Weakest Preconditions

Part 2: Calculating wp, wlp; Domain Predicates

CS 536: Science of Programming, Spring 2021

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

- How to add error domain predicates to the wlp of a loop-free program to obtain its wp.

C. Calculating wlp for Loop-Free Programs

- It’s easy to calculate the wp and wlp of a loop-free/error-free program S especially since for such programs, the wp and wlp are identical.

- The following algorithm takes S and q and syntactically calculates a particular predicate for wlp(S, q), which is why it’s described using \( \text{wlp}(S, q) \equiv \ldots \) instead of \( \text{wp}(S, q) \equiv \ldots \).

  - \( \text{wlp}(\text{skip}, q) = q \)
  - \( \text{wlp}(v := e, Q(v)) = Q(e) \) where \( Q \) is a predicate function over one variable.
    - The operation that takes us from \( Q(v) \) to \( Q(e) \) is called syntactic substitution; we’ll look at it in more detail soon, but in the simple case, we simply inspect the definition of \( Q \), search its text for occurrences of the variable \( v \) and replace them with copies of \( e \).

  - \( \text{wlp}(S_1 ; S_2, q) = \text{wlp}(S_1, \text{wlp}(S_2, q)) \)
  - \( \text{wlp}(\text{if } B \text{ then } S_1 \text{ else } S_2 fi, q) = (B \rightarrow w_1) \land (\neg B \rightarrow w_2) \) where \( w_1 = \text{wlp}(S_1, q) \) and \( w_2 = \text{wlp}(S_2, q) \).
    - Since it’s equivalent, you can also use \( (B \land w_1) \lor (\neg B \land w_2) \).
  - \( \text{wlp}(B_1 \rightarrow S_1 \bigcirc B_2 \rightarrow S_2 fi, q) = (B_1 \rightarrow w_1) \land (B_2 \rightarrow w_2) \) where \( w_1 = \text{wlp}(S_1, q) \) and \( w_2 = \text{wlp}(S_2, q) \).
    - For the nondeterministic if, you must use \( (B_1 \rightarrow w_1) \land (B_2 \rightarrow w_2) \), not \( (B_1 \land w_1) \lor (B_2 \land w_2) \), because they’re not equivalent (unlike the deterministic if statement).
    - When \( B_1 \) and \( B_2 \) are both true, either \( S_1 \) or \( S_2 \) can run, so we need \( B_1 \land B_2 \rightarrow w_1 \land w_2 \), and this is implied by \( (B_1 \rightarrow w_1) \land (B_2 \rightarrow w_2) \).
    - Using \( (B_1 \land w_1) \lor (B_2 \land w_2) \) fails because it allows for the possibility that \( B_1 \) and \( B_2 \) are both true but only one of \( w_1 \) and \( w_2 \) is true. This isn’t a problem when \( B_2 \leftrightarrow \neg B_1 \), which is why we can use \( (B \land w_1) \lor (\neg B \land w_2) \) with deterministic if statements.
D. Some Examples of Calculating wp/wlp:

- The programs in these examples never end in “state” \( \bot \), so the \( wp \) and \( wlp \) are equivalent.

  - **Example 2:** \( wlp(x := x+1, x \geq 0) = x+1 \geq 0 \)
  
  - **Example 3:** \( wlp(y := y+x; x := x+1, x \geq 0) \)
    
    \[ wlp(y := y+x, wlp(x := x+1, x \geq 0)) \]
    
    \[ wlp(y := y+x, x+1 \geq 0) = x+1 \geq 0 \]
  
  - **Example 4:** \( wlp(y := y+x; x := x+1, x \geq y) \)
    
    \[ wlp(y := y+x, wlp(x := x+1, x \geq y)) \]
    
    \[ wlp(y := y+x, x+1 \geq y) \]
    
    \[ x+1 \geq y+x \]
  
  - **Example 5:** (Swap the two assignments in Example 4)
    
    \[ wlp(x := x+1; y := y+x, x \geq y) \]
    
    \[ wlp(x := x+1, wlp(y := y+x, x \geq y)) \]
    
    \[ wlp(x := x+1, x \geq y+x) \]
    
    \[ x+1 \geq y+x+1 [\Leftrightarrow y \leq 0 \text{ if you want to logically simplify}] \]
  
  - **Example 6:** \( wlp(\text{if } y \geq 0 \text{ then } x := y \text{ fi}, x \geq 0) \)
    
    \[ wlp(\text{if } y \geq 0 \text{ then } x := y \text{ else skip } \text{ fi}, x \geq 0) \]
    
    \[ (y \geq 0 \rightarrow wlp(x := y, x \geq 0)) \land (y < 0 \rightarrow wlp(\text{skip}, x \geq 0)) \]
    
    \[ (y \geq 0 \rightarrow y \geq 0) \land (y < 0 \rightarrow x \geq 0) \]
  
  - If we want to simplify logically, we can continue with
    
    \[ y \geq 0 \lor (y < 0 \land x \geq 0) \]
    
    \[ (y \geq 0 \lor y < 0) \land (y \geq 0 \lor x \geq 0) \]
    
    \[ y \geq 0 \lor x \geq 0 \text{ (which is also } y < 0 \rightarrow x \geq 0, \text{ if you prefer)} \]

E. Domain Predicates for Avoiding Runtime Errors in Expressions

- To avoid runtime failure of \( \sigma(e) \), we'll take the context in which we're evaluating \( e \) and augment it with a predicate that guarantee non-failure of \( \sigma(e) \). For example, for \( \{P(e)\} v := e \{P(v)\} \), we'll augment the precondition to guarantee that evaluation of \( e \) won't fail.
  
  - For each expression \( e \), we will define a **domain predicate** \( D(e) \) such that \( \sigma = D(e) \) implies \( \sigma(e) \neq \bot \).
    
    - This predicate has to be defined recursively, since we need to handle complex expressions like \( D(b[b[k]]) = 0 \leq k < \text{size}(b) \land 0 \leq b[k] < \text{size}(b) \).
    
    - As with \( wp \) and \( sp \), the domain predicate for an expression is unique only up to logical equivalence. For example, \( D(x/y + u/v) = y \neq 0 \land v \neq 0 \equiv v \neq 0 \land y \neq 0 \).

  - **Definition** (Domain predicate \( D(e) \) for expression \( e \)) We must define \( D \) for each kind of expression that can cause a runtime error:
• $D(c) = D(v) = T$ if where $c$ is a constant and $v$ is a variable.
  • Evaluation of a variable or constant doesn't cause failure.
• $D(b[e]) = D(e) \land 0 \leq e < \text{size}(b)$
• $D(e_1 / e_2) = D(e_1) \land D(e_2) \land e_2 \neq 0$
• $D(\text{sqrt}(e)) = D(e) \land e \geq 0$
  • And so on, depending on the datatypes and operations being used.
• The various operations (+, -, etc.) and relations (≤, =, etc.) don't cause errors but we still have to check their subexpressions:
• $D(e_1 \text{ op } e_2) = D(e_1) \land D(e_2)$, except for op = / or %
• $D(\text{op } e) = D(e)$, unless you add an operator that can cause runtime failure.
• $D(\text{if } B \text{ then } e_1 \text{ else } e_2 \text{ fi}) = D(B) \land (B \rightarrow D(e_1)) \land (\neg B \rightarrow D(e_2))$
  • (For a conditional expression, we only need safety of the one branch we execute.)

**Example 7:** $D(b[b[k]]) = D(b[k]) \land 0 \leq b[k] < \text{size}(b)$
  $\equiv D(k) \land 0 \leq k < \text{size}(b) \land 0 \leq b[k] < \text{size}(b)$

**Example 8:** $D((-b + \text{sqrt}(b^2 - 4*a^2*c))/(2*a))$
  $\equiv D(e) \land D(2*a) \land 2*a \neq 0$  
  where $e = -b + \text{sqrt}(b^2 - 4*a^2*c)$
  $\equiv D(-b) \land D(\text{sqrt}(b^2 - 4*a^2*c)) \land D(2*a) \land 2*a \neq 0$
  $\equiv D(\text{sqrt}(b^2 - 4*a^2*c)) \land 2*a \neq 0$  
  since $D(-b) = D(2*a) = T$
  $\equiv D(b^2 - 4*a^2*c) \land (b^2 - 4*a^2*c \geq 0) \land 2*a \neq 0$
  $\equiv b^2 - 4*a^2*c \geq 0 \land 2*a \neq 0$

**Example 9:** $D(\text{if } 0 \leq k < \text{size}(b) \text{ then } b[k] \text{ else } 0 \text{ fi})$
  $\equiv D(B) \land (B \rightarrow D(b[k])) \land (\neg B \rightarrow D(0))$  
  where $B = 0 \leq k < \text{size}(b)$
  $\equiv (B \rightarrow D(b[k])) \land (\neg B \rightarrow T)$  
  since $D(B)$ and $D(0) = T$
  $\equiv B \rightarrow D(b[k])$  
  since everything implies true
  $\equiv B \rightarrow D(k) \land 0 \leq k < \text{size}(b)$  
  expanding $D(b[k])$
  $\equiv B \rightarrow T \land B$  
  since $B = 0 \leq k < \text{size}(b)$
  $\equiv T$

**F. Domain Predicates for Avoiding Runtime Errors in Programs**

• Recall that we extended our notion of operational semantics to include $\langle S, \sigma \rangle \rightarrow^* \langle E, \perp \rangle$ to indicate that evaluation of $S$ causes a runtime failure.
• We can avoid runtime failure of statements by adding domain predicates to the preconditions of statements. Though for loops we can't in general calculate the $wlp/wp$, we can calculate the domain predicate for them.
• **Definition:** For statement $S$, the predicate $D(S)$ gives a sufficient condition to avoid runtime errors.
  • $D(\text{skip}) = T$
  • $D(v := e) = D(e)$

© James Sasaki, 2021
• \( D(b[e_1] := e_2) = D(b[e_1]) \land D(e_2) \)

• \( D(S_1 ; S_2) = D(S_1) \land \text{wp}(S_1, D(S_2)) \)

[Running \( S_1 \) when \( D(S_1) \) holds tells us \( S_1 \) won't cause an error. Running \( S_1 \) when \( \text{wp}(S_1, D(S_2)) \) holds tells us that \( S_1 \) will establish \( D(S_2) \), so running \( S_2 \) won't cause an error.]

• If \( \sigma \models D(S_1) \) then \( \bot_e \notin M(S_1, \sigma) \).

• If \( \sigma \models \text{wp}(S_1, D(S_2)) \), then \( M(S_1, \sigma) \models D(S_2) \).

• If \( M(S_1, \sigma) \models D(S_2) \), then \( \bot_e \notin M(S_2, M(S_1, \sigma)) \).

• \( D(\text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}, q) = D(B) \land (B \rightarrow D(S_1)) \land (\neg B \rightarrow D(S_2)) \)

• \( D(\text{if } B_1 \rightarrow S_1 \square B_2 \rightarrow S_2 \text{ fi}, q) = D(B_1 \lor B_2) \land (B_1 \lor B_2) \land (B_1 \rightarrow D(S_1)) \land (B_2 \rightarrow D(S_2)) \)

• Note we need \( (B_1 \lor B_2) \) to avoid failure of the nondeterministic \textit{if-fi} due to none of the guards holding.

• This definition extends easily to \textit{if-fi} with more than two guarded commands.

• \( D(\text{while } B \text{ do } S_1 \text{ od}) = D(B) \land (B \rightarrow D(S_1)) \)

• \( D(\text{do } B_1 \rightarrow S_1 \square B_2 \rightarrow S_2 \text{ od}) = D(B_1 \lor B_2) \land (B_1 \rightarrow D(S_1)) \land (B_2 \rightarrow D(S_2)) \)

• The domain predicate for nondeterministic \textit{do-od} is like that for \textit{if-fi} except that having none of the guards hold does not cause an error.

\[ \textbf{Calculating wp for loop-free programs} \]

• With the domain predicates, it's easy to extend \( \text{wp} \) for \( \text{wp} \) for loop-free programs because we don't have to argue for termination of a loop.

• \textbf{Definition:} \( \text{wp}(S, q) = D(S) \land D(w) \land w \), where \( w = \text{wp}(S, q) \).

• \( D(S) \) tells us that running \( S \) won't cause an error, \( D(w) \) tells us that \( w \) makes sense, and \( w \) tells us that running \( S \) will establish \( q \) (if \( S \) terminates).

• \textbf{Example 10:} If a program does a division, then the \( \text{wp} \) and \( \text{wp} \) can differ.

• Let \( p_2 = \text{wp}(x := y; z := v/x, z > x+2) = \text{wp}(x := y, p_1) \)

• Where \( p_1 = \text{wp}(z := v/x, z > x+2) = D(z := v/x) \land D(w) \land w \)

where \( w = \text{wp}(z := v/x, z > x+2) = v/x > x+2 \)

\( p_1 = D(z := v/x) \land D(v/x > x+2) \land v/x > x+2 = x \neq 0 \land x \neq 0 \land v/x > x+2 \iff x \neq 0 \land v/x > x+2 \)

• So \( p_2 = \text{wp}(x := y, p_1) = \text{wp}(x := y, x \neq 0 \land v/x > x+2) = \text{wp}(x := y, x \neq 0 \land v/x > x+2) \), since \( x := y \) causes no errors

\[ y = y \neq 0 \land v/y > y+2 \]

• \textbf{Example 11:} Let's calculate \( p_0 = \text{wp}(x := b[k], \text{sqrt}(x) \geq 1) \). When \( S = x := b[k] \) and \( q = \text{sqrt}(x) \geq 1 \), then

• \( p_0 = \text{wp}(S, q) = D(S) \land D(w) \land w \)

where \( w = \text{wp}(S, q) = \text{wp}(x := b[k], q = \text{sqrt}(x) \geq 1) = \text{sqrt}(b[k]) \geq 1 \)

• Breaking this down,
• $wlp(S, q) = wlp(x := b[k], \sqrt{x} \geq 1) \iff \sqrt{b[k]} \geq 1$, so

• $D(wlp(S, q)) = D(\sqrt{b[k]} \geq 1)) = D(b[k]) \wedge b[k] \geq 0 = 0 \leq k < \text{size}(b) \wedge b[k] \geq 0$.

• $D(S) = D(x := b[k]) = D(k) \wedge 0 \leq k < \text{size}(b) \iff 0 \leq k < \text{size}(b)$

• Combining, we get

$$wp(x := b[k], \sqrt{x} \geq 1)$$

$$= D(wp(x := b[k], \sqrt{x} \geq 1)) \wedge wp(x := b[k], \sqrt{x} \geq 1) \wedge D(x := b[k])$$

$$\iff (\sqrt{b[k]} \geq 1) \wedge (0 \leq k < \text{size}(b) \wedge b[k] \geq 0) \wedge (0 \leq k < \text{size}(b))$$

$$\iff 0 \leq k < \text{size}(b) \wedge b[k] \geq 0 \wedge \sqrt{b[k]} \geq 1$$

(which, if we decide to simplify numerically),

$$\iff 0 \leq k < \text{size}(b) \wedge b[k] \geq 1$$
Weakest Preconditions

Part 2: Calculating wp, wlp; Domain Predicates

CS 536: Science of Programming, Spring 2021

A. Why

• The weakest precondition and weakest liberal preconditions are the most general preconditions that a program needs in order to run correctly.

B. Objectives

At the end of this activity you should be able to

• Describe the relationship between \(wp(S, q_1 \lor q_2), wp(S, q_1), \) and \(wp(S, q_2)\) and how it differs for deterministic and nondeterministic programs.

• Be able to calculate the \(wlp\) of a simple loop-free program.

C. Problems

1. How are \(wp(S, q_1 \lor q_2)\) and \(wp(S, q_1)\) and \(wp(S, q_2)\), related if \(S\) is deterministic? If \(S\) is nondeterministic?

For Problems 2 – 4, just syntactically calculate the \(wlp\); don't also logically simplify the result.

2. Calculate the \(wlp\) in each of the following cases.

   a. \(wlp(k := k - s, n = 3 \land k = 4 \land s = -7)\).
   b. \(wlp(n := n^*(n-k), n = 3 \land k = 4 \land s = -7)\).
   c. \(wlp(n := n^*(n-k); k := k - s, n > k + s)\)

3. Let \(Q(k, s) = 0 \leq k \leq n \land s = sum(0,k)\) where \(sum(u, v)\) is the sum of \(u, u+1, \ldots, v\) (when \(u \leq v\)) or 0 (when \(u > v\)).

   a. Calculate \(wp(k := k+1; s := s+k, Q(k, s))\).
   b. Calculate \(wp(s := s+k+1; k := k+1, Q(k, s))\).
   c. Calculate \(wp(s := s+k; k := k+1, Q(k, s))\). (This one isn't compatible with \(s = sum(0, k)\).)

4. Calculate the \(wp\) below.

   a. \(wp(if B then x := x/2 fi; y := x, x = 5 \land y = z)\).
   b. \(wp(if x \geq 0 then x := x^2 else x := y fi; x := c^x, a \leq x < y)\)

For Problems 5 and 6, don't forget the domain predicates. You can logically simplify as you go.

5. Calculate \(p\) to be the \(wp\) in \(\{p\} x := y/b[k] \{x > 0\}\).

6. Calculate \(p_1\) and \(p_2\) to be the \(wp\)'s in \(\{p_1\} y := sqrt(b[k]) \{z < y\}\) and \(\{p_2\} k := x/k \{p_1\}\).
Solution to Practice 11 (Weakest Preconditions, pt. 2)

1. For deterministic \( S \), \( wp(S, q_1 \lor q_2) \leftrightarrow wp(S, q_1) \lor wp(S, q_2) \).
   For nondeterministic \( S \), we have \( \Rightarrow \) but not \( \Leftarrow \).

2. (Calculate \( wlp \))
   a. \( wlp(k := k - s, n = 3 \land k = 4 \land s = -7) = n = 3 \land k-s = 4 \land s = -7 \)
   b. \( wlp(n := n*(n-k), n = 3 \land k = 4 \land s = -7) = n*(n-k) = 3 \land k = 4 \land s = -7 \)
   c. \( wlp(n := n*(n-k); k := k-s, n > k+s) = wlp(n := n*(n-k), wlp(k := k-s, n > k+s)) \)
      = \( wlp(n := n*(n-k), n > k+s) \)
      = \( n*(n-k) > k-s+s \)

3. (\( wp \) involving sums) We have \( Q(k, s) = 0 \leq k \leq n \land s = sum(0, k) \).
   a. \( wp(k := k+1; s := s+k, Q(k, s)) = wp(k := k+1, wp(s := s+k, Q(k, s))) = wp(k := k+1, Q(k, s+k)) = wp(k := k+1, 0 \leq k \leq n \land s+k = sum(0, k)) = 0 \leq k+1 \leq n \land s+k+1 = sum(0, k+1) \)
   b. \( wp(s := s+k+1; k := k+1, Q(k, s)) = wp(s := s+k+1, wp(k := k+1, Q(k, s))) = wp(s := s+k+1, Q(k+1, s)) = wp(s := s+k+1, 0 \leq k+1 \leq n \land s = sum(0, k+1)) = 0 \leq k+1 \leq n \land s+k+1 = sum(0, k+1) \)
   c. \( wp(s := s+k; k := k+1, Q(k, s)) = wp(s := s+k, wp(k := k+1, Q(k, s))) = wp(s := s+k, Q(k+1, s)) = wp(s := s+k, 0 \leq k+1 \leq n \land s = sum(0, k+1)) = 0 \leq k+1 \leq n \land s+k = sum(0, k+1) \) [which isn't compatible with \( s = sum(0, k) \)]

4. (\( wp \) of if-then)
   a. \( wp(if B then x := x/2 fi; y := x, x = 5 \land y = z) = wp(if B then x := x/2 fi; wp(y := x, x = 5 \land y = z)) = wp(if B then x := x/2 fi, x = 5 \land x = z) = (B \rightarrow wp(x := x/2, x = 5 \land x = z)) \land (\neg B \rightarrow wp(skip, x = 5 \land x = z)) = (B \rightarrow x/2 = 5 \land x/2 = z) \land (\neg B \rightarrow x = 5 \land x = z) \)
b. \( \text{wp}(\text{if } x \geq 0 \text{ then } x := x \times 2 \text{ else } x := y \text{ fi; } x := c \times x, \ a \leq x < y) \)
\[ = \text{wp}(S, \text{wp}(x := c \times x, \ a \leq x < y)) \text{ where } S \text{ is the if statement} \]
\[ = \text{wp}(S, \ a \leq c \times x < y) \]
\[ = \text{wp}(\text{if } x \geq 0 \text{ then } x := x \times 2 \text{ else } x := y \text{ fi, } a \leq c \times x < y) \]
\[ = (x \geq 0 \rightarrow \text{wp}(x := x \times 2, \ a \leq c \times x < y)) \land (x < 0 \rightarrow \text{wp}(x := y, \ a \leq c \times x < y)) \]
\[ = (x \geq 0 \rightarrow a \leq c \times (x \times 2) < y) \land (x < 0 \rightarrow a \leq c \times y < y) \]

5. For \( \{p\} x := y/b[k] \ \{x > 0\}, \)
\[ \text{let } p \leftrightarrow \text{wp}(x := y/b[k], \ x > 0) \]
\[ = \text{wp}(x := y/b[k], \ x > 0) \land D(x := y/b[k]) \]
\[ = y/b[k] > 0 \land b[k] \neq 0 \land D(b[k]) \]
\[ = y/b[k] > 0 \land b[k] \neq 0 \land 0 \leq k < \text{size}(b) \]

6. For \( \{p_1\} y := \sqrt{b[k]} \ \{z < y\}, \)
\[ \text{let } p_1 \leftrightarrow \text{wp}(y := \sqrt{b[k]), \ z < y) \]
\[ = \text{wp}(y := \sqrt{b[k]}, \ z < y) \land D(y := \sqrt{b[k])} \]
\[ = z < \sqrt{b[k]} \land b[k] \geq 0 \land D(b[k]) \]
\[ = z < \sqrt{b[k]} \land b[k] \geq 0 \land 0 \leq k < \text{size}(b) \]

For \( \{p_2\} k := x/k; \ \{p_1\}, \)
\[ \text{let } p_2 \leftrightarrow \text{wp}(k := x/k, \ p_1) \]
\[ = \text{wp}(k := x/k, \ p_1) \land D(k := x/k) \]
\[ = p_1[x/k / k] \land k \neq 0 \]
\[ = (z < \sqrt{b[k]} \land b[k] \geq 0 \land 0 \leq D(b[k) < \text{size}(b)) \land [x/k / k] \land k \neq 0 \]
\[ = z < \sqrt{b[x/k]} \land b[x/k] \geq 0 \land 0 \leq D(b[x/k) < \text{size}(b)) \land k \neq 0 \]