

# Computer Science 203 Programming Languages Fall 2005 - Lecture 3

## Some Proof Techniques for Language Analysis

Cormac Flanagan

University of California, Santa Cruz

## Induction

- Probably the single most important technique for the study of formal semantics and type systems of programming languages.
- Of several kinds
  - mathematical induction (the simplest)
  - well-founded induction (the most general)
  - structural induction (the most widely used in this context)

## Mathematical Induction

- Goal: prove that  $\forall n \in \mathbb{N}. P(n)$
- Strategy: (2 steps)
  - Base case: prove that  $P(0)$
  - Inductive case:
    - pick an arbitrary  $n \in \mathbb{N}$
    - assume that  $P(n)$  holds
    - prove that  $P(n+1)$
    - or, formally, prove that  $\forall n \in \mathbb{N}. P(n) \Rightarrow P(n+1)$

## Mathematical Induction: Notes

- The inductive step looks similar to the goal but it is simpler because of the assumption that  $P(n)$  holds.
 
$$\forall n \in \mathbb{N}. P(n-1) \Rightarrow P(n) \quad \text{vs.} \quad \forall n \in \mathbb{N}. P(n)$$
- Why does mathematical induction work?
  - The key property of  $\mathbb{N}$  is that there are no infinite descending chains of naturals.
  - For each  $n$ ,  $P(n)$  can be obtained from the base case and  $n$  uses of the inductive case.

## Example of Mathematical Induction

- Recall the evaluation rules for IMP commands.
- Prove that if  $\sigma(x) \leq 6$  then
 
$$\langle \text{while } x \leq 5 \text{ do } x := x + 1, \sigma \rangle \Downarrow \sigma[x := 6]$$
- Reformulate the claim:
  - Let  $W = \text{while } x \leq 5 \text{ do } x := x + 1$
  - Let  $\sigma_i = \sigma[x := 6 - i]$
  - Claim:  $\forall i \in \mathbb{N}. \langle W, \sigma_i \rangle \Downarrow \sigma_0$
- Now the claim looks provable by mathematical induction on  $i$ .

## Example of Mathematical Induction (Base Case)

- Base case:  $i = 0$  or  $\langle W, \sigma_0 \rangle \Downarrow \sigma_0$ 
  - To prove an evaluation judgment, construct a derivation tree:

$$\frac{\sigma_0(x) = 6}{\langle x, \sigma_0 \rangle \Downarrow 6} \quad \frac{\langle x, \sigma_0 \rangle \Downarrow 6 \quad \langle 5, \sigma_0 \rangle \Downarrow 5}{\langle x \leq 5, \sigma_0 \rangle \Downarrow \text{false}} \quad \frac{\langle x \leq 5, \sigma_0 \rangle \Downarrow \text{false}}{\langle \text{while } x \leq 5 \text{ do } x := x + 1, \sigma_0 \rangle \Downarrow \sigma_0}$$

- This completes the base case.



## Structural Induction

- Recall  $e ::= n \mid e_1 + e_2 \mid e_1 * e_2 \mid x$
- Define  $\prec \subseteq \text{Aexp} * \text{Aexp}$  such that
  - $e_1 \prec e_1 + e_2$
  - $e_2 \prec e_1 + e_2$
  - $e_1 \prec e_1 * e_2$
  - $e_2 \prec e_1 * e_2$
- and no other elements of  $\text{Aexp} * \text{Aexp}$  are related by  $\prec$
- 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  $\forall e_1, e_2 \in \text{Aexp}. P(e_1) \wedge P(e_2) \Rightarrow P(e_1 + e_2)$
  - Prove  $\forall e_1, e_2 \in \text{Aexp}. P(e_1) \wedge P(e_2) \Rightarrow P(e_1 * e_2)$

CMPS201 Lecture 3

13

## Structural Induction: Notes

- It is called structural induction because proofs are guided by the structure of the expression.
- In a proof, there are as many cases as there are expression forms:
  - Atomic expressions (with no subexpressions) are all base cases.
  - Composite expressions are the inductive cases.
- This is the most useful form of induction in the study of programming languages.

CMPS201 Lecture 3

14

## Example of Induction on Structure of Expressions

- Let
  - $L(e)$  be the number of literals and variable occurrences in  $e$
  - $O(e)$  be the number of operators in  $e$
- Prove that  $\forall e \in \text{Aexp}. L(e) = O(e) + 1$
- 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 induction hypothesis  $L(e_1) = O(e_1) + 1$  and  $L(e_2) = O(e_2) + 1$
    - Thus  $L(e) = O(e) + 1$
  - Case  $e = e_1 * e_2$ : same as the case for  $+$ .

CMPS201 Lecture 3

15

## Another Proof

- Prove that IMP is deterministic
  - $\forall e \in \text{Aexp}. \forall \sigma \in \Sigma. \forall n, n' \in \mathbb{N}. \langle e, \sigma \rangle \Downarrow n \wedge \langle e, \sigma \rangle \Downarrow n' \Rightarrow n = n'$
  - $\forall b \in \text{Bexp}. \forall \sigma \in \Sigma. \forall t, t' \in \mathbb{B}. \langle b, \sigma \rangle \Downarrow t \wedge \langle b, \sigma \rangle \Downarrow t' \Rightarrow t = t'$
  - $\forall c \in \text{Comm}. \forall \sigma, \sigma', \sigma'' \in \Sigma. \langle c, \sigma \rangle \Downarrow \sigma' \wedge \langle c, \sigma \rangle \Downarrow \sigma'' \Rightarrow \sigma' = \sigma''$
- No immediate way to use mathematical induction
- For commands we cannot use induction on the structure of the command
  - Consider the rule for while. Its evaluation does not depend only on the evaluation of its strict subexpressions

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

CMPS201 Lecture 3

16

## Induction on the Structure of Derivations

- Key idea: The hypothesis gives not only a  $c \in \text{Comm}$  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:
 
$$\frac{\langle x, \sigma_{i+1} \rangle \Downarrow 5 - i \quad 5 - i \leq 5 \quad \frac{\langle x + 1, \sigma_{i+1} \rangle \Downarrow 6 - i}{\langle x := x + 1, \sigma_{i+1} \rangle \Downarrow \sigma_i} \quad \langle W, \sigma_i \rangle \Downarrow \sigma_0}{\langle x \leq 5, \sigma_{i+1} \rangle \Downarrow \text{true} \quad \langle x := x + 1; W, \sigma_{i+1} \rangle \Downarrow \sigma_0}}{\langle \text{while } x \leq 5 \text{ do } x := x + 1, \sigma_{i+1} \rangle \Downarrow \sigma_0}$$
- We adapt the structural induction principle to work on the structure of derivations.

CMPS201 Lecture 3

17