**Weakest Preconditions**

*Part 1: Definitions and Basic Properties*

CS 536: Science of Programming, Fall 2021

A. Why

- Weakest liberal preconditions ($\text{wlp}$) and weakest preconditions ($\text{wp}$) are the most general requirements that a program must meet to be correct.

B. Objectives

At the end of today you should understand

- What $\text{wlp}$ and $\text{wp}$ are and how they are related to preconditions in general.

C. The Weakest Precondition of $S$ and $q$

- Say we have a triple $\models_{\text{tot}} \{p_0\} S \{q\}$ and a predicate $p_1$. If $p_1 \rightarrow p_0$, then we can strengthen the precondition of our triple to $\models_{\text{tot}} \{p_1\} S \{q\}$.
  - But if the implication works the other way, $p_0 \rightarrow p_1$, then in general we can't simply weaken the precondition of our triple by replacing $p_0$ with $p_1$.
  - We can do the substitution if $\{p_1 \land \neg p_0\} S \{q\}$ holds, because we already know $\{p_1 \land p_0\} S \{q\}$\(^1\), so we can combine the two triples and get $\{p_1 \land (\neg p_0 \lor p_0)\} S \{q\}$, which simplifies to $\{p_1\} S \{q\}$.
  - (Note if $p_0 \leftrightarrow p_1$, then $\{p_0\} S \{q\}$ and $\{p_1\} S \{q\}$ are equivalent, so all this is only interesting if $p_0$ is strictly weaker than $p_1$; that is, $p_0 \rightarrow p_1$ but $p_1 \not\rightarrow p_0$)
  - In general, we can imagine a sequence of strictly weaker preconditions $p_0, p_1, p_2, \ldots$ for $S$ and $q$.
    - If $T$ appears in the sequence, it has to be at the end because no predicate is strictly weaker than true. It turns out (we won't prove this) that a sequence of strictly weakest preconditions always has a limit (possibly at infinity). This limit point has interesting properties.

  - Definition: The weakest precondition for $S$ and $q$, written $\text{wp}(S, q)$, is the condition $w$ where $w$ is a valid precondition for $q$ (under total correctness) and no strictly weaker valid precondition exists (i.e., all other preconditions are stronger than $w$). In symbols, $\models_{\text{tot}} \{w\} S \{q\}$, and for all $w'$, $(\models_{\text{tot}} \{w'\} S \{q\}$ if and only if $w' \rightarrow w$).
    - The $\Leftarrow$ part of the iff (sufficiency of $w' \rightarrow w$) holds because we're simply strengthening $w$ to $w'$.
    - The $\Rightarrow$ part of the iff (necessity of $w' \rightarrow w$) is the important part, the one that says $w$ is the weakest precondition under total correctness.

---

\(^1\){$p_1 \land p_0$} $S$ \{q\} holds because $\{p_0\} S \{q\}$.
• We'll write \( wp(S, q) \) as a predicate, but technically \( wp(S, q) \) is a set of states (the set of all states that are preconditions of \( S \) and \( q \) under total correctness). As sets, there are \( wp(S, q) \) that don't correspond well to writable predicates, and in those cases we'll have to write predicates that approximate \( wp(S, q) \).

• As a predicate, a \( wp \) is unique but only "up to logical equivalence" — any predicate logically equivalent to a \( wp \) is also a \( wp \). Later, we'll see a syntactic algorithm that helps us calculate some \( wp \)'s; in those cases, we'll prefer the representation produced by the algorithm.

D. The Weakest Liberal Precondition, \( wlp \)

• The weakest liberal precondition is analogous to the \( wp \) but for partial correctness instead of total correctness.

Definition: The weakest liberal precondition for \( S \) and \( q \), written \( wlp(S, q) \), is the condition \( w \) where \( w \) is a valid precondition for \( q \) under partial correctness and no strictly weaker valid precondition exists (i.e., all other preconditions are stronger than \( w \)). In symbols, \( \vdash \{w\} S \{q\} \), and for all \( w' \), \( \vdash \{w'\} S \{q\} \) if and only if \( w' \rightarrow w \).

Connection between \( wp \) and \( wlp \): Since total correctness is partial correctness plus termination, we know \( wp(S, q) \subseteq wlp(S, q) \land wp(S, T) \). In particular, \( wp(S, q) \Rightarrow wlp(S, q) \), and the contrapositive \( \neg wlp(S, q) \Rightarrow \neg wp(S, q) \) is also useful: If a state fails to be in the \( wlp \) of \( S \) and \( q \), it fails to be in the \( wp \) also.

E. Examples of \( wp \) and \( wlp \)

• Example 1: If \( x \) is an integer, then the weakest precondition and the weakest liberal precondition of \( y := x*x \) and \( (x \geq 0 \land y \geq 4) \leftrightarrow x \geq 2 \).
- The sequence \( x \geq 5, x \geq 4, x \geq 3, x \geq 2 \) is an example of strictly weaker preconditions, starting with \( x \geq 5 \) and ending with \( x \geq 2 \).
- Since our program can't diverge and can't cause a runtime error, the \( wp \) and \( wlp \) are the same.

• Example 2: The \( wp \) and \( wlp \) of if \( y \leq x \) then \( m := x \) else skip fi and \( m = \max(x, y) \) are \( (y > x \rightarrow m = y) \).
- The true branch sets up the postcondition when \( y \leq x \). The false branch (implicitly else skip) does nothing, so it has to be run in a state that satisfies the postcondition already.

• Example 3: The weakest precondition of while \( x \neq 0 \) do \( x := x-1 \) od and \( x = 0 \) is \( x \geq 0 \). Starting with \( x \geq 0 \) terminates with \( x = 0 \), and starting with \( x < 0 \) doesn't terminate.
- The \( wlp \) of the loop and postcondition is simply \( T \). Since we're ignoring termination, the body of the loop doesn't affect the fact that for while \( x \neq 0 \) ... to exit, \( x \) must be zero.
- Our loop terminates iff run with \( x \geq 0 \), so if \( W \) is our loop, then \( wp(W, T) \leftrightarrow x \geq 0 \).
- We can verify \( x \geq 0 \leftrightarrow wp(W, x = 0) \leftrightarrow wlp(W, x = 0) \land wp(W, T) \leftrightarrow T \land x \geq 0 \leftrightarrow x \geq 0 \).
• **Example 4**: The weakest precondition of \( W \equiv \text{while } x > 0 \text{ do } x := x - 1 \text{ od} \) and \( x \leq 0 \) is \( T \) (true). Again, starting with \( x \geq 0 \) terminates with \( x = 0 \), and if we want to terminate with some particular value of \( x < 0 \), we can just start with \( x \) equal to that value because the loop terminates immediately.
  
  • Since \( \wp(W, x \leq 0) \) is true, \( \wp(W, x \leq 0) \land \wp(W, T) \) must also be true, so both \( \wp(W, x \leq 0) \) and \( \wp(W, T) \) must be true. Semantically, we can also justify this by arguing that while \( x > 0 \) … terminates iff \( x \leq 0 \).

• **Example 5**: For \( S \) in general, the weakest precondition of \( S \) and \( T \) is the set of states in which running \( S \) terminates. I.e., \( \sigma \models \wp(S, T) \iff \bot \not\in M(S, \sigma) \).

  • For the \( \wp \) we find \( \wp(S, T) \iff T \). If we run \( S \) in any state \( \sigma \) then either \( S \) terminates (satisfying true) or it doesn’t terminate. Either way, \( \sigma \models \wp(S, T) \).

### F. Why Are \( \wp \) and \( \wp \) Important?

• To see why \( \wp \) and \( \wp \) are important, let’s look at the difference between them and arbitrary valid preconditions \( p \) for \( S \) and \( q \). Knowing \( \models_{\text{tot}} \{p\} S \{q\} \) or \( \models \{p\} S \{q\} \) doesn’t tell us anything about \( \{\neg p\} S \{??\} \) in general.
  
  • If we run start \( S \) in a state that doesn’t satisfy \( p \), then \( S \) might diverge, it might yield a runtime error, or it might or might not terminate in a state that satisfies \( q \).

• What make \( \wp \) and \( \wp \) special that their negations make for invalid preconditions: We know \( \not\models_{\text{tot}} \{\neg \wp(S, q)\} S \{q\} \) and \( \not\models \{\wp(S, q)\} S \{q\} \).

• **Notation**: \( D \) stands for a deterministic program, \( N \) for a nondeterministic program, and \( S \) for a program whose determinism is unspecified.
  
  • If \( \sigma \not\models \wp(D, q) \), then running \( D \) either doesn’t terminate or terminates in a state satisfying \( \neg q \). (Either way, you have a bug.)

  • If \( \sigma \not\models \wp(N, q) \), you’re guaranteed that \( M(N, \sigma) \not\models q \), but that only implies that some state in \( M(N, \sigma) \) is \( \bot \) or satisfies \( \neg q \), not that all of them do. In general, it’s possible that for some other execution path (i.e., a different \( \tau \in M(N, \sigma) \) you get), you might terminate satisfying \( q \).

### G. \( \wp \) and \( \wp \) for Deterministic and Nondeterministic Programs

• Table 1 shows the various possibilities for a state \( \sigma \) to be in the \( \wp \) or \( \wp \) of \( S \) and \( q \) or \( \neg q \).

• The first three outcomes describe \( M(S, \sigma) \) that satisfy \( q \) or \( \neg q \) or never terminate.
  
  • Row 1: \( M(S, \sigma) \models q \), so \( \sigma \) is in the \( \wp \) and \( \wp \) of \( S \) and \( q \).
  
  • Row 2: \( M(S, \sigma) \models \neg q \), so \( \sigma \) is in the \( \wp \) and \( \wp \) of \( S \) and \( \neg q \).
  
  • Row 3: \( M(S, \sigma) = \{\bot\} \), so \( \sigma \) is in the \( \wp \) of \( S \) and both \( q \) and \( \neg q \), but in the \( \wp \) of neither.

---

\(^2\) Recall that for a nondeterministic program to satisfy \( q \) (or \( \neg q \)), all its states must satisfy \( q \) (or \( \neg q \)).
A nondeterministic program can behave according to rows 1 - 3, but more generally, different execution paths can have different outcomes. The two combinations of outcomes that are compatible with $wlp$ are:

- **Row 4:** $M(N, \sigma) - \bot \models q$ I.e., he members of $M(N, \sigma)$ each $\models q$ or are $\bot$, so $\sigma \models wlp(N, q)$.
- **Row 5:** $M(N, \sigma) - \bot \models \neg q$ I.e., the members of $M(N, \sigma)$ each $\models \neg q$ or are $\bot$, so $\sigma \models wlp(N, \neg q)$.

The final two combinations of results are for nondeterministic programs that satisfy none of the combinations of $wp$ or $wlp$ of $N$ and $q$ or $\neg q$.

- **Row 6:** $\bot \in M(N, \sigma)$ and there are $\tau_1$ and $\tau_2 \in M(N, \sigma)$ where $\tau_1 \models q$ and $\tau_2 \models \neg q$.
  - We get $\sigma \models \neg wp(N, q) \land \neg wp(N, \neg q)$.
  - In addition, $\sigma \models \neg wp(N, q) \land \neg wp(N, \neg q)$ by contraposition of how $wp$ implies $wlp$.
- **Row 7** is row 6 with $\bot \in M(N, \sigma)$ included. Again, $\sigma \models \neg wp(N, q) \land \neg wp(N, \neg q)$ and $\sigma \models \neg wp(N, q) \land \neg wp(N, \neg q)$.

### H. Properties of wp and wlp for Deterministic and Nondeterministic Programs

- There are a number of properties connecting the $wp$, $wlp$, $\neg wp$, and $\neg wlp$ of $q$ and $\neg q$.
- Some properties are common to both deterministic and nondeterministic programs:
  1. $M(S, \sigma) = \{ \bot \} \Rightarrow wp(S, q) \land wp(S, \neg q)$
     - $M(S, \sigma) - \bot = \emptyset$, so $it \models q$ and $\models \neg q$, so $\sigma \models wlp(S, q) \land wlp(S, \neg q)$.
  2. $M(S, \sigma) = \{ \bot \} \Rightarrow \neg wp(S, q) \land \neg wp(S, \neg q)$
     - $M(S, \sigma) = \{ \bot \} \not\equiv q$ and $\not\equiv \neg q$, so $\sigma \models \neg wp(S, q) \land \neg wp(S, \neg q)$.
  3. $wlp(S, q) \land wlp(S, \neg q) \Rightarrow M(S, \sigma) = \{ \bot \}$

#### Table 1: wp and wlp and Possible Final States

<table>
<thead>
<tr>
<th>Row</th>
<th>$\sigma \models ...$</th>
<th>$\tau \in M(S, \sigma)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>2</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>3</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>4</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>5</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>6</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>7</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Row</th>
<th>$\sigma \models ...$</th>
<th>$\tau \in M(S, \sigma)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>2</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>3</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>4</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>5</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>6</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>7</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
</tbody>
</table>

- $\checkmark$ indicates that $\tau$ is a possible final state.

CS Dept., Illinois Institute of Technology - 4 - © J. Sasaki 2021
• For \( \sigma \models wp(S, q) \land wp(S, \neg q) \), we must have \( M(S, \sigma) - \bot = q \land \neg q \). So, \( M(S, \sigma) - \bot = \emptyset \), so \( M(S, \sigma) = \{ \bot \} \).

4. \( wp(S, q) \Rightarrow wp(S, \neg q) \)
   - If \( \sigma \models wp(S, q) \), then \( M(S, \sigma) \models q \), so \( M(S, \sigma) - \bot = q \), and so \( \sigma \models wp(S, q) \).

5. \( wp(S, q) \Rightarrow \neg wp(S, \neg q) \)
   - If \( \sigma \models wp(S, q) \), then \( M(S, \sigma) - \bot = q \), so for all \( \tau \in M(S, \sigma) - \bot \), \( \tau \not\models \neg q \). Since \( \bot \not\models \neg q \), we know \( M(S, \sigma) \not\models \neg q \), so \( \sigma \not\models wp(S, \neg q) \).

6. \( wp(S, q) \Rightarrow \neg wp(S, \neg q) \)
   - If \( \sigma \models wp(S, q) \), then \( \sigma \not\models \bot \) and \( M(S, \sigma) \models q \), so \( \bot \not\in M(S, \sigma) \). Since \( M(S, \sigma) \not= \emptyset \), there exists a \( \tau \in M(S, \sigma) - \bot \) with \( \tau \models q \). This tells us \( \tau \not\models \neg q \), and having such a \( \tau \) is enough to establish \( M(S, \sigma) - \bot \not\models \neg q \), so \( \sigma \not\models wp(S, \neg q) \). Since \( \sigma \not\models \bot \), we know \( \sigma \not\models \neg wp(S, \neg q) \).

• There are also properties that hold for deterministic programs but not nondeterministic programs.

7a. \( \neg wp(D, q) \land \neg wp(D, \neg q) \Rightarrow M(D, \sigma) = \{ \bot \} \)
   - For a deterministic program, \( M(D, \sigma) = \{ \tau \} \), where either \( \tau = \bot \), \( \tau \models q \), or \( \tau \models \neg q \). But \( \sigma \models \neg wp(D, q) \land \neg wp(D, \neg q) \) implies that \( M(D, \sigma) \not\models q \) and \( M(D, \sigma) \not\models \neg q \), which leaves \( M(D, \sigma) = \{ \bot \} \) as the only possibility.

7b. \( \neg wp(N, q) \land \neg wp(N, \neg q) \) doesn’t imply \( M(S, \sigma) = \{ \bot \} \)
   - For a nondeterministic program, if \( M(N, \sigma) \not\models q \) and \( M(N, \sigma) \not\models \neg q \), it’s still possible for \( M(N, \sigma) \) to contain non-\( \bot \) states. A simple counterexample is \( M(N, \sigma) = \{ \tau_1, \tau_2 \} \) where \( \tau_1 \models q \) and \( \tau_2 \models \neg q \). Note it’s possible that \( \bot \not\in M(N, \sigma) \), which definitely makes \( M(N, \sigma) \not= \{ \bot \} \).

8a. \( \neg wp(D, q) \Rightarrow wp(D, \neg q) \)
   - \( M(D, \sigma) = \{ \tau \} \) where \( \tau = \bot \), \( \tau \models q \), or \( \tau \models \neg q \). If \( \sigma \models \neg wp(D, q) \), then \( \tau \not\models q \) fails, which leaves \( \tau = \bot \) or \( \tau \models \neg q \), in which case \( M(D, \sigma) - \bot \not\models \neg q \), so \( \sigma \not\models wp(D, \neg q) \).

8b. \( \neg wp(N, q) \) doesn’t imply \( wp(D, \neg q) \)
   - When \( M(N, \sigma) \not\models q \), there can still be a \( \tau_1 \in M(N, \sigma) \) with \( \tau_1 \models q \), in which case \( \sigma \not\models wp(S, \neg q) \).

I. **Disjunctive Postconditions Behave Differently Under Nondeterminism**

• The \( wp \) and \( wp \) of conjunctive postconditions \( q_1 \land q_2 \) have a nice relationship with the \( wp \) and \( wp \) of \( q_1 \) and \( q_2 \) separately: \( wp(S, q_1 \land q_2) \Leftrightarrow wp(S, q_1) \land wp(S, q_2) \) and similarly for \( wp \).
  - This holds both for deterministic and nondeterministic programs.
• The situation is slightly different for disjunctive postconditions \( q_1 \lor q_2 \).
• The discussion below uses \( wp \) everywhere, but all the results also hold for \( wp \) (the arguments are similar).
• For both deterministic and nondeterministic programs, \( wp(S, q_1) \lor wp(S, q_2) \Rightarrow wp(S, q_1 \lor q_2) \)
  - Each of \( M(S, \sigma) \models q_1 \) and \( M(S, \sigma) \models q_2 \) imply \( M(S, \sigma) \models q_1 \lor q_2 \).
• However, the other direction, $wp(S, q_1 \lor q_2) \Rightarrow wp(S, q_1) \lor wp(S, q_2)$ only holds for deterministic programs.
  
  • $M(D, \sigma) = \{\tau\} \models q_1 \lor q_2$ implies $M(D, \sigma) = \{\tau\} \models q_1$ or $M(D, \sigma) = \{\tau\} \models q_2$.

• But for nondeterministic programs, $wp(S, q_1 \lor q_2) \Rightarrow wp(S, q_1) \lor wp(S, q_2)$ doesn’t have to hold.
  
  • Let $M(N, \sigma) = \Sigma_1 \cup \Sigma_2$ where $\Sigma_1 \models q_1$ and $\Sigma_2 \models q_2$ and $\Sigma_1 \neq \Sigma_2$. Then $\Sigma_1 \cup \Sigma_2 \models q_1 \lor q_2$, but unless one of $\Sigma_1$ and $\Sigma_2$ is empty, we don’t have $\Sigma_1 \cup \Sigma_2 \models q_1$ or $\Sigma_1 \cup \Sigma_2 \models q_2$.

• The standard example for this property is a coin-flip program.

• **Example 11**: Let $flip = if T \rightarrow x := 0 \Box T \rightarrow x := 1 fi$.
  
  • Let heads $= x = 0$ as and tails $= x = 1$, then $M(flip, \emptyset) = \{x = 0\}, \{x = 1\}$, which $\models heads \lor tails$ but $\not\models heads$ and $\not\models tails$. So $wp(flip, heads \lor tails) = T$ but $wp(flip, heads) = wp(flip, tails) = F$. 