma' \text{ and } c, \sigma \Downarrow \sigma'' \text{ then } \sigma' = \sigma'' \]

How do we prove it?

- Prove that IMP is deterministic
  - If \( e, \sigma \Downarrow n \text{ and } e, \sigma \Downarrow n' \text{ then } n = n' \)
  - If \( b, \sigma \Downarrow t \text{ and } b, \sigma \Downarrow t' \text{ then } t = t' \)
  - No immediate way to use mathematical induction
  - For commands we cannot use induction on the structure of the command
    - Can you see why?

\[ \langle b, \sigma \rangle \Downarrow \text{true} \]
\[ \langle c, \sigma \rangle \Downarrow \sigma' \]
\[ \text{while } b \text{ do } c, \sigma \Downarrow \sigma'' \]
\[ \text{while } b \text{ do } c, \sigma \Downarrow \sigma' \]
How do we prove it?

- Prove that IMP is deterministic
  If \(<e, \sigma> \Downarrow n\) and \(<e, \sigma> \Downarrow n'\) then \(n = n'\)
  If \(<e, \sigma> \Downarrow n'\) then \(e = e'\)
- No immediate way to use mathematical induction
- For commands we cannot use induction on the structure of the command
  while's evaluation does not depend only on the evaluation of its strict subexpressions
  \(<b, \sigma> \Downarrow \text{true} \quad \langle c, \sigma > \Downarrow \sigma'\>

We need something new!

Some more powerful form of induction
With all the bells and whistles!

Recall Proof Systems

- Operational semantics assigns meanings to programs by listing rules of inference that allow to prove judgments by constructing derivations.
- A derivation is a tree-structured object made up of valid instances of inference rules.

Find the keyword in the above. Ideas?

Induction on the Structure of Derivations

- Key idea: The hypothesis does not just assume a \(c \in \text{Com}\) but the existence of a derivation of \(<c, \sigma> \Downarrow \sigma'\>
- Derivation trees are also defined inductively, just like expression trees
- A derivation is built of subderivations
- Adapt the structural induction principle to work on the structure of derivations

Notation: Naming Derivations

- Write \(D :: \text{Judgment}\) to mean "\(D\) is the derivation that proves \(\text{Judgment}\)"
- Example:
  \[D :: <e_1 + e_2, \sigma> \Downarrow n_1 + n_2\]
Proving Com Evaluation is Deterministic

- Note: recall that meta-variables are universally-quantified (i.e., the above is for all $c \in \text{Com}, \sigma, \sigma', \sigma'' \in \Sigma$, derivations $D$ and $D'$.
- Start by picking each to be arbitrary.

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