Solution - HW 2 - Types, Expressions, States, Quantified Predicates

CS 536: Science of Programming, Fall 2019

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. `match(b1, b2, n)` is legal. From the comment, `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` = *value* but `u[1]` is an expression.
   c. { `r` = *one*, `s` = *four*, `t` = `r + s` } is ill-formed; the bindings of `r` and `s` are okay, but the binding `t` = 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 = *value*, s = *value* implies that r and s are expression variables.)

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

4. (State satisfying predicate) $\varphi \equiv x = y*z \wedge y = 3*z \wedge z = b[0] + b[2] \wedge 3 < b[1] < b[2] < 6$

   We're to expand σ = {`x` = ____, `y` = ____, `z` = 5, `b` = __________ } so that $\sigma \models \varphi$. From z = 5 we know `y` = 3 × 5 = 15, so `x` = 15 × 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 σ = {`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 = an array value. If b[0] is supposed to have the value 0, then we need b = (0) 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. $\sigma[x \mapsto 5] = \{ x = 2, y = 4, b = (-1, 0, 4, 2) \} [x \mapsto 5] = \{ x = 5, y = 4, b = (-1, 0, 4, 2) \} [x \mapsto 5]$ 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 $\sigma \vDash (\exists x\,.\,\exists j\,.\,b[j] < x < y)$ using j = 0 and x = 5. The state $\sigma[j \mapsto 0]\,[x \mapsto 5] = \{x = 5, y = 6, b = (4, 2, 8), j = 0\}$ satisfies $b[j] < x < y$, since it reduces to $4 < 5 < 6$. It's important to remember that updating σ so that x = 5 replaces the x = 4 binding of σ.

      We can also use j = 1 as a witness value: it works with x = 3, 4, or 5.

   b. Let $\tau = \{x = 0, y = 7, b = (4, 2, 8)\}$, then $\tau \nvDash (\forall x\,.\,\forall k\,.\,0 < k < 3 \rightarrow x < b[k])$ because to satisfy $\forall x\,.\,\forall k\,.\ldots$, the value $\tau(x) = 0$ 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 $\tau \nvDash (\forall x\,.\,\forall k\,.\ldots)$, 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: $\tau[x \mapsto 3][k \mapsto 1] \nvDash 0 < k < 3 \rightarrow x < b[k]$.

8. (Invalid $Q\,x\,.\,\varphi$)

   a. $\nvDash (\forall x \in U\,.\,(\exists y \in V\,.\,P(x, y)))$ holds when there is some state σ 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., $\sigma[x \mapsto \alpha]\,[y \mapsto \beta] \nvDash P(x, y)$. Note that if no variables with bindings in σ are used in $P(x, y)$, then we can take $\sigma = \varnothing$.

   b. $\nvDash \forall y\,.((\exists x \in U\,.\,P(x, y)) \rightarrow (\exists y \in U\,.\,Q(x, y)))$ means $\nvDash \forall y.\,(p_1 \rightarrow p_2)$ where $p_1 \equiv \exists x \in U\,.\,P(x, y)$ and $p_2 \equiv \exists y \in U\,.\,Q(x, y)$.

      For $\nvDash \forall y.\,(p_1 \rightarrow p_2)$, we need a state σ and a value α such that $\sigma[y \mapsto \alpha] \vDash p_1$ but also $\sigma[y \mapsto \alpha] \nvDash p_2$.

For $\sigma[y \mapsto \alpha] \vDash p_1 \equiv \exists\, x \in U\,.\; P(x, y)$, we need a $\beta \in U$ such that $\sigma[y \mapsto \alpha][x \mapsto \beta] \vDash P(x, y)$. (I.e., P is true on values β and α for x and y.)

For $\sigma[y \mapsto \alpha] \nvDash p_2 \equiv \exists\, y \in U\,.\; Q(x, y)$, we need that for all $\delta \in U$, $\sigma[y \mapsto \alpha]\,[y \mapsto \delta] \nvDash Q(x, y)$. Since $\sigma[y \mapsto \alpha]\,[y \mapsto \delta] = \sigma[y \mapsto \delta]$, we're saying that Q is false about values $\sigma(x)$ and δ. (The reason we use δ instead of $\sigma(y)$ for y is that the y in $p_2 \equiv \exists\, y\ldots$ hides the y in the outer $\forall y$. $(\ldots \to \ldots)$

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: $R(x, y) \equiv 0 < x < n \wedge 0 < y < n$.

   Then we can define $P(b, c, d, s, t) \equiv R(c, d) \wedge R(s, t) \wedge \forall\, c \le i < d\,.\,\exists\, s \le j < t\,.\, b[i] < b[j]$. In English, this says that c and d are legal indexes, r and s 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].