

Computer Science 203
Programming Languages
Fall 2004 - Lecture 4

Due today:
Homework 2
Winskel Ch. 3+4

Cormac Flanagan
University of California, Santa Cruz

Review of Induction

- Mathematical induction
 - $\forall n \in \mathbb{N}. P(n)$
- Well-founded induction
 - For well-founded relation $< \subseteq A \times A$
 - To prove $\forall x \in A. P(x)$ it is enough to prove $\forall x \in A. [\forall y < x \Rightarrow P(y)] \Rightarrow P(x)$
- Structural induction over syntax
 - For each kind of expression e ,
 - assume P holds on all direct subtrees of e
 - prove that P holds on e

Limits of Structural Induction over Program Syntax

- Prove that IMP is deterministic
 - $\forall e \in Aexp. \forall \sigma \in \Sigma. \forall n, n' \in \mathbb{N}. \langle e, \sigma \rangle \Downarrow n \wedge \langle e, \sigma \rangle \Downarrow n' \Rightarrow n = n'$
 - $\forall b \in Bexp. \forall \sigma \in \Sigma. \forall t, t' \in \mathbb{B}. \langle b, \sigma \rangle \Downarrow t \wedge \langle b, \sigma \rangle \Downarrow t' \Rightarrow t = t'$
 - $\forall c \in Comm. \forall \sigma, \sigma', \sigma'' \in \Sigma. \langle c, \sigma \rangle \Downarrow \sigma' \wedge \langle c, \sigma \rangle \Downarrow \sigma'' \Rightarrow \sigma' = \sigma''$
- Cannot use induction on the structure of commands
 - Consider the rule for while. Its evaluation does not depend only on the evaluation of its strict subexpressions

$$\frac{\langle b, \sigma \rangle \Downarrow true \quad \langle c, \sigma \rangle \Downarrow \sigma' \quad \langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma''}{\langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma''}$$

Induction on the Structure of Derivations

- Key idea: The hypothesis gives not only a $c \in Comm$ but also the existence of a derivation of $\langle c, \sigma \rangle \Downarrow \sigma'$.
- Derivation trees are also defined inductively, just like expression trees.
- A derivation is built of subderivations:

$$\frac{\langle x, \sigma_{i+1} \rangle \Downarrow 5-i \quad 5-i \leq 5 \quad \frac{\langle x+1, \sigma_{i+1} \rangle \Downarrow 6-i}{\langle x:=x+1, \sigma_{i+1} \rangle \Downarrow \sigma_i} \quad \langle W, \sigma_i \rangle \Downarrow \sigma_0}{\langle x \leq 5, \sigma_{i+1} \rangle \Downarrow true \quad \langle x:=x+1; W, \sigma_{i+1} \rangle \Downarrow \sigma_0}}{\langle \text{while } x \leq 5 \text{ do } x := x+1, \sigma_{i+1} \rangle \Downarrow \sigma_0}$$
- We adapt the structural induction principle to work on the structure of derivations.

Induction on Derivations

- To prove that for all derivation D of a judgment, property $P(D)$ holds:
1. For each derivation rule of the form

$$\frac{H_1 \dots H_n}{C}$$
 2. Assume that P holds for a derivation of H_i ($i = 1, \dots, n$).
 3. Prove the the property holds for the derivation obtained from the derivations of H_i using the given rule.

Example of Induction on Derivations (I)

- Prove that evaluation of commands is deterministic:
 - $\langle c, \sigma \rangle \Downarrow \sigma'$ and $\langle c, \sigma \rangle \Downarrow \sigma'' \Rightarrow \sigma' = \sigma''$
- Pick arbitrary c, σ, σ' and $D :: \langle c, \sigma \rangle \Downarrow \sigma'$
- To prove: $\langle c, \sigma \rangle \Downarrow \sigma'' \Rightarrow \sigma' = \sigma''$
- Proof by induction on the structure of the derivation D
- Case: the last rule used in D was the one for skip:

$$D :: \frac{}{\langle \text{skip}, \sigma \rangle \Downarrow \sigma}$$
 - This means that $c = \text{skip}$, and $\sigma' = \sigma$.
 - Therefore $\langle c, \sigma \rangle \Downarrow \sigma''$ uses the rule for skip. Thus $\sigma'' = \sigma$.
 - This is a base case in the induction.

Example of Induction on Derivations (II)

- Case: the last rule used in D was the one for sequencing

$$D :: \frac{D_1 :: \langle c_1, \sigma \rangle \Downarrow \sigma_1 \quad D_2 :: \langle c_2, \sigma_1 \rangle \Downarrow \sigma'}{\langle c_1; c_2, \sigma \rangle \Downarrow \sigma'}$$

- Pick arbitrary σ'' such that $D'' :: \langle c_1; c_2, \sigma \rangle \Downarrow \sigma''$.
 - By inversion D'' uses the rule for sequencing
 - and has subderivations $D''_1 :: \langle c_1, \sigma \rangle \Downarrow \sigma''_1$ and $D''_2 :: \langle c_2, \sigma''_1 \rangle \Downarrow \sigma''$
- By induction hypothesis on D_1 (with D''_1): $\sigma_1 = \sigma''_1$
 - Now $D''_2 :: \langle c_2, \sigma_1 \rangle \Downarrow \sigma''$
- By induction hypothesis on D_2 (with D''_2): $\sigma'' = \sigma'$
- This is a simple inductive case.

Example of Induction on Derivations (III)

- Case: the last rule used in D was the one for while true

$$D :: \frac{D_1 :: \langle b, \sigma \rangle \Downarrow \text{true} \quad D_2 :: \langle c, \sigma \rangle \Downarrow \sigma_1 \quad D_3 :: \langle \text{while } b \text{ do } c, \sigma_1 \rangle \Downarrow \sigma'}{\langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma'}$$

- Pick arbitrary σ'' such that $D'' :: \langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma''$
 - By inversion and determinism of boolean expressions, D'' also uses the rule for while true
 - and has subderivations $D''_2 :: \langle c, \sigma \rangle \Downarrow \sigma''_1$ and $D''_3 :: \langle W, \sigma''_1 \rangle \Downarrow \sigma''$
- By induction hypothesis on D_2 (with D''_2): $\sigma_1 = \sigma''_1$
 - Now $D''_3 :: \langle \text{while } b \text{ do } c, \sigma_1 \rangle \Downarrow \sigma''$
- By induction hypothesis on D_3 (with D''_3): $\sigma'' = \sigma'$

Induction on Derivation: Notes

- If we have to 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 $D :: P(x)$
 - and 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.
- There are often several choices for induction.
 - Choosing the right one is a trial-and-error process.
 - A lot of practice can help a lot!

Equivalence

- Two expressions (commands) are equivalent if they yield the same result from all states

$$e_1 \approx e_2 \text{ iff } \forall \sigma \in \Sigma. \forall n \in \mathbb{Z}. \langle e_1, \sigma \rangle \Downarrow n \text{ iff } \langle e_2, \sigma \rangle \Downarrow n$$

and for commands

$$c_1 \approx c_2 \text{ iff } \forall \sigma, \sigma' \in \Sigma. \langle c_1, \sigma \rangle \Downarrow \sigma' \text{ iff } \langle c_2, \sigma \rangle \Downarrow \sigma'$$

Notes on Equivalence

- Equivalence is like validity:
 - It must hold in all states.
 - $2 \approx 1 + 1$ is like " $2 = 1 + 1$ is valid".
 - $2 \approx 1 + x$ might or might not hold.
 - So, 2 is not equivalent to $1 + x$
- Equivalence (for IMP) is undecidable.
 - If it were decidable we could solve the halting problem. (How?)
- Equivalence justifies code transformations:
 - compiler optimizations
 - code instrumentation
 - abstract modeling
- Semantics is the basis for proving equivalence.

Equivalence Examples

- skip; $c \approx c$
- $(x := e_1; x := e_2) \approx x := e_2$. (When is this true?)
- while b do $c \approx$ if b then c ; while b do c else skip
- If $e_1 \approx e_2$ then $x := e_1 \approx x := e_2$
- while true do skip \approx while true do $x := x + 1$
- If c is
 - while $x \neq y$ do
 - if $x \geq y$ then $x := x - y$ else $y := y - x$
- then $(x := 221; y := 527; c) \approx (x := 17; y := 17)$

Proving an Equivalence

- Prove that "skip; c \approx c" for all c.
- Assume that $D :: \langle \text{skip}; c, \sigma \rangle \Downarrow \sigma'$.
- By inversion (twice) we have that:

$$D :: \frac{\langle \text{skip}, \sigma \rangle \Downarrow \sigma \quad D_1 :: \langle c, \sigma \rangle \Downarrow \sigma'}{\langle \text{skip}; c, \sigma \rangle \Downarrow \sigma'}$$

- Thus, we have $D_1 :: \langle c, \sigma \rangle \Downarrow \sigma'$.
- The other direction is similar.

Proving an Inequivalence

- Prove that $x := y \not\approx x := z$ when $y \neq z$.
- It suffices to produce a *witness* store σ in which the two commands yield different results
- Let $\sigma(y) = 0$ and $\sigma(z) = 1$.
- Then $\langle x := y, \sigma \rangle \Downarrow \sigma[x := 0]$
- and $\langle x := z, \sigma \rangle \Downarrow \sigma[x := 1]$.

Homework 3 (for Tuesday October 11)

- Read a small example proof by induction, by Faron Moller:
<http://www.soe.ucsc.edu/classes/cms203/Fall05/mollerproof.ps>
1. In IMP, prove or disprove the equivalence of the two commands:
 $t := x; x := y; y := t$
and
 $t := y; y := x; x := t$
(where x, y, and t are distinct locations).
 2. In IMP, prove that if:
 $\langle \text{while } b \text{ do } y := y - x, \sigma \rangle \Downarrow \sigma'$
then there exists an integer k such that:
 $\sigma'(y)$ equals $\sigma(y)$ plus k times $\sigma(x)$.
(Please make it explicit if/when you reason by induction on derivations, stating your induction hypothesis.)