**Strength; Weakest Preconditions pt. I**

CS 536: Science of Programming, Fall 2018

9/26: pp. 7, 8, 9/28: updated activity

A. Why

- To combine correctness triples, we need to weaken and strengthen conditions.
- A weakest precondition is the most general requirement that a program must meet to produce the right result.

B. Objectives

At the end of today you should understand

- What weakening and strengthening are.
- What weakest preconditions (wp) are and how they are related to preconditions in general.

C. Stronger and Weaker Predicates

- **Generalizing the Sequence Rule:** We’ve already seen that two triples \( \{ p \} S_1 \{ q \} \) and \( \{ q \} S_2 \{ r \} \) can be combined to form the sequence \( \{ p \} S_1 \; S_2 \{ r \} \).
- Say we want to combine two triples that don’t have a common middle condition, \( \{ p \} S_1 \{ q \} \) and \( \{ q' \} S_2 \{ r \} \).
  - We can do this iff \( q \rightarrow q' \). If \( S_1 \) terminates in state \( \tau \) and \( \{ p \} S_1 \{ q \} \) is valid, then \( \tau \models q \). If \( \models q \rightarrow q' \), then \( \tau \models q' \), so if \( \{ q' \} S_2 \{ r \} \), then if running \( S_2 \) in \( \tau \) terminates, it terminates in a state satisfying \( r \).
  - This reasoning works both for partial and total correctness.
- Our earlier rule with \( q \equiv q' \) is a special case of this more general rule (since \( q \) always implies itself).
- **Definition:** If \( q \rightarrow r \) then \( q \) is **stronger than** \( r \) and \( r \) is **weaker than** \( q \).
  - I.e., the set of states that satisfy \( q \) is \( \subseteq \) the set of states that satisfy \( r \).
  - (Technically, we should say “stronger than or equal to” and “weaker than or equal to,” since we can have \( q \equiv r \), but it’s too much of a mouthful. Then “strictly stronger” means “stronger than or equal to and not equal to;” strictly weaker is similar.)
- **Note:** One notation for implication is \( p \supset q \); it’s important not to read that \( \supset \) as a superset symbol, since the set of states for \( p \subseteq \) the set of states for \( q \).
- **Example 1:** \( x = 0 \) is stronger than \( x = 0 \lor x = 1 \), which is stronger than \( x \geq 0 \).
- A predicate corresponds to the set of states that satisfy it, so we can use Venn diagrams with sets of states to display comparisons of predicate strength.
  - In Figure 1, \( p \rightarrow q \) because the set of states for \( p \subseteq \) set of states for \( q \). We don’t have \( p \rightarrow r, q \rightarrow r, r \rightarrow p, \) or \( r \rightarrow q \), but we do have \( p \rightarrow \neg r \) (all of \( p \) is outside \( r \)), \( r \rightarrow \neg p \), and \( \neg q \rightarrow \neg p \).
  - In Figure 2, we see that stronger predicates stand for smaller sets of states. No states satisfy \( \mathcal{F} \) (false), so it is the strongest predicate. \( \mathcal{F} \) stands for \( \emptyset \), the empty set of states. Weaker states stand for larger sets of states. Since every state satisfies \( \mathcal{T} \) (true), \( \mathcal{T} \) stands for \( \Sigma \), the set of all states.
D. Strengthening and Weakening Conditions

- This idea of freely going from stronger to weaker predicates applies to individual triples, not just sequences.
  Both of the following properties are valid for both partial and total correctness of triples (i.e., for \( \models \) and \( \equiv_{\text{tot}} \)).
  - **Preconditions can always be strengthened**: If \( p_0 \rightarrow p_1 \) and \( \{ p_1 \} S \{ q \} \), then \( \{ p_0 \} S \{ q \} \).
  - **Postconditions can always be weakened**: If \( q_0 \rightarrow q_1 \) and \( \{ p \} S \{ q_0 \} \), then \( \{ p \} S \{ q_1 \} \).
  - **Example 2**: If \( \{ x \geq 0 \} S \{ y = 0 \} \) is valid, then so are
    - \( \{ x \geq 0 \} S \{ y = 0 \lor y = 1 \} \) (Weak. postcondition)
    - \( \{ x = 0 \} S \{ y = 0 \} \) (Str. precondition)
    - \( \{ x = 0 \} S \{ y = 0 \lor y = 1 \} \) (Str. precond. and weak. postcond.)
  - Note these rules still apply if \( p_0 \leftrightarrow p_1 \) or \( q_0 \leftrightarrow q_1 \).
  - **Example 3**: Let \( p_0 \equiv s = \text{sum}(0, k) \) and \( p_1 \equiv s + k + 1 = \text{sum}(0, k + 1) \).
    Since \( p_0 \leftrightarrow p_1 \), we can “strengthen” \( \{ p_1 \} S \{ q \} \) to \( \{ p_0 \} S \{ q \} \) (where \( q \equiv s = \text{sum}(0, k) \)).
  - Just because we can strengthen preconditions and weaken postconditions doesn’t mean we **should**. In the limit, strengthening preconditions and weakening postconditions gives us uninteresting correctness triples:
    - \( \sigma \models \{ \text{F} \} S \{ q \} \) and \( \sigma \equiv_{\text{tot}} \{ \text{F} \} S \{ q \} \) have the strongest possible preconditions
    - \( \sigma \models \{ p \} S \{ \text{T} \} \) has the weakest possible postcondition
    - \( \sigma \equiv_{\text{tot}} \{ p \} S \{ \text{T} \} \) has the weakest possible postcondition but it does tell us that \( S \) terminates when you start it in \( p \).
  - From the programmer’s point of view, if \( \{ p \} S \{ q \} \) has a bug, then strengthening \( p \) or weakening \( q \) can get rid of the bug without changing \( S \).
    - **Example 4**: If \( \{ p \} S \{ q \} \) causes an error if \( x = 0 \), we can tell the user to use \( \{ p \land x \neq 0 \} S \{ q \} \).
    - **Example 5**: If \( \{ p \} S \{ q \} \) causes an error because \( S \) terminates with \( y = 1 \) (and \( y = 1 \) does not imply \( q \)),
      we can tell the user to use \( \{ p \} S \{ q \lor y = 1 \} \).
  - From the user’s point of view, weaker preconditions and stronger postconditions make triples more useful:
    - **Weaker precondition adds starting states**: If \( p \rightarrow q \) then going from \( \{ p \} S \{ r \} \) to \( \{ q \} S \{ r \} \) provides
      more information because it increases the set of states that we say \( S \) can start in and end with \( r \).
      Basically, we combine \( \{ p \} S \{ r \} \) and \( \{ \neg p \lor q \} S \{ r \} \) to get \( \{ q \} S \{ r \} \).
• **Stronger postconditions remove ending states**: If \( q \rightarrow r \) then going from \( \{ p \} S \{ r \} \) to \( \{ p \} S \{ q \} \) provides more information because it decreases the set of states we say \( S \) will end in.

### E. The Weakest Precondition (wp)

- If \( \{ p \} S \{ q \} \) is valid, it’s often the case that \( p \) only describes a subset of the set of all states that lead to satisfaction of \( q \). I.e., we can often weaken \( p \) to some other predicate that still works as a precondition for \( S \) and \( q \). Say \( \{ p \} S \{ q \} \) is valid but \( \{ \neg p \} S \{ q \} \) is satisfiable: Say \( \{ r \} S \{ q \} \) is valid, where \( r \rightarrow \neg p \).
  - Since \( \{ p \} S \{ q \} \) and \( \{ r \} S \{ q \} \) are both valid, so is \( \{ p \lor r \} S \{ q \} \).
  - Since \( r \rightarrow \neg p \), we know \( p \rightarrow p \lor r \) is valid but \( p \lor r \rightarrow p \) is not, so \( p \) is strictly stronger than \( p \lor r \). The set of states that satisfy \( p \lor r \) is a strict superset of the set of states that satisfy \( p \).
  - This means that going from \( \{ p \} S \{ q \} \) to \( \{ p \lor r \} S \{ q \} \) weakens the precondition for \( S \) and \( q \): It increases the set of states we can start in and end up with \( q \) satisfied.

- Though we can strengthen a precondition arbitrarily far (to \( \mathbb{F} \) in the limit), we usually can’t weaken a precondition all the way to \( \top \): We keep weakening \( p \) while maintaining \( \{ p \} S \{ q \} \), but eventually we get to a \( p \) that can’t be weakened any more and still get us to \( q \).
  - In particular, if \( \{ \neg p \} S \{ q \} \) is unsatisfiable, then there’s no \( r \) that implies \( \neg p \) that we can use to weaken \( p \) to \( p \lor r \) while maintaining satisfaction of \( q \).

- **Example 6**:
  - If \( x \in \mathbb{N} \) (not \( \mathbb{Z} \) for once) then \( \{ x \geq 6 \} x := x \times x \{ x \geq 4 \} \) is valid, but it’s also the case that \( \{ x < 6 \} x := x \times x \{ x \geq 4 \} \) is satisfiable (e.g., when \( x = 5 \)), so we can extend \( x \geq 6 \) with \( x \geq 5 \) (i.e., \( x \geq 6 \lor x = 5 \)) and maintain validity.
  - On the other hand, \( \{ x \geq 2 \} x := x \times x \{ x \geq 4 \} \) is valid and \( \{ x < 2 \} x := x \times x \{ x \geq 4 \} \) is unsatisfiable, so there’s no way to weaken \( x \geq 2 \) and maintain validity.

- **Definition**: The **weakest precondition** of program \( S \) and postcondition \( q \), written \( wp(S, q) \), is the predicate \( w \) such that \( \models_{stt} \{ w \} S \{ q \} \) and for every \( \sigma \models \neg w \), we have \( M(S, \sigma) \not\models q \) (i.e., \( M(S, \sigma) \) includes \( \bot \) or for some \( \tau \in M(S, \sigma) \), we have \( \tau \models \neg q \).
  - **Example 7**: \( wp(x := y \times y, x \geq 4) \Leftrightarrow y \times y \geq 4 \). If \( \sigma \models y \times y \geq 4 \) then \( M(S, \sigma) \models x \geq 4 \). If \( \sigma \not\models y \times y \geq 4 \) then \( M(S, \sigma) \not\models x \geq 4 \).
  - **Example 8**: If \( x \in \mathbb{Z} \), then \( wp(\textbf{if } x \geq 0 \textbf{ then } y := x \textbf{ else } y := -x \textbf{ fi}, y = \text{abs}(x)) \Leftrightarrow \top \). No matter what state we start in, this program sets \( y \) to the absolute value of \( x \). Equivalently, in no state does it not set \( y \) to the absolute value of \( x \). Note that since you can’t weaken \( \top \), if \( \top \) is a valid precondition for \( S \) and \( q \), then it’s also the weakest precondition for \( S \) and \( q \).

- The \( wp(S, q) \) concept is important, so it’s good to look more into it.
  - First, as a predicate, \( wp(S, q) \) is only unique up to logical equivalence. In some sense, a predicate is just a name for the set of states that satisfy it, which is why (e.g.) \( x = 0 \Leftrightarrow x - 1 = 1 \). So “let \( w \Leftrightarrow wp(S, q) \)” means that \( w \) can be any of the logically equivalent predicates that are satisfied by exactly the set of states \( \{ \sigma \in \Sigma \mid M(S, \sigma) \models q \} \), which is the definition of \( wp(S, q) \) as a set of states.
If \( w \Leftrightarrow \text{wp}(S, q) \), then \( w \) has two properties. First, it’s a \textit{precondition} for \( S \) and \( q \): \( \equiv_{\text{tot}} \{ w \} S \{ q \} \). Note since \( w \) is a precondition, any stronger \( p \) that implies \( w \) is also a precondition: If \( \models \ p \rightarrow w \) then if \( \sigma \models p \) then \( \sigma \models w \), so \( M(S, \sigma) \models q \). Since \( \sigma \equiv_{\text{tot}} \{ p \} S \{ q \} \) holds for any \( \sigma \), we know \( p \) is a valid precondition.

The second property for \( w \), being the \textit{weakest} precondition, can be characterized a couple of ways:

- In the definition of \( w \Leftrightarrow \text{wp}(S, q) \), the weakest precondition part is \( (\sigma \models \neg w \Rightarrow M(S, \sigma) \not\models q) \).
- This turns out to be equivalent to saying that any valid precondition for \( S \) and \( q \) must be stronger than \( w \): If \( \models_{\text{tot}} \{ p \} S \{ q \} \), then \( \models p \rightarrow w \).
- To see this, take the contrapositive of \( (\sigma \models \neg w \Rightarrow M(S, \sigma) \not\models q) \), which is \( (M(S, \sigma) \models q \Rightarrow \sigma \models \neg w) \), which in turn implies \( \sigma \models w \), since \( \sigma \) is a state. If \( \sigma \equiv_{\text{tot}} \{ p \} S \{ q \} \), then \( \sigma \models p \) implies \( M(S, \sigma) \models q \), which implies \( \sigma \models w \), so \( \sigma \models p \rightarrow w \). Since this holds for any \( \sigma \) that satisfies \( p \), we know \( p \rightarrow w \) is valid.
- So if \( w \) is any valid precondition \( \models_{\text{tot}} \{ w \} S \{ q \} \), then \( \models p \rightarrow w \Leftrightarrow \sigma \equiv_{\text{tot}} \{ p \} S \{ q \} \). Only if \( w \) is the weakest precondition do we get the other direction, \( \sigma \equiv_{\text{tot}} \{ p \} S \{ q \} \Leftrightarrow \models p \rightarrow w \).

\[ \text{F. wp and Deterministic Programs} \]

- If \( S \) is deterministic, then \( S \) leads to a unique result: \( M(S, \sigma) = \{ \tau \} \) for some \( \tau \in \Sigma_\perp \).
- If \( S \) terminates normally \( (\tau \in \Sigma) \), then the start state \( \sigma \) is part of either \( \text{wp}(S, q) \) or \( \text{wp}(S, \neg q) \), depending on whether \( \tau \) satisfies \( q \) or \( \neg q \).
- Since \( \text{wp}(S, q) \) is the set of states that lead to satisfaction of \( q \), \( \neg \text{wp}(S, q) \) is the set of states that lead to an error or to satisfaction of \( \neg q \). Similarly, \( \neg \text{wp}(S, \neg q) \) is the set of states that lead to an error or to satisfaction of \( q \). The intersection of these two sets, \( \neg \text{wp}(S, q) \land \neg \text{wp}(S, \neg q) \), is the set of states that lead to an error.
- Since \( \sigma \) must lead \( S \) either to termination satisfying \( q \), termination satisfying \( \neg q \), or nontermination, every state satisfies exactly one of \( \text{wp}(S, q) \), \( \text{wp}(S, \neg q) \), and \( \neg \text{wp}(S, q) \land \neg \text{wp}(S, \neg q) \).
- Let \( E \equiv \neg \text{wp}(S, q) \land \neg \text{wp}(S, \neg q) \), then we get the identities
  - \( \neg \text{wp}(S, q) \Leftrightarrow E \lor \text{wp}(S, \neg q) \). The negation of “\( S \) terminates with \( q \) true” is “\( S \) doesn’t terminate or it terminates with \( q \) false”.
  - \( \neg \text{wp}(S, \neg q) \Leftrightarrow E \lor \text{wp}(S, q) \) is symmetric: The negation of “\( S \) terminates with \( q \) false” is “\( S \) doesn’t terminate or it terminates with \( q \) true”.
- If \( S \) contains a loop and \( M(S, \sigma) \) diverges \( (\sigma \in \Sigma) \), then \( \sigma \models \neg \text{wp}(S, q) \land \neg \text{wp}(S, \neg q) \). (See Figure 3.)
- On the left of Figure 3 is the set of all states \( \Sigma \) broken up into three partitions
  - The states that establish \( q \) form \( \text{wp}(S, q) = \{ \sigma \in \Sigma \mid M(S, \sigma) = \{ \tau \} \land \tau \models q \} \)
  - The states that establish \( \neg q \) form \( \text{wp}(S, \neg q) = \{ \sigma \in \Sigma \mid M(S, \sigma) = \{ \tau \} \land \tau \models \neg q \} \)
  - The states that lead to \( \perp \) form \( \neg \text{wp}(S, q) \land \neg \text{wp}(S, \neg q) = \{ \sigma \in \Sigma \mid M(S, \sigma) = \{ \perp \} \} \)
- The arrows indicate that starting from \( \text{wp}(S, q) \) or \( \text{wp}(S, \neg q) \) yields a state that satisfies \( q \) or \( \neg q \) respectively. Starting from a state outside both weakest preconditions leads to an error.
Disjunctive Postconditions

There are some relationships that hold between the two states \( \Sigma \) together cover together all possible states.

As with deterministic programs, the three predicates \( wp(S, q) \), \( wp(S, \neg q) \), and \( \neg wp(S, q) \land \neg wp(S, \neg q) \) together cover together all possible states.

- With deterministic programs, \( \sigma \models \neg wp(S, q) \land \neg wp(S, \neg q) \) iff running \( S \) in \( \sigma \) causes an error. This is because \( M(S, \sigma) \) is a singleton set \( \{ \tau \} \), and \( \tau \) must either be \( \bot \) or satisfy \( q \) or satisfy \( \neg q \).
- For deterministic \( S \), \( \sigma \models \neg wp(S, q) \land \neg wp(S, \neg q) \) iff \( M(S, \sigma) = \{ \bot \} \).
- But with nondeterministic programs, even if \( M(S, \sigma) \) doesn’t include \( \bot \), (i.e., \( \not\models_{\text{tot}} \{ T \} S \{ T \} \)), it can still mix states that satisfy \( q \) with states that satisfy \( \neg q \). A set like that neither satisfies \( q \) nor \( \neg q \). Figure 4 illustrates \( M(S, \sigma) \neq q \) and \( M(S, \sigma) \neq \neg q \) because \( M(S, \sigma) \supseteq \{ \tau_1, \tau_2 \} \) where \( \tau_1 \models q \) and \( \tau_2 \models \neg q \).
- For nondeterministic \( S \), \( \sigma \models \neg wp(S, q) \land \neg wp(S, \neg q) \) iff (1) \( \bot \in M(S, \sigma) \) or (2) There exist \( \tau_1, \tau_2 \in M(S, \sigma) \) with \( \tau_1 \models q \) and \( \tau_2 \models \neg q \).

**Example 9:** Let \( S \equiv \textbf{if } x \geq 0 \rightarrow x := 10 \ \textbf{else } x := 20 \) \textbf{fi}, and let \( \Sigma_0 = M(S, \{ x = 0 \}) \) be the set with two states \( \{ \{ x = 10 \}, \{ x = 20 \} \} \). Then \( \Sigma_0 \not\models x = 10, x \neq 10, x = 20, \) and \( x \neq 20 \). (We do have \( \Sigma_0 \models x = 10 \lor x = 20. \))

**H. Disjunctive Postconditions**

- There are some relationships that hold between the \( wp \) of a predicate and the \( wp \)'s of its subpredicates.
• E.g., if you start in a state that is guaranteed to lead to a result that satisfies $q_1$ and $q_2$ separately, then the result will also satisfy $q_1 \land q_2$, and vice versa. In symbols, $wp(S, q_1) \land wp(S, q_2) \iff wp(S, q_1 \land q_2)$.
  
  • This relationship holds for both deterministic and nondeterministic $S$.
  
  • The relationship between $wp(q_1 \lor q_2)$ and $wp(q_1)$ and $wp(q_2)$ differs for deterministic and nondeterministic $S$.

• Deterministic $S$: For all $S$, $wp(S, q_1) \lor wp(S, q_2) \iff wp(S, q_1 \lor q_2)$
  
• Nondeterministic $S$: For all $S$, $wp(S, q_1) \lor wp(S, q_2) \Rightarrow wp(S, q_1 \lor q_2)$, but $\iff$ doesn't hold for some $S$.
  
• For deterministic $S$, $M(S, \sigma) = \{ \tau \}$ for some $\tau \in \Sigma_\tau$. If $\tau \models q_1 \land q_2$ then either $\tau \models q_1$ or $\tau \models q_2$ (or both).

• So if $M(S, \sigma) \neq \{ \bot \}$, then $M(S, \sigma) \models q_1 \lor q_2$ iff $M(S, \sigma) \models q_1$ or $M(S, \sigma) \models q_2$.
  
• Because of this, $wp(S, q_1) \lor wp(S, q_2) \iff wp(S, q_1 \lor q_2)$.
  
• For nondeterministic $S$, we still have $wp(S, q_1) \lor wp(S, q_2) \Rightarrow wp(S, q_1 \lor q_2)$. I.e., if you start in a state that's guaranteed to terminate satisfying $q_1$, or guaranteed to terminate satisfying $q_2$, then that state is guaranteed to terminate satisfying $q_1 \lor q_2$.

• For nondeterministic $S$, the other direction, $wp(S, q_1) \lor wp(S, q_2) \iff wp(S, q_1 \lor q_2)$, doesn't always hold: $S$ can guarantee establishing $q_1 \lor q_2$ without leaving any way to guarantee satisfaction of just $q_1$ or just $q_2$.

• **Example 10:** Let $CoinFlip \equiv \textbf{if} \ T \Rightarrow x := 0 \ \textbf{else} \ x := 1 \ \textbf{fi}$.
  
  • For all $\sigma$, $M(CoinFlip, \sigma) = \{ \{ x = 0, x = 1 \} \}$, which $\models x = 0 \lor x = 1$ but $\nvdash x = 0$ and $\nvdash x = 1$.
  
  • Let $Heads \iff wp(CoinFlip, x = 0)$, $Tails = wp(CoinFlip, x = 1)$, and $Heads_or_Tails = wp(CoinFlip, x = 0 \lor x = 1)$. We find $Heads \iff Tails \Rightarrow F$ but $Heads_or_Tails \iff T$.
  
  • Altogether, ($Heads \lor Tails$) $\Rightarrow$ (but not $\iff$) $Heads_or_Tails$.

• So for nondeterministic $S$, even though $\models_{tot} \{ wp(S, q) \} S \{ q \}$, if $q$ is disjunctive, it's possible for you to run $S$ in a state $\sigma$ $\models \lnot wp(S, q)$ but still terminate without error in a state satisfying $q$. (For deterministic $S$, this won't happen.) E.g., if $S \equiv \textbf{if} B \ \textbf{then} \ x := 0 \ \textbf{else} \ x := 1 \ \textbf{fi}$, then $M(S, \sigma) \models x = 0$ or $M(S, \sigma) \models x = 1$ (tails), $wp(S, x = 0) \Rightarrow B$ and $wp(S, x = 1) \Rightarrow \lnot B$.

### I. The Weakest Liberal Precondition ($wlp$)

• The relationship between the **weakest precondition** ($wp$) and the **weakest liberal precondition** ($wlp$) is the same as total vs partial correctness.
  
  • $wp(S, q)$ is the set of start states that guarantee termination establishing $q$.
  
  • $wlp(S, q)$ is the set of start states that guarantee (causing an error or termination establishing $q$).

• **Definition:** The **weakest liberal precondition** of $S$ and $q$, written $wlp(S, q)$, is the predicate $w$ such that $\models \{ w \} S \{ q \}$ and for every $\sigma$ $\models \lnot w$, $\bot \notin M(S, \sigma)$ and $M(S, \sigma) \not\models q$.

• If we start in a state $\sigma$ satisfying $wlp(S, q)$ then either some execution path for $S$ in $\sigma$ causes an error or else all execution paths for $S$ in $\sigma$ lead to final states that $\not\models q$. If we start in a $\sigma$ satisfying $\lnot wlp(S, q)$, then every execution path for $S$ in $\sigma$ leads to a final state and at least one of the final states $\models \lnot q$.

• We always have $wp(S, q) \Rightarrow wlp(S, q)$; the other direction, $wp(S, q) \not\Leftrightarrow wlp(S, q)$, only holds if $S$ never causes an error.
• **Example 11:** Let \( W \equiv \textbf{while } x \neq 0 \textbf{ do } x := x-1; \ y := 0 \textbf{ od}, \) then for \( M(W, \sigma), \)
  
  \( \textbf{• If } \sigma \vdash x = 0 \textbf{ then } M(W, \sigma) = \{ \sigma \}. \) \( \text{Note if } \sigma \vdash x = 0 \land y = 0 \textbf{ then } M(W, \sigma) = \{ \sigma \} \) [9/26]

  \( \textbf{• If } \sigma \vdash x > 0 \textbf{ then } M(W, \sigma) = \{ \sigma[x \mapsto 0][y \mapsto 0] \} \)

  \( \text{Note the only way } W \textbf{ terminates with } y \neq 0 \textbf{ is if we run it in } x = 0 \land y \neq 0. \)

  \( \textbf{• If } \sigma \vdash x < 0 \textbf{ then } M(W, \sigma) = \{ \bot \} \textbf{ so for any postcondition } q, \ x < 0 \rightarrow \neg wp(W, q) \) and \( x < 0 \rightarrow \neg wp(W, q). \)

  \( \textbf{• If we look at a particular postcondition, say } q \equiv x = 0 \land y = 0, \) we find \( wp(W, q) \Leftrightarrow x > 0 \lor x = y = 0 \lor x < 0 \textbf{ and } wp(W, q) \Leftrightarrow x > 0 \lor x = y = 0. \) \( \text{For } \neg q \Leftrightarrow x \neq 0 \lor y \neq 0, \) since \( W \) can never terminate with \( x \neq 0, \) we find \( wp(W, \neg q) \Leftrightarrow wp(W, y \neq 0) \Leftrightarrow x = 0 \lor y \neq 0 \lor x < 0 \) and \( wp(W, \neg q) \Leftrightarrow wp(W, y \neq 0) \Leftrightarrow x = 0 \land y \neq 0. \)

  \( \text{The “being weakest” property of } wp \textbf{ is similar to that for } wp, \) but for partial correctness: \( \models \{ \text{wp}(S, q) \} S \{ q \} \) and for all \( p, \models \{ p \} S \{ q \} \textbf{ iff } p \rightarrow \text{wp}(S, q). \)

**J. Calculating wp for Loop-Free Programs**

• It’s easy to calculate the \( \text{wp} \) of a loop-free program.

  \( \textbf{• If a loop-free program cannot cause a runtime error then its } wp \textbf{ and } wp \textbf{ are the same, which is also nice.} \)

• The following algorithm takes \( S \) and \( q \) where \( S \) has no loops and syntactically calculates a particular predicate for \( wp(S, q), \) which is why it’s described using \( wp(S, q) \equiv \ldots \) instead of \( wp(S, q) \Leftrightarrow \ldots. \)

  \( \textbf{• wp(skip, } q) \equiv q \)

  \( \textbf{• wp(v := e, } Q(v)) \equiv Q(e) \textbf{ where } Q \textbf{ is a predicate function over one variable} \)

  \( \textbf{• The operation that takes us from } Q(v) \textbf{ to } Q(e) \textbf{ is called } \text{syntactic substitution}; \) we’ll look at it in more detail soon, but in the simple case, we simply inspect the definition of \( Q, \) searching its text for occurrences of the variable \( v \) and replacing them with copies of \( e. \)

  \( \textbf{• wp(S_1 : S_2, } q) \equiv wp(S_1, wp(S_2, q)) \)

  \( \textbf{• wp(if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi, } q) \equiv (B \rightarrow w_1) \land (\neg B \rightarrow w_2) \) where \( w_1 \equiv wp(S_1, q) \) and \( w_2 \equiv wp(S_2, q). \) \( \text{If you want, you can write } (B \land w_1) \lor (\neg B \land w_2), \) which is equivalent.

  \( \textbf{• wp(if } B_1 \rightarrow S_1 \sqcap B_2 \rightarrow S_2 \textbf{ fi, } q) \equiv (B_1 \rightarrow w_1) \land (B_2 \rightarrow w_2) \) where \( w_1 \equiv wp(S_1, q) \) and \( w_2 \equiv wp(S_2, q). \)

  \( \textbf{• For the nondeterministic } if, \) don’t write \( (B_1 \land w_1) \lor (B_2 \land w_2) \) instead of \( (B_1 \rightarrow w_1) \land (B_2 \rightarrow w_2); \) they aren’t logically equivalent. \( \text{When } B_1 \text{ and } B_2 \text{ are both true, either } S_1 \text{ or } S_2 \text{ can run, so we need } B_1 \land B_2 \rightarrow w_1 \land w_2. \)

  \( \textbf{• Using } (B_1 \land w_1) \lor (B_2 \land w_2) \) fails because it allows for the possibility that \( B_1 \text{ and } B_2 \text{ are both true but one of } w_1 \text{ and } w_2 \text{ is not true.} \) \( \text{This isn’t a problem when } B_2 \Leftrightarrow \neg B_1, \) which is why we can use \( (B \land w_1) \lor (\neg B \land w_2) \) with deterministic \( if \) statements.
A. Why

- The weakest precondition is the most general precondition that a program needs in order to run correctly.

B. Objectives

At the end of this activity you should be able to

- Define what a weakest preconditions is and how it's related to (and different from) preconditions in general
- Be able to calculate the \( wp \) of a simple loop-free program.

C. Problems

1. Let \( w \equiv wp(S, q) \) and let \( S \) be deterministic.
   a. For which \( \sigma \models w \) do we have \( \sigma \models_{\text{tot}} \{ w \} S \{ q \} \)?
   b. For which \( \sigma \models \neg w \) do we have \( \sigma \models \{ \neg w \} S \{ q \} \)?
   c. For which \( \sigma \models w \) do we have \( \sigma \models_{\text{tot}} \{ w \} S \{ \neg q \} \)?
   d. For which \( \sigma \models \neg w \) do we have \( \sigma \models \{ \neg w \} S \{ \neg q \} \)?
   e. If \( S \) is nondeterministic, how do we have to modify the statement in part (d)?

2. If \( \sigma \models w \) and \( \sigma \models \{ w \} S \{ q \} \) and \( \sigma \not\models_{\text{tot}} \{ w \} S \{ q \} \),
   a. What can we conclude about \( M(S, \sigma) \)?
   b. If in addition, \( S \) is deterministic, what more can we conclude about \( M(S, \sigma) \)?

3. For an arbitrary \( p \) (not necessarily one that implies \( w \)), what \( \models \) and \( \models_{\text{tot}} \) properties relationships do the triples
   a. \( \{ p \wedge w \} S \{ q \} \) and \( \{ \neg p \wedge w \} S \{ q \} \) have?
   b. \( \{ p \wedge \neg w \} S \{ \neg q \} \) and \( \{ \neg p \wedge \neg w \} S \{ \neg q \} \) have, if \( S \) is deterministic?
   c. \( \{ p \wedge \neg w \} S \{ q \} \) and \( \{ \neg p \wedge \neg w \} S \{ q \} \) have, if \( S \) is nondeterministic?

4. How are \( wp(S, q_1 \lor q_2) \) and \( wp(S, q_1) \cup wp(S, q_2) \), related if \( S \) is deterministic? If \( S \) is nondeterministic?

5. Calculate the \( wp \) in each of the following cases. (Just calculate the syntactically \( wp \); don't also logically simplify the result.)
   a. \( wp(k := k - s, n = 3 \land k = 4 \land s = -7) \).
   b. \( wp(n := n*(n-k), n = 3 \land k = 4 \land s = -7) \).
   c. \( wp(n := n*(n-k); k := k - s, n > k + s) \).
Solution to Activity 10 (Strength; Weakest Preconditions, pt. 1)

1. (Properties of weakest preconditions)
   a. For all $\sigma \models w$, we have $\sigma \models_{tot} \{w\} S \{q\}$, since $w$ is a precondition for $\models_{tot} \{\ldots\} S \{q\}$.
   b. For no $\sigma \models \neg w$ do we have $\sigma \models \{\neg w\} S \{q\}$ because for $w$ to be the weakest precondition for $S$ and $q$, it cannot be that $M(S, \sigma) \models q$.
   c. For no $\sigma \models w$ do we have $\sigma \models_{tot} \{w\} S \{\neg q\}$ because $w$ is a precondition for $\models_{tot} \{\ldots\} S \{q\}$.
   d. For all $\sigma \models \neg w$, we have $\sigma \models \{\neg w\} S \{\neg q\}$ because for $w$ to be the weakest precondition for $S$ and $q$, $\sigma \models \neg w$ implies $M(S, \sigma) \not\models q$. Since $S$ is deterministic, either $M(S, \sigma) = \{\bot\}$ or $M(S, \sigma) \models \neg q$. Either way, $\sigma \models \{\neg w\} S \{\neg q\}$.
   e. If $S$ is nondeterministic and $M(S, \sigma) \not\models q$, then as in the deterministic case, nontermination is a possibility ($\bot \in M(S, \sigma)$ can happen). Regardless, we no longer know $M(S, \sigma) \models \neg q$ because we can have $M(S, \sigma) \not\models q$ and $M(S, \sigma) \not\models \neg q$ simultaneously.

2. (Partial but not total correctness when the wp is satisfied)
   a. If $\sigma \models w$ and $\sigma \models \{w\} S \{q\}$ then $M(S, \sigma) \models \{\bot\} \models q$. If $\sigma \not\models_{tot} \{w\} S \{q\}$ then $M(S, \sigma) \not\models q$. This can only happen if $\bot \in M(S, \sigma)$. (I.e., $S$ can diverge under $\sigma$.)
   b. If in addition $S$ is deterministic, then we don’t just have $\bot \in M(S, \sigma)$, we have $\{\bot\} = M(S, \sigma)$. (I.e., $S$ diverges under $\sigma$.)

3. (Intersection with wp)
   a. $\models_{tot} \{p \land w\} S \{q\}$ and $\models_{tot} \{\neg p \land w\} S \{q\}$ follow from $w$ being a precondition under $\models_{tot}$.
   b. Because $w$ is weakest, we have for all $\sigma \models p \land \neg w$, that $\sigma \not\models_{tot} \{p \land \neg w\} S \{q\}$. If $S$ is deterministic, this implies $\sigma \not\models \{p \land \neg w\} S \{\neg q\}$. Similarly, for all $\sigma \models \neg p \land \neg w$, we have $\sigma \not\models \{p \land \neg w\} S \{\neg q\}$.
   c. If $S$ is nondeterministic then if $\sigma \models p \land \neg w$, we still know $\sigma \not\models_{tot} \{p \land \neg w\} S \{q\}$ but both $\sigma \models$ and $\sigma \not\models \{p \land \neg w\} S \{\neg q\}$ are possible. Similarly, if $\sigma \models \neg p \land \neg w$, we know $\sigma \not\models_{tot} \{\neg p \land \neg w\} S \{q\}$, but both $\sigma \models$ and $\sigma \not\models \{p \land \neg w\} S \{\neg q\}$ are possible.

4. For deterministic $S$, $wp(S, q_1 \lor q_2) = wp(S, q_1) \cup wp(S, q_2)$. For nondeterministic $S$, we have $\supseteq$ instead of $=$. (See Example 10.)

5. (Calculate wp)
   a. $wp(k := k - s, n = 3 \land k = 4 \land s = -7) \equiv n = 3 \land k - s = 4 \land s = -7$
   b. $wp(n := n \times (n - k), n = 3 \land k = 4 \land s = -7) \equiv n \times (n - k) = 3 \land k = 4 \land s = -7$
   c. $wp(n := n \times (n - k), k := k - s, n > k + s)$
      $\equiv wp(n := n \times (n - k), wp(k := k - s, n > k + s))$
      $\equiv wp(n := n \times (n - k), n > k - s + s)$
      $\equiv n \times (n - k) > k - s + s$