Programming Languages and Computation

# Week 5: Denotational Semantics and Induction

## 1 Denotational Semantics

* 1. Compute the value of the following arithmetic expressions in the state $\sigma = [x \mapsto -1, y \mapsto 11, z \mapsto 10]$. Your answer should be explicit about the steps you take and make reference to the definition of the denotation function.

(a) $(x + y) - z$

(b) $x * (12 - z)$

(c) $x - (z - 1)$

Solution

(a)

$$\begin{aligned}
[\![(x+y)-z]\!]_{\mathcal{A}}(\sigma) &= [\![x+y]\!]_{\mathcal{A}}(\sigma) - [\![z]\!]_{\mathcal{A}}(\sigma) \\
&= ([\![x]\!]_{\mathcal{A}}(\sigma) + [\![y]\!]_{\mathcal{A}}(\sigma)) - [\![z]\!]_{\mathcal{A}}(\sigma) \\
&= (\sigma(x) + \sigma(y)) - \sigma(z) \\
&= ((-1) + 11) - 10 \\
&= 0
\end{aligned}$$

(b)

$$\begin{aligned}
[\![x*(12-z)]\!]_{\mathcal{A}}(\sigma) &= [\![x]\!]_{\mathcal{A}}(\sigma) \cdot [\![12-z]\!]_{\mathcal{A}}(\sigma) \\
&= [\![x]\!]_{\mathcal{A}}(\sigma) \cdot ([\![12]\!]_{\mathcal{A}}(\sigma) - [\![z]\!]_{\mathcal{A}}(\sigma)) \\
&= \sigma(x) \cdot (12 - \sigma(z)) \\
&= (-1) \cdot (12 - 10) \\
&= -2
\end{aligned}$$

(c)

$$\begin{aligned}
[\![x-(z-1)]\!]_{\mathcal{A}}(\sigma) &= [\![x]\!]_{\mathcal{A}}(\sigma) - [\![z-1]\!]_{\mathcal{A}}(\sigma) \\
&= [\![x]\!]_{\mathcal{A}}(\sigma) - ([\![z]\!]_{\mathcal{A}}(\sigma) - [\![1]\!]_{\mathcal{A}}(\sigma)) \\
&= \sigma(x) - (\sigma(z) - 1) \\
&= (-1) - (10 - 1) \\
&= -10
\end{aligned}$$

* 2. Compute the value of the following arithmetic expressions in the state $\sigma = [x \mapsto 11, y \mapsto 12]$. Your answer should be explicit about the steps you take and make reference to the definition of the denotation function.

(a) $(x + y) - z$

(b) $x * (12 - z)$

(c) $x - (z - 1)$

Solution

This question depends on the convention that variables not explicitly mentioned by the state are assigned the value 0.

(a)

$$\begin{aligned}
[\![(x+y)-z]\!]_{\mathcal{A}}(\sigma) &= [\![x+y]\!]_{\mathcal{A}}(\sigma) - [\![z]\!]_{\mathcal{A}}(\sigma) \\
&= [\![x]\!]_{\mathcal{A}}(\sigma) + [\![y]\!]_{\mathcal{A}}(\sigma) - [\![z]\!]_{\mathcal{A}}(\sigma) \\
&= \sigma(x) + \sigma(y) - \sigma(z) \\
&= (11 + 12) - 0 \\
&= 23
\end{aligned}$$

(b)

$$\begin{aligned}
[\![x*(12-z)]\!]_{\mathcal{A}}(\sigma) &= [\![x]\!]_{\mathcal{A}}(\sigma) \cdot [\![12-z]\!]_{\mathcal{A}}(\sigma) \\
&= [\![x]\!]_{\mathcal{A}}(\sigma) \cdot ([\![12]\!]_{\mathcal{A}}(\sigma) - [\![z]\!]_{\mathcal{A}}(\sigma)) \\
&= \sigma(x) \cdot (12 - \sigma(z)) \\
&= 11 \cdot (12 - 0) \\
&= 132
\end{aligned}$$

(c)

$$\begin{aligned}
[\![x-(z-1)]\!]_{\mathcal{A}}(\sigma) &= [\![x]\!]_{\mathcal{A}}(\sigma) - [\![z-1]\!]_{\mathcal{A}}(\sigma) \\
&= [\![x]\!]_{\mathcal{A}}(\sigma) - ([\![z]\!]_{\mathcal{A}}(\sigma) - [\![1]\!]_{\mathcal{A}}(\sigma)) \\
&= \sigma(x) - (\sigma(z) - 1) \\
&= 11 - (0 - 1) \\
&= 12
\end{aligned}$$

* 3. Compute the value of the following Boolean expressions in the state $\sigma = [x \mapsto 11, y \mapsto 12]$. Your answer should be explicit about the steps you take and make reference to the definition of the denotation function.

(a) $(x + y \leq z)$ && true

(b) $!(x * y = z) \,||\, x = 11$

(c) $x \leq y$ && $y \leq x$

Solution

(a)

$$\begin{aligned}
[\![(x + y \leq z) \text{ \&\& true}]\!]_{\mathcal{B}}(\sigma) &= [\![x + y \leq z]\!]_{\mathcal{B}}(\sigma) \wedge [\![\text{true}]\!]_{\mathcal{B}}(\sigma) \\
&= ([\![x + y]\!]_{\mathcal{A}}(\sigma) \leq [\![z]\!]_{\mathcal{A}}(\sigma)) \wedge \top \\
&= ([\![x]\!]_{\mathcal{A}}(\sigma) + [\![y]\!]_{\mathcal{A}}(\sigma) \leq [\![z]\!]_{\mathcal{A}}(\sigma)) \wedge \top \\
&= (\sigma(x) + \sigma(y) \leq \sigma(z)) \ \wedge \top \\
&= (11 + 12 \leq 0) \wedge \top \\
&= \bot \wedge \top \\
&= \bot
\end{aligned}$$

(b)

$$\begin{aligned}
[\![!(x * y = z) \,||\, x = 11]\!]_{\mathcal{B}}(\sigma) &= [\![!(x * y = z)]\!]_{\mathcal{B}}(\sigma) \vee [\![x = 11]\!]_{\mathcal{B}}(\sigma) \\
&= \neg [\![x * y = z]\!]_{\mathcal{B}}(\sigma) \vee ([\![x]\!]_{\mathcal{A}}(\sigma) = [\![11]\!]_{\mathcal{A}}(\sigma)) \\
&= \neg([\![x * y]\!]_{\mathcal{A}}(\sigma) = [\![z]\!]_{\mathcal{A}}(\sigma)) \vee ([\![x]\!]_{\mathcal{A}}(\sigma) = [\![11]\!]_{\mathcal{A}}(\sigma)) \\
&= \neg([\![x]\!]_{\mathcal{A}}(\sigma) * [\![y]\!]_{\mathcal{A}}(\sigma) = [\![z]\!]_{\mathcal{A}}(\sigma)) \vee ([\![x]\!]_{\mathcal{A}}(\sigma) = [\![11]\!]_{\mathcal{A}}(\sigma)) \\
&= \neg(\sigma(x) * \sigma(y) = \sigma(z)) \vee (\sigma(x) = 11) \\
&= \neg(11 * 12 = 0) \vee (11 = 11) \\
&= \neg\bot \vee \top \\
&= \top \vee \top \\
&= \top
\end{aligned}$$

(c)

$$\begin{aligned}
[\![x \leq y \text{ \&\& } y \leq x]\!]_{\mathcal{B}}(\sigma) &= [\![x \leq y]\!]_{\mathcal{B}}(\sigma) \wedge [\![y \leq x]\!]_{\mathcal{A}}(\sigma) \\
&= [\![x]\!]_{\mathcal{A}}(\sigma) \leq [\![y]\!]_{\mathcal{A}}(\sigma) \wedge ([\![y]\!]_{\mathcal{A}}(\sigma) \leq [\![x]\!]_{\mathcal{A}}(\sigma)) \\
&= \sigma(x) \leq \sigma(y) \wedge \sigma(y) \leq \sigma(x) \\
&= 11 \leq 12 \wedge 12 \leq 11 \\
&= \top \wedge \bot \\
&= \bot
\end{aligned}$$

The next questions relate to when arithmetic and Boolean expressions are *syntactically equivalent* or *semantically equivalent*. Two arithmetic or Boolean expressions are syntactically equivalent if they have exactly the same abstract syntax tree and semantically equivalent is they denote the same function. Remember that two functions are equal if, and only if, they have the same value on every input.

* 4. Which of the following arithmetic expressions are syntactically equivalent and which are semantically equivalent?

(a) $x * 3$

(b) $x * 1$

(c) $x + (x + x)$

(d) $(x + (x)) + x$

(e) $x + ((x) + x)$

(f) $x + (y * 0)$

Solution

The only syntactically equivalent expressions are (c) and (e). However, the expressions (a), (c), (d), and (e) are semantically equivalent, and likewise so are (b) and (f).

* 5. Consider the Boolean expression $x \leq 2$ && $y \leq 3$. Find a semantically equivalent expression that does not use the && operator.

Solution

Lots of possible answers, e.g. !(!($x \leq 2$) || !($x \leq 3$)).

* 6. Find a state in which the arithmetic expressions $x * 2$ and $x + 3$ evaluate to the same value. Explain why they are *not* semantically equivalent.

Solution

A state in which they evaluate to the same value is $[x \mapsto 3]$. They are not semantically equivalent, however, as they evaluate to distinct values in the state $[x \mapsto 1]$.

** 7. Suppose that $e_1 \in \mathcal{A}$ and $e_2 \in \mathcal{A}$ are semantically equivalent arithmetic expressions. Prove that $e_1 + e_2$ is semantically equivalent to $e_1 * 2$. Your answer should make explicit reference to the denotation function.

Solution

To show that $e_1 + e_2$ is semantically equivalent to $e_1 * 2$, we must show that, for any state $\sigma \in \text{State}$, $[\![e_1 + e_2]\!]_{\mathcal{A}}(\sigma) = [\![e_1 * 2]\!]_{\mathcal{A}}(\sigma)$. Let $\sigma \in \text{State}$ be a state. By expanding the definitions, we can see that $[\![e_1 + e_2]\!]_{\mathcal{A}}(\sigma) = [\![e_1]\!]_{\mathcal{A}}(\sigma) + [\![e_2]\!]_{\mathcal{A}}(\sigma)$ and that $[\![e_1 * 2]\!]_{\mathcal{A}}(\sigma) = [\![e_1]\!]_{\mathcal{A}}(\sigma) \cdot 2$. As $e_1$ and $e_2$ are semantically equivalent, we know that $[\![e_1]\!]_{\mathcal{A}}(\sigma) = [\![e_2]\!]_{\mathcal{A}}(\sigma)$. Therefore, we have that:

$$\begin{aligned} [\![e_1 + e_2]\!]_{\mathcal{A}}(\sigma) &= [\![e_1]\!]_{\mathcal{A}}(\sigma) + [\![e_2]\!]_{\mathcal{A}}(\sigma) \\ &= [\![e_1]\!]_{\mathcal{A}}(\sigma) + [\![e_1]\!]_{\mathcal{A}}(\sigma) \\ &= [\![e_1]\!]_{\mathcal{A}}(\sigma) * 2 \\ &= [\![e_1 * 2]\!]_{\mathcal{A}}(\sigma) \end{aligned}$$

Therefore, we can conclude that the two expressions are semantically equivalent as required.

** 8. Suppose that $e_1 + 1$ and $e_2 + 1$ are two arithmetic expressions that are semantically equivalent for some $e_1, e_2 \in \mathcal{A}$. Prove that $e_1$ and $e_2$ are also semantically equivalent.

Solution

To show that $e_1$ and $e_2$ are semantically equivalent let $\sigma \in \mathsf{State}$ be an arbitrary state. We have that $[\![e_1 + 1]\!]_{\mathcal{A}}(\sigma) = [\![e_2 + 1]\!]_{\mathcal{A}}(\sigma)$. It follows, by definition, that $[\![e_1]\!]_{\mathcal{A}}(\sigma) + 1 = [\![e_2]\!]_{\mathcal{A}}(\sigma) + 1$. Therefore, $[\![e_1]\!]_{\mathcal{A}}(\sigma) = [\![e_2]\!]_{\mathcal{A}}(\sigma)$ as required.

** 9. Suppose that $e_1 * e_2$ and $e_1 * 2$ are two arithmetic expressions that are *not* semantically equivalent for some arithmetic expressions $e_1, e_2 \in \mathcal{A}$.

(a) Prove that $e_2$ cannot be semantically equivalent to 2.

(b) Find concrete examples of expressions $e_1, e_2 \in \mathcal{A}$ such that $e_1 * e_2$ and $e_1 * 2$ are not semantically equivalent but where there exists a state $\sigma \in \mathsf{State}$ such that $[\![e_2]\!]_{\mathcal{A}}(\sigma) = 2$.

Solution

(a) There exists a state $\sigma \in \mathsf{State}$ such that $[\![e_1]\!]_{\mathcal{A}}(\sigma) \cdot [\![e_2]\!]_{\mathcal{A}}(\sigma) \neq [\![e_1]\!] \cdot 2$. Therefore, there exists a state in which $[\![e_2]\!]_{\mathcal{A}}(\sigma) \neq 2$. Thus, $e_2$ cannot be semantically equivalent to 2.

(b) There are many possible answers, but the point is that even though $e_2$ cannot be equivalent to 2 there may still be states in which it evaluates to 2. For example, if $e_1 = 1$ and $e_2 = x$, then we have that both $e_1 * e_2$ and $e_1 * 2$ evaluate to 2 in the state $\sigma = [x \mapsto 2]$.

** 10. Let us suppose we want to add a new construct to the language of arithmetic expressions:

$$A \to x \mid n \mid \cdots \mid \mathsf{let}\ x = A\ \mathsf{in}\ A$$

An expression let $x = e_1$ in $e_2$ using this construct should evaluate the sub-expression $e_2$ in a state where the variable $x$ is mapped to the value of $e_1$. For example, the expression let $x = 2$ in $x + y$ when evaluated in the state $[x \mapsto 3, y \mapsto 2]$ should be 4.

Extend the definition of the denotation function $[\![\cdot]\!]_{\mathcal{A}}$ with an equation for this construct. You may find it useful to use the notation $\sigma[x \mapsto n]$ to represent the state $\sigma$ updated such that $(\sigma[x \mapsto n])(x) = n$ and $(\sigma[x \mapsto n])(y) = \sigma(y)$ for all $y \neq x$. You do not need to change any other equations.

Solution

The required equation is:

$$[\![\mathsf{let}\ x = e_1\ \mathsf{in}\ e_2]\!]_{\mathcal{A}}(\sigma) = [\![e_2]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_1]\!]_{\mathcal{A}}(\sigma)])$$

The key things to note are that the state is updated according to the value of $e_1$ as specified and that the equation is recursive in that the denotation for new construct is defined in terms of the denotation of its sub-expressions.

** 11. Now let us suppose we extend the language of arithmetic expressions with a different operator:

$$A \to x \mid n \mid \cdots \mid B\ ?\ A : A$$

An instance of this *ternary operator* $e_1 ? e_2 : e_3$ for some Boolean expression $e_1 \in \mathcal{B}$ and arithmetic expressions $e_2, e_3 \in \mathcal{A}$ behaves as $e_2$ in states where $e_1$ is true and behaves as $e_3$ otherwise.

Extend the definition of the denotation function $[\![\cdot]\!]_{\mathcal{A}}$ with an equation for this construct. Your answer may make reference to the denotation function for Boolean expressions.

Solution

The required equation is:

$$[\![e_1\ ?\ e_2 : e_3]\!]_{\mathcal{A}}(\sigma) = \begin{cases} [\![e_2]\!]_{\mathcal{A}}(\sigma) & \text{if } [\![e_1]\!]_{B}(\sigma) \\ [\![e_3]\!]_{\mathcal{A}}(\sigma) & \text{otherwise} \end{cases}$$

## 2 Proof by Induction

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

$$x^0 = 1$$
$$x^{n+1} = x \cdot x^n$$

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.

Solution

We shall prove that $(x \cdot y)^z = x^z \cdot y^z$ for any $x, y, z \in \mathbb{N}$ by induction over $z \in \mathbb{N}$.

- In the base case, we have that $(x \cdot y)^0 = 1$ and $x^0 \cdot y^0 = 1 \cdot 1 = 1$. Therefore, $(x \cdot y)^0 = x^0 \cdot y^0$ as required.
- Let us suppose $(x \cdot y)^z = x^z \cdot y^z$ holds for some $z \in \mathbb{N}$. We must show that $(x \cdot y)^{z+1} = x^{z+1} \cdot y^{z+1}$ holds. By definition, $(x \cdot y)^{z+1} = (x \cdot y) \cdot (x \cdot y)^z$. It then follows from our induction hypothesis that $(x \cdot y)^{z+1} = (x \cdot y) \cdot x^z \cdot y^z$. Therefore, $(x \cdot y)^{z+1} = x^{z+1} \cdot y^{z+1}$ as required.

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

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

(a) Prove by structural induction over arithmetic expressions that $\mathsf{height}(e) > 0$ for all arithmetic expressions $e \in \mathcal{A}$.

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

Solution

(a) We shall prove that $\mathsf{height}(e) > 0$ for all arithmetic expressions $e \in \mathcal{A}$ by induction as follows:
- In the case of a variable or a numeric literal, the height is clearly greater than 0.
- In the inductive cases, we have that induction hypotheses $\mathsf{height}(e_1), \mathsf{height}(e_2) > 0$. It then follows that $1 + \max\{\mathsf{height}(e_1), \mathsf{height}(e_2)\} > 0$ as required.

(b) We shall prove that $2^{\mathsf{height}(e)-1} \geq \#\mathsf{FV}(e)$ for all arithmetic expressions $e \in \mathcal{A}$ by induction as follows:
- In the case of a variable $x \in \mathsf{Var}$, the height is 1 and $\#\mathsf{FV}(x) = 1$. Therefore, $2^{\mathsf{height}(x)-1} = 2^0 = 1$ and thus $2^{\mathsf{height}(x)-1} \geq 1$ as required.
- Similarly, in the case of a numeric literal $n \in \mathbb{Z}$, the height is 1 and $\#\mathsf{FV}(x) = 0$. Therefore, $2^{\mathsf{height}(n)-1} = 2^0 = 1$ and thus $2^{\mathsf{height}(n)-1} \geq 0$ as required.
- Now consider an expression of the form $e_1 + e_2$ where, inductively, we know that $2^{\mathsf{height}(e_1)-1} \geq \#\mathsf{FV}(e_1)$ and $2^{\mathsf{height}(e_2)-1} \geq \#\mathsf{FV}(e_2)$. By definition, $\mathsf{height}(e_1 + e_2)$ is equal to $1 + \max\{\mathsf{height}(e_1), \mathsf{height}(e_2)\}$. Therefore, $2^{\mathsf{height}(e_1+e_2)-1}$ is at least as large as both $2^{\mathsf{height}(e_1)}$ and $2^{\mathsf{height}(e_2)}$. It then follows from the induction hypotheses, that $2^{\mathsf{height}(e_1+e_2)-1}$ is at least as large as both $2 \cdot \#\mathsf{FV}(e_1)$ and $2 \cdot \#\mathsf{FV}(e_2)$. Finally, as $\#\mathsf{FV}(e_1) + \#\mathsf{FV}(e_2) \geq \#\mathsf{FV}(e_1 + e_2)$, we have that $2^{\mathsf{height}(e_1+e_2)-1} \geq \#\mathsf{FV}(e_1 + e_2)$ as required.
- The cases for subtraction and multiplication are analogous to that of addition.

** 14. If $x$ is a variable and $e_1$ and $e_2$ are arithmetic expressions, then we write $e_1[e_2/x]$ 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:

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

(a) Compute the value of the expression $(y - x)[z/x]$ in the state $[x \mapsto 1, y \mapsto 2, z \mapsto 3]$.

(b) Find a state $\sigma$ such that $[\![y - x]\!]_{\mathcal{A}}(\sigma)$ 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 $\sigma \in \mathsf{State}$, any pair of arithmetic expressions $e_1, e_2 \in A$ and any variable $x \in \mathsf{Var}$, we have that:

$$[\![e_1[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![e_1]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)]).$$

Remember that $e_1$ may be an *arbitrary* variable.

Solution

(a) The expression $(y - x)[z/x]$ is by definition equal to $y - z$. Therefore, evaluating in the state $[x \mapsto 1, y \mapsto 2, z \mapsto 3]$ leads to the value $-1$.

(b) Under the state $[x \mapsto 3, y \mapsto 2, z \mapsto 3]$, the expression $y - x$ evaluates to $-1$. This state is derived by replacing the value for $x$ with the value for $z$, thus capturing the behaviour of the substitution $[z/x]$.

(c) We shall prove by structural induction that:

$$[\![e_1[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![e_1]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)]).$$

for any state $\sigma \in \mathsf{State}$, any pair of arithmetic expressions $e_1, e_2 \in A$ and any variable $x \in \mathsf{Var}$ by structural induction over $e_1$.

- Suppose $e_1$ is a variable $y \in \mathsf{Var}$. In order to correctly apply the substitution, we need to know whether the variable $y$ is equal to the variable $x$. Therefore, there are two subcases to consider:
  - If $y = x$, then we have that $[\![y[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![e_2]\!]_{\mathcal{A}}(\sigma)$. On the other hand, $[\![y]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)])$ is by definition $[\![e_2]\!]_{\mathcal{A}}(\sigma)$. Therefore, $[\![y[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![y]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)])$ as required.
  - Otherwise, let us suppose that $y \neq x$. In this case, $[\![y[e_2/x]]\!]_{\mathcal{A}}(\sigma) = \sigma(y)$ as the expression $y$ is unaffected by the substitution Likewise, $[\![y]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)])$ is equal to $\sigma(y)$ as both the states $\sigma$ and $\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)]$ assign the same value to the variable $y$. Therefore, we have that $[\![y[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![y]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)])$ as required.

- Now let us suppose $e_1$ is a numeric literal $n \in \mathbb{Z}$. In this case, $[\![n[e_2/x]]\!]_{\mathcal{A}}(\sigma) = n$ and, likewise, $[\![n]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]]) = n$. Therefore, $[\![n[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![n]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]])$ as required.
- Now let us suppose $e_1$ is of the form $e_3 + e_4$ for some arithmetic expressions $e_3$, $e_4 \in \mathcal{A}$. Then $[\![(e_3 + e_4)[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![e_3[e_2/x]]\!]_{\mathcal{A}}(\sigma) + [\![e_4[e_2/x]]\!]_{\mathcal{A}}(\sigma)$ by definition. Our induction hypotheses tell us that $[\![e_3[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![e_3]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]])$ and likewise for $e_4$. Therefore, we have that $[\![(e_3 + e_4)[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![e_3]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)]) + [\![e_4]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)])$. It follows then that $[\![(e_3 + e_4)[e_2/x]]\!]_{\mathcal{A}}(\sigma) = [\![e_3 + e_4]\!]_{\mathcal{A}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)])$ as required.
- The cases of subtraction and multiplication are analogous to that of addition.

** 15. 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/notes/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.

Solution

The induction principle for Boolean expressions states that $P(e)$ is true of all $e \in \mathcal{B}$ whenever:

- $P(\mathsf{true})$ is true;
- $P(\mathsf{false})$ is true;
- $P(e_1 \;\&\&\; e_2)$ is true for any $e_1$, $e_2 \in \mathcal{B}$ such that $P(e_1)$ and $P(e_2)$ is true;
- $P(e_1 \;\|\; e_2)$ is true for any $e_1$, $e_2 \in \mathcal{B}$ such that $P(e_1)$ and $P(e_2)$ is true;
- $P(!e)$ is true for any $e \in \mathcal{B}$ such that $P(e)$ is true;
- $P(e_1 = e_2)$ is true for any $e_1$, $e_2 \in \mathcal{A}$;
- And, $P(e_1 \leq e_2)$ is true for any $e_1$, $e_2 \in \mathcal{A}$.

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

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

(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 $\mathsf{FV}(e_1) \neq \mathsf{FV}(e_2)$.

(b) Prove by induction that for *all* Boolean expressions $e \in \mathcal{B}$ and pair of states $\sigma, \sigma' \in \mathsf{State}$ that:

$$[\![e]\!]_{\mathcal{B}}(\sigma) = [\![e]\!]_{\mathcal{B}}(\sigma')$$

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

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

Solution

(a) The Boolean expressions $\mathsf{true}$ and $\mathsf{true} \,||\, x \leq y$ are semantically equivalent but have a different set of free variables.

(b) We shall prove by induction that for all Boolean expressions $e \in \mathcal{B}$ and pair of states $\sigma, \sigma' \in \mathsf{State}$ that:

$$[\![e]\!]_{\mathcal{B}}(\sigma) = [\![e]\!]_{\mathcal{B}}(\sigma')$$

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

- In the case of the constant $\mathsf{true}$, we have that $[\![\mathsf{true}]\!]_{\mathcal{B}}(\sigma) = \top$ regardless of $\sigma$. In particular, $[\![\mathsf{true}]\!]_{\mathcal{B}}(\sigma) = [\![\mathsf{true}]\!]_{\mathcal{B}}(\sigma')$ for any two states $\sigma, \sigma' \in \mathsf{State}$.
- The case of the constant $\mathsf{false}$ is analogous to that of $\mathsf{true}$.
- Now consider a Boolean expression of the form $e_1 \leq e_2$ where $e_1 \in \mathcal{A}$ and $e_2 \in \mathcal{A}$ are arithmetic expressions. Let $\sigma, \sigma' \in \mathsf{State}$ be states such that $\forall x \in \mathsf{FV}(e_1 \leq e_2).\, \sigma(x) = \sigma'(x)$. By definition, $\mathsf{FV}(e_1 \leq e_2) = \mathsf{FV}(e_1) \cup \mathsf{FV}(e_2)$. Therefore, we also know that $\forall x \in \mathsf{FV}(e_1).\, \sigma(x) = \sigma'(x)$ and likewise for $e_2$. It then follows that $[\![e_1]\!]_{\mathcal{A}}(\sigma) = [\![e_1]\!]_{\mathcal{A}}(\sigma')$ and likewise for $e_2$. Thus, $[\![e_1 \leq e_2]\!]_{\mathcal{B}}(\sigma) = [\![e_1 \leq e_2]\!]_{\mathcal{B}}(\sigma')$ as required.
- The case of Boolean expressions of the form $e_1 = e_2$ is analogous to the preceding case.
- Now consider a Boolean expression of the form $!e$ where, inductively, we know that $[\![e]\!]_{\mathcal{B}}(\sigma) = [\![e]\!]_{\mathcal{B}}(\sigma')$ whenever $\forall x \in \mathsf{FV}(e).\, \sigma(x) = \sigma'(x)$. Let $\sigma, \sigma' \in \mathsf{State}$ be states such that $\forall x \in \mathsf{FV}(!e).\, \sigma(x) = \sigma'(x)$. As $\mathsf{FV}(!e) = \mathsf{FV}(e)$, we have that $\forall x \in \mathsf{FV}(e).\, \sigma(x) = \sigma'(x)$. Therefore, the induction hypothesis applies. It then follows that:

$$\begin{aligned} [\![!e]\!]_{\mathcal{B}}(\sigma) &= \neg [\![e]\!]_{\mathcal{B}}(\sigma) \\ &= \neg [\![e]\!]_{\mathcal{B}}(\sigma') \\ &= [\![!e]\!]_{\mathcal{B}}(\sigma') \end{aligned}$$

  as required.
- Now consider a Boolean expression of the form $e_1 \,\&\&\, e_2$ where, inductively, we know that $[\![e_1]\!]_{\mathcal{B}}(\sigma) = [\![e_1]\!]_{\mathcal{B}}(\sigma')$ whenever $\forall x \in \mathsf{FV}(e_1).\, \sigma(x) = \sigma'(x)$ and likewise for $e_2$. Let $\sigma, \sigma' \in \mathsf{State}$ be states such that $\forall x \in \mathsf{FV}(e_1 \,\&\&\, e_2).\, \sigma(x) = \sigma'(x)$. As $\mathsf{FV}(e_1 \,\&\&\, e_2) \supseteq \mathsf{FV}(e_1), \mathsf{FV}(e_2)$, we have that $\forall x \in \mathsf{FV}(e_1).\, \sigma(x) = \sigma'(x)$ and likewise

for $e_2$. Therefore, the induction hypotheses apply. It then follows that:

$$\begin{aligned}[\![e_1 \;\&\&\; e_2]\!]_{\mathcal{B}}(\sigma) &= [\![e_1]\!]_{\mathcal{B}}(\sigma) \wedge [\![e_2]\!]_{\mathcal{B}}(\sigma) \\ &= [\![e_1]\!]_{\mathcal{B}}(\sigma') \wedge [\![e_2]\!]_{\mathcal{B}}(\sigma') \\ &= [\![e_1 \;\&\&\; e_2]\!]_{\mathcal{B}}(\sigma')\end{aligned}$$

as required.

- The case of Boolean expressions of the form $e_1 \parallel e_2$ is analogous to the preceding case.

** 17. Define a recursive function for substitution acting on Boolean expressions, you may wish to model your answer on substitution for arithmetic expressions from Question 14.
Prove that your definition satisfies the property:

$$[\![e_1[e_2/x]]\!]_{\mathcal{B}}(\sigma) = [\![e_1]\!]_{\mathcal{B}}(\sigma[x \mapsto [\![e_2]\!]_{\mathcal{A}}(\sigma)]).$$

for any Boolean expression $e_1 \in \mathcal{B}$, arithmetic expression $e_2 \in \mathcal{A}$, and any variable $x \in \mathsf{Var}$. You may assume the analogous property shown in Question 14.

Solution

The definition should be:

$$\begin{aligned}\mathsf{true}[e/x] &= \mathsf{true} \\ \mathsf{false}[e/x] &= \mathsf{false} \\ (!e_1)[e_2/x] &= !(e_1[e_2/x]) \\ (e_1 \;\&\&\; e_2)[e_3/x] &= e_1[e_3/x] \;\&\&\; e_2[e_3/x] \\ (e_1 \parallel e_2)[e_3/x] &= e_1[e_3/x] \parallel e_2[e_3/x] \\ (e_1 = e_2)[e_3/x] &= e_1[e_3/x] = e_2[e_3/x] \\ (e_1 \leq e_2)[e_3/x] &= e_1[e_3/x] \leq e_2[e_3/x]\end{aligned}$$

The proof of correctness is by structural induction and is analogous to Question 14, with cases for equality and comparison being derived from the property shown in Question 14.

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

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

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 "$\varepsilon$" 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.

Solution

(a) The expression $(y + \varepsilon)[x + x]$ is equal to $y + x + x$ and the expression $(y + \varepsilon)[x * 2]$ is equal to the expression $y + (x * 2)$. Both these expressions denote the function that maps a state $\sigma$ to the integer $\sigma(y) + 2\sigma(x)$. Therefore, they are semantically equivalent.

(b) Let $e_1$ and $e_2$ be semantically equivalent arithmetic expressions. The expression $(y + \varepsilon)[e_1]$ maps a state $\sigma$ to the value $\sigma(y) + [\![e_1]\!]_{\mathcal{A}}(\sigma)$ and, likewise the expression $(y + \varepsilon)[e_2]$ maps a state $\sigma$ to the value $\sigma(y) + [\![e_2]\!]_{\mathcal{A}}(\sigma)$. As $e_1$ and $e_2$ are semantically equivalent, we know that $[\![e_1]\!]_{\mathcal{A}}(\sigma) = [\![e_2]\!]_{\mathcal{A}}(\sigma)$ for all states. It then follows that $(y + \varepsilon)[e_1]$ and $(y + \varepsilon)[e_2]$ are semantically equivalent as required.

(c) We shall prove 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 by induction on the context:

- In the base case with a context $\varepsilon$, we must show that $\varepsilon[e_1]$ is semantically equivalent to $\varepsilon[e_2]$ given that $e_1$ and $e_2$ are semantically equivalent. As the expression $\varepsilon[e_1]$ is equal to $e_1$ and likewise for $e_2$, this case is trivial.
- Now consider a context of the form $e_3 + C$. We must show that $(e_3 + C)[e_1]$ is semantically equivalent to $(e_3 + C)[e_2]$ given that $e_1$ and $e_2$ are semantically equivalent. By definition, $(e_3 + C)[e_1] = e_3 + C[e_1]$ and likewise $(e_3 + C)[e_2] = e_3 + C[e_2]$. By induction, we know that $C[e_1]$ is semantically equivalent to $C[e_2]$. It then follows that $e_3 + C[e_1]$ is semantically equivalent to $e_3 + C[e_2]$ and, therefore, that $(e_3 + C)[e_1]$ is semantically equivalent to $(e_3 + C)[e_2]$ as required.
- The other cases are analogous to the preceding case.