

Week 7: Proof by Induction & Operational Semantics

1 Proof by Induction

- * 1. Consider the exponential function for natural numbers with the following recursive definition:

$$\begin{aligned}x^0 &= 1 \\ x^{n+1} &= x \cdot x^n\end{aligned}$$

Prove by induction that $(x \cdot y)^z = x^z \cdot y^z$ for any $x, y, z \in \mathbb{N}$. You may assume that multiplication satisfies the usual laws of associativity and commutativity.

- ** 2. The *height* of an arithmetic expression is defined recursively as follows:

$$\begin{aligned}\text{height}(n) &= 1 \\ \text{height}(x) &= 1 \\ \text{height}(e_1 + e_2) &= 1 + \max\{\text{height}(e_1), \text{height}(e_2)\} \\ \text{height}(e_1 - e_2) &= 1 + \max\{\text{height}(e_1), \text{height}(e_2)\} \\ \text{height}(e_1 * e_2) &= 1 + \max\{\text{height}(e_1), \text{height}(e_2)\}\end{aligned}$$

- (a) Prove by structural induction over arithmetic expressions that $\text{height}(e) > 0$ for all arithmetic expressions $e \in \mathcal{A}$.
- (b) Prove by structural induction over arithmetic expressions that $2^{\text{height}(e)-1} \geq \#FV(e)$ for all arithmetic expressions $e \in \mathcal{A}$ where $\#FV(e)$ is the number of free variables appearing in that expression.

Hint: Try using the facts that, if $x \geq 2y$, $2z$, then $x \geq y + z$, and that $\#A + \#B \geq \#(A \cup B)$.

- ** 3. If x is a variable and e_1 and e_2 are arithmetic expressions, then we write $e_1[x \mapsto e_2]$ for the expression that results from *substituting* e_2 for x in the expression e_1 . Formally, this operation it

is defined by recursion over the expression e_1 as follows:

$$\begin{aligned}
 n[x \mapsto e] &= n \\
 y[x \mapsto e] &= \begin{cases} e & \text{if } x = y \\ y & \text{otherwise} \end{cases} \\
 (e_1 + e_2)[x \mapsto e] &= e_1[x \mapsto e] + e_2[x \mapsto e] \\
 (e_1 - e_2)[x \mapsto e] &= e_1[x \mapsto e] - e_2[x \mapsto e] \\
 (e_1 * e_2)[x \mapsto e] &= e_1[x \mapsto e] * e_2[x \mapsto e]
 \end{aligned}$$

- (a) Compute the value of the expression $(y - x)[x \mapsto z]$ in the state $[x \mapsto 1, y \mapsto 2, z \mapsto 3]$.
- (b) Find a state σ such that $\llbracket y - x \rrbracket_{\mathcal{A}}(\sigma)$ evaluates to the same answer you got in part (a). What is the relationship between this state and the state $[x \mapsto 1, y \mapsto 2, z \mapsto 3]$?
- (c) Prove by structural induction over expressions, for any state $\sigma \in \text{State}$, any pair of arithmetic expressions $e_1, e_2 \in \mathcal{A}$ and any variable $x \in \text{Var}$, we have that:

$$\llbracket e_1[x \mapsto e_2] \rrbracket_{\mathcal{A}}(\sigma) = \llbracket e_1 \rrbracket_{\mathcal{A}}(\sigma[x \mapsto \llbracket e_2 \rrbracket_{\mathcal{A}}(\sigma)]).$$

Remember that e_1 may be an *arbitrary* variable.

- ** 4. Write down the induction principle for Boolean expressions. Try to generalise from the induction principle for arithmetic expressions as it appears in the reference notes (<https://uob-coms20007.github.io/reference/semantics/induction.html>).

Hint: the cases for Boolean expressions of the form $e_1 \leq e_2$ and $e_1 = e_2$ are not inductive cases as the sub-expressions are not actually Boolean expressions.

- ** 5. We extend the notion of *free variables* of an arithmetic expression to Boolean expressions. Formally, we define a function $\text{FV} : \mathcal{B} \rightarrow \mathcal{P}(\text{Var})$ from Boolean expressions to sets of variables by recursion over the structure of expressions as follows:

$$\begin{aligned}
 \text{FV}(\text{true}) &= \emptyset \\
 \text{FV}(\text{false}) &= \emptyset \\
 \text{FV}(e_1 \leq e_2) &= \text{FV}(e_1) \cup \text{FV}(e_2) \\
 \text{FV}(e_1 = e_2) &= \text{FV}(e_1) \cup \text{FV}(e_2) \\
 \text{FV}(!e) &= \text{FV}(e) \\
 \text{FV}(e_1 \ \&\& \ e_2) &= \text{FV}(e_1) \cup \text{FV}(e_2) \\
 \text{FV}(e_1 \parallel e_2) &= \text{FV}(e_1) \cup \text{FV}(e_2)
 \end{aligned}$$

- (a) Find two Boolean expressions $e_1, e_2 \in \mathcal{B}$ that are semantically equivalent, i.e. they evaluate to the same value on all states, but for which $\text{FV}(e_1) \neq \text{FV}(e_2)$.
- (b) Prove by induction that for *all* Boolean expressions $e \in \mathcal{B}$ and pair of states $\sigma, \sigma' \in \text{State}$

that:

$$\llbracket e \rrbracket_{\mathcal{B}}(\sigma) = \llbracket e \rrbracket_{\mathcal{B}}(\sigma')$$

where $\forall x \in \text{FV}(e). \sigma(x) = \sigma'(x)$.

You may assume the fact that the analogous result holds for arithmetic expressions in your answer.

** 6. The set of *contexts* is defined by the following grammar:

$$C \rightarrow \varepsilon \mid A + C \mid C + A \mid A - C \mid C - A \mid A * C \mid C * A$$

where A is an arbitrary arithmetic expression. We write \mathcal{C} for the set of contexts.

Given a context $C \in \mathcal{C}$ and an arithmetic expression $e \in \mathcal{A}$, we write $C[e] \in \mathcal{A}$ for the arithmetic expression that is derived by replacing the “ ε ” in C with the expression e . For example, $(x + \varepsilon)[y]$ is the expression $x + y$. Formally, this operation is defined by recursion over contexts:

$$\begin{aligned} \varepsilon[e_1] &= e_1 \\ (e_2 + C)[e_1] &= e_2 + C[e_1] \\ (C + e_2)[e_1] &= C[e_1] + e_2 \\ (e_2 - C)[e_1] &= e_2 - C[e_1] \\ (C - e_2)[e_1] &= C[e_1] - e_2 \\ (e_2 * C)[e_1] &= e_2 * C[e_1] \\ (C * e_2)[e_1] &= C[e_1] * e_2 \end{aligned}$$

- (a) Consider the arithmetic expressions $x + x$ and $x * 2$ and the context $y + \varepsilon$. Show that $(y + \varepsilon)[x + x]$ and $(y + \varepsilon)[x * 2]$ are semantically equivalent.
- (b) Now suppose e_1 and e_2 are arbitrary arithmetic expressions that are semantically equivalent. Show that $(y + \varepsilon)[e_1]$ and $(y + \varepsilon)[e_2]$ are semantically equivalent as well.
- (c) Prove by structural induction that, for any context $C \in \mathcal{C}$, and any two semantically equivalent arithmetic expressions $e_1 \in \mathcal{A}$ and $e_2 \in \mathcal{A}$, that $C[e_1]$ and $C[e_2]$ are semantically equivalent.

*** 7. *Strong induction* is a variation on proof by induction where the induction hypothesis applies to *all* smaller values rather than just the predecessor or the subtrees. Formally, it can be stated as following principle:

In order to prove $\forall n \in \mathbb{N}. P(n)$, prove:

1. $P(0)$;

2. And, $P(n + 1)$ under the assumption that $P(m)$ holds for all $m \leq n$.

Prove that strong induction follows from standard induction. That is, given a property P of the natural numbers that satisfies the requirements (1) and (2), prove that $P(n)$ is true for all $n \in \mathbb{N}$.

Hint: Try proving the property $P'(n) = \forall m \leq n. P(m)$ by induction.

2 Operational Semantics

This section is about the big-step operational semantics of While programs as given by the relation $\Downarrow \subseteq \mathcal{S} \times \text{State} \times \text{State}$, which is defined inductively by these inference rules:

$$\begin{array}{c}
\frac{}{\text{skip}, \sigma \Downarrow \sigma} \qquad \frac{}{x \leftarrow e, \sigma \Downarrow \sigma[x \mapsto \llbracket e \rrbracket_{\mathcal{A}}(\sigma)]} \\
\\
\frac{S_1, \sigma_1 \Downarrow \sigma_2 \quad S_2, \sigma_2 \Downarrow \sigma_3}{S_1; S_2, \sigma_1 \Downarrow \sigma_3} \qquad \frac{S_1, \sigma_1 \Downarrow \sigma_2}{\text{if } e \text{ then } S_1 \text{ else } S_2, \sigma_1 \Downarrow \sigma_2} \llbracket e \rrbracket_{\mathcal{B}}(\sigma_1) = \top \\
\\
\frac{S_2, \sigma_1 \Downarrow \sigma_2}{\text{if } e \text{ then } S_1 \text{ else } S_2, \sigma_1 \Downarrow \sigma_2} \llbracket e \rrbracket_{\mathcal{B}}(\sigma_1) = \perp \qquad \frac{S, \sigma_1 \Downarrow \sigma_2}{\text{while } e \text{ do } S, \sigma_1 \Downarrow \sigma_2} \llbracket e \rrbracket_{\mathcal{B}}(\sigma_1) = \perp \\
\\
\frac{S, \sigma_1 \Downarrow \sigma_2 \quad \text{while } e \text{ do } S, \sigma_2 \Downarrow \sigma_3}{\text{while } e \text{ do } S, \sigma_1 \Downarrow \sigma_3} \llbracket e \rrbracket_{\mathcal{B}}(\sigma_1) = \top
\end{array}$$

Figure 1: Inference rules for operational semantics.

- * 8. Write down a derivation for the judgement $x \leftarrow 1; \{x \leftarrow 2; x \leftarrow 3\}, [] \Downarrow [x \mapsto 3]$ using the inference rules in Figure 1.
- * 9. Write down a derivation for the judgement $\{x \leftarrow 1; x \leftarrow 2\}; x \leftarrow 3, [] \Downarrow [x \mapsto 3]$ using the inference rules in Figure 1.
- * 10. Write down a derivation for the judgement $\{x \leftarrow 1; x \leftarrow 2\}; x \leftarrow 3, [] \Downarrow [x \mapsto 3]$ using the inference rules in Figure 1.
- * 11. Compute the final state for the program $\text{if } x \leq y \text{ then } x \leftarrow y \text{ else } y \leftarrow x$ when executed in each of the following states:
 - $[]$
 - $[x \mapsto 2, y \mapsto 3]$
 - $[x \mapsto 4, y \mapsto 2]$

- * 12. Find a state σ such that $x \leftarrow 1; y \leftarrow x * 2, [] \Downarrow \sigma$. You must also write down the derivation of the statement.
- * 13. Find a state $\sigma \in \text{State}$ for which there exists a derivation of the judgement $\text{while } !(x \leq -1) \text{ do } x \leftarrow x + d, [d \mapsto -1] \Downarrow \sigma$. You should provide the derivation.
- ** 14. Find a state σ such that $x \leftarrow 2; y \leftarrow x * y, \sigma \Downarrow [x \mapsto 2, y \mapsto 4]$. You must write down the derivation of the statement.
- * 15. Suppose $e \in \mathcal{B}$ is a Boolean expression that is semantically equivalent to false. Prove that $\text{while } e \text{ do } S, \sigma \Downarrow \sigma$ for any state $\sigma \in \text{State}$.
- ** 16. Suppose $S_1, S_2 \in \mathcal{S}$ are two statements such that $S_1, \sigma \Downarrow \sigma'$ and $S_2, \sigma \Downarrow \sigma'$ for some states $\sigma, \sigma' \in \text{State}$. Prove that if e then S_1 else $S_2, \sigma \Downarrow \sigma'$ for any Boolean expression $e \in \mathcal{B}$.
- ** 17. Suppose we introduce a new language construct $\text{do } S \text{ while } e$ where $S \in \mathcal{S}$ is a statement and $e \in \mathcal{B}$ is a Boolean expression. The operational semantics for this construct is given by the following inference rules:

$$\frac{S, \sigma_1 \Downarrow \sigma_2}{\text{do } S \text{ while } e, \sigma_1 \Downarrow \sigma_2} \llbracket e \rrbracket_{\mathcal{B}}(\sigma_2) = \perp \quad \frac{S, \sigma_1 \Downarrow \sigma_2 \quad \text{do } S \text{ while } e, \sigma_2 \Downarrow \sigma_3}{\text{do } S \text{ while } e, \sigma_1 \Downarrow \sigma_3} \llbracket e \rrbracket_{\mathcal{B}}(\sigma_2) = \top$$

- (a) Find a state $\sigma \in \text{State}$ such that $\text{do } x \leftarrow x + 1 \text{ while } x \leq 1, [] \Downarrow \sigma$ and give the associated derivation.
- (b) For a given statement $S \in \mathcal{S}$ and a Boolean expression $e \in \mathcal{B}$ find a While program that is equivalent to the program $\text{do } S \text{ while } e$ but does not use the new construct. That is, find a statement $S' \in \mathcal{S}$ such that:

$$S', \sigma \Downarrow \sigma' \iff \text{do } S \text{ while } e, \sigma \Downarrow \sigma'$$

You do not need to prove that your answer is correct but should provide a derivation of judgement $S', [] \Downarrow \sigma$ where S is given to be the statement $x \leftarrow x + 1$, e is given to be the expression $x \leq 1$ and σ is the state from part (a).

- *** 18. Suppose we introduce a new language construct $\text{for } x \text{ do } S$ where $S \in \mathcal{S}$ is a statement and $x \in \text{Var}$ is a variable. The operational semantics for this construct is given by the following inference rules:

$$\frac{}{\text{for } x \text{ do } S, \sigma \Downarrow \sigma} \sigma(x) \leq 0 \quad \frac{S, \sigma_1 \Downarrow \sigma_2 \quad \text{for } x \text{ do } S, \sigma_2[x \mapsto \sigma_2(x) - 1] \Downarrow \sigma_3}{\text{for } x \text{ do } S, \sigma_1 \Downarrow \sigma_3} \sigma_1(x) > 0$$

- (a) Find a state $\sigma \in \text{State}$ such that $\text{for } x \text{ do } y \leftarrow y + x; x \leftarrow x - 2, [x \mapsto 3] \Downarrow \sigma$ and give the associated derivation.
- (b) For a given statement $S \in \mathcal{S}$ and a variable $x \in \text{Var}$ find a While program that is equivalent

to the program `for x do S` but does not use the new construct. That is, find a statement $S' \in \mathcal{S}$ such that:

$$S', \sigma \Downarrow \sigma' \Leftrightarrow \text{for } x \text{ do } S$$

You do not need to prove that your answer is correct but should provide a derivation of judgement $S', [x \mapsto 3] \Downarrow \sigma$ where S is given to be the statement $y \leftarrow y + x; x \leftarrow x - 2, x$ is given to be the variable x and σ is the state from part (a).