Weakest Preconditions

Part 1: Definitions and Basic Properties

CS 536: Science of Programming, Spring 2020

A. Why

• Weakest liberal preconditions (wp) and weakest preconditions (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 wp and 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 \( \{p_1 \land p_0\} S \{q\} \)', 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.

\* \( \{p_1 \land p_0\} S \{q\} \) 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 develop approximations to \( wp(S, q) \) that we can write well.

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, \( \models \{ w \} S \{ q \} \), and for all \( w' \), \( (\models \{ w' \} S \{ q \} \iff w' \rightarrow w) \).

**Connection between \( wp \) and \( wlp \):** Since total correctness is partial correctness plus termination, we know \( wp(S, q) \models wlp(S, q) \land wp(S, T) \). In particular, \( wp(S, q) \models wlp(S, q) \), and the contrapositive \( \lnot wlp(S, q) \models \lnot wp(S, q) \) 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 \( y := x \times x \) and \( (x \geq 0 \land y \geq 4) \) is \( x \geq 2 \).
  - 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 \( wp(W, T) \iff x \geq 0 \).
  - We can verify \( x \geq 0 \iff wp(W, x = 0) \iff wlp(W, x = 0) \land wp(W, T) \iff T \land x \geq 0 \iff x \geq 0 \).
• **Example 4:** The weakest precondition of \( W = \textbf{while } x > 0 \textbf{ do } x := x - 1 \textbf{ od } \) 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, \( wp(W, x \leq 0) \land wp(W, \tau) \) must also be true, so both \( wlp(W, x \leq 0) \) and \( wp(W, \tau) \) must be true. Semantically, we can also justify this by arguing that \( \textbf{while } x > 0 \ldots \) 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., \( \sigma \models wp(S, T) \iff \bot \notin M(S, \sigma) \).

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

**F. Why Are** \( wp \) and \( wlp \) **Important?**

- To see why \( wp \) and \( wlp \) are important, let’s look at the difference between them and an arbitrary precondition \( p \) for \( S \) and \( q \).

  - Though \( \{p\} \ S \{q\} \) holds, in general, this doesn’t tell us anything about \( \{\neg p\} \ S \{??\} \).

  - 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 terminate with \( q \) true or with \( q \) false.

- Since \( wp \) and \( wlp \) preconditions, we know starting in a state satisfying them yields a correct result: \( \models_{\text{tot}} \{wp(S, q)\} \ S \{q\} \) and \( \models \{wlp(S, q)\} \ S \{q\} \). If \( S \) is nondeterministic, this even applies to all possible execution paths.

  - The difference with \( wp \) and \( wlp \) is that we know starting in a state not satisfying \textbf{will not} always yield a correct result: \( \not\models_{\text{tot}} \{\neg wp(S, q)\} \ S \{q\} \) 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 \( \sigma \not\models wp(N, q) \), the situation is slightly more complicated: You’re guaranteed that \( M(N, \sigma) \not= q \), but that only implies that \textbf{some} state in \( M(N, \sigma) \) is \( \bot \) or satisfies \( \neg q \), not that all of them do. Depending on the execution path (i.e., which \( \tau \in M(N, \sigma) \) you get), you might terminate with \( q \) satisfied.

**G.** \( wp \) and \( wlp \) **for Deterministic and Nondeterministic Programs**

- Table 1 below shows the various possibilities for a state \( \sigma \) to be in the \( wp \) or \( wlp \) 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 \( wlp \) of \( S \) and \( q \).
  * Row 2: \( M(S, \sigma) \models \neg q \), so \( \sigma \) is in the \( wp \) and \( wlp \) of \( S \) and \( \neg q \).
  * Row 3: \( M(S, \sigma) = \{\bot\} \), so \( \sigma \) is in the \( wlp \) of \( S \) and both \( q \) and \( \neg q \), but in the \( wp \) of neither.

\(^\dagger\) Recall that for a nondeterministic program to satisfy \( q \) or \( \neg q \), all its states must satisfy \( q \) or \( \neg q \).
Some properties are common to both deterministic and nondeterministic programs:

- 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: \( M(N, \sigma) - \bot \models q \) (the members of \( M(N, \sigma) \) each \( \models q \) or are \( \bot \)), so \( \sigma \models wlp(N, q) \).
  - Row 5: \( M(N, \sigma) - \bot \models \neg q \) (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: \( \bot \notin M(N, \sigma) \) 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 \( \sigma \models \neg wlp(N, q) \wedge \neg wlp(N, \neg q) \).
    - In addition, \( \sigma \models \neg wp(N, q) \wedge \neg wp(N, \neg q) \) by contraposition of how \( wp \) implies \( wlp \).
  - Row 7 is row 6 with \( \bot \in M(N, \sigma) \) included. Again, \( \sigma \models \neg wp(N, q) \wedge \neg wp(N, \neg q) \) and \( \sigma \models \neg wlp(N, q) \wedge \neg wlp(N, \neg q) \).

### 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. \( M(S, \sigma) = \{ \bot \} \Rightarrow wp(S, q) \wedge wlp(S, \neg q) \)  
     - \( M(S, \sigma) - \bot \models \emptyset \), so it \( \models q \) and \( \models \neg q \), so \( \sigma \models wlp(S, q) \wedge wlp(S, \neg q) \).
  2. \( M(S, \sigma) = \{ \bot \} \Rightarrow \neg wp(S, q) \wedge \neg wp(S, \neg q) \)  
     - \( M(S, \sigma) = \{ \bot \} \not\models q \) and \( \not\models \neg q \), so \( \sigma \models \neg wp(S, q) \wedge \neg wp(S, \neg q) \).
  3. \( wp(S, q) \wedge wlp(S, \neg q) \Rightarrow M(S, \sigma) = \{ \bot \} \)  
     - For \( \sigma \models wp(S, q) \wedge wlp(S, \neg q) \), we must have \( M(S, \sigma) - \bot \models q \) and \( \models \neg q \). To do this, \( M(S, \sigma) - \bot \) must be \( \emptyset \), so \( M(S, \sigma) = \{ \bot \} \).
4. \(wp(S, q) \Rightarrow wlp(S, q)\)
   - If \(\sigma \models wp(S, q)\), then \(M(S, \sigma) \models q\), so \(M(S, \sigma) - \bot \models q\), and so \(\sigma \models wlp(S, q)\)

5. \(wp(S, q) \Rightarrow \lnot wp(S, -q)\) [9/28]
   - if \(\sigma \models wlp(S, q)\), then \(M(S, \sigma) - \bot \models q\), so for all \(\tau \in M(S, \sigma) - \bot\), \(\tau \not\models -q\). Since \(\bot \not\models -q\) and \(M(S, \sigma) \not\models \emptyset\), we know \(M(S, \sigma) \not\models -q\), so [9/28] \(\sigma \not\models wlp(S, -q)\).

6. \(wp(S, q) \Rightarrow \lnot wlp(S, -q)\)
   - [11/12 rewritten] If \(\sigma \models wp(S, q)\), then \(\sigma \not\models \bot\), \(M(S, \sigma) \models q\), and \(\bot \not\in M(S, \sigma)\). Since \(M(S, \sigma) \not\models \emptyset\), there exists a \(\tau \in M(S, \sigma) - \bot\) with \(\tau \models q\). This tells us \(\tau \not\models -q\), and having such a \(\tau\) is enough to establish \(M(S, \sigma) - \bot \not\models -q\), so \(\sigma \not\models wlp(S, -q)\). Since \(\sigma \not\models \bot\), we know \(\sigma \models \lnot wlp(S, -q)\).
   - There are also properties that hold for deterministic programs but not nondeterministic programs. [9/28 some rewriting starts]

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

7b. \(\lnot wp(N, q) \land \lnot wp(N, -q)\) doesn’t imply \(M(S, \sigma) = \{\bot\}\)
   - For a nondeterministic program, if \(M(N, \sigma) \not\models q\) and \(\not\models -q\), 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 \(\tau_2 \models -q\). Note it’s possible that \(\bot \not\in M(N, \sigma)\), which definitely makes \(M(N, \sigma) = \{\bot\}\) false.

8a. \(\lnot wp(D, q) \Rightarrow wlp(D, -q)\)
   - Again, \(M(D, \sigma) = \{\tau\}\) where \(\tau = \bot\), \(\tau \models q\), or \(\tau \models -q\). If \(\sigma \models \lnot wp(S, q)\), then \(\tau \models q\) fails, which leaves \(\tau = \bot\) or \(\tau \models -q\), in which case \(M(D, \sigma) - \bot \models -q\), so \(\sigma \models wlp(S, -q)\).

8b. \(\lnot wp(N, q)\) doesn’t imply \(wlp(D, -q)\)
   - Again, if \(M(N, \sigma) \not\models q\), it’s still possible for a \(\tau_1 \in M(N, \sigma)\) to \(\models q\), in which case \(\sigma \not\models wlp(S, -q)\).

[I. Disjunctive Postconditions Behave Differently Under Nondeterminism]

- The \(wp\) and \(wlp\) of conjunctive postconditions \((q_1 \land q_2)\) have a nice relationship with the \(wp\) and \(wlp\) 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 \(wlp\).
  - 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 \(wlp\) (the arguments are similar).
- For both deterministic and nondeterministic programs, \(wp(S, q_1) \lor wp(S, q_2) \Rightarrow wp(S, q_1 \lor q_2)\)
  - \(M(S, \sigma) \models q_k\) implies \(M(S, \sigma) \models q_1 \lor q_2\) (where \(k = 1\) or \(2\)).
- 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.

---

© James Sasaki, 2020
• $M(S, \sigma) = \{ \tau \} \models q_1 \lor q_2$ implies $M(S, \sigma) = \{ \tau \} \models q_1$ or $M(S, \sigma) = \{ \tau \} \models q_2$.

• But for nondeterministic programs, $wp(S, q_1 \lor q_2) \implies wp(S, q_1) \lor wp(S, q_2)$ doesn’t have to hold.

• Let $M(N, \sigma) = \Sigma_1 \cup \Sigma_2$ where $\Sigma_1 \models q_1$ and $\Sigma_2 \models q_2$. 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 $\text{flip} = \text{if } T \rightarrow x := 0 \text{ then } x := 1 \text{ fi}$.

• Let $\text{heads} = x = 0$ as and $\text{tails} = x = 1$, then $M(\text{flip}, \emptyset) = \{ x = 0 \}, \{ x = 1 \}$, which $\models \text{heads} \lor \text{tails}$ but $\not\models \text{heads}$ and $\not\models \text{tails}$.