A. Why

• A predicate is satisfied or unsatisfied relative to a state.
• A predicate is valid if it is satisfied in all states.
• State updates occur when we introduce new variables or change the values of existing variables.

B. Outcomes

At the end of today, you should

• Know how to check a predicate for satisfaction in a state, how to check a predicate for validity, and know how to update a state.

C. Questions

1. Say u and v stand for variables (possibly the same variable) and α and β are values (possibly equal). When is \( \sigma[u \mapsto \alpha][v \mapsto \beta] = \sigma[v \mapsto \beta][u \mapsto \alpha] \)? Hint: There are four cases because maybe \( u \equiv v \) and maybe \( \alpha = \beta \).

2. Let \( \sigma(b) = (7, 5, 12, 16) \). Assume out-of-bound indexes cause runtime errors.
   a. Does \( \sigma \models \exists k . 0 \leq k \land k + 1 < \text{size}(b) \land b[k] < b[k + 1] \)? If so, what was your witness value for \( k \)?
   b. Does \( \sigma \models \exists k . 0 \leq k - 1 \land k + 1 < \text{size}(b) \land b[k - 1] < b[k] < b[k + 1] \)? If so, what was your witness value for \( k \)?
   c. Does \( \sigma \models \forall k . 0 \leq k < 4 \rightarrow b[k] > 0 ? \)
   d. If \( \sigma(k) = -5 \), then does \( \sigma \models \exists k . b[k] > 0 ? \)

3. For each of the situations below, fill in the blanks to describe when the situation holds.
   Fill in _____ ₁ with “some”, “every”, or “this”
   Fill in _____ ₂ with “some” or “every”
   Fill in _____ ₃ with “\( \sigma(x) \) must be undefined”, “\( \sigma(x) \) must be defined and \( \sigma \models p \)”, or “nothing of \( \sigma(x) \)”
   Fill in _____ ₄ with “\( \models p \)” or “\( \not\models p \)”
   a. \( \sigma \models (\exists x \in U. p) \) iff for _____ ₁ state \( \sigma \) and _____ ₂ \( \alpha \in U, \sigma[x \mapsto \alpha] _____ ₄ \)
   b. \( \sigma \models (\forall x \in U. p) \) iff for _____ ₁ state \( \sigma \) and _____ ₂ \( \alpha \in U, \sigma[x \mapsto \alpha] _____ ₄ \)
   c. \( \sigma \models (\exists x \in U. p) \) requires _____ ₃ .
   d. \( \sigma \models (\forall x \in U. p) \) requires _____ ₃ .
   e. \( \sigma \not\models (\exists x \in U. p) \) iff for _____ ₁ state \( \sigma \) for _____ ₂ \( \alpha \in U, \sigma[x \mapsto \alpha] _____ ₄ \)

© James Sasaki, 2021
f. \( \sigma \not\models (\forall x \in U. p) \) iff for \( \alpha \in U \), \( \sigma[x \mapsto \alpha] \) __________

g. \( \not\models (\forall x \in U. p) \) iff for \( \alpha \in U \), we have \( \sigma \) __________ \( (\forall x \in U. p) \).

h. \( \not\models (\exists x \in U. p) \) iff for \( \alpha \in U \), we have \( \sigma \) __________ \( (\exists x \in U. p) \).

i. \( \not\models (\forall x \in U. p) \) iff for \( \alpha \in U \), and for \( \alpha \in U \), we have \( \sigma[x \mapsto \alpha] \) __________

j. \( (\exists x \in U. (\forall y \in V. p)) \) iff for \( \alpha \in U \), and for \( \beta \in V \), we have \( \sigma \) __________ \( (\forall x \in U. p) \).

k. \( \not\models (\exists x \in U. (\forall y \in V. p)) \) iff for \( \alpha \in U \), and for \( \beta \in V \), we have \( \sigma \) __________ \( (\forall x \in U. p) \).

l. \( (\forall x \in U. (\exists y \in V. p)) \) iff for \( \alpha \in U \), and for \( \beta \in V \), we have \( \sigma \) __________ \( (\exists y \in V. p) \).

m. \( \not\models (\forall x \in U. (\exists y \in V. p)) \) iff for \( \alpha \in U \), and for \( \beta \in V \), we have \( \sigma \) __________

4. Let \( p = \exists y. \forall x. f(x) > y \) and \( q = \forall x. \exists y. f(x) > y \). (As usual, assume a domain of \( \mathbb{Z} \).)

a. Is it the case that (regardless of the definition of \( f \)), if \( p \) is valid then so is \( q \)? If so, explain why. If not, give a definition of \( f(x) \) and show \( \models p \) but \( \not\models q \).

b. (The converse.) Is it the case that (regardless of the definition of \( f \)), if \( q \) is valid then so is \( p \)? If so, explain why. If not, give a definition of \( f(x) \) and show \( \models q \) but \( \not\models p \).
CS 536: Solution to Activity 4 (Satisfaction, Validity, and State Updates)

1. \( \sigma[u \mapsto \alpha][v \mapsto \beta] = \sigma[v \mapsto \beta][u \mapsto \alpha] \) iff \( u \neq v \) or \( u = v \) and \( \alpha = \beta \). Another way to phrase this is \( \alpha = \beta \) or \( u \neq v \)

2. (Quantified statements over arrays) Let \( \sigma(b) = (7, 5, 12, 16) \).
   a. Yes, \( \sigma \models \exists k . 0 \leq k \land k+1 < \text{size}(b) \land b[k] < b[k+1] \) with 1 and 2 as possible witnesses for \( k \).
   b. Yes, \( \sigma \models \exists k . 0 \leq k-1 \land k+1 < \text{size}(b) \land b[k-1] < b[k] < b[k+1] \) with 2 as the only witness that works.
   c. Yes, \( \sigma \models \forall k . b[k] > 0 \)
   d. Yes, if \( \sigma(k) = -5 \), we still have \( \sigma \models \exists k . b[k] > 0 \), with witnesses 0, 1, 2, 3. The key is that for \( \sigma \) to satisfy the existential with witness call it \( \alpha \), then we need \( \sigma[k \mapsto \alpha] = b[k] > 0 \), which doesn't depend on \( \sigma(k) \) because the update of \( \sigma \) uses \( k = \alpha \), not \( k = \) whatever \( \sigma(k) \) happens to be.

   Here's a step-by-step explanation (this is way too much detail for appearing on a test):

   \[
   \begin{align*}
   \sigma[k \mapsto \alpha] &= b[k] > 0 \\
   \text{iff } \sigma[k \mapsto \alpha](b[k]) &= \sigma[k \mapsto \alpha](0) & \text{defn state } \models \text{ relational test} \\
   \text{iff } (\sigma[k \mapsto \alpha](b))(\sigma[k \mapsto \alpha](k)) &= 0 & \text{the value of 0 is zero} \\
   \text{iff } (\sigma(b))(\sigma[k \mapsto \alpha](k)) &= 0 & \sigma[k \mapsto \alpha](b) = \sigma(b) \text{ because } b \neq k \\
   \text{iff } (\sigma(b))(\alpha) &= 0 & \sigma[k \mapsto \alpha](k)) = \alpha \\
   \text{iff } 7, 5, 12, \text{or } 16 &= 0 & \text{depending on } \alpha = 0, 1, 2, \text{or } 3
   \end{align*}
   \]

3. (Validity/invalidity of quantified predicates)
   a. this \( \sigma \), some \( \alpha \), \( \models p \)
   b. this \( \sigma \), every \( \alpha \), \( \models p \)
   c. nothing of \( \sigma(x) \)
   d. nothing of \( \sigma(x) \)
   e. this \( \sigma \), every \( \alpha \), \( \not\models p \)
   f. this \( \sigma \), some \( \alpha \), \( \not\models p \)
   g. some \( \sigma \), \( \not\models \forall x \in U. \ p \)
   h. some \( \sigma \), every \( \alpha \), \( \not\models p \)
   i. some \( \sigma \), some \( \alpha \), \( \not\models p \)
   j. every \( \sigma \), some \( \alpha \), every \( \beta \), \( \models p \)
   k. some \( \sigma \), every \( \alpha \), some \( \beta \), \( \not\models p \)
   l. every \( \sigma \), every \( \alpha \), some \( \beta \), \( \models p \)
   m. some \( \sigma \), some \( \alpha \), every \( \beta \), \( \not\models p \)
4. (∃ ∀ predicates versus ∀ ∃ predicates, specifically \( p = \exists y \ . \forall x \ . f(x) > y \), and \( q = \forall x \ . \exists y \ . f(x) > y \))

   a. The relation does hold: \( \models p \) implies \( \models q \). The short explanation is that for each value \( \alpha \) for \( x \), we need to find a value \( \beta \) for \( y \) that satisfies the body, but \( p \) says that there's a value that works for every \( \alpha \), so we can use that value for \( \beta \). In more detail, assume \( p \) is valid: for every state \( \sigma \), there is some value \( \beta \) where for every value \( \alpha \), \( \sigma[y \rightarrow \beta][x \rightarrow \alpha] \models f(x) > y \). To show that \( q \) is valid, take an arbitrary state \( \tau \) with value \( \alpha \) for \( x \). We need a witness value for the \( \exists y \); using \( p \) with \( \tau \) for \( \sigma \), we get a \( \beta \) for the \( \exists y \) of \( p \) and use that as the witness for the \( \exists y \) in \( q \). So then we need \( \tau[x \rightarrow \alpha][y \rightarrow \beta] \models f(x) > y \). Substituting \( \sigma \) for \( \tau \) and swapping the order of the updates, we need \( \sigma[y \rightarrow \beta][x \rightarrow \alpha] \models f(x) > y \). But that's exactly what \( p \) provided.

   b. The relation does not hold: We can have \( \models q \) but \( \not\models p \). The easiest example is \( f(x) = x \), then validity of \( p \) would require us to find an integer value for \( y \) that is > every possible integer value of \( x \), but no such value exists.

   As an aside, you can use an arbitrary predicate over \( x \) and \( y \) instead of \( f(x) > y \) as the body of the \( \exists ∀ \) and \( ∀ ∃ \) predicates. I use \( f(x) > y \) here just because it's nice and concrete.