CS 611
Advanced Programming Languages

Andrew Myers
Cornell University

Lecture 5: Inductive proofs
4 Sep 00

Administration

- Homework 1 due September 11

Equivalence of expressions

- Last time: equivalence of two semantics for same language
- What about equivalence of two expressions in language?
  - IMP: expressions are commands, arithmetic, boolean exprs
  - Useful for program transformations
- Idea: programs *observationally* equivalent if they permit the same executions
- Example: x := y + y ~ x := 2*y ; z := z

Formalizing Equivalence

- Program equivalence:
  $c_1 \sim c_2 \Leftrightarrow \forall\sigma.\ (\langle c_1, \sigma\rangle \Downarrow \sigma' \Leftrightarrow \langle c_2, \sigma\rangle \Downarrow \sigma')$
- *Expressions* e_1, e_2 are observationally equivalent if every program containing one (e.g., e_1) is equivalent to the same program with the other (e.g., e_2) substituted for it
  - Let C[ ] be an expression *context*: any program with a *hole* [ ] where an expression can go
  - Example: x := 0; while x < 10 do [ ]
  - Let $C[e_i]$ be the program with e_i instead of hole
  - $e_1 \sim e_2 \Leftrightarrow \forall C[\,].\ C[e_1] \sim C[e_2]$
  - IMP: two notions of equivalence identical for commands (not true for all languages)

Contexts

- To capture idea of "all contexts", define command context *C*[ ] with BNF:

C[ ] ::= [ ] | *C*[ ] **;** *c* | *c* **;** *C*[ ]
  | **if** *b* **then** *C*[ ] **else** *c*
  | **if** *b* **then** *c* **else** *C*[ ]
  | **while** *b* **do** *C*[ ]

- Can use inductive definition of context to construct proofs of expression equivalence

Inductive proofs

- Some things we'd like to prove
  - equivalence of different semantics
    - small-step vs. large-step
  - equivalence of different expressions
    - c; while ¬b do c vs. do c until b
  - termination of expressions
  - deterministic evaluation of expressions, programs
- In general, need inductive proofs

Proving termination

- Assertion: Arithmetic expressions always terminate: $\exists n.\langle a, \sigma\rangle \to^* \langle n, \sigma\rangle$
- An argument:
  - Expressions of the form X or n always terminate in one step (evaluation defined by axioms)
  - Expressions of the form $a_1 + a_2$, $a_1 \times a_2$, $a_1 - a_2$ terminate if their constituent expressions a_1, a_2 terminate
- Problem: circular!

Ordinary Induction

- Mathematical induction: a property P(n) holds for all $n \geq 1$ if

  P(1) (base case)

  $\forall_{n\geq 1}$ P(n) $\Rightarrow$ P(n+1) (inductive step)
- Inductive hypothesis: P(n)
- Strategy:
  1. prove base case
  2. show P(n+1) is true if inductive hypothesis P(n) holds

Course-of-values induction

- Course-of-values induction: a property P(n) holds for all $n \geq 1$ if

  P(1) (base case)

  $\forall_{n\geq 1} (\forall_{n'\in 1..n} \mathrm{P}(n')) \Rightarrow \mathrm{P}(n+1)$ (inductive step)
- Inductive hypothesis: $\forall_{n'\in 1..n} \mathrm{P}(n')$
- Often easier to prove

Soundness

- Course-of-values rule:

$$\frac{\mathrm{P}(1) \qquad \forall_{n\geq 1} (\forall_{n'\in 1..n} \mathrm{P}(n')) \Rightarrow \mathrm{P}(n+1)}{\forall_{n\geq 1} \mathrm{P}(n)}$$

- Idea: introduce new predicate P'(n):

$$\mathrm{P'}(n) = \forall_{n'\in 1..n} \mathrm{P}(n')$$

- Lemmas: P'(n) $\Rightarrow$ P(n), P(1) $\Rightarrow$ P'(1)

Proof via ordinary induction

$$\frac{\forall_{n\geq 1} (\forall_{n'\in 1..n} \mathrm{P}(n')) \Rightarrow \mathrm{P}(n+1)}{\forall_{n\geq 1} (\forall_{n'\in 1..n} \mathrm{P}(n')) \Rightarrow (\mathrm{P}(n+1) \wedge (\forall_{n'\in 1..n} \mathrm{P}(n')))}$$

$$\frac{\dfrac{\mathrm{P}(1)}{\mathrm{P'}(1)} \qquad \dfrac{\forall_{n\geq 1} (\forall_{n'\in 1..n} \mathrm{P}(n')) \Rightarrow (\forall_{n'\in 1..n+1} \mathrm{P}(n'))}{\forall_{n\geq 1} \mathrm{P'}(n) \Rightarrow \mathrm{P'}(n+1)}}{\dfrac{\forall_{n\geq 1} \mathrm{P'}(n)}{\forall_{n\geq 1} \mathrm{P}(n)}}$$

Structural Induction

- Property **P**(e) holds for all exprs e if
  - **P**(e) holds for all expression forms e with no sub-expressions (e.g. n, X)
  - Given expression form e with sub-expressions e_i (e.g., $a_0 + a_1$), **P**(e) holds assuming **P**(e_i) holds for all e_i
- $\mathbf{P}(a_0+a_1) = \exists n.\langle a_0+a_1, \sigma\rangle \to^* \langle n, \sigma\rangle$
- Assume:

  $\exists n_0.\langle a_0, \sigma\rangle \to^* \langle n_0, \sigma\rangle \Rightarrow \exists n_0.\langle a_0+a_1, \sigma\rangle \to^* \langle n_0+a_1, \sigma\rangle$

  $\exists n_1.\langle a_1, \sigma\rangle \to^* \langle n_1, \sigma\rangle \Rightarrow \exists n_1.\langle n_0+a_1, \sigma\rangle \to^* \langle n_0+n_1, \sigma\rangle$

  (axiom) $\exists n.\langle n_0+n_1, \sigma\rangle \to \langle n, \sigma\rangle$

  ($\to^*$ lemmas) $\exists n.\langle a_0+a_1, \sigma\rangle \to^* \langle n, \sigma\rangle$ $\therefore$

Alternative: course-of-values

- Use course-of-values induction on size of expression (height of abstract syntax)
- P(n) is "all expressions of size n terminate"
- P(1) clearly true (n, X)
- Induction step: prove $a_0 + a_1$ of size n terminates
- Induction hypothesis: a_0, a_1 terminate (must be smaller than n)

Induction on Derivations

- Sometimes proof requires induction on height of derivation
- Example: commands in IMP are deterministic
- Want to show:

$$\forall \sigma, \sigma_1, \sigma'_1, c\ .$$
$$(\langle c, \sigma\rangle \Downarrow \sigma_1\ \&\ \langle c, \sigma\rangle \Downarrow \sigma'_1 \Rightarrow \sigma_1 = \sigma'_1)$$

Proof of Determinism

- Every command that terminates has a large-step semantics derivation (proof tree) with finite height
- Height of derivation tree is longest chain from conclusion (root) to any axiom (leaf)
- Let P(n) be statement "all commands whose derivation has height n are deterministic"

$$P(n) = \forall d, d'\ .\ \Big(\text{height}(d) = n\ \&\ d = \frac{\cdots}{\langle c, \sigma\rangle \Downarrow \sigma_1}\ \&$$
$$d' = \frac{\cdots}{\langle c, \sigma\rangle \Downarrow \sigma'_1}\Big) \Rightarrow \sigma_1 = \sigma'_1$$

Base Case

- Statement about derivations ($\forall n$) implies desired statement about commands
- P(1): skip, $X := a$
- Inductive step: consider derivations of $\langle c, \sigma\rangle \Downarrow \sigma_1$ with height n for commands ; , if, while

Inductive step for ;

- Now suppose d is derivation of $\langle c_0 ; c_1, \sigma\rangle \Downarrow \sigma_1$ with height n, d' derivation of $\langle c_0 ; c_1, \sigma\rangle \Downarrow \sigma'_1$
- Inductive hypothesis: $\sigma_2 = \sigma'_2$, then $\sigma_1 = \sigma'_1$

$$d = \dfrac{\dfrac{\cdots}{\langle c_0, \sigma\rangle \Downarrow \sigma_2} \quad \dfrac{\cdots}{\langle c_1, \sigma_1\rangle \Downarrow \sigma_1}}{\langle c_0 ; c_1, \sigma\rangle \Downarrow \sigma_1}$$

height < n

height = n

$$d' = \dfrac{\dfrac{\cdots}{\langle c_0, \sigma\rangle \Downarrow \sigma'_2} \quad \dfrac{\cdots}{\langle c_1, \sigma'_1\rangle \Downarrow \sigma'_1}}{\langle c_0 ; c_1, \sigma'\rangle \Downarrow \sigma'_1}$$

Inductive step for if

- Assume $\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \Downarrow \sigma_1$, $\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \Downarrow \sigma'_1$
- Assume booleans are deterministic: b evaluates the same way for both
- WLOG derivations look like

$$\dfrac{\dfrac{\cdots}{\langle b, \sigma\rangle \Downarrow \text{true}} \quad \dfrac{\cdots}{\langle c_0, \sigma\rangle \Downarrow \sigma_1}}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \Downarrow \sigma_1} \qquad \dfrac{\dfrac{\cdots}{\langle b, \sigma\rangle \Downarrow \text{true}} \quad \dfrac{\cdots}{\langle c_0, \sigma\rangle \Downarrow \sigma'_1}}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \Downarrow \sigma'_1}$$

- Sub-derivations: (height < n) $\dfrac{\cdots}{\langle c_0, \sigma\rangle \Downarrow \sigma'}$ $\dfrac{\cdots}{\langle c_0, \sigma\rangle \Downarrow \sigma''}$
- Therefore, $\sigma' = \sigma''$

Inductive step for while

- Assume $\langle \text{while } b \text{ do } c, \sigma\rangle \Downarrow \sigma_1$, $\langle \text{while } b \text{ do } c, \sigma\rangle \Downarrow \sigma'_1$
- Derivations look like

$$\frac{\dfrac{\ldots}{\langle b, \sigma\rangle \Downarrow \text{true}} \quad \dfrac{\ldots}{\langle c, \sigma\rangle \Downarrow \sigma_2} \quad \dfrac{\ldots}{\langle \text{while } b \text{ do } c, \sigma_2\rangle \Downarrow \sigma_1}}{\langle \text{while } b \text{ do } c, \sigma\rangle \Downarrow \sigma_1}$$

$$\frac{\dfrac{\ldots}{\langle b, \sigma\rangle \Downarrow \text{true}} \quad \dfrac{\ldots}{\langle c, \sigma\rangle \Downarrow \sigma'_2} \quad \dfrac{\ldots}{\langle \text{while } b \text{ do } c, \sigma'_2\rangle \Downarrow \sigma'_1}}{\langle \text{while } b \text{ do } c, \sigma\rangle \Downarrow \sigma'_1}$$

while

- Assume $\langle \text{while } b \text{ do } c, \sigma\rangle \Downarrow \sigma_1$, $\langle \text{while } b \text{ do } c, \sigma\rangle \Downarrow \sigma_2$
- Derivations look like

$\sigma_2 = \sigma'_2$, $\sigma_1 = \sigma'_1$

$$\frac{\dfrac{\ldots}{\langle b, \sigma\rangle \Downarrow \text{true}} \quad \dfrac{\ldots}{\langle c, \sigma\rangle \Downarrow \sigma_2} \quad \dfrac{\ldots}{\langle \text{while } b \text{ do } c, \sigma_2\rangle \Downarrow \sigma_1}}{\langle \text{while } b \text{ do } c, \sigma\rangle \Downarrow \sigma_1}$$

$$\frac{\dfrac{\ldots}{\langle b, \sigma\rangle \Downarrow \text{true}} \quad \dfrac{\ldots}{\langle c, \sigma\rangle \Downarrow \sigma'_2} \quad \dfrac{\ldots}{\langle \text{while } b \text{ do } c, \sigma'_2\rangle \Downarrow \sigma'_1}}{\langle \text{while } b \text{ do } c, \sigma\rangle \Downarrow \sigma'_1}$$

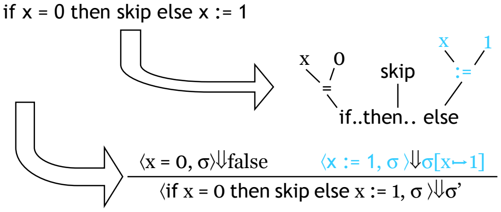
Induction

- Structural induction:
  - prove that a property holds of all language atoms
  - prove that it holds for each kind of expression if it holds of the parts of the expression
  - ⇒ property holds for *all* expressions
- Induction on derivations
  - prove it holds for derivations that are axioms
  - prove property holds if it holds for every *derivation* (evaluation*)* of parts of an expression
  - ⇒ property holds for *all* derivations

Observation

- These two forms of induction are very similar — both operate on *inductively defined sets* (syntax, evaluations)

if x = 0 then skip else x := 1



$$\frac{\langle x = 0, \sigma\rangle \Downarrow \text{false} \qquad \langle x := 1, \sigma\rangle \Downarrow \sigma[x \mapsto 1]}{\langle \text{if } x = 0 \text{ then skip else } x := 1, \sigma\rangle \Downarrow \sigma'}$$

Expression inference rules

BNF spec for arithmetic expressions in IMP:

$$a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

Let A be the set of all arithmetic expressions.
Inductive definition of A via inference rules:

Axioms: $\dfrac{}{n}$ $\qquad \dfrac{}{X}$

Rules: $\dfrac{a_0 \quad a_1}{a_0 + a_1}$ $\qquad \dfrac{a_0 \quad a_1}{a_0 - a_1}$ $\qquad \dfrac{a_0 \quad a_1}{a_0 \times a_1}$

Expression derivation tree

- Every legal expression now has a derivation tree.
- Example: (2+3) × (4-x)

$$\frac{\dfrac{2 \quad 3}{2+3} \qquad \dfrac{4 \quad x}{4 - x}}{(2+3) \times (4 - x)}$$

- Structural induction is induction on syntactic derivations!

Summary

- Any proof system (inference rules) is an inductive definition of a set
- Rule induction can be applied to any inductive definition
- Examples: structural induction, induction on derivations are both instances of this approach
- We will use rule induction for other proof systems in course (*e.g.*, type-checking rules)