Types, Expressions, and States
CS 536: Science of Programming, Fall 2021

A. Why

- Expressions represent values relative to a state.
- Types describe common properties of sets of values.
- The value of an array is a function value from index values to array values.

B. Outcomes

At the end of today, you should

- Be able to read and write expressions we'll be using in our language.
- Be able to read and write states.
- Be able to evaluate an expression relative to a state.
- Be able to handle array names, array indexing expressions, and their values relative to a state.

C. Questions

1. Which of the following expressions are legal or illegal according to the syntax we're using?

   Assume \( x, y, z \) are integer variables and \( b \) is an array name.
   a. \((x \leq y ? x : y)^*\) do you need assumptions as to the types of \( x \) and \( y \)?
   b. \((x < y ? -1 : (x = y ? 0 : 1))^*\)
   c. \((y = 0 ? f : g)\(17)^*\)
   d. \(b[0][1]^*\) What type must \( b \) have for this to be legal?*
   e. \(b^*\) Remember we're given that \( b \) is an array *
   f. \(f(b, b[0]) < 3^*\) Also, if this is legal, what is the type of \( f ?^*\)
   g. \((x < 3 ? x : F)^*\)

2. Which of the following are legal ways to write out a state? (And if not, why not?)

   a. \(\{x = 5, y = 2\}\)
   b. \(\{x = \text{five}, y = \text{one plus one}\}\)
   c. \(\{x = 5, y = x \text{ minus 3}\}\)
   d. \(\{x = 5, y = \alpha - 3\} \text{ where } \alpha = 5\)
   e. \(\{x = 5, y = (\text{the value of } x \text{ in this environment minus 3})\}\)
   f. \(\{\}\)
3. Consider the state $\sigma_2$ described graphically below.

\[\begin{array}{c|c|c|c|c}
    x & 12 & y & T & b \\
    \\hline
    0 & 1 & 2 & 3 & 8
\end{array}\]

a. Write a definition for $\sigma_2 = \{ \ldots \}$ using four ways described in section **E: Arrays and Their Values**.

b. Calculate $\sigma_2(e)$ where $e = y \land x > b[x/5]$. Assume integer division truncates.

4. Let $\sigma_3 = \{ z = 4, b[0] = 1, b[1] = 5, b[2] = 8 \}$.
   a. Abbreviate this using tuple notation for the value of $b$ (i.e., $b = (...)$).
   b. Write out the graphical representation of $\sigma_3$ (a memory diagram as in Problem 2).
   c. Calculate $\sigma_3(e)$ where $e = b[b[z-4]] > z \land z+2 : z-2$. (Hint: Give names to parts of $e$ and calculate the values of those parts first.)

5. Let $e_4 = x = y+1 \land y = z^2 - 3 \land z = 6$. Write out the textual definition of a state $\sigma_4$ in which $e_4$ evaluates to true. Use only bindings that map variables to constants. $\sigma_4 = \{ x = 34, y = 33, z = 6 \}$

6. Which of the following states are well-formed and also proper for the expression $b[i] + 0 * y$? If ill-formed, why? If taking the value might cause a runtime error, why?
   a. $\{i = 0, b = (3, 4, 8), y = 3, z = 5\}$
   b. $\{i = 0, b = (6), y = 5\}$
   c. $\{i = 0, b = 6, y = 5\}$
   d. $\{i = 1, b = (3, 4, 8)\}$
   e. $\{i = 1, i = 2, y = 0, b = (2, 6)\}$
   f. $\{i = 5, b = (1, 2), y = 4\}$
CS 536 Solution to Practice 3 (Types, Expressions, and States)

1. (Legal and illegal expressions)
   a. \((x > y ? x : y)\) is legal
   b. \((x < y ? -1 : (x = y ? 0 : 1))\) is legal
   c. \((y = 0 ? f : g)(17)\) is illegal because the conditional expression can't yield a function
   d. \(b[0][1]\) is legal (\(b\) must be a 2-dimensional array)
   e. \(b\) (all by itself) is illegal, since \(b\) we've assumed is an array
   f. \(f(b, b[0]) < 3\) is legal (the name \(b\) is being used as an argument to a function). We infer that \(f\) has type (int array) × int → int.
   g. \((x < 3 ? x : F)\) is illegal because \(x\) and \(F\) have different types. (I.e., the expression doesn't have a fixed type because the types of its arms don't match.)

2. (Legal ways to represent states)
   a. \(\{x = 5, y = 2\}\) is legal
   b. \(\{x = five, y = one plus one\}\) is legal because “five” and “one” etc. refer to semantic objects.
   c. \(\{x = 5, y = x \text{ minus } 3\}\) is illegal: To be legal, “\(x\) minus 3” has to be a value, so “\(x\)” has to be a value (it has to be the name of a mathematical object like 5). But the binding \(x = 5\) tells us “\(x\)” is a variable that can appear in an expression, so “\(x\)” is a syntactic object. It can't be syntactic and semantic at the same time.
   Also, in this nicely word-processed document, “\(x\)” is presented in this font, so we know it's supposed to be a syntactic object. On paper, you see the difference between “\(x\)” and “\(x\)”. Even so, if someone wrote \(\{ x = 5, y = x \text{ minus } 1 \}\) on the blackboard*, it would have to be illegal because of using \(x\) in two incompatible ways.
   d. \(\{x = 5, y = \alpha - 3\}\) where \(\alpha = 5\) — is legal. We infer that symbols \(x\) and \(y\) are syntactic objects and \(\alpha\) is the name of the semantic object 5.
   e. \(\{x = 5, y = (\text{the value of } x \text{ in this environment, minus } 3)\}\) is legal. Since “the value of \(x\) in this environment” is just another name (albeit complicated) for the mathematical object 5, it's legal to use here.
   f. \(\{\}\) is legal, since it's just another way to write \(\emptyset\), the empty state.

3. (Graphically defined state)
   a. \(\sigma_2 = \{b = \beta, x = 12, y = T\}\) where \(\beta = (2, 4, 3, 8)\).
   \(\sigma_2 = \{b[0] = 2, b[1] = 4, b[2] = 3, b[3] = 8, x = 12, y = T\}\).
   \(\sigma_2 = \{b = \beta, x = 12, y = T\}\) where \(\beta = \{(0, 2), (1, 4), (2, 3), (3, 8)\}\).
   \(\sigma_2 = \{b = \beta, x = 12, y = T\}\) where \(\beta(0) = 2, \beta(1) = 4, \beta(2) = 3, \beta(3) = 8\).

* Using a felt marker, apparently :-)

b. You can write out these kinds of calculations to different levels of detail, but a brief answer is that \( \sigma_2(e) = \sigma_2(y \land x > b(x/5)) = T \land 12 > 3 = T \) You can certainly show intermediate steps:
\[
\begin{align*}
\sigma_2(e) &= \sigma_2(y) \land \sigma_2(x) > \sigma_2(b)(\sigma_2(x/5)) \\
&= T \land 12 > \sigma_2(b)(12/5) \\
&= T \land 12 > \sigma_2(b)(2) = T \land 12 > 3 = T
\end{align*}
\]

4. (Alternative ways to represent a state with an array value)
   a. \( \sigma_3 = \{ z = 4, b = (1, 5. 8) \} \)
   b.
   \[
   \begin{array}{ccc}
   \sigma_3 & 0 & 1 & 2 \\
   z & 4 & b & 1 & 5 & 8
   \end{array}
   \]
   c. We have \( e = b[b[z-4]] > z ? z+2 : z-2 \). To make this easier to deal with, let's break it down.
   Let \( e = e_1 > z ? z+2 : z-2 \) where \( e_1 = b[e_2] \) and \( e_2 = b[z-4] \).
   - First, \( \sigma_3(e_2) = \sigma_3(b[z-4]) = (\sigma_3(b))(\sigma_3(z-4)) = (\sigma_3(b))(4-4) = (\sigma_3(b))(0) = 1 \)
   - So \( \sigma_3(e_1) = \sigma_3(b[e_2]) = (\sigma_3(b))(\sigma_3(e_2)) = (\sigma_3(b))(1) = 5 \)
   - Then \( \sigma_3(e_1 > z) = (\sigma_3(e_1) > \sigma_3(z)) = 5 > 4 = F \)
   - So \( \sigma_3(e) = \sigma_3(e_1 > z ? z+2 : z-2) = \sigma_3(z+2) \) because the test \( \sigma_3(e_1 > z) = F \)
   - So finally, \( \sigma_3(e) = \sigma_3(z+2) = \sigma_3(z) + \sigma_3(2) = 4 + 2 = 6 \)

5. \( \sigma_4 = \{ z = 6, y = 33, x = 34 \} \)

6. (Proper states)
   a. (Well-formed and) Proper: The extra binding for \( z \) isn't a problem
   b. (Well-formed and) Proper: The value of \( b \) is an array of length 0.
   c. (Well-formed but) Improper: The value of \( b \) can't be an integer.
   d. (Well-formed but) Improper: We need a binding for \( y \) even though we're multiplying it by zero. [So our semantics uses eager evaluation, not lazy evaluation.]
   e. Ill-formed: We have two bindings for \( i \).
   f. (Well-formed and) Proper but causes a runtime error, since \( b \) has size 2.