State Updates, Satisfaction of Quantified Predicates

CS 536: Science of Programming, Fall 2022

Updated 2022-09-01

A. Why?

- A predicate is satisfied relative to a state; it 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 this class, you should

- Know what it means to update a state.
- Know what it means for a quantified predicate to be valid of be satisfied in a state.

C. "Updating" States

- To check quantified predicates for satisfaction, we need to look at different states that are related to, but not identical to, our starting state.
- ***Example 1***: For $\{y = 1\} \vDash \forall x \in \mathbb{Z} \,.\, x^2+1 \geq y-1$, we need to know that $\{y = 1, x = \alpha\} \vDash x^2+1 \geq y-1$ for every $\alpha \in \mathbb{Z}$. I.e., we need to know that
    - ....
    - $\{y = 1, x = -1\} \vDash x^2+1 \geq y-1$
    - $\{y = 1, x = 0\} \vDash x^2+1 \geq y-1$
    - $\{y = 1, x = 1\} \vDash x^2+1 \geq y-1$
    - $\{y = 1, x = 2\} \vDash x^2+1 \geq y-1$
    - ....
  - Similarly, for $\{z = 4\} \vDash \exists x \in \mathbb{Z} \,.\, x \geq z$, we need $\{z = 4, x = \alpha\} \vDash x \geq z$ for some particular integer α ($\alpha = 5$ works nicely).
- There is a complicating factor. If the quantified variable already appears in the state, then we need to ***replace*** its binding with one that gives the value we're interested in checking.
- ***Example 2***: We already know $\{z = 4\} \vDash \exists x \in \mathbb{Z} \,.\, x \geq z$ because $\{z = 4, x = 5\} \vDash x \geq z$. If we start with the state $\{z = 4, x = -15\}$, which already has a binding for x, we ignore it because at the time we test for satisfaction of $x \geq z$, we're using $z = 4$. In other words, we test for $\{z = 4, x = 5\} \vDash x \geq z$ regardless of whether we started with a value for x or not (i.e., $\{z = 4, x = -15\}$ or $\{z = 4\}$).

- In ***Example*** 2, the x that appears in $\{z = 4, x = 5\}$ is not the same x that appears within $\exists x \in \mathbb{Z} . x \geq z$. However, the two x's in "$\{z = 4, x = 5\} \vDash x \geq z$" ***are*** the same x. Giving the two x's the same name causes the confusion. We would have to give the x's different names:: Let xo be the "outer" x and xi be the "inner" x, then

$$\{z = 4, xo = -15\} \vDash \exists xi \in \mathbb{Z} . xi \geq z$$

because

$$\{z = 4, xo = -15, xi = 5\} \vDash xi \geq z$$

- But there really isn't any use for us to keep xo because there's no way to access it. We would need it more complicated languages where you have code that can take the then-current xo and save it for later use.

- ***Definition***: For any state σ, variable x, and value α, the ***update**** ***of*** σ ***at*** x ***with*** α (written $\sigma[x \mapsto \alpha]$) is the state that is a copy of σ except that it binds variable x to value α.
  - Let $\tau = \sigma[x \mapsto \alpha]$, then $\tau(x) = \alpha$; if variable $y \not\equiv x$, then $\tau(y) = \sigma(y)$.
  - Note $\tau(x) = \alpha$ regardless of whether $\sigma(x)$ is defined or not. If $\sigma(x)$ is defined, its type and exact value are irrelevant.
- Set theoretically,
  - If x has no binding in σ, then $\sigma[x \mapsto \alpha]$ is $\sigma \cup \{x = \alpha\}$: It's like σ but has been extended with $x = \alpha$.
  - [2022-09-01] If x has a binding in σ, say $\sigma = \{x = \beta\} \cup \sigma_0$ where σ_0 is the rest of σ, then $\sigma[x \mapsto \alpha]$ is $\sigma_0 \cup \{x = \alpha\}$. It's like σ but has the binding $x = \alpha$, not $x = \beta$. (Having two bindings for x would be illegal.)
- ***Important***: Calling it the "update" of σ is kind of misleading because we're not modifying σ.
  - Taking $\sigma[x \mapsto \alpha]$ ***does not do*** an update in place; if we define $\tau = \sigma[x \mapsto \alpha]$, then σ is still σ.
  - Conceptually, we aren't modifying σ, we're looking at a state much like it.
- We're not required to give $\sigma[x \mapsto \alpha]$ a new name; we can write it out explicitly:
  - If $x \equiv v$ where v stands for a variable (not literally the variable v) then if $v \equiv x$, then $\sigma[x \mapsto \alpha](v) = \sigma[x \mapsto \alpha](x) = \alpha$, otherwise (if $x \not\equiv v$), then $\sigma[x \mapsto \alpha](v) = \sigma(v)$.
  - (You have to read $\sigma[x \mapsto \alpha](v)$ left-to-right — we're taking the function $\sigma[x \mapsto \alpha]$ and applying it to v. I.e., $\sigma[x \mapsto \alpha](v) = (\sigma[x \mapsto \alpha])(v)$, where the left pair of parentheses are for grouping and the ones around v are for the function call.)
- ***Example 3***: If $\sigma = \{x = 2, y = 6\}$, then $\sigma[x \mapsto 0] = \{x = 0, y = 6\}$, so
  - $\sigma[x \mapsto 0](x) = 0$ (Even though $\sigma(x) = 2$)
  - $\sigma[x \mapsto 0](y) = \sigma(y) = 6$ (Since we didn't update y)
  - $\sigma[x \mapsto 0](x+y) = 0+6 = 6$ (Since the x in $x+y$ gets evaluated to 0)
  - $\sigma[x \mapsto 0] \vDash x^2 \leq 0$ (Even though our starting $\sigma \nvDash x^2 \leq 0$)

---

* Unfortunately, "update" is the traditional name, and for myself, I can't find any word that's exactly right. We're not always *extending* σ, we're not always *superseding* σ, ….

- The value part of an update has to be a semantic value, not a syntactic one, so if you wanted to add one to x, you can't use "$\sigma[x \mapsto x+1]$" because it isn't well-formed (the x on the left side of $\mapsto$ must be syntactic, the x on the right side of $\mapsto$ has to be semantic, and x can't be both).
  - On the other hand, "$\sigma[x \mapsto \sigma(x+1)]$" or "σ[$x$ ↦ α plus one] where $\alpha = \sigma(x)$" do make sense.

Multiple Updates

- We can do a sequence of updates on a state. E.g., $\sigma[x \mapsto 0][y \mapsto 8]$ is a doubly updated state. Sequences of updates are read left-to-right, so this is $(\sigma[x \mapsto 0])[y \mapsto 8]$.
- ***Example 4***: If $\sigma = \{x = 2, y = 6\}$, then $\sigma[x \mapsto 0][y \mapsto 8] = \{x = 0, y = 6\}[y \mapsto 8] = \{x = 0, y = 8\}$.
- ***Example 5***: $\sigma[x \mapsto 0][y \mapsto 8] = \sigma[y \mapsto 8][x \mapsto 0]$ because he order of update doesn't matter if you have two different variables.
- ***Example 6***: $\sigma[x \mapsto 0][x \mapsto 17] = \sigma[x \mapsto 17] \neq \sigma[x \mapsto 17][x \mapsto 0] = \sigma[x \mapsto 0]$: If you update the same variable twice, the second update supersedes the first.
- Of course, if the second update is identical to the first, nothing happens: $\sigma[x \mapsto \alpha][x \mapsto \alpha] = \sigma[x \mapsto \alpha]$
- If you have to evaluate an expression, be sure to do it in the correct state.
  - Let $\sigma(x) = 1$ and let $\tau = \sigma[x \mapsto 2]$, then $\tau[z \mapsto \sigma(x)+10]$ maps z to $\sigma(x)+10 = 1+10 = 11$. We can omit τ and also write $\sigma[x \mapsto 2][z \mapsto \sigma(x)+10]$, which gives the same state as τ.
  - On the other hand, look at $\tau[z \mapsto \tau(x)+10]$. Since $\tau = \sigma[x \mapsto 2]$, the value of $\tau(x)+10 = 12$, so $\tau[z \mapsto \tau(x)+10] = \tau[z \mapsto 12]$.
  - If we hadn't given the name $\tau = \sigma[x \mapsto 2]$, then we would had to write $\sigma[x \mapsto 2][z \mapsto \sigma[x \mapsto 2](x)+10]$. (This is pretty ugly, so giving $\sigma[x \mapsto 2]$ a name like τ makes things more readable.)

D. Updating Array Values

- Updating array elements like *b[0]* is a bit more complicated than updating simple variables like x and y. First, let's extend our notion of updating states to updating general functions.
- ***Definition***: If δ is a function on one argument and α and β are valid members of the domain and range of δ respectively, then the ***update of*** δ ***at α with β***, written $\delta[\alpha \mapsto \beta]$, is the function defined by $\delta[\alpha \mapsto \beta](\gamma) = \beta$ if $\gamma = \alpha$ and $\delta[\alpha \mapsto \beta](\gamma) = \delta(\gamma)$ if $\gamma \neq \alpha$. The name α should be a semantic constant (like $\underline{0}$ or $\underline{\text{zero}}$).
- ***Definition***: If σ is a (proper) state for an array b and α is a valid index value for b, then $\sigma[b[\alpha] \mapsto \beta]$ means $\sigma[b \mapsto \eta[\alpha \mapsto \beta]]$ where η = the function $\sigma(b)$. In words, if σ includes the binding b = function η, then the updating σ at $b[\alpha]$ with β is just like updating σ at b with an updated version of η, namely $\eta[\alpha \mapsto \beta]$.
- ***Example 7***: Say $\sigma = \{x = 3, b = (2, 4, 6)\}$, then $\sigma[b[0] \mapsto 8] = \{x = 3, b = (8, 4, 6)\}$. Here, $\sigma(b)$ is the function (2, 4, 6) (which means {(0, 2), (1, 4), (2, 6)]), so $\sigma(b)[0 \mapsto 8]$ (the update of function $\sigma(b)$) is the function $(2, 4, 6)[0 \mapsto 8] = (8, 4, 6)$.
- The notation $\sigma[b[\alpha] \mapsto \beta]$ is a bit of a hack: The name b is syntactic but α is semantic. The restriction that α be a constant like $\underline{0}$ or $\underline{\text{zero}}$ avoids the complications that result if you allow α to

be the name for a complicated semantic expression like $\tau(e)$. The intuition is that α models the memory offset from b[0] that we need to find in order to do the update.

E. Satisfaction of Quantified Predicates

- One use of updated states is for describing how assignment works. (We'll see this later.) The other use for updated states is for defining when quantified predicates are satisfied.
- ***Definition***: $\sigma \vDash \exists\, x \in S \,.\, p$ if for one or more ***witness*** values $\alpha \in S$, it's the case that $\sigma[x \mapsto \alpha] \vDash p$. Note we're asking a hypothetical question: "If we were to calculate $\sigma[x \mapsto \alpha]$, would we find that it satisfies p?"
    - ***Example 8a***: For any state σ, we can show $\sigma \vDash \exists\, x \,.\, x^2 \leq 0$ using 0 as the witness: $\sigma[x \mapsto 0] \vDash x^2 \leq 0$, since $\sigma[x \mapsto 0](x^2 \leq 0) = \sigma[x \mapsto 0](x^2) \leq \sigma[x \mapsto 0](0) = (0^2 \leq 0) = \mathrm{T}$.
- Remember, $\sigma(x)$ is irrelevant, since $\sigma[x \mapsto \alpha]$ overrides any value for $\sigma(x)$.
    - ***Example 8b***: If $\sigma(x)$ is, say 5, it's still the case that $\sigma \vDash \exists\, x . x^2 \leq 0$ using 0 as the witness because we $\sigma[x \mapsto 0] \vDash x^2 \leq 0$, regardless of $\sigma(x) = 5$.
- If there are many successful witness values, we don't have to specify all of them; we just need one.
    - [2022-09-01] ***Example 9***: If $\sigma(y) = 3$, then $\sigma \vDash \exists\, x \,.\, x^2 \leq y$ with $x = 0$ or 1 (or -1) as possible witness values.
- ***Definition***: $\sigma \vDash \forall\, x \in S \,.\, p$ if for every value $\alpha \in S$, we have $\sigma[x \mapsto \alpha] \vDash p$. (Again, this is hypothetical: "If for every α, we were to calculate $\sigma[x \mapsto \alpha]$, would we find that it satisfies p?"
    - ***Example 10***: To know $\sigma \vDash \forall\, x \in \mathbb{Z} \,.\, x^2 \geq x$, we need to know $\sigma[x \mapsto \alpha] \vDash x^2 \geq x$ for every $\alpha \in \mathbb{Z}$. Since for every integer α, indeed α^2 is $\geq \alpha$, this does hold. Recall that it doesn't matter what $\sigma(x)$ is, since we're interested in $\sigma[x \mapsto \alpha]$.
- When asking if σ satisfies $\forall\, x \in S \,.\, q$ or $\exists\, x \in S \,.\, q$, we don't care about $\sigma(x)$. For a predicate p in general, for the question "Does $\sigma \vDash p$?" only depends on how σ operates on the non-quantified variables of p.
    - ***Example 11***: Since the body of $\forall\, x \in \mathbb{Z} \,.\, x^2 \geq x$ uses only the quantified variable x, it doesn't matter what bindings σ has when checking $\sigma \vDash \forall\, x \in \mathbb{Z} \,.\, x^2 \geq x$. Even $\sigma = \varnothing$ works: $\varnothing \vDash \forall\, x \in \mathbb{Z} \,.\, x^2 \geq x$.
- Note with nested quantifiers, the notation does get more complicated.
- [2022-09-01] ***Example 12***: $\sigma \vDash \forall\, x \,.\, x > y^2 \rightarrow \exists\, z . z \geq x+y^2$ iff (for every $\alpha \in \mathbb{Z}$, if $\alpha > \sigma(y)^2$, then there is some $\beta \in \mathbb{Z}$ such that $\beta \geq \alpha + \sigma(y)^2$).

    $\sigma \vDash \forall\, x > y^2 \,.\, \exists\, z . z \geq x+y^2$

iff $\sigma \vDash \forall\, x . x > y^2 \rightarrow \exists\, z . z \geq x+y^2$	defn bounded $\forall$
iff for every $\alpha \in \mathbb{Z}$, $\sigma[x \mapsto \alpha] \vDash x > y^2 \rightarrow \exists\, z . z \geq x+y^2$,	defn $\vDash \forall$

    - Now, $\sigma[x \mapsto \alpha] \vDash x > y^2 \rightarrow \exists\, z . z \geq x+y^2$

iff $\sigma[x \mapsto \alpha] \vDash x > y^2$ implies $\sigma[x \mapsto \alpha] \vDash \exists\, z . z \geq x+y^2$	defn $\vDash \rightarrow$
iff $\gamma = \sigma(y)$ and $\alpha > \gamma^2$ implies $\sigma[x \mapsto \alpha] \vDash \exists\, z . z \geq x+y^2$	~~where $\gamma = \sigma(y)$~~

iff $y = \sigma(y)$ and $\alpha > y^2$ implies for some β, $\sigma[x \mapsto \alpha][z \mapsto \beta] \vDash z \geq x+y^2$ defn $\vDash \exists$
iff $y = \sigma(y)$ and $\alpha > y^2$ implies for some β, $\beta \geq$ to $\alpha+y^2$ defn $\vDash \geq$

- Taking $\beta = 2\alpha$ for our witness value, we need $\alpha > y^2$ implies $2\alpha \geq \alpha+y^2$, which is true.
- Note defining intermediate names like "let $\tau = \sigma[x \mapsto \alpha][z \mapsto \beta]$" is allowed, if you wish.

Justifying DeMorgan's Laws for Quantified Predicates

- In general, we want our systems of reasoning to be ***sound***: We want the textual transformations that make up logical equivalence to reflect truths about how our semantics work.
- ***Example 15***: Here is a check of DeMorgan's law for existentials, which says $\neg\exists x.p \Leftrightarrow \forall x.\neg p$. Semantically, we want each of these to be valid if and only if the other is. So we need $\sigma \vDash \neg\exists x.p$ if and only if $\sigma \vDash \forall x.\neg p$.

  $\sigma \vDash \neg\exists x \in S.p$

iff $\sigma \nvDash \exists x.p$	defn of $\sigma \vDash \neg$predicate
[2022-09-01]	
iff not (there is an $\alpha \in S$ do such that $\sigma[x \mapsto \alpha] \vDash p$)	defn of $\sigma \vDash$ existential
iff for no $\alpha \in S$ do we have $\sigma[x \mapsto \alpha] \vDash p$	(rephrasing)
iff for every $\alpha \in S$ we have $\sigma[x \mapsto \alpha] \nvDash p$	equivalence of "no $\vDash$" vs "every $\nvDash$"
iff for every $\alpha \in S$ we have $\sigma[x \mapsto \alpha] \vDash \neg p$	defn of $\sigma \vDash \neg$predicate
iff $\sigma \vDash \forall x.\neg p$	defn of $\sigma \vDash$ universal.

- Showing the semantic property that $\vDash \neg\exists x.p \leftrightarrow \forall x.\neg p$ gives us a justification for adding $\neg\exists x.p \Leftrightarrow \forall x.\neg p$ as a proof rule.

- [2022-09-01] In class
  - Validity: $\vDash p$ means $\sigma \vDash p$ for all σ — "p is valid"
    - $\vDash x + 1 > x$
  - Not valid $\nvDash p$ means for some σ, $\sigma \nvDash p$ — σ is the "counterexample"
    - $\nvDash x^2 > 0$
  - If $\vDash \forall x . p$ means "for all σ, $\sigma \vDash \forall x . p$"
  - then $\nvDash \forall x . p$ means "for some σ, $\sigma \nvDash \forall x . p$"
    - $\nvDash \forall x . x^2 > y$ because $\{y = 5\} \nvDash \forall x . x^2 > y$
    - $\nvDash \forall x . x^2 > y$ means "for some σ, σ, there is no β such that $\sigma[x \mapsto \beta] \vDash x^2 > y$"
      - counterexample was $\sigma = \{y = 5\}$
  - $\nvDash \forall x. \exists y. p(x, y)$ means
    - For some σ, $\sigma \nvDash \forall x. \exists y. p(x, y)$
    - for some σ, for some α, $\sigma[x \mapsto \alpha] \nvDash \exists y. p(x, y)$
    - for some σ, for some α, for all β, $\sigma[x \mapsto \alpha][y \mapsto \beta] \nvDash p(x, y)$