Weakest Preconditions

Part 1: Definitions and Basic Properties CS 536: Science of Programming, Spring 2023

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 $\neg q$ or to \perp .)
- The predicate r intersects p, so states within $p \wedge r$ are guaranteed to lead (via S) to states in q.
- States in $\neg p \land r$ might lead via *S* to *p* or $\neg p$ or to \bot , but if all of them lead to *p*, then we could extend our precondition *p* and we'd have $\{p \lor \neg p \land r\}S\{q\}$, which simplifies to $\{p \lor r\}S\{q\}$.
 - A shorter way to say this is if $\models_{tot} \{p\} S\{q\}$ and $\models_{tot} \{\neg p \land r\} S\{q\}$, then $\models_{tot} \{p \lor r\} S\{q\}$.
 - Of course, in general, we don't know $\vDash_{tot} \{ \neg p \land r \} S \{ q \}$, but if we can prove it, we can weaken the precondition p to r, which provides the user with more flexibility for running S. But sometimes, we can't weaken the precondition any more than it is already.
- *Definition: w is the weakest precondition of S and q* (we write *w* = *wp*(*S*, *q*)) if *w* is a pre-condition that can't be weakened. I.e., ⊨_{tot} {*w*}*S*{*q*} and there is no *r* strictly stronger than *w* such that ⊨_{tot} {*r*}*S*{*q*}.
 - The converse holds, since it's just precondition strengthening: If $\vDash_{tot} \{w\} S\{q\}$, then knowing $r \rightarrow w$ lets us conclude (is *sufficient* for) $\vDash_{tot} \{r\} S\{q\}$.
 - Being the weakest precondition makes $r \rightarrow w$ a *necessary* condition for $\vDash_{tot} \{r\} S\{q\}$.
 - So if *w* is the weakest precondition, then $\{r\}S\{q\}$ iff $w \rightarrow r$.
 - In terms of states, $wp(S,q) = \{\sigma \in \Sigma \mid M(S,\sigma) \models q\}$



Figure 1: Extending Precondition of {p}S{q}

- Recall that in general, $\vDash_{tot} \{p\} S\{q\}$ doesn't tell us anything about $M(S, \sigma)$ 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 $\models \{\neg w\}S\{\neg q\}$. If $\sigma \models \neg w$ then $M(S, \sigma) = \{\tau\}$ where $\tau = \bot$ or $\tau \models \neg q$.
- Writing *wp*(*S*, *q*) as a predicate is convenient, but technically the weakest precondition of *S* under *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 ⇔ w, then u is also a wp. For example, if the wp (S, q) is x > 0, then x ≥ 1 and 0 < x and so on are also wp's.
- Later we'll see a syntactic algorithm that helps us calculate some wp's; in those cases, we'll write $wp(S,q) \equiv w$ where w is the syntactic 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 $\models \{w\}S\{q\}$ and for all $u, \models \{u\}S\{q\}$ if and only if $\models u \rightarrow w$.
 - In terms of states, $wlp(S, q) = \{\sigma \in \Sigma \mid M(S, \sigma) \bot \models q\}$.

$$wlp(S,q) \left\{ \begin{array}{c} \sigma \in wp(S,q) \text{ iff } M(S,\sigma) = \{\tau\} \vDash q \\ \\ M(S,\sigma) = \{\bot\} \\ \\ \sigma \in wp(S,\neg q) \text{ iff } M(S,\sigma) = \{\tau\} \vDash \neg q \end{array} \right\} wlp(S,\neg q)$$

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): Those states in $M(S, \sigma)$ satisfy q.
- The bottom third shows the states in $wp(S, \neg q)$: Those states in $M(S, \sigma)$ satisfy $\neg q$.
- 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 $\neg wp(S, q) \Leftrightarrow wlp(S, \neg q)$.
- The relationship: $wlp(S,q) \land wlp(S, \neg q)$ 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. $wp(y := x * x, x \ge 0 \land y \ge 4) \Leftrightarrow wlp(y := x * x, x \ge 0 \land y \ge 4) \Leftrightarrow x \ge 2.$
- **Example 2**: The wp and wlp of if $y \le x$ then m := x else skip fi and m = max(x, y) are $(y > x \rightarrow m = y)$.
 - Later, we'll see how to calculate the wp in this instance, but for now, let's look at it intuitively. The true branch sets up the postcondition when $y \le x$. The false branch (the implicit *else skip*) runs when y > x and doesn't change the state, so we need the postcondition m = y to already be satisfied.

- **Example 3**: The weakest precondition of **while** $x \neq 0$ **do** x := x 1 **od** and x = 0 is $x \ge 0$. Starting with $x \ge 0$ terminates with x = 0. 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 $x \ge 0$, so if W is our loop, then $wp(W, T) \Leftrightarrow x \ge 0$.
 - We can verify $x \ge 0 \Leftrightarrow wp(W, x = 0) \Leftrightarrow wlp(W, x = 0) \land wp(W, T) \Leftrightarrow T \land x \ge 0 \Leftrightarrow x \ge 0$.
- *Example 4*: The weakest precondition of $W \equiv$ *while* x > 0 *do* x := x 1 *od* and $x \le 0$ is T (true). Again, starting with $x \ge 0$ terminates with x = 0. We can terminate with any negative value for x simply by running the loop with that value; the loop terminates immediately without changing x.
 - Since $T \Leftrightarrow wp(W, x \le 0) \Leftrightarrow wlp(W, x \le 0) \land wp(W, T)$, both $wlp(W, x \le 0)$ and $wp(W, T) \Leftrightarrow T$. Semantically, we can also justify this by arguing that *while* x > 0 ... terminates immediately iff $x \le 0$.
- **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 have that $wp(S, F) \Leftrightarrow F$. (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 $M(S,\sigma) \models q$ or equivalently $\models_{tot} \{p\}S\{q\}$ iff $\models wp(S,q) \rightarrow p$.
 - $\sigma \in wlp(S,q)$ iff $M(S,\sigma) \perp \models q$ or equivalently $\models \{p\} S\{q\}$ iff $\models wlp(S,q) \rightarrow p$.
- 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 $\tau \models q$.

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. $M(S,\sigma) = \{\bot\} \Rightarrow wlp(S,q) \land wlp(S,\neg q)$

- $M(S, \sigma) \rightarrow = \emptyset$, so it $\models q$ and $\models \neg q$, so $\sigma \models wlp(S, q) \land wlp(S, \neg q)$.
- 2. $M(S, \sigma) = \{\bot\} \Rightarrow \neg wp(S, q) \land \neg wp(S, \neg q)$
 - $M(S, \sigma) = \{ \perp \} \nvDash q \text{ and } \nvDash \neg q, \text{ so } \sigma \vDash \neg wp(S, q) \land \neg wp(S, \neg q).$
- 3. $wlp(S,q) \land wlp(S,\neg q) \Rightarrow M(S,\sigma) = \{\bot\}$
 - For $\sigma \models wlp(S, q) \land wlp(S, \neg q)$, we must have $M(S, \sigma) \bot \models q \land \neg q$. Since no actual state satisfies $\models q \land \neg q$, that implies that $M(S, \sigma) \bot \models \emptyset$, 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. $wlp(S,q) \Rightarrow \neg wp(S,\neg q)$
 - If $\sigma \models wlp(S, q)$, then $M(S, \sigma) \bot \models q$, so for all $\tau \in M(S, \sigma) \bot$, we have $\tau \nvDash \neg q$. If $\bot \in M(S, \sigma)$ then $M(S, \sigma) \nvDash \neg q$, so $\tau \nvDash \neg q$.
- 6. $wp(S,q) \Rightarrow \neg wlp(S,\neg q)$
 - If σ is in wp(S, q) then $M(S, \sigma) \models q$. For σ to be in $wlp(S, \neg q)$, we need every $\tau \in M(S, \sigma)$ to be either \bot or to satisfy $\neg q$. But every $\tau \in M(S, \sigma)$ satisfies q, so $\tau \neq \bot$ and τ doesn't satisfy $\neg q$. 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 $\neg wp(S,q) \land \neg wp(S,\neg q) \Rightarrow M(S,\sigma) = \{\bot\}$.
 - For deterministic *S*, we know $M(S, \sigma) = \text{some } \{\tau\}$, where $\tau = \bot, \tau \models q$, or $\tau \models \neg q$. But $\sigma \models \neg wp(S, q) \land \neg wp(S, \neg q)$ implies that $M(S, \sigma) \nvDash q$ and $M(S, \sigma) \nvDash \neg q$, which leaves $M(S, \sigma) = \{\bot\}$ as the only possibility.
 - 7b. If *S* is nondeterministic, then $\neg wp(S, q) \land \neg wp(S, \neg q)$ doesn't imply $M(S, \sigma) = \{\bot\}$.
 - For a nondeterministic program, if $M(S, \sigma) \neq q$ and $M(S, \sigma) \neq \neg q$, it's still possible for $M(S, \sigma)$ to contain non- \bot states. A simple counterexample is $M(S, \sigma) = \{\tau_1, \tau_2\}$ where $\tau_1 \models q$ and $\tau_2 \models \neg q$. Note it's possible that $\bot \notin M(S, \sigma)$, which definitely makes $M(S, \sigma)$ unequal to $\{\bot\}$.
 - 8a. If *S* is deterministic, then $\neg wp(S, q) \Rightarrow wlp(S, \neg q)$.
 - $M(S, \sigma) = \{\tau\}$ where $\tau = \bot$, $\tau \models q$, or $\tau \models \neg q$. If $\sigma \models \neg wp(S, q)$, then $\tau \models q$ fails, which leaves $\tau = \bot$ or $\tau \models \neg q$, in which case $M(S, \sigma) \bot \models \neg q$, so $\sigma \models wlp(S, \neg q)$.
 - 8b. If *S* is nondeterministic, then $\neg wp(S, q)$ doesn't imply $wlp(S, \neg q)$.
 - Since $M(S, \sigma) \neq q$ says only that not every value in $M(S, \sigma)$ satisfies q, so there can still be a $\tau_1 \in M(S, \sigma)$ with $\tau_1 \models q$, in which case $\sigma \not\models wlp(S, \neg q)$.

G. Disjunctive Postconditions Under Nondeterminism are Different

- For deterministic and nondeterministic both, the *wp/wlp* of a conjunction is the same as the conjunction of the *wp/wlp* 's.
 - $wp(S, q_1) \land wp(S, q_2) \Leftrightarrow wp(S, q_1 \land q_2).$

- $wlp(S, q_1) \land wlp(S, q_2) \Leftrightarrow wlp(S, q_1 \land q_2).$
- Also, the disjunction of the *wp/wlp* 's is sufficient to imply the *wp/wlp* of the disjunction:
 - $wp(S,q_1) \lor wp(S,q_2) \Rightarrow wp(S,q_1 \lor q_2).$
 - $wlp(S, q_1) \lor wlp(S, q_2) \Rightarrow wlp(S, q_1 \lor q_2).$
- Necessity of the *wp/wlp* of the disjunction holds for deterministic programs:
 - $wp(S, q_1 \lor q_2) \Rightarrow wp(S, q_1) \lor wp(S, q_2).$
 - $wlp(S, q_1 \lor q_2) \Rightarrow wlp(S, q_1) \lor wlp(S, q_2).$
- But for nondeterministic programs, $wp(S, q_1 \lor q_2) \Rightarrow wp(S, q_1) \lor wp(S, q_2)$ can be invalid. The standard example for this property is the coin-flip program we've seen before.
- **Example 11**: Let $flip \equiv if T \rightarrow x := 0 \square T \rightarrow x := 1$ fi.
 - Let *heads* = x = 0 as and *tails* = x = 1, then M(*flip*, Ø) = { { x = 0 }, { x = 1 } }. Though { x = 0 } and { x = 1 } both satisfy *heads* ∨ *tails*, neither of them satisfies *heads* or *tails*. So wp(*flip*, *heads* ∨ *tails*) = T but wp(*flip*, *heads*) = wp(*flip*, *tails*) = F.
- Let's look at the situation in terms of sets of states. Assume $\perp \notin M(S, \sigma)$ and let $M(S, \sigma) = \Sigma_1 \cup \Sigma_2$ where $\Sigma_1 \models q_1$ and $\Sigma_2 \models q_2$. We have that $\Sigma_1 \cup \Sigma_2 \models q_1 \lor q_2$, but if neither Σ_1 nor Σ_2 are \emptyset , then $\Sigma_1 \cup \Sigma_2$ includes at least one element satisfies q_1 and one that satisfies q_2 , so $\Sigma_1 \cup \Sigma_2$ satisfies neither q_1 nor q_2 .