1 Induction

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

R\text{ULE} 1 \quad 8 \in \text{Quux} \\
R\text{ULE} 2 \quad 5 \in \text{Quux} \\
R\text{ULE} 3 \quad a \in \text{Quux} \quad b \in \text{Quux} \quad c = a + b + 1

(a) Of the rules above (i.e., R\text{ULE} 1, R\text{ULE} 2, and R\text{ULE} 3), 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 ∈ 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 ∈ Quux, there exists i ∈ \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 \sigma_0 be a store that maps all program variables to zero.

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

(b) What is the sequence of configurations that \langle \text{foo} := 5; (\text{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 \text{foo} := 5; (\text{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 \sigma' such that \langle ((6+(\text{foo} := (\text{bar} := 3:5);1+\text{bar}))+\text{bar}) \times \text{foo}, \sigma_0 \rangle \rightarrow^* \langle n, \sigma' \rangle.

(d) Is the relation \rightarrow 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 \sigma_0 be a store that maps all program variables to zero.

(a) Show a derivation that \langle 3 + (5 \times \text{bar}), \sigma_0 \rangle \Downarrow \langle 3, \sigma_0 \rangle.
(b) Find an integer \( n \) and store \( \sigma' \) such that \( \langle \text{foo} := 5; (\text{foo} + 2) \times 7, \sigma_0 \rangle \Downarrow \langle n, \sigma' \rangle \).

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

(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 \( \sigma_0 \) be a store that maps all program variables to zero.

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

(b) What is the sequence of configurations that

\( \langle \text{foo := bar + 3; if foo < bar then skip else bar := 1, } \sigma_0 \rangle \)

steps to? (You don’t need to show the derivations for each step, just show what configuration \( \langle \text{foo := bar + 3; if foo < bar then skip else bar := 1, } \sigma_0 \rangle \) 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 \( \sigma_0 \) be a store that maps all program variables to zero.

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

(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 \( \textbf{not } b \).

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