Weakest Preconditions, v.2

Part 1: Definitions and Basic Properties

CS 536: Science of Programming, Fall 2022

A. Why

- Weakest liberal preconditions (*wlp*) 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 *wlp* and *wp* are and how they are related to preconditions in general.

Part 1: The Deterministic Case

C. Weakening the Precondition of $\vDash_{tot}$ { p } S { q }

- Let's assume that *S* is deterministic. Figure 1 illustrates how $\vDash_{tot}$ *{ p } S { q }* works: If you take any state in *p* and follow the arrow by applying *S*, you end in a state that satisfies *q*.
  - (To illustrate partial correctness, we would add arrows from *p* to ¬*q* or to ⊥.)
- The predicate *r* intersects *p*, so states within *p* ∧ *r* are guaranteed to lead (via *S*) to states in *q*.
- Now, states in ¬*p* ∧ *r* might lead via *S* to *p* or ¬*p* or to ⊥, but if all of them lead to *p*, then we could extend our precondition *p* and we'd have *{ p* ∨ ¬*p* ∧ *r } S { q }*, which simplifies to *{* p ∨ r *} S { q }*.
  - A shorter way to say this is if $\vDash_{tot}$ *{ p } S { q }* and $\vDash_{tot}$ *{* ¬*p* ∧ *r } S { q }*, then $\vDash_{tot}$ *{* p ∨ r *} S { q }*.
  - Of course, in general, we don't know $\vDash_{tot}$ *{* ¬*p* ∧ *r } S { q }*, but if we can prove it, we can weaken the precondition *p* to *r*, which provides the user with more flexibility for running *S*.
- ***Definition: w is the weakest precondition of S and q*** (we write *w = wp(S, q)*) if *w* is a precondition that can't be weakened. I.e., $\vDash_{tot}$ *{ w } S { q }* and there is no *r* strictly stronger than *w* such that $\vDash_{tot}$ *{ r } S { q }*.
  - We already know the converse; we've been calling it precondition strengthening: If $\vDash_{tot}$ *{ w } S { q }*, then knowing *r* → *w* lets us conclude (is ***sufficient*** for) $\vDash_{tot}$ *{ r } S { q }*.
  - Being the weakest precondition makes *r* → *w* a ***necessary*** condition for $\vDash_{tot}$ *{ r } S { q }*.
  - So if *w* is the weakest precondition, then *{ r } S { q }* iff *w* → *r*.
  - In terms of states, $wp(S, q) = \{ \sigma \in \Sigma \mid M(S, \sigma) \vDash q \}$

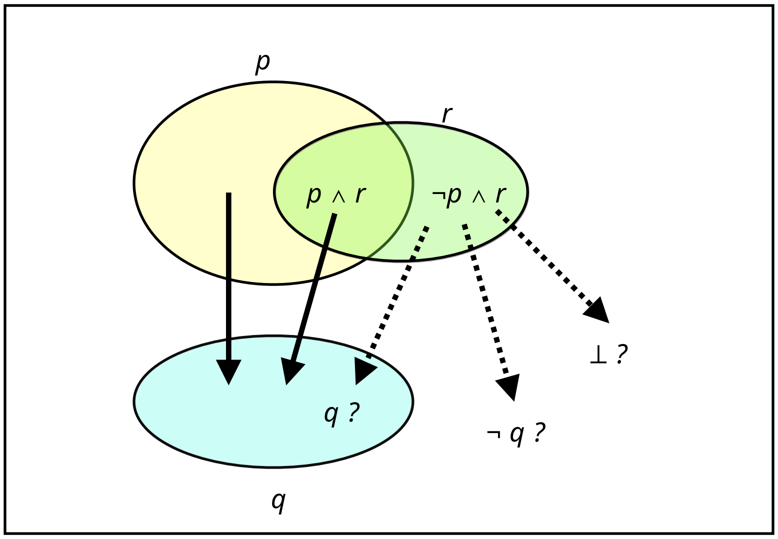

Figure 1: Extending Precondition of { p } S { q }
(Under total correctness)

- Recall that in general, $\vDash_{tot}$ *{ p } S { q }* doesn't tell us anything about *M(S, σ)* if $\sigma \nvDash p$. But if *p* is weakest, we know $M(S, \sigma) \nvDash q$.
  - For deterministic programs, we can state this using partial correctness: If *w = wp(S, q)* and S is deterministic then $\vDash \{\neg w\}\ S\ \{\neg q\}$. If $\sigma \vDash \neg w$ then $M(S, \sigma) = \{ \tau \}$ where $\tau = \bot$ or $\tau \vDash \neg q$.
- 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)*.
- Usually, we talk about "the" *wp(S, q),* but as a predicate, a *wp* is unique only "up to logical equivalence": If $u \Leftrightarrow w$, then *u* is also a wp. So in general if we have that the *wp(S, q)* is $x > 0$, then $x \geq 1$ and $0 < x$, etc. are also wp's.
- But 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 a valid precondition for *q* under partial correctness where no strictly weaker valid precondition exists.
  - In symbols, $w = wlp(S, q)$ iff $\vDash \{w\}\ S\ \{q\}$ and for all *u*, $\vDash \{u\}\ S\ \{q\}$ if and only if $\vDash u \rightarrow w$.
  - In terms of states, $wlp(S, q) = \{ \sigma \in \Sigma \mid M(S, \sigma) - \bot \vDash q \}$.

$wlp(S, q)$	$\sigma \in wp(S, q)$ iff $M(S, \sigma) = \{ \tau \} \vDash q$	
	$M(S, \sigma) = \{ \bot \}$	$wlp(S, \neg q)$
	$\sigma \in wp(S, \neg q)$ iff $M(S, \sigma) = \{ \tau \} \vDash \neg q$	

Figure 2: The Weakest Liberal Precondition for Deterministic S

Relationships Between wp and wlp

- Figure 2 illustrates the relationships between *wp* and *wlp* for deterministic programs.
- The top third shows the states in $wp(S, q)$: For them, $M(S, \sigma)$ satisfies q.
- The bottom third shows the states in $wp(S, \neg q)$: For them, $M(S, \sigma)$ satisfies $\neg q$.
- The middle third shows that states that cause nontermination.
  - Adding the nonterminating states to $wp(S, q)$ gives $wlp(S, q)$.
  - Adding the nonterminating states to $wp(S, \neg q)$ gives $wlp(S, \neg q)$.
  - Subsequently, $\neg wp(S, \neg q) \Leftrightarrow wlp(S, q)$ and $\neg wp(S, q) \Leftrightarrow wlp(S, \neg q)$.
- And one more relationship: $wlp(S, q) \wedge wlp(S, \neg q)$ describes the states that cause nontermination.

Why Are wp and wlp Important?

- The reason *wp* and *wlp* are important is that if you have a precondition and can show that it's the weakest precondition, you have the most general solution to "What states can I start in and successfully end in q ?
  - With *wp*, "successfully end" means "terminates satisfying q". With *wlp*, it means "if we terminate, we terminate satisfying q".
- The solution is most general in the sense that any state not satisfying the *wp* or *wlp* is guaranteed to ***not*** successfully end in q.
- Compare with non-weakest preconditions, where starting in a state not satisfying the precondition might end successfully or end not successfully (satisfying $\neg q$) or not terminate.

E. Examples of wp and wlp

- ***Example 1***: The assignment $y := x*x$ always terminates, so wp and wlp behave identically on it. $wp(y := x*x,\ x \geq 0 \wedge y \geq 4) \Leftrightarrow wlp(y := x*x,\ x \geq 0 \wedge y \geq 4) \Leftrightarrow x \geq 2$.
- ***Example 2***: The *wp* and *wlp* of ***if*** $y \leq x$ ***then*** $m := x$ ***else skip fi*** and $m = max(x, y)$ are $(y > x \rightarrow m = y)$.
  - Later, we'll see an algorithm for calculating the *wp* in this instance, but for now, intuitively, the true branch sets up the postcondition when $y \leq x$. The false branch (implicitly ***else skip***) runs when $y > x$ but it does nothing, we need to already be in a state that satisfies the postcondition, namely $m = y$.
- ***Example 3***: The weakest precondition of ***while*** $x \neq 0$ ***do*** $x := x\text{-}1$ ***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 $x \geq 0 \Leftrightarrow wp(W, x = 0) \Leftrightarrow wlp(W, x = 0) \wedge wp(W, T) \Leftrightarrow T \wedge x \geq 0 \Leftrightarrow x \geq 0$.

- ***Example 4***: The weakest precondition of $W \equiv$ ***while*** $x > 0$ ***do*** $x := x\text{-}1$ ***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 $T \Leftrightarrow wp(W, x \leq 0) \Leftrightarrow wlp(W, x \leq 0) \wedge wp(W, T)$, both $wlp(W, x \leq 0)$ and $wp(W, T) \Leftrightarrow T$. Semantically, we can also justify this by arguing that ***while*** $x > 0$ … terminates immediately iff $x \leq 0$.
- ***Example 5***: For any S and σ, either we terminate (in a state satisfying true) or we don't terminate. Therefore $wlp(S, T) \Leftrightarrow T$. Also, since $wlp(S, T) \Leftrightarrow \neg wp(S, \neg T) \Leftrightarrow T$, we see $wp(S, F) \Leftrightarrow F$. (In Figure 2 terms, the bottom third of the diagram is empty because running S in σ never terminates in a state satisfying false.)

Part 2: The Nondeterministic Case

- With nondeterministic programs, *wp* and *wlp* are more complicated (of course). The basic definitions are the same:
  - $\sigma \in wp(S, q)$ iff $M(S, \sigma) \vDash q$ or equivalently $\vDash_{tot} \{ p \}\ S\ \{ q \}$ iff $\vDash wp(S, q) \rightarrow p$
  - $\sigma \in wlp(S, q)$ iff $M(S, \sigma) - \bot \vDash q$ or equivalently $\vDash \{ p \}\ S\ \{ q \}$ iff $\vDash wlp(S, q) \rightarrow p$
- Let $\Sigma_0 = M(S, \sigma)$ or $M(S, \sigma) - \bot$ depending on whether we're discussing *wp* or *wlp*.
- Since Σ_0 satisfies q iff every individual state in Σ_0 satisfies q, nonsatisfaction only requires one counterexample state:
  - $\sigma \notin wp(S, q)$ iff for some $\tau \in M(S, \sigma)$, we have $\tau = \bot$ or $\tau \nvDash q$ (and since τ is a state, $\tau \vDash \neg q$).
  - $\sigma \notin wlp(S, q)$ iff for some $\tau \in M(S, \sigma)$, we have $\tau \nvDash q$ (and since τ is a state, $\tau \vDash \neg q$).
- But there are no constraints on other members of Σ_0, so $\sigma \notin wp(S, q)$ and $\sigma \notin wlp(S, q)$ are both compatible with having $\tau \in M(S, \sigma)$ with $\tau \vDash q$.

F. 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. $M(S, \sigma) = \{\bot\} \Rightarrow wlp(S, q) \wedge wlp(S, \neg q)$
     - $M(S, \sigma) - \bot = \varnothing$, so it $\vDash q$ and $\vDash \neg q$, so $\sigma \vDash wlp(S, q) \wedge wlp(S, \neg q)$.
  2. $M(S, \sigma) = \{\bot\} \Rightarrow \neg wp(S, q) \wedge \neg wp(S, \neg q)$
     - $M(S, \sigma) = \{\bot\} \nvDash q$ *and* $\nvDash \neg q$, *so* $\sigma \vDash \neg wp(S, q) \wedge \neg wp(S, \neg q)$.

3. $wlp(S, q) \wedge wlp(S, \neg q) \Rightarrow M(S, \sigma) = \{\perp\}$
   - For $\sigma \vDash wlp(S, q) \wedge wlp(S, \neg q)$, we must have $M(S, \sigma) - \perp \vDash q \wedge \neg q$. So, $M(S, \sigma) - \perp = \varnothing$, so $M(S, \sigma) = \{\perp\}$.
4. $wp(S, q) \Rightarrow wlp(S, q)$
   - If $\sigma \vDash wp(S, q)$, then $M(S, \sigma) \vDash q$, so $M(S, \sigma) - \perp \vDash q$, and so $\sigma \vDash wlp(S, q)$.
5. $wlp(S, q) \Rightarrow \neg wp(S, \neg q)$
   - If $\sigma \vDash wlp(S, q)$, then $M(S, \sigma) - \perp \vDash q$, so for all $\tau \in M(S, \sigma) - \perp$, $\tau \nvDash \neg q$. If $\perp \in M(S, \sigma)$ then it $\nvDash \neg q$, so $\tau \nvDash \neg q$.
6. $wp(S, q) \Rightarrow \neg wlp(S, \neg q)$
   - If σ is in $wp(S, q)$ then $M(S, \sigma) \vDash q$. For σ to be in $wlp(S, \neg q)$, we need every $\tau \in M(S, \sigma)$ to be either $\perp$ or to satisfy $\neg q$. But every $\tau \in M(S, \sigma)$ satisfies q, so it's neither $\perp$ nor satisfies $\neg q$. So if σ is in $wp(S, q)$, it's not in $wlp(S, \neg q)$, it's in $\neg wlp(S, \neg q)$.

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

7a. If S is deterministic, then $\neg wp(S, q) \wedge \neg wp(S, \neg q) \Rightarrow M(S, \sigma) = \{\perp\}$.
   - For deterministic S, $M(S, \sigma)$ = some $\{\tau\}$, where either $\tau = \perp$, $\tau \vDash q$, or $\tau \vDash \neg q$. But $\sigma \vDash \neg wp(S, q) \wedge \neg wp(S, \neg q)$ implies that $M(S, \sigma) \nvDash q$ and $M(S, \sigma) \nvDash \neg q$, which leaves $M(S, \sigma) = \{\perp\}$ as the only possibility.

7b. If S is nondeterministic, then $\neg wp(S, q) \wedge \neg wp(S, \neg q)$ doesn't imply $M(S, \sigma) = \{\perp\}$.
   - For a nondeterministic program, if $M(S, \sigma) \nvDash q$ and $M(S, \sigma) \nvDash \neg q$, it's still possible for $M(S, \sigma)$ to contain non-$\perp$ states. A simple counterexample is $M(S, \sigma) = \{\tau_1, \tau_2\}$ where $\tau_1 \vDash q$ and $\tau_2 \vDash \neg q$. Note it's possible that $\perp \notin M(S, \sigma)$, which definitely makes $M(S, \sigma) \neq \{\perp\}$.

8a. If S is deterministic, then $\neg wp(S, q) \Rightarrow wlp(S, \neg q)$
   - $M(S, \sigma) = \{\tau\}$ where $\tau = \perp$, $\tau \vDash q$, or $\tau \vDash \neg q$. If $\sigma \vDash \neg wp(S, q)$, then $\tau \vDash q$ fails, which leaves $\tau = \perp$ or $\tau \vDash \neg q$, in which case $M(S, \sigma) - \perp \vDash \neg q$, so $\sigma \vDash wlp(S, \neg q)$.

8b. If S is nondeterministic, then $\neg wp(S, q)$ doesn't imply $wlp(S, \neg q)$
   - When $M(S, \sigma) \nvDash q$, there can still be a $\tau_1 \in M(S, \sigma)$ with $\tau_1 \vDash q$, in which case $\sigma \nvDash wlp(S, \neg q)$.

G. Disjunctive Postconditions Behave Differently Under Nondeterminism

- For deterministic and nondeterministic both, the *wp/wlp* of a conjunction is the same as the conjunction of the *wp/wlp*'s.
  - $wp(S, q_1) \wedge wp(S, q_2) \Leftrightarrow wp(S, q_1 \wedge q_2)$
  - $wlp(S, q_1) \wedge wlp(S, q_2) \Leftrightarrow wlp(S, q_1 \wedge q_2)$
- Also, the disjunction of the *wp/wlp*'s implies the *wp/wlp* of the disjunction:
  - $wp(S, q_1) \vee wp(S, q_2) \Rightarrow wp(S, q_1 \vee q_2)$
  - $wlp(S, q_1) \vee wlp(S, q_2) \Rightarrow wlp(S, q_1 \vee q_2)$

- However, the other direction only works for deterministic programs:
  - For deterministic S only,
    - $wp(S, q_1 \vee q_2) \Rightarrow wp(S, q_1) \vee wp(S, q_2)$
    - $wlp(S, q_1 \vee q_2) \Rightarrow wlp(S, q_1) \vee wlp(S, q_2)$
- But for nondeterministic programs, $wp(S, q_1 \vee q_2) \Rightarrow wp(S, q_1) \vee wp(S, q_2)$ doesn't have to hold.
- The standard example for this property is a coin-flip program.
- ***Example 11***: Let $flip \equiv$ *if* $T \rightarrow x := 0 \ \square\ T \rightarrow x := 1$ *fi*.
  - Let $heads \equiv x = 0$ as and $tails \equiv x = 1$, then $M(flip, \varnothing) = \{\{x = 0\}, \{x = 1\}\}$, which $\vDash heads \vee tails$ but $\nvDash heads$ and $\nvDash tails$. So $wp(flip, heads \vee tails) = T$ but $wp(flip, heads) = wp(flip, tails)$

- In general, let $M(S, \sigma) = \Sigma_1 \cup \Sigma_2$ where $\Sigma_1 \vDash q_1$ and $\Sigma_2 \vDash q_2$. If q_1 and q_2 are not $\Leftrightarrow$ under σ. then $\Sigma_1 \neq \Sigma_2$. Also, assume Σ_1 and Σ_2 are not $\varnothing$. Then $\Sigma_1 \cup \Sigma_2 \vDash q_1 \vee q_2$, but since $\Sigma_1 \cup \Sigma_2$ includes elements that satisfy q_1 and q_2, we don't have $\Sigma_1 \cup \Sigma_2 \vDash q_1$ or $\Sigma_1 \cup \Sigma_2 \vDash q_2$. separately.

