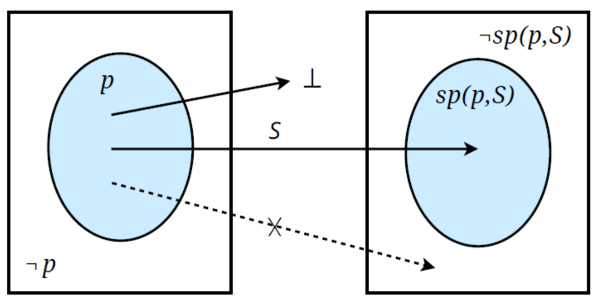
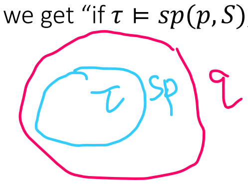
The Strongest Postconditions

- Given a precondition p and program S, **the strongest postcondition** of p and S , written as $sp(p,S)$ is (the predicate that stands for) the set of states we could terminate in if we run S starting in a state that satisfies p.
  - In symbols, using the language of states: $sp(p,S) = \{\tau \mid \tau \in M(S,\sigma) - \bot \text{ for some } \sigma \text{ where } \sigma \vDash p\}$, or equivalently $sp(p,S) = \bigcup_\sigma (M(S,\sigma) - \bot) \text{ where } \sigma \vDash p$.
  - From the definition of the strongest postcondition we can see that $\vDash \{p\}\ S\ \{sp(p,S)\}$; in other words, this postcondition only guarantees a valid triple under partial correctness. The reason is easy to see: the precondition p is given, and there could be states that satisfy p make program S diverge or create error.



- From the above figure, we can see:
  - If $\sigma \vDash p$, then $\forall \tau.\, \tau \in M(S,\sigma) \to \tau = \bot \vee \tau \vDash sp(p,S)$.
  - If $\sigma \vDash \neg p$, we don't know anything interesting about $M(S,\sigma)$.

1. Prove that $\vDash \{p\}\ S\ \{q\} \Leftrightarrow (sp(p,S) \Rightarrow q)$.
   It looks trivial, but our definition of $sp(p,S)$ didn't say anything about this postcondition the strongest. In this example, let us prove that $sp(p,S)$ is the strongest postcondition.
   - $\Leftarrow$: By the definition of $sp(p,S)$, we have $\vDash \{p\}\ S\ \{sp(p,S)\}$. And since $sp(p,S) \Rightarrow q$, then $\vDash \{p\}\ S\ \{q\}$ (weakening the postcondition).
   - $\Rightarrow$: Let τ be a state such that $\tau \vDash sp(p,S)$. By the definition of $sp(p,S)$, there exists some $\sigma \vDash p$ such that $\tau \in M(S,\sigma) - \bot$. $\vDash \{p\}\ S\ \{q\}$ implies that $M(S,\sigma) - \bot \vDash q$, thus $\tau \vDash q$. To sum up, we get "if $\tau \vDash sp(p,S)$, then $\tau \vDash q$", which implies "$sp(p,S) \Rightarrow q$".



Calculate sp for Loop-free Programs

Like wlp, we can use some algorithm/rule to calculate $sp(p,S)$ textually.

- $sp(p, \mathbf{skip}) \equiv p$.
- $sp(p, v \coloneqq e) \equiv p[v_0 \,/\, v] \wedge v = e[v_0 \,/\, v]$, where v_0 is the aged v (in other words, the old value of v before executing $v \coloneqq e$).
  - This is the forward assignment rule, so actually this rule can produce the strongest postcondition.
- $sp(p, S_1; S_2) \equiv sp(sp(p,S_1), S_2)$.

2. Calculate the following sp's.
   a. $sp\ (x > y,\ x \coloneqq x + k) \equiv x_0 > y \wedge x = (x + k)[x_0\ /\ x] \equiv x_0 > y \wedge x = x_0 + k$
   b. $sp\ (x_0 > y \wedge x = x_0 + k,\ y \coloneqq y + k) \equiv x_0 > y_0 \wedge x = x_0 + k \wedge y = y_0 + k$

   c. $sp\ (x > y,\ x \coloneqq x + k;\ y \coloneqq y + k) \equiv x_0 > y_0 \wedge x = x_0 + k \wedge y = y_0 + k$ #Combine a. and b.
   - By losing x_0 and y_0, we can slightly weaken the postcondition to $x > y$.

   d. $sp(x > f(x, y), x \coloneqq x + 1; x \coloneqq x + x)$
   - $sp(x > f(x, y),\ x \coloneqq x + 1) \equiv \big(x > f(x, y)\big)[x_0\ /\ x] \wedge x = (x + 1)[x_0\ /\ x]$
     $\equiv x_0 > f(x_0, y) \wedge x = x_0 + 1$

   - $sp(x > f(x, y), x \coloneqq x + 1; x \coloneqq x + x)$
     $\equiv sp(x_0 > f(x_0, y) \wedge x = x_0 + 1,\ x \coloneqq x + x)$
     $\equiv (x_0 > f(x_0, y) \wedge x = x_0 + 1)[x_1\ /\ x] \wedge x = (x + x)[x_1\ /\ x]$
     $\equiv (x_0 > f(x_0, y) \wedge x_1 = x_0 + 1) \wedge x = x_1 + x_1$

- Let us think about sp in a conditional statement with an example:
  $sp(T,\ \textbf{if}\ x \geq y + z\ \textbf{then}\ x \coloneqq x - 1\ \textbf{else}\ y \coloneqq y + 2\ \textbf{fi}) \equiv ?$

  Following intuition, it is quite straightforward to come up with the following solution:
  - When the if condition is true, we should have $sp(T \wedge x \geq y + z,\ x \coloneqq x - 1) \equiv x_0 \geq y + z \wedge x = x_0 - 1$.
  - When the if condition is false, we should have $sp(T \wedge x < y + z,\ y \coloneqq y + 2) \equiv x < y_0 + z \wedge y = y_0 + 2$.
  - The sp for the whole statement should one of the above, thus:
    $"sp"(T,\ \textbf{if}\ x \geq y + z\ \textbf{then}\ x \coloneqq x - 1\ \textbf{else}\ y \coloneqq y + 2\ \textbf{fi})$
    $\equiv (x_0 \geq y + z \wedge x = x_0 - 1) \vee (x < y_0 + z \wedge y = y_0 + 2)$

  - Is this postcondition the strongest? No, it can be stronger since we didn't include that y is not updated in the true branch and x is not updated in the false branch. We need to add this information as well, and we need to be careful which variables are aged.

- To calculate the sp for a conditional statement, we need to calculate some variable sets first:
  - $lhs(S) =$ the set of variables that appear as the lhs of assignments in statement S.
  - $rhs(S) =$ the set of variables that appear as the rhs of assignments in statement S.
  - $free(p) =$ the set of variables that are free in precondition p.
  - $aged(p, S) = lhs(S) \cap \big(rhs(S) \cup\ free(p)\big)$ is the set of variables whose assignments cause aging.

- Let $IF \equiv\ \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}$, and let $aged(p, IF) = \{x, y, \dots\}$. Then $sp(p, IF) \equiv sp(p_0 \wedge B,\ S_1) \vee sp(p_0 \wedge \neg B,\ S_2)$, where $p_0 = p \wedge x = x_0 \wedge y = y_0 \dots$
- Let $NF \equiv\ \textbf{if}\ B_1 \rightarrow S_1\ \square\ B_1 \rightarrow S_2\ \textbf{fi}$, and let $aged(p, NF) = \{x, y, \dots\}$. Then $sp(p, NF) \equiv sp(p_0 \wedge B_1,\ S_1) \vee sp(p_0 \wedge B_1,\ S_2)$, where $p_0 = p \wedge x = x_0 \wedge y = y_0 \dots$

3. Calculate $sp(T,\ \textbf{if}\ x \geq y + z\ \textbf{then}\ x \coloneqq x - 1\ \textbf{else}\ y \coloneqq y + 2\ \textbf{fi})$
   - Let $p \equiv T, S \equiv\ \textbf{if}\ x \geq y + z\ \textbf{then}\ x \coloneqq x - 1\ \textbf{else}\ y \coloneqq y + 2\ \textbf{fi}$
     - $lhs(S) = \{x, y\}$
     - $rhs(S) = \{x, y\}$
     - $free(p) = \emptyset$
     - $aged(p, S) = \{x, y\}$

- $sp(T \wedge x = x_0 \wedge y = y_0 \wedge x \geq y + z,\ x \coloneqq x - 1)$
  $\equiv x_0 = x_0 \wedge y = y_0 \wedge x_0 \geq y + z \wedge x = x_0 - 1$

- $sp(T \wedge x = x_0 \wedge y = y_0 \wedge x < y + z,\ y \coloneqq y + 2)$
  $\equiv x = x_0 \wedge y_0 = y_0 \wedge x < y_0 + z \wedge y = y_0 + 2$

- $sp(p, S)$
  $\equiv (x_0 = x_0 \wedge y = y_0 \wedge x_0 \geq y + z \wedge x = x_0 - 1) \vee (x = x_0 \wedge y_0 = y_0 \wedge x < y_0 + z \wedge y = y_0 + 2)$
  $\Leftrightarrow (y = y_0 \wedge x_0 \geq y + z \wedge x = x_0 - 1) \vee (x = x_0 \wedge x < y_0 + z \wedge y = y_0 + 2)$

4. Calculate $sp(p, S)$ where $p \equiv (x = y)$ and $S \equiv \mathbf{if}\ y \geq 1 \rightarrow x \coloneqq 1 \ \square\ y \leq 1 \rightarrow z \coloneqq 0\ \mathbf{fi}$.
   - $lhs(S) \equiv \{x, z\}$
   - $rhs(S) \cup free(p) \equiv \{x, y\}$
   - $aged(p, S) \equiv \{x\}$

   - $sp(x = y \wedge x = x_0 \wedge y \geq 1,\ x \coloneqq 1) \equiv (x_0 = y \wedge x_0 = x_0 \wedge y \geq 1) \wedge (x = 1)$
   - $sp(x = y \wedge x = x_0 \wedge y \leq 1,\ z \coloneqq 0) \equiv x = y \wedge x = x_0 \wedge y \leq 1 \wedge z = 0$
   - $sp(p, S) \equiv (x_0 = y \wedge x_0 = x_0 \wedge y \geq 1 \wedge x = 1) \vee (x = y \wedge x = x_0 \wedge y \leq 1 \wedge z = 0)$

<u>Forward Assignment vs. Backward Assignment</u>

- With backward assignment rule, we can get valid triple $\{q[e\ /\ v]\}\ v \coloneqq e\ \{q\}$; and with forward assignment rule we get valid triple $\{p\}\ v \coloneqq e\ \{p[v_0\ /\ v] \wedge v = e[v_0\ /\ v]\}$. The preconditions and postconditions calculated this way happen to be the weakest and strongest, respectively. Here we will show that these two rules can derive from each other. This proof will only show that these two rules are equally strong: if one is correct the other is also correct, and it does not show anything interesting between sp and wlp in general.

- First, let us calculate the $sp(q[e\ /\ v], v \coloneqq e)$, where this precondition is calculated from the backward assignment rule.
  - $sp(q[e\ /\ v], v \coloneqq e) \quad \equiv q[e\ /\ v][v_0\ /\ v] \wedge v = e[v_0\ /\ v]$ # After $q[e\ /\ v]$, every v must be from e
    $\Leftrightarrow q[e[v_0\ /\ v]\ /\ v] \wedge v = e[v_0\ /\ v]$
    $\Rightarrow q[v\ /\ v]$
    $\Leftrightarrow q$
    - $sp(q[e\ /\ v], v \coloneqq e) \Rightarrow q$. Given an assignment statement S and postcondition q, then backward assignment rule + forward assignment rule can give you a postcondition that's stronger that q.

- Then, let us calculate the $wlp(v \coloneqq e,\ p[v_0\ /\ v] \wedge v = e[v_0\ /\ v])$, where this postcondition is calculated from the forward assignment rule.
  - $wlp(v \coloneqq e,\ p[v_0\ /\ v] \wedge v = e[v_0\ /\ v]) \quad \equiv (p[v_0\ /\ v] \wedge v = e[v_0\ /\ v])[e\ /\ v]$
    $\equiv p[v_0\ /\ v] \wedge e = e[v_0\ /\ v]$
    - $p \wedge v = v_0 \quad \Leftrightarrow p \wedge T \wedge v = v_0$
      $\Leftrightarrow p \wedge e = e \wedge v = v_0$
      $\Leftrightarrow p[v\ /\ v] \wedge e = e[v\ /\ v] \wedge v = v_0$
      $\Rightarrow p[v_0\ /\ v] \wedge e = e[v_0\ /\ v]$
      $\equiv wlp(v \coloneqq e,\ p[v_0\ /\ v] \wedge v = e[v_0\ /\ v])$

- $(p \wedge v = v_0) \Rightarrow wlp(v \coloneqq e,\ p[v_0 \,/\, v] \wedge v = e[v_0 \,/\, v])$. Given a precondition p (and $v = v_0$) and an assignment statement S, then forward assignment rule + backward assignment rule can give you a precondition that's weaker than $p \wedge v = v_0$.

<u>Provability of Triples</u>

- Remember that we want to use valid correctness triples to show a program works as expected. In other words, given a program S, given a precondition p that can be provided to this program before it executes, and given a postcondition q that we expect to get after this program executes, we need to show this correctness triple $\{p\}\ S\ \{q\}$ is valid; denoted as $\vDash \{p\}\ S\ \{q\}$ or $\vDash_{tot} \{p\}\ S\ \{q\}$ (depend on what correctness level we need).

- In fact, not all triples can be decided to be valid or invalid. This is like not everything true can be proved to be true, or not all yes-or-no problems have an algorithm that can guarantee a solution (This is taught in $CS530$).

- If a triple $\{p\}\ S\ \{q\}$ can be proved to be valid, then we say this triple is **provable**, denoted as $\vdash \{p\}\ S\ \{q\}$ or $\vdash_{tot} \{p\}\ S\ \{q\}$ (depend on what correctness level we need).
  - In this course we care about provable triples. We focus on creating proofs for provable triples; we don't focus on deciding whether a valid triple is provable or not.