Harvard School of Engineering and Applied Sciences — CS 152: Programming Languages

Induction; Small-step operational semantics; Large-step operational semantics; IMP Section and Practice Problems

Feb 11-12, 2016

---

1 Induction

Let's inductively define a set of integers **Quux** with the following inference rules.

$$\text{RULE1}\;\frac{}{8 \in \textbf{Quux}} \qquad \text{RULE2}\;\frac{}{5 \in \textbf{Quux}} \qquad \text{RULE3}\;\frac{a \in \textbf{Quux} \quad b \in \textbf{Quux}}{c \in \textbf{Quux}}\; c = a + b + 1$$

(a) Of the rules above (i.e., RULE1, RULE2, and RULE3), which are axioms and which are inductive rules?

(b) Give a derivation showing that 11 is in the set **Quux**.

(c) Give a derivation showing that 20 is in the set **Quux**.

(d) Write down the inductive reasoning principle for **Quux**. That is, if you wanted to prove that for some property P, for all $a \in \textbf{Quux}$ we have $P(a)$, what would you need to show? (See Lecture 2 §5.6 and Lecture 3 §1.2.)

(e) Prove that for all $a \in \textbf{Quux}$, there exists $i \in \mathbb{Z}$ such that $a = 3 \times i - 1$.

Make sure that you follow the Recipe for Inductive Proofs! See Lecture 3 §1.2. What set are you inducting on? What is the property you are trying to prove? Go through each case.

(f) Is 2 in the set **Quux**? If so, give a derivation proving it.

2 Small-step operational semantics

Consider the small-step operational semantics for the language of arithmetic expressions (Lectures 1 and 2). Let σ_0 be a store that maps all program variables to zero.

(a) Show a derivation that $\langle 3 + (5 \times \mathsf{bar}), \sigma_0 \rangle \longrightarrow \langle 3 + (5 \times 0), \sigma_0 \rangle$.

(b) What is the sequence of configurations that $\langle \mathsf{foo} := 5;\ (\mathsf{foo} + 2) \times 7, \sigma_0 \rangle$ steps to? (You don't need to show the derivations for each step, just show what configuration $\langle \mathsf{foo} := 5;\ (\mathsf{foo} + 2) \times 7, \sigma_0 \rangle$ steps to in one step, then two steps, then three steps, and so on, until you reach a final configuration.)

(c) Find an integer n and store σ' such that $\langle ((6 + (\mathsf{foo} := (\mathsf{bar} := 3; 5); 1 + \mathsf{bar})) + \mathsf{bar}) \times \mathsf{foo}, \sigma_0 \rangle \longrightarrow^* \langle n, \sigma' \rangle$.

(d) Is the relation $\longrightarrow$ reflexive? Is it symmetric? Is it anti-symmetric? Is it transitive?

(For each of these questions, if the answer is "no", what is a suitable counterexample? If any of the answers are "yes", think about how you would prove it.)

3 Large-step operational semantics

Consider the large-step operational semantics for the language of arithmetic expressions (Lectures 3 and 4). Let σ_0 be a store that maps all program variables to zero.

(a) Show a derivation that $\langle 3 + (5 \times \mathsf{bar}), \sigma_0 \rangle \Downarrow \langle 3, \sigma_0 \rangle$.

(b) Find an integer n and store σ' such that $\langle \mathsf{foo} := 5;\ (\mathsf{foo} + 2) \times 7, \sigma_0 \rangle \Downarrow \langle n, \sigma' \rangle$.

If you have time and a big piece of paper, give the derivation of $\langle \mathsf{foo} := 5;\ (\mathsf{foo} + 2) \times 7, \sigma_0 \rangle \Downarrow \langle n, \sigma' \rangle$.

(c) Is the relation $\Downarrow$ reflexive? Is it symmetric? Is it anti-symmetric? Is it transitive?

(For each of these questions, if the answer is "no", what is a suitable counterexample? If any of the answers are "yes", think about how you would prove it.)

4 IMP

Consider the small-step operational semantics for IMP given in Lecture 5. Let σ_0 be a store that maps all program variables to zero.

(a) Find a configuration $\langle c, \sigma' \rangle$ such that $\langle \textbf{if } 8 < 6 \textbf{ then } \mathsf{foo} := 2 \textbf{ else } \mathsf{bar} := 8, \sigma_0 \rangle \longrightarrow \langle c, \sigma' \rangle$ and give a derivation showing that $\langle \textbf{if } 8 < 6 \textbf{ then } \mathsf{foo} := 2 \textbf{ else } \mathsf{bar} := 8, \sigma_0 \rangle \longrightarrow \langle c, \sigma' \rangle$.

(b) What is the sequence of configurations that

$$\langle \mathsf{foo} := \mathsf{bar} + 3; \textbf{if } \mathsf{foo} < \mathsf{bar} \textbf{ then skip else } \mathsf{bar} := 1, \sigma_0 \rangle$$

steps to? (You don't need to show the derivations for each step, just show what configuration $\langle \mathsf{foo} := \mathsf{bar} + 3; \textbf{if } \mathsf{foo} < \mathsf{bar} \textbf{ then skip else } \mathsf{bar} := 1, \sigma_0 \rangle$ steps to in one step, then two steps, then three steps, and so on, until you reach a final configuration.)

Now consider the large-step operational semantics for IMP given in Lecture 5. Let σ_0 be a store that maps all program variables to zero.

(c) Find a store σ' such that $\langle \textbf{while } \mathsf{foo} < 3 \textbf{ do } \mathsf{foo} := \mathsf{foo} + 2, \sigma_0 \rangle \Downarrow \sigma'$ and give a derivation showing that $\langle \textbf{while } \mathsf{foo} < 3 \textbf{ do } \mathsf{foo} := \mathsf{foo} + 2, \sigma_0 \rangle \Downarrow \sigma'$.

(d) Suppose we extend boolean expressions with negation.

$$b ::= \cdots \mid \textbf{not } b$$

i. Give an inference rule or inference rules that show the (large step) evaluation of **not** b.

ii. Show that **if** b **then** c_1 **else** c_2 is equivalent to **if not** b **then** c_2 **else** c_1. (See Lecture 5 §2.1.)