Syntactic Substitution

CS 536: Science of Programming, Spring 2023

A. Why

- Syntactic substitution is used in the assignment rules to calculate weakest preconditions (and later, strongest postcondition).

B. Objectives

At the end of today's class you should

- Know what syntactic substitution is and how to do it.
- Be able to carry out substitution on an expression or predicate.

C. Syntactic Substitution

- Recall that $wp(v := e, P(v)) \equiv P(e)$
- The operation of going from $P(v)$ to $P(e)$ is called ***syntactic substitution***.
- A common notation is $p[e/v]$. The advantage of this notation is that it's easier to do multiple ("iterated") substitutions. There are other notations people use, such as $p[v := e]$, $p[v \mapsto e]$, and p^{e}_{v}.

D. Substitution Into An Expression

- As part of substitution into a predicate, we need to be able to ***substitute into an expression***; the idea is to take an expression e and replace its occurrences of variable v with expression e'.
- ***Notation***: We write $e[e'/v]$, pronounced "e with e' (substituted) for v". We'll treat the substitution brackets as having very high precedence, so we'll need parentheses around e for complex expressions.
- ***Example 1***: $x + y[5/x] \equiv x + (y[5/x]) \equiv x + y$ but $(x + y)[5/x] \equiv 5 + y$.
- For the language at hand, substitution into expressions is very simple because we don't have anything that introduces a local variable (like ***let*** $x = e_1$ ***in*** e_2).
- To carry out $e[e'/v]$, we go through e. Everywhere we see an occurrence of v, we replace it by (e'). If the parentheses are redundant, we can omit them.
  - If e has no occurrence of v (there's no v to replace), then $e[e'/v] \equiv e$. Another way to say this is that if e only uses variables $\neq v$, then $e[e'/v] \equiv e$.
- Note: Substitution is a textual operation. For example, $(x + x)[2/x] \equiv 2 + 2$, which equals 4 in any state, but $(x + x)[2/x] \neq 4$.
- ***Example 2***: $(a - x)[2/x] \equiv a - (2) \equiv a - 2$ (the parentheses are redundant)

 © James Sasaki, 2023

- ***Example 3***: $(x*(x+1))[b-c/x] \equiv (b-c)*(b-c+1)$ (the parentheses are required).
- ***Example 4***: $(b[x*y])[x+3/x] \equiv b[(x+3)*y]$
- ***Example 5***: $(y+b[x])\ [x*3/x] \equiv y+b[x*3]$
- ***Example 6***: $(\textbf{if } x>0 \textbf{ then } -x \textbf{ else } 0 \textbf{ fi})[z+2/x] \equiv \textbf{if } z+2>0 \textbf{ then } -(z+2) \textbf{ else } 0 \textbf{ fi}$
- ***Example 7***: $(b[x*(x+1)/2])[y+4/x] \equiv b[(y+4)*((y+4)+1)/2] \equiv b[(y+4)*(y+4+1)/2]$.
- The technical definition of $e[e'/v]$ is done by cases on the structure of e. Briefly, we have constants and variables as base cases and expressions with subexpressions as recursive cases.

Definition of $e[e'/v]$ by Structural Induction

- ***Case 1*** (base cases)
  - $c[e'/v] \equiv c$ if c is a constant
  - $v[e'/v] \equiv (e')$
  - If $v \neq w$, then $w[e'/v] \equiv w$.
- ***Case 2*** (recursive cases): Consider the expressions that have subexpressions: function calls $f(e_1, e_2, \ldots)$, array indexing expressions $b[e_1, e_2, \ldots]$, parenthesized expressions (e_1), unary operations $\oplus e_1$, binary operations $e_1 \oplus e_2$ and ternary operations $e_1 ? e_2 : e_3$ (or ***if*** e_1 ***then*** e_2 ***else*** e_3 ***fi***), we recursively process each subexpression.
  - Let $e_1' \equiv (e_1)[e'/v]$, $e_2' \equiv (e_2)[e'/v]$, etc.
    - Then $(f(e_1, e_2, \ldots))[e'/v] \equiv f(e_1', e_2', \ldots)$
    - And $(b[e_1, e_2, \ldots])[e'/v] \equiv b[e_1', e_2', \ldots]$
    - And $(e_1 \oplus e_2)[e'/v] \equiv e_1' \oplus e_2'$
    - And so on.

E. Substitution Into A Predicate

- ***Notation***: $p[e/v]$ is pronounced "p with e (substituted) for v" and stands for the result of substituting e for each (free) occurrence of v in p. (Don't worry about free and bound occurrences of a variable until we get to quantified predicates.)
- Substitution into expressions and predicates is a syntactic operation. For example, $(x>0)[1/x] \equiv 1>0$, which $\Leftrightarrow$ true, but $(x>0)[1/x] \neq T$.

Substitution Case 1: Non-Quantified Predicate

- For a predicate that is not quantified, we substitute recursively in its sub-predicates or expressions. (Note the predicate might ***contain*** a quantified subpredicate, but those predicates will get covered in the other cases, where a predicate ***is*** a quantified predicate.)
  - $(\neg p)[e/v] \equiv \neg(p[e/v])$
  - $(p_1 \wedge p_2)[e/v] \equiv p_1[e/v] \wedge p_2[e/v]$, and similarly for $\vee$, $\rightarrow$, and $\leftrightarrow$.
  - $(e_1 < e_2)[e/v] \equiv (e_1[e/v]) < (e_2[e/v])$, and similarly for the other relational operators.

 © James Sasaki, 2023

- ***Example 8***: $(x > 0 \to y \geq x/2)[z+1/x]$
  $\equiv (x > 0)[z+1/x] \to (y \geq x/2)[z+1/x]$
  $\equiv (z+1 > 0 \to y \geq (z+1)/2)$. (The parentheses around $z+1$ are necessary)
- ***Note:*** If p contains no occurrences at all of v, then $p[e/v] \equiv p$. E.g., $(x < y)[e/z] \equiv x < y$.
  - This is especially true of predicates that only contain constants, such as $2 + 2 = 4$.
- This case of substitution continues recursively until we come across a quantified predicate.
  - To cover the quantified predicate case, we need to know that only some occurrences of variables do get substituted for (the "free" occurrences). The others (the "bound" occurrences) do not get substituted for.
  - However, within a predicate, the same variable can be used in different ways. This complicates things.

F. Free and Bound Variables and Occurrences of Variables

- ***Notation***: Q stands for a quantifier ($\forall$ or $\exists$).
- For the definition of $(Q\,x\,.\,q)[e/v]$, our natural instinct is to think that $(Q\,x\,.\,q)[e/v]$ should $\equiv (Q\,x\,.\,(q[e/v]))$, but in fact this isn't always true because of a distinction between "free" and "bound" occurrences of variables.
- ***Definition***: If an occurrence of a variable v in a predicate is within the scope of a quantifier over v, then it is a ***bound occurrence***, else it is a ***free occurrence***. A variable v ***is free in*** (= ***occurs free in***) p iff it has a free occurrence in p. Similarly, v ***is bound in*** (= ***occurs bound in***) p iff it has a bound occurrence in p. (In computer science terms, local variables have bound occurrences, and non-local variables have free occurrences.)
- For any variable v and predicate p, there are four possibilities:
  - v is neither free nor bound in p (this case applies when v doesn't occur at all in p).
  - v is free but not bound in p: v occurs at least once in p, and all the occurrences of v are free.
  - v is not free but is bound in p: v occurs at least once in p, and all the occurrences of v are bound.
  - v is free and bound in p: v occurs at least twice in p with at least one occurrence being free and at least one occurrence being bound.
- ***Example 9***: If $p \equiv x > z \wedge \exists\, x\,.\,\exists\, y\,.\,y \leq f(x, y)$, then
  - x is free and bound in p. (Its first occurrence is free; its second is bound.)
  - y is bound in p but not free in p.
  - z is free in p but not bound in p.
  - w is neither free nor bound in p.
- The reason we're interested in occurrences of variables being free or bound in a predicate is that ***we only substitute for free occurrences*** of a variable. In computer science terms, we're looking for non-local variables, not local variables.

 © James Sasaki, 2023

- Taking polynomials as an example, $p(x) = x^2 + a * x + y$. If we want to substitute 17 for y, that's fine: $p(x) = x^2 + a * x + 17$; substituting expressions with variables that aren't bound in the definition is okay too: substituting $(z^3 + 1)$ for y gives us $p(x) = x^2 + a * x + (z^3 + 1)$. But if we want to substitute something like $(x + 3)$ for y (note: x is the defined parameter variable), we ***don't*** want $p(x) = x^2 + a * x + (x + 3)$. But if we had defined $p(w) = w^2 + a * w + y$, then substituting $(x + 3)$ for y gives us $p(w) = w^2 + a * w + (x + 3)$.

G. Substitution Into A Quantified Predicate

- In case 1 of the definition of substitution, the major operator of the predicate was not a quantifier, it was a conjunction or disjunction, etc.
- In the remaining cases, we substitute into a quantified predicate: $(Q\,x\,.\,q)[e/v]$.

Substitution Case 2: Quantified Variable ≡ Variable to Replace

- In the simplest quantifier case, the quantified variable matches the variable we're substituting for. I.e., we have $(Q\,v\,.\,q)[e/v]$.
- Since all the occurrences in q of v are bound, there are no free occurrences of v in $Q\,v\,.\,q$, so there's nothing to replace: $(Q\,v\,.\,q)[e/v] \equiv Q\,v\,.\,q$.
- ***Example 10***: $(x > 0 \wedge \exists\, x\,.\,x \le f(y))[17/x] \equiv 17 > 0 \wedge \exists\, x\,.\,x \le f(y)$. Here, the first occurrence of x (in $x > 0$) is free, so we replace it with 17, but the second occurrence of x is bound, so we don't do any replacement.

Substitution Case 3: Quantified Variable Doesn't Occur in Replacement Expression

- If $x \neq v$ and x does not occur in e, then $(Q\,x\,.\,q)[e/v] \equiv (Q\,x\,.\,(q[e/v]))$. Here, we go through the text of q and replace its free occurrences of v with e.
- ***Example 11***: $(y \ge 0 \to \forall\, x\,.\,x > y \to x * x > y \wedge \exists y\,.\,f(y) > x))[17/y]$
  $\equiv 17 \ge 0 \to \forall\, x\,.\,(x > y \to x * x > y \wedge \exists y\,.\,f(y) > x)[17/y]$
  $\equiv 17 \ge 0 \to \forall\, x\,.\,x > 17 \to x * x > 17 \wedge \exists y\,.\,f(y) > x$.
  Note the y in $f(y)$ is bound, so there's no substituting for it.
- In case 3, the restriction that the quantified variable not appear in e keeps us from having a "capture" problem, where occurrences of x in e are free, but when we we replace an occurrence of v by e in $Q\,x\,.\,q[e/v]$, the occurrences of x in e become bound, which changes their meaning.
- ***Example 12:*** $(\exists y\,.\,y = v^2)[x + 1/v] \equiv \exists y\,.\,y = (x + 1)^2$. If we were to let $(\exists\, x\,.\,x = v^2)[x + 1/v]$ be $\exists\, x\,.\,x = (x + 1)^2$, then the x in $x + 1$ would become bound to the $\exists x$ (= the x would be "***captured***").
  - (Before the substitution, the x in $\ldots[x + 1/v]$ was not quantified, so after the substitution, we also want x to not be quantified.)

 © James Sasaki, 2023

- The way out of this problem is to ***rename the quantified variable*** from x to something not in e; that way the quantifier can't capture occurrences of x.

Substitution Case 4: Quantified Variable Does Occur in Replacement Expression

- This case is the most complicated one. If $x \neq v$ and x occurs in e, then what we do is replace the quantified variable with one that doesn't appear in the quantifier's body. Then we proceed as in Case 3.
  - So, $(Q\,x\,.\,q)[e/v] \equiv (Q\,z\,.\,q[z/x])[e/v] \equiv (Q\,z\,.\,(q[z/x][e/v]))$
    where z is a ***fresh variable*** (one not used in e or q).
- ***Example 13***: Using z as a fresh variable, we have

  $(g(x,v) < 0 \wedge (\exists\, x\,.\,x = v^2) \wedge h(y,v) > 0)[x+1/v]$

  $\equiv g(x,x+1) < 0 \wedge (\exists\, z\,.\,((x = v^2)[z/x])[x+1\,/\,v]) \wedge h(y,x+1) > 0$

  // Pick fresh variable, quantify over it and then substitute for it in the body

  $\equiv\ g(x,x+1) < 0 \wedge (\exists\, z\,.\,z = v^2)[x+1\,/\,v] \wedge h(y,x+1) > 0$

  $\equiv\ g(x,x+1) < 0 \wedge (\exists\, z\,.\,z = (x+1)^2) \wedge h(y,x+1) > 0$

  - Note there's some ambiguity in the definition: Which "fresh" variable should we choose?

- Substitution into a predicate is also how application of a predicate function works.
- ***Example 14:*** Define $member(x,b) \equiv \exists\, 0 \le k < size(b)\,.\,x = b[k]$. Then $member(12,\ b1)$ is calculated as $(\exists\, 0 \le k < size(b)\,.\,x = b[k])[12\,/\,x][b1\,/\,b] \equiv\ \exists\, 0 \le k < size(b1)\,.\,12 = b[k]$. Renaming occurs when an argument uses a variable that's quantified in the body.

  $member(k*c, b2)\ \equiv\ (\exists\, 0 \le k < member(b)\,.\,x = b[k])[k*c\,/\,x][b2\,/\,b]$

  $\equiv\ (\exists\, 0 \le k1 < size(b)\,.\,x = b[k1])[k*c\,/\,x][b2\,/\,b]$ -- Renaming k to $k1$

  $\equiv\ (\exists\, 0 \le k1 < size(b2)\,.\,k*c = b2[k1])$

 © James Sasaki, 2023