Weakest precondition I

Farzaneh Derakhshan
based on material by Stefan Muller and Jim Sasaki

CS 536: Science of Programming, Fall 2023
Lecture 11

We learned in Lectures 9 and 10 that a program can be annotated with different conditions and thus be a part of multiple Hoare triples. In the next couple of lectures, we will explore annotations that are considered "better" than others.

1 Weakest precondition

Let's start with an example. Consider the program S to be $x := y + 1$. We want to show that the postcondition $x > 0$ holds for this program. We need to figure out the preconditions $\mathbf{P}$ that make the following total correctness triple valid:

$$\vDash [\ \mathbf{P}\ ]\ S\ [\ x > 0\ ]$$

In other words, we need to answer the question: *what precondition on the initial state ensures that the program will terminate in a state where x has a positive value?* It turns out there are many such preconditions, ranging from the trivial one (F) to $y \geq 100$ to $y \geq 0$. Although all of the conditions are technically correct, they are not equally useful. For instance, the conditions F and $y \geq 100$ are more restrictive about the initial state; they limit the range of values of y for which the postcondition is guaranteed. Generally, we want to find the precondition that is least restrictive and ensures the postcondition for the largest range of inputs. The least restrictive precondition that we can choose is indeed $y \geq 0$ – the least restriction that we can put on the initial state to ensure that the post-condition holds. $y \geq 0$ is the *weakest precondition*: the least restrictive precondition needed to establish the postcondition. The terms "weak" and "strong" refer to how restrictive an assertion is. The weaker an assertion is, the less restrictive it is; the stronger it is, the more restrictive.

We write $P = wp(S, Q)$ to indicate that P is the weakest precondition for statement S and postcondition Q. In our example, $y \geq 0$ is the weakest precondition. The precondition $y \geq 100$ is not as weak because it allows only a subset of the values accepted by $y \geq 0$. The precondition F is the strongest of these three formulas because it allows no value to be considered.

The strength of formulas coincides with the logical consequence: if $P' \Rightarrow P$, then P is weaker than P' (or equivalently, P' is stronger than P). Figure 1 illustrates the concept of the weakest precondition using the logical consequence. Based on the figure, we can conclude that $\vDash [\ wp(S, Q)\ ]\ S\ [\ Q\ ]$ and also if $\vDash [\ P\ ]\ S\ [\ Q\ ]$, then $P \Rightarrow wp(S, Q)$.

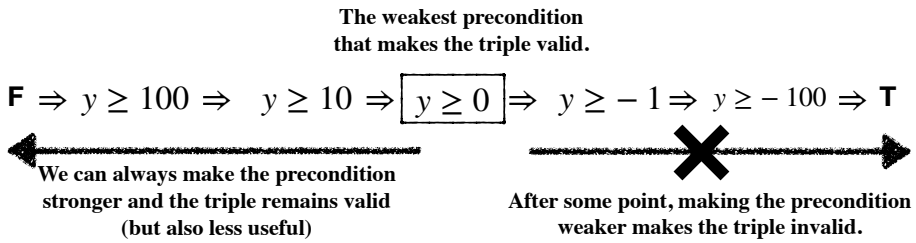

Figure 1: Weakest precondition and logical consequence

More formally for a program S and postcondition Q we define $wp(S, Q) = P$ as:

1. For any state $\sigma \vDash P$, we have $M(S, \sigma) = \sigma'$ and $\sigma' \vDash Q$.
2. For any state $\sigma \nvDash P$, either
   (a) $M(S, \sigma) = \sigma'$ such that $\sigma' \nvDash Q$, or
   (b) $M(S, \sigma) = \perp$, or ($\langle S, \sigma \rangle$ results in an error)
   (c) $M(S, \sigma) = \{\}$. ($\langle S, \sigma \rangle$ diverges)

Example 1. Consider the program $y := x * x$, and the postcondition $x \geq 0 \wedge y \geq 4$.We have

$$wp(y := x * x, x \geq 0 \wedge y \geq 4) = x \geq 2.$$

Why? With the definition above we need to show the following two conditions:

1. For any state $\sigma \vDash x \geq 2$, we have $M(y := x * x, \sigma) = \sigma[y \mapsto \sigma(x) * \sigma(x)]$, and we can observe that $\sigma[y \mapsto \sigma(x) * \sigma(x)] \vDash x \geq 0 \wedge y \geq 4$, since $\sigma(x) \geq 2$.
2. For any state $\sigma \nvDash x \geq 2$, we have $M(y := x * x, \sigma) = \sigma[y \mapsto \sigma(x) * \sigma(x)]$. But if $\sigma(x) < 0$, then $\sigma[y \mapsto \sigma(x) * \sigma(x)] \nvDash x \geq 0$, and if $0 \leq \sigma(x) < 2$, then $\sigma[y \mapsto \sigma(x) * \sigma(x)] \nvDash y \geq 4$. In both of these cases, we have $\sigma[y \mapsto \sigma(x) * \sigma(x)] \nvDash x \geq 0 \wedge y \geq 4$

The reasoning we used in Example 1 helps us establish whether a given precondition is the weakest precondition. However, this reasoning doesn't help us to create the weakest possible precondition. In the next lecture, we will introduce the rules that will enable us to generate the weakest precondition.

The weakest precondition is unique upto logical equivalence. Let's put $P_1 = wp(S, Q)$ and $P_2 = wp(S, Q)$. We know that the weakest precondition makes the total correctness triple valid, i.e., $\vDash [\ P_1\ ]\ S\ [\ Q\ ]$ and $\vDash [\ P_2\ ]\ S\ [\ Q\ ]$. Recall that if $\vDash [\ P\ ]\ S\ [\ Q\ ]$, then $P \Rightarrow wp(S, Q)$. This gives us $P_1 \Rightarrow wp(S, Q)$ and $P_2 \Rightarrow wp(S, Q)$, which we can rewrite as $P_1 \Rightarrow P_2$ and $P_2 \Rightarrow P_1$.

2 Weakest liberal precondition

The weakest liberal precondition is the weakest precondition's counterpart for partial correctness. It is the weakest precondition **P** that makes the following partial correctness triple valid:

$$\vDash \{\mathbf{P}\}\ S\ \{Q\}$$

In other words, $wlp(S, Q)$ is called the weakest liberal precondition on the initial state such that S starting from the initial state either terminates in a final state satisfying the postcondition Q, diverges or returns a run-time error.

More formally for a program S and postcondition Q we define $wlp(S, Q) = P$ as

1. For any state $\sigma \vDash P$ either
   (a) $M(S, \sigma) = \sigma'$ and $\sigma' \vDash Q$, or
   (b) $M(S, \sigma) = \perp$, or
   (c) $M(S, \sigma) = \{\}$
2. For any state $\sigma \nvDash P$, we have $M(S, \sigma) = \sigma'$ and $\sigma' \nvDash Q$.

Exercise 1. Using the definition above, show that $wlp(y := x * x, x \geq 0 \wedge y \geq 4) = x \geq 2$. How is your reasoning different from Example 1?

If the program S always terminates successfully, i.e., it does not diverge or result in a run-time error, then we have $wp(S, Q) = wlp(S, Q)$. In fact, in this case, the definitions of the weakest and weakest liberal preconditions conincide as $wlp(S, Q) = P$ iff:

1. For any state $\sigma \vDash P$ we have $M(S, \sigma) = \sigma'$ and $\sigma' \vDash Q$, and
2. For any state $\sigma \nvDash P$, we have $M(S, \sigma) = \sigma'$ and $\sigma' \nvDash Q$.

Example 2. Put program S_1 to be if $y \leq x$ then $m := x$ else skip fi. We want to find the weakest precondition for S_1 and the postcondition $m = \max(x, y)$, i.e., we want to figure out what restrictions on the initial state are necessary to make sure that S_1 calculates the maximum of x and y. Our conjecture is:

$$wp(S_1, m = \max(x, y)) = y > x \to m = y.$$

1. Consider the state σ such that $\sigma \vDash y > x \to m = y$. The program terminates, so for some σ', we have $M(S_1, \sigma) = \sigma'$. We proceed by cases. First, assume $\sigma \vDash y \leq x$, then we know that $\sigma'(m) = \sigma'(x) = \max(\sigma'(x), \sigma'(y))$. Now assume, $\sigma \vDash y > x$, then by $\sigma \vDash y > x \to m = y$, we know that $\sigma'(m) = \sigma(m) = \sigma(y) = \max(\sigma(x), \sigma(y)) = \max(\sigma'(x), \sigma'(y))$. In both cases, we get $\sigma' \vDash m = \max(x, y)$.
2. Consider the state $\sigma \nvDash y > x \to m = y$. From $\sigma \nvDash y > x \to m = y$, we get $\sigma \vDash \neg(\neg y > x \vee m = y)$, which by the deMorgan's law and double negation rule is equivalent to $\sigma \vDash y > x \wedge m \neq y$. The program terminates, so for some σ', we have $M(S_1, \sigma) = \sigma'$. By the $y > x$ condition, we know that the program takes the else branch, so $\sigma(m) = \sigma'(m)$, and $\sigma(x) = \sigma'(x)$, and $\sigma(y) = \sigma'(y)$. Moreover, we know that $\sigma' \vDash y = \max(x, y)$. This along with the condition $m \neq y$ gives us $\sigma' \vDash m \neq \max(x, y)$, or equivalently $\sigma' \nvDash m = \max(x, y)$.

Exercise 2. Find $wlp(S_1, y > x \to m = y)$.

Example 3. Consider program $S_2 ::=$ while $x \neq 0$ do $x := x - 1$ od. It will always ends in the final state satisfying $x = 0$ unless it doesn't terminate. For the program to terminate, the initial state must satisfy $x \geq 0$. we have

$$wp(S_2, x = 0) = x \geq 0,$$

On the other hand, the weakest liberal precondition, which allows the program to diverge, does not need to put any restriction on the initial state for the same postcondition:

$$wlp(S_2, x = 0) = T.$$

Note 1. In Example 3, since the final state always satisfies the postcondition $x = 0$, the weakest precondition $wp(S_2, x = 0)$ provides the condition under which S terminates. We can generalize this observation for any program S, by using the postcondition T (which always holds in the final state as well): **wp(S, T) is the condition under which S terminates.**

Note 2. In Example 3 the weakest precondition and weakest liberal precondition are different, since the program might be nonterminating for some inputs. However, we can observe that the weakest precondition ($x \geq 0$) implies the weakest liberal precondition ($\top$); we have $x \geq 0 \Rightarrow T$. We can generalize this observation to all programs S and postconditions Q as:

$$wp(S, Q) \Rightarrow wlp(S, Q).$$

Note 3. If $\sigma \vDash wlp(S, Q)$ but $\sigma \nvDash wp(S, Q)$, then program S starting from the initial state σ, either diverges or results in a run-time error, i.e., $M(S, \sigma) = \{\}$ or $M(S, \sigma) = \perp$. In Example 3, consider the initial state $\{x = -1\}$, the program S_2 diverges when starting from this initial state, and we have $\{x = -1\} \nvDash wp(S_2, x = 0)$, but $\{x = -1\} \vDash wlp(S_2, x = 0)$.

Exercise 3. Provide a proof for the observation in Note 3.