Syntactic Substitution

- In backward assignment rule, we replace $Q(v)$ by $Q(e)$ to get the wlp for statement $v \coloneqq e$ and postcondition $Q(v)$. The operation of going from $Q(v)$ to $Q(e)$ is called **syntactic substitution**. This substitution procedure is totally textual.

(Substitution in an Expression)

- **Substitution in an expression** is the following operation: take an expression e and replace its occurrence of variable v with expression e', written as $e[e'/\, v]$ and pronounced "e with e' for v".
  - o Like state updating, the squared parenthesis has very high hierarchy and when there are multiple substitutions, we calculate from left to right.

1. Finish the following substitution in expressions.
   a. $(x+y)[5\,/\,x] \equiv 5+y$
   b. $x+y\,[5\,/\,x] \equiv x+y$
   - o Substitution has very high hierarchy.
   - o If there is no variable v in e, then $e[e'/v] \equiv e$.

   c. $(x+x)[2\,/\,x] \equiv 2+2$
   - o The substitution operation is done; and $(x+x)[2\,/\,x] \not\equiv 4$.

   d. $\big(x*(x+1)\big)\,[b-c\,/\,x\,] \equiv (b-c)*(b-c+1)$
   - o Add parentheses accordingly.

   e. $\textbf{if } x>0 \textbf{ then} -x \textbf{ else } 0 \textbf{ fi } [z+2\,/\,x] \equiv \textbf{if } z+2>0 \textbf{ then} -(z+2) \textbf{ else } 0 \textbf{ fi}$
   f. $b[(x+1) \div 2\,]\,[a\,/\,b] \equiv a[(x+1)\div 2\,]$
   g. $f(b)\,[a\,/\,b] \equiv f(a)$, here f is a function and a, b are arrays

- In general, $e[e'/v]$ can be calculated using the following recursive algorithm:

$$e[e'/v] \equiv \begin{cases} c, & \text{if } e \text{ is a constant } c \\ e', & \text{if } e \equiv v \\ e, & \text{if } e \text{ is a variable and } e \not\equiv v \\ b\big[e_1[e'/\,v]\big]\big[e_2[e'/\,v]\big] \ldots \big[e_n[e'/\,v]\big], & \text{if } e \equiv b[e_1][e_2] \ldots [e_n] \\ op\,(e_1[e'/v], e_2[e'/v], \ldots, e_n[e'/\,v]), & \text{if } e \equiv op(e_1, e_2, \ldots, e_n) \\ f\,(e_1[e'/v], e_2[e'/v], \ldots, e_n[e'/\,v]), & \text{if } e \equiv f\,(e_1, e_2, \ldots, e_n) \end{cases}$$

(Substitution in a Predicate)

- **Substitution in a predicate** is the following operation: take a predicate p and replace each ***free occurrence*** of variable v with expression e', written as $p[e'/\,v]$ and pronounced "p with e' for v".
  - o Again, the substitution is textual, so $(x>0)[1\,/\,x] \equiv 1>0$, but $(x>0)[1\,/\,x] \not\equiv T$.

- We don't need to worry about the word "free" if the predicate is not quantified.

2. $(x > 0 \to y \geq x \,/\, 2)[z + 1 \,/\, x] \equiv (x > 0)[z + 1 \,/\, x] \to (y \geq x \,/\, 2)[z + 1 \,/\, x]$
$$\equiv z + 1 > 0 \to y \geq (z + 1)/2$$

- What if the predicate is quantified, or part of the predicate is quantified (such as $y < 0 \to \exists x. x \geq y$)?
Let Q stand for a quantifier ∀ or ∃. In our intuition, $(Qx. q)\ [e \,/\, v]$ should be syntactically equivalent to $Qx. (q[e \,/\, v])$, but in fact this is not always true. We need to consider whether a variable is *free* or *bound*.

- 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** p iff it has a free occurrence in p. Similarly, v **is bound in** p iff it has a bound occurrence in p.

3. Decide for each of the following variables, whether it is free or bound in p.
$$p \equiv x > z \wedge \exists x. \exists y. y \leq f(x, y)$$
   a. Variable x is free and bound in p.
   b. Variable y is bound in p.
   c. Variable z is free in p.
   d. Variable u is neither free nor bound in p.

   - From the above example we can see that each occurrence of some variable can be either free or bound. But for a variable, it can be both/either/neither free and/or/nor bound in a predicate.

- To syntactic substitute in a quantified predicate, we **only substitute for free occurrences**.
- But there still are other problems. For example, if we have predicate $\exists x. x = v^2$ and we want to replace v by $y + 1$, then $(\exists x. x = v^2)[y + 1 \,/\, v] \equiv \exists x. x = (y + 1)^2$, and it looks correct to us. However, if we let $(\exists x. x = v^2)[x + 1 \,/\, v] \equiv \exists x. x = (x + 1)^2$, it totally changes the semantic of this predicate; we don't want to have such a substitution.

- To $(Qx. q)[e \,/\, v]$, there are three different cases:
  - Case 1: $x \equiv v$.
  - Case 2: $x \not\equiv v$, and x does not appear in e.
  - Case 3: $x \not\equiv v$, and x appears in e.

- For Case 1, we don't need to do anything, since we don't substitute for a bound occurrence: $(Qv. q)[e \,/\, v] \equiv Qv. q$.
- For Case 2, we just need to need substitute the free occurrences of v in q: $(Qx. q)[e \,/\, v] \ \equiv Qx. (q[e \,/\, v])$

4. Finish the following substitution in predicates.
   a. $\big(x > 0 \wedge \exists\, x. x \leq f(y)\big)[17 \,/\, x] \ \equiv 17 > 0 \wedge \exists\, x. x \leq f(y)$.

   b. $(y \geq 0 \to \forall x. x > y \to x * x > y \wedge \exists\, y. f(y) > x)\ [17 \,/\, y]$
   $\equiv 17 \geq 0 \to \forall x. x > 17 \to x * x > 17 \wedge \exists\, y. f(y) > x$

- For Case 3, we need to rename the quantified variable from x to something **fresh variable** (a variable that is not in e or q, such as z in this example): $(Qx. q)[e \,/\, v] \equiv (Qz. q[z \,/\, x])\ [e \,/\, v] \equiv Qz. q[z \,/\, x][e \,/\, v]$.

- A fresh variable is defined to be a variable that is not in e or q. However, to avoid unexpected problems, if possible, simply choose some variable that is not used anywhere in the whole predicate.

  For example: $(\exists x. x > 0 \rightarrow (\forall y. y + z < v^2) \wedge x > y)[y + 1 / v]$. When we only look at $(\forall y. y + z < v^2)[y + 1 / v]$, we are in case 3, and we can choose x as a fresh variable here to replace y; but x is used in other places in the predicate. To avoid future ambiguity, it is better to use variable w here.

5. $\big((\exists x. x = v^2) \wedge h\,(y, v) > 0\big)\,[x + 1 / v]$
$$\begin{aligned}&\equiv (\exists x. x = v^2)\,[x + 1 / v] \wedge (h\,(y, v) > 0)[x + 1 / v]\\ &\equiv (\exists z. (x = v^2)[z / x])\,[x + 1 / v] \wedge (h\,(y, v) > 0)[x + 1 / v]\\ &\equiv (\exists z. z = v^2)\,[x + 1 / v] \wedge (h\,(y, v) > 0)[x + 1 / v]\\ &\equiv (\exists z. z = (x + 1)^2) \wedge h\,(y, x + 1) > 0\end{aligned}$$

6. Let $member(x, b) \equiv \exists 0 \le k < size(b). x = b[k]$.
   a. What is $member(12, b_1)$?
   In fact, $member(12, b_1) \equiv member(x, b)[12 / x][b_1 / b]$
$$\begin{aligned}&\equiv (\exists 0 \le k < size(b). x = b[k])[12 / x][b_1 / b]\\ &\equiv (\exists 0 \le k < size(b). 12 = b[k])[b_1 / b]\\ &\equiv (\exists k. 0 \le k < size(b) \wedge 12 = b[k])[b_1 / b]\\ &\equiv \exists k. 0 \le k < size(b_1) \wedge 12 = b_1[k]\end{aligned}$$

   b. What is $member(k * c, b_2)$?
   $member(k * c, b_2) \equiv member(x, b)[k * c / x][b_2 / b]$
$$\begin{aligned}&\equiv (\exists 0 \le k < size(b). x = b[k])[k * c / x][b_2 / b] \quad \text{\# case 3}\\ &\equiv (\exists 0 \le k_0 < size(b). (x = b[k])[k_0 / k])\,[k * c / x][b_2 / b]\\ &\equiv (\exists 0 \le k_0 < size(b). x = b[k_0])[k * c / x][b_2 / b]\\ &\equiv (\exists 0 \le k_0 < size(b). k * c\ = b[k_0])[b_2 / b]\\ &\equiv (\exists 0 \le k_0 < size(b_2). k * c = b_2[k_0])\end{aligned}$$

<u>Forward Assignment Rule</u>

- Quick review, with partially valid triple $\{Q(e)\}\ v \coloneqq e\ \{Q(v)\}$ we got the backward assignment rule: $wlp\big(v \coloneqq e, Q(v)\big) \equiv Q(e)$.
  - With the knowledge of syntactical substitution, we can generalize the **backward assignment rule** as follows: $wlp(v \coloneqq e, q) \equiv q[e / v]$.

- Now, we have another question. What can be used as a postcondition for $\{p\}\ v \coloneqq e\ \{?\}$ and what can be the strongest postcondition?
  - Intuition tells us $\{p\}\ v \coloneqq e\ \{p \wedge v = e\}$ looks true, and it does work in some situations; but in general, it is not correct.

7. Are the following triples valid (under either correctness)?
   a. $\{x > y\}\ z \coloneqq 2\ \{x > y \wedge z = 2\}$.
   Yes. To justify it is valid, we can apply the backward assignment rule:
$$wlp(z \coloneqq 2, x > y \wedge z = 2) \equiv (x > y \wedge z = 2)[2 / z] \equiv x > y \wedge (2 = 2) \Leftrightarrow x > y$$

   b. $\{x > 0\}\ x \coloneqq x - 2\ \{x > 0 \wedge x = x - 2\}$.
   Clearly this is not valid, since $(x = x - 2) \Leftrightarrow F$.

- Why $\{p\}\ v := e\ \{p \land v = e\}$ works in example 7.a but not 7.b? The difference is whether v has appears in e or p.

- Here is how we solve this problem: we "record" the old value of v in precondition with a **logical constant** (which is a constant that only appears in pre- and/or post- conditions that helps us to reason): $\{x > 0 \land x = x_0\}\ x := x - 2\ \{x_0 > 0 \land x = x_0 - 2\}$. We say that we ***age*** the variable x in the precondition.

- **Aging** a variable x is to introduce in the precondition a logical constant to name the value of x before executing the statement.

- **[Forward Assignment Rule]** $p[v_0 / v] \land v = e[v_0 / v]$ is a postcondition for *partially valid triple* $\{p \land v = v_0\}\ v := e\ \{q\}$. Note that, even if we omit the $v = v_0$ in the precondition, it is still understood.

8. Create a valid triple $\{x > 0\}\ x := x - 1\ \{q\}$.
   - With the forward assignment rule, we can age the variable that is being assigned to, then we have:
   $$\{x > 0 \land x = x_0\}\ x := x - 1\ \{x_0 > 0 \land x = x_0 - 1\}$$

   - We can use existential to represent the postcondition to remove the aging of x in the precondition:
   $$\{x > 0\}\ x := x - 1\ \{\exists x_0 . x_0 > 0 \land x = x_0 - 1\}$$

   - x_0 only appears in the postcondition; so, dropping of the quantifier and only keeping the body already implies the existence of such x_0. Then we have:
   $$\{x > 0\}\ x := x - 1\ \{x_0 > 0 \land x = x_0 - 1\}$$

- From the above example, we can see that, it is okay to omit the aging part in the precondition and use fresh logic variable x_0 for x in the postcondition directly to represent the old value of variable x. It is still understood.