Lecture 3: Types, Expressions, and Arrays

1. (Expression syntax & type)
   a. \((x < y \ ? \ x \ : \ F)\) is illegal: The types of the two clauses of the conditional don’t match
      \(x\) is an int, \(F\) is a boolean.
   b. \(b[0] + b[1][1]\) is illegal: \(b[0]\) needs one more index, since \(b\) is 2-dimensional
   c. \(\text{match}(b1, b2, n)\) is legal. From the comment, \text{match} returns a boolean.

2. (Well-formed states?)
   a. \(\{x = (2), y = 4\}\) is well-formed (\(b\) is an array of length 1)
   b. \(\{u = (3, 4), v = 0, w = u[1]\}\) is ill-formed; we need \(w = \text{value}\) but \(u[1]\) is an expression.
   c. \(\{r = \text{one}, s = \text{four}, t = r + s\}\) is ill-formed; the bindings of \(r\) and \(s\) are okay, but the
      binding \(t = \text{the expression } r + s\) is illegal. (Even if we don’t write it in this font, \(r + s\)
      has to be an expression because using \(r = \text{value}, s = \text{value}\) implies that \(r\) and \(s\) are
      expression variables.)

3. (Array representations) We have \(\sigma = \{x = 2, b = \beta\}\) where \(\beta = (\text{five, two plus two, 6})\).
   a. \(\sigma = \{x = 2, b = (5, 4, 6)\}\) is one way; writing \(\{\ldots, b = (\text{five, two plus two, 6})\}\) is okay too, since
      five, and two plus two must stand for semantic values.
   b. \(\sigma = \{x = 2, b[0] = 5, b[1] = 4, b[2] = 6\}\) is one way.

4. (State satisfying predicate) \(\varphi \equiv x = y^*z \land y = 3^*z \land z = b[0] + b[2] \land 3 < b[1] < b[2] < 6\)
   We’re to expand \(\sigma = \{x = \ldots, y = \ldots, z = 5, b = \ldots\}\) so that \(\sigma \models \varphi\). From \(z = 5\) we
   know \(y = 3 \times 5 = 15\), so \(x = 15 \times 5 = 75\). Since \(3 < b[1] < b[2] < 6\), we know \(b[1] = 4\) and \(b[2] = 5\),
   so \(z = b[0] + b[2]\) implies \(5 = b[0] + 5\) so \(b[0] = 0\). Altogether, we get \(\sigma = \{x = 75, y = 15, z = 5,\)
   \(b = (0, 4, 5)\}\).

5. (State and result for expression.) For a state to be proper for \(0^*b[b[j]]\), it has to have \(j\) an
   integer and \(b\) an array value. For the expression to use valid indexes for \(b\), we need the
   values of \(j\) and \(b[j]\) to be legal indexes for \(b\).
   a. \(\{j = 0, b = (3, 2, 5, 4), c = (3), d = 8\}\) is well-formed, proper, and evaluates yields zero.
c. \(|j = 0, b = 0| is well-formed, but not proper. (We need \(b = \text{an array value. If } b[0] \text{ is supposed to have the value } 0, \text{ then we need } b = (0) \text{ or } b[0] = 0\).)

**Lecture 4: State Updates, Satisfaction of Quantified Predicates**

6. (State updates) We have \(\sigma = \{x = 2, y = 4, b = (-1, 0, 4, 2)\}\).
   a. \(\sigma[z \mapsto 1] = \sigma \cup \{(z, 1)\}\) by definition because \(\sigma(z)\) is undefined.
   b. \(\sigma[x \mapsto 5] = \{x = 2, y = 4, b = (-1, 0, 4, 2)\}\) because \(\sigma(x)\) is defined. On the other hand, \(\sigma \cup \{(x, 5)\} = \{x = 2, y = 4, b = (-1, 0, 4, 2), x = 5\}\), which has two bindings for \(x\), so it is ill-formed.

7. (Q x . \(\varphi\) satisfaction)
   a. Let \(\sigma = \{x = 4, y = 6, b = (4, 2, 8)\}\), then \(\sigma \not\models (\exists x . \exists j . b[j] < x < y)\) using \(j = 0\) and \(x = 5\). The state \(\sigma[j \mapsto 0][x \mapsto 5] = \{x = 5, y = 6, b = (4, 2, 8)\}\) satisfies \(b[j] < x < y\), since it reduces to \(4 < 5 < 6\). It’s important to remember that updating \(\sigma\) so that \(x = 5\) replaces the \(x = 4\) binding of \(\sigma\).
   
   We can also use \(j = 1\) as a witness value: it works with \(x = 3, 4, \text{ or } 5\).
   b. Let \(\tau = \{x = 0, y = 7, b = (4, 2, 8)\}\), then \(\tau \not\models (\forall x . \forall k . 0 < k < 3 \rightarrow x < b[k])\) because \(\tau(x) = 0\) is irrelevant. From \(0 < k < 3\) we know \(k = 1\) or \(2\), but either way there are plenty of possible \(x\) values that are not \(< \tau(b)(1) = 4\) or not \(< \tau(b)(2) = 8\). (Note to show \(\tau \not\models (\forall x . \forall k . . . )\), we have to show that there is at least one set of counterexamples. I.e., for some \(x\) there is some \(k\) such that the body is not satisfied. The bindings \(x = 3\) and \(k = 1\) work: \(\tau[x \mapsto 3][k \mapsto 1] \not\models 0 < k < 3 \rightarrow x < b[k]\).

8. (Invalid Q x . \(\varphi\))
   a. \(\not\models (\forall x \in U . (\exists y \in V . P(x, y)))\) holds when there is some state \(\sigma\) and some value \(\alpha \in U\) for \(x\) where for every value \(\beta \in V\) for \(y\), the body \(P(x, y)\) is not satisfied. I.e., \(\sigma[x \mapsto \alpha][y \mapsto \beta] \not\models P(x, y)\). Note that if no variables with bindings in \(\sigma\) are used in \(P(x, y)\), then we can take \(\sigma = \emptyset\).
   b. \(\not\models \forall y . ((\exists x \in U . P(x, y)) \rightarrow (\exists y \in U . Q(x, y)))\) means \(\not\models \forall y . (p_1 \rightarrow p_2)\) where \(p_1 = \exists x \in U . P(x, y)\) and \(p_2 = \exists y \in U . Q(x, y)\).
   For \(\not\models \forall y . (p_1 \rightarrow p_2)\), we need a state \(\sigma\) and a value \(\alpha\) such that \(\sigma[y \mapsto \alpha] \models p_1\) but also \(\sigma[y \mapsto \alpha] \not\models p_2\).
For $\sigma[y \mapsto \alpha] = p_1 = \exists x \in U. \ P(x, y)$, we need a $\beta \in U$ such that $\sigma[y \mapsto \alpha][x \mapsto \beta] \models P(x, y)$. (I.e., $P$ is true on values $\beta$ and $\alpha$ for $x$ and $y$.)

For $\sigma[y \mapsto \alpha] \not\models p_2 = \exists y \in U. \ Q(x, y)$, we need that for all $\delta \in U$, $\sigma[y \mapsto \alpha][y \mapsto \delta] \not\models Q(x, y)$. Since $\sigma[y \mapsto \alpha][y \mapsto \delta] = \sigma[y \mapsto \delta]$, we’re saying that $Q$ is false about values $\sigma(x)$ and $\delta$. (The reason we use $\delta$ instead of $\sigma(y)$ for $y$ is that the $y$ in $p_2 = \exists y...$ hides the $y$ in the outer $\forall y. (... \rightarrow ...) $)

9. (Predicate function) To make life easier, first I’ll define a helper predicate $R(x, y)$ that is true if both $x$ and $y$ are legal indexes for $b$: $R(x, y) = 0 < x < n \land 0 < y < n$.

Then we can define $P(b, c, d, s, t) = R(c, d) \land R(s, t) \land \forall c \leq i < d. \exists s \leq j < t. b[i] < b[j]$.

In English, this says that $c$ and $d$ are legal indexes, $s$ and $t$ are legal indexes, and for all indexes $i$ between $c$ and $d$, there is some index $j$ between $s$ and $t$ such that $b[i] < b[j]$. 