Weakest Preconditions, pt. 2

CS 536: Science of Programming, Fall 2018

A. Why

- A weakest precondition is the most general requirement that a program must meet to produce the right result.

B. Objectives

At the end of today you should understand

- How the domain predicate of an expression ensures it doesn’t cause runtime failure.
- How to calculate the $wp$ and $wlp$ of loop-free programs, with the help of domain predicates.

C. Review: Calculating $wlp$ for Loop-Free Programs

- $wlp(skip, q) \equiv q$
- $wlp(v := e, Q(v)) \equiv Q(e)$ where $Q$ is a predicate function over one variable
- $wlp(S_1 ; S_2, q) \equiv wlp(S_1, wlp(S_2, q))$
- $wlp(if B then S_1 else S_2 fi, q) \equiv (B \rightarrow w_1) \land (\neg B \rightarrow w_2) \lor (B \land w_1) \lor (\neg B \land w_2)$ where $w_1 \equiv wlp(S_1, q)$ and $w_2 \equiv wlp(S_2, q)$.
- $wlp(if B_1 \rightarrow S_1 \land B_2 \rightarrow S_2 fi, q) \equiv (B_1 \rightarrow wlp(S_1, q)) \land (B_2 \rightarrow wlp(S_2, q))$.

D. Some Examples of Calculating $wp/wlp$:

- The programs in these examples don’t end in “state” $\perp$, so the $wp$ and $wlp$ are equivalent.
- Example 1: $wp(x := x+1, x \geq 0) \equiv x+1 \geq 0$
- Example 2: $wp(y := y+x; x := x+1, x \geq 0)$
  $\quad \equiv wp(y := y+x, wp(x := x+1, x \geq 0))$
  $\quad \equiv wp(y := y+x, x+1 \geq 0) \equiv x+1 \geq 0$
- Example 3: $wp(y := y+x; x := x+1, x \geq y)$
  $\quad \equiv wp(y := y+x, wp(x := x+1, x \geq y))$
  $\quad \equiv wp(y := y+x, x+1 \geq y)$
  $\quad \equiv x+1 \geq y+x$
  - If we were asked only to calculate the $wp$, we’d stop here. If we also wanted to logically simplify the $wp$ then $x+1 \geq y+x \iff y \leq 1$.
- Example 4: (Swap the two assignments in Example 14)
  $wp(x := x+1; y := y+x, x \geq y)$
  $\quad \equiv wp(x := x+1, wp(y := y+x, x \geq y))$
  $\quad \equiv wp(x := x+1, x \geq y+x)$
  $\quad \equiv x+1 \geq y+x+1 [\iff y \leq 0 \text{ if you want to logically simplify}]$
- Example 5: $wp(if y \geq 0 then x := y fi, x \geq 0)$
  $\quad \equiv wp(if y \geq 0 then x := y else skip fi, x \geq 0)$
\[(y \geq 0 \land wp(x := y, x \geq 0)) \lor (y < 0 \land wp(\text{skip}, x \geq 0))\]
\[(y \geq 0 \land y \geq 0) \lor (y < 0 \land x \geq 0).\]

If we want to simplify logically, we can continue with

\[
\begin{align*}
\equiv & \ y \geq 0 \lor (y < 0 \land x \geq 0) \\
\equiv & \ (y \geq 0 \lor y < 0) \land (y \geq 0 \lor x \geq 0) \\
\equiv & \ y \geq 0 \lor x \geq 0 \text{ (which is also } \equiv y < 0 \rightarrow x \geq 0, \text{ if you prefer)}
\end{align*}
\]

E. **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 guarantees 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 \(\sigma \models D(e)\) implies \(\sigma(e) \neq \bot\).
  - This predicate has to be defined recursively, since we need to handle complex expressions like \(D(b[b[i]]) \equiv 0 \leq i < \text{size}(b) \land 0 \leq b[i] < \text{size}(b)\).
  - As with \(\text{wp}\) and \(\text{sp}\), the domain predicate for an expression is unique only up to logical equivalence. For example, \(D(x/y + u/v) \equiv y \neq 0 \land v \neq 0 \iff v \neq 0 \land y \neq 0\).

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

**Example 6:** \(D(b[b[i]]) \equiv D(b[i]) \land 0 \leq b[i] < \text{size}(b)\)

\[
\begin{align*}
\equiv & \ D(i) \land 0 \leq i < \text{size}(b) \land 0 \leq b[i] < \text{size}(b) \\
\equiv & \ 0 \leq i < \text{size}(b) \land 0 \leq b[i] < \text{size}(b)
\end{align*}
\]

**Example 7:** \(D((-b + \text{sqrt}(b*b - 4*a*c))/(2*a))\)

\[
\begin{align*}
\equiv & \ D(e) \land D(2*a) \land 2*a \neq 0 \\
\equiv & \ D(-b) \land D(\text{sqrt}(b*b - 4*a*c)) \land D(2*a) \land 2*a \neq 0 \\
\equiv & \ D(\text{sqrt}(b*b - 4*a*c)) \land 2*a \neq 0 \quad \text{Since } D(-b) \equiv D(2*a) \equiv T
\end{align*}
\]
\[
\equiv D(b \cdot b - 4 \cdot a \cdot c) \land (b \cdot b - 4 \cdot a \cdot c \geq 0) \land 2 \cdot a \neq 0
\]
\[
\Leftrightarrow b \cdot b - 4 \cdot a \cdot c \geq 0 \land 2 \cdot a \neq 0
\]

- **Example 8:** \(D(\text{if } 0 \leq i < \text{size}(b) \text{ then } b[i] \text{ else } 0 \text{ fi})\)
  \[
  \equiv D(B) \land (B \rightarrow D(b[i])) \land (\neg B \rightarrow D(0)) \quad \text{where } B \equiv 0 \leq i < \text{size}(b) \\
  \equiv (B \rightarrow D(b[i])) \land (\neg B \rightarrow T) \quad \text{since } D(B) \text{ and } D(0) \equiv T \\
  \Leftrightarrow B \rightarrow D(b[i]) \quad \text{since everything implies true} \\
  \equiv B \rightarrow D(i) \land 0 \leq i < \text{size}(b) \quad \text{expanding } D(b[i]) \\
  \Leftrightarrow B \rightarrow B \quad \text{since } B \equiv 0 \leq i < \text{size}(b) \\
  \Leftrightarrow T
  \]

**F. Avoiding Runtime Errors in Programs**

- Recall that we extended our notion of operational semantics to include \(\langle S, \sigma \rangle \rightarrow^* \langle E, \bot \rangle\) 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. Unlike \(wlp\), we can even do this for loops.
- **Definition:** For statement \(S\), the predicate \(D(S)\) gives a sufficient condition to avoid runtime errors
  - \(D(\text{skip}) \equiv T\)
  - \(D(v := e) \equiv D(e)\)
  - \(D(b[e_1] := e_2) \equiv D(b[e_1]) \land D(e_2)\)
  - \(D(S_1; S_2, q) \equiv D(S_1) \land D(S_2)\)
  - \(D(\text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}, q) \equiv D(B) \land (B \rightarrow D(S_1)) \land (\neg B \rightarrow D(S_2))\)
  - \(D(\text{if } B_1 \rightarrow S_1 \land B_2 \rightarrow S_2 \text{ fi}, q) \equiv D(B_1 \lor B_2) \land (B_1 \lor B_2) \land (B_1 \rightarrow D(S_1)) \land (B_2 \rightarrow D(S_2))\)
    - The condition \((B_1 \lor B_2)\) avoids failure of the nondeterministic \textbf{if-fi} due to none of the guards holding. This definition extends easily to \textbf{if-fi} with more than two guarded commands.
  - \(D(\text{while } B \text{ do } S_1 \text{ od}) \equiv D(B) \land (B \rightarrow D(S_1))\)
  - \(D(\text{do } B_1 \rightarrow S_1 \land B_2 \rightarrow S_2 \text{ od}) \equiv D(B_1 \lor B_2) \land (B_1 \lor B_2) \land (B_1 \rightarrow D(S_1)) \land (B_2 \rightarrow D(S_2))\)
    - The domain predicate for nondeterministic \textbf{do-od} is like that for \textbf{if-fi} except that having none of the guards hold does not cause an error.
- With the domain predicates, it's easy to extend \(wlp\) for \(wp\) for loop-free programs, since we would also want to show termination of a loop.
- **Definition:** If \(S\) is not a loop, then \(wp(S, q) \Leftrightarrow wlp(S, D(S) \land q)\)
- **Example 9:** If a program does a division, then the \(wp\) and \(wlp\) can differ:
  - \(D(x := y; z := v/x)\)
    \[
    \equiv D(x := y) \land D(z := v/x)
    \]
    \[
    \Leftrightarrow D(v/x) \quad \text{since } D(x := y) \equiv D(y) \equiv T
    \]
    \[
    \Leftrightarrow x \neq 0 \quad \text{Technically, } D(v/x) \equiv D(v) \land D(x) \land x \neq 0
    \]
• $wp(x := y; z := v/x, z > x+2)$
  $\equiv wp(x := y, wp(z := v/x, z > x+2 \land D(x := y; z := v/x)))$

$\iff wp(x := y, wp(z := v/x, z > x+2 \land x \neq 0))$ // Substituting from above

$\equiv wp(x := y, v/x > x+2 \land x \neq 0)$

$\equiv v/y > y+2 \land y \neq 0$

• Will add example 10 back later [10/1]


Weakest Preconditions, pt. 2
CS 536: Science of Programming

A. Why

- The weakest precondition is the most general precondition that a program needs in order to run correctly.

B. Objectives

At the end of this activity you should

- Be able to calculate the wp of a simple loop-free program.

C. Problems

1. Let \( Q(i, s) \equiv 0 \leq i \leq n \land s = \text{sum}(0, i) \) where \( \text{sum}(u, v) \) is the sum of \( u \), \( u+1 \), ..., \( v \) (when \( u \leq v \)) or 0 (when \( u > v \)).
   a. Calculate \( wp(i := i+1; s := s+i, Q(i, s)) \).
   b. Calculate \( wp(s := s+i+1; i := i+1, Q(i, s)) \).
   c. Calculate \( wp(s := s+i; i := i+1, Q(i, s)) \). (This one isn’t compatible with \( s = \text{sum}(0, i) \).)

2. Calculate the \( wp \) below. (Again, just calculate the syntactically \( wp \) without logically simplifying the result.)
   a. \( wp(\text{if } B \text{ then } x := x/2 \text{ fi}; y := x, x = 5 \land y = z) \).
   b. \( wp(\text{if } x \geq 0 \text{ then } x := x*2 \text{ else } x := y \text{ fi}; x := c*x, a \leq x < y) \)

9. Let \( w \equiv wp(S, q) \). The definition of \( wp \) tells us that if \( \models \neg w \), then \( M(S, \sigma) \subseteq \Sigma \) and \( M(S, \sigma) \not\models q \). Using this, show that \( w \) is the weakest precondition for partial correctness: if \( \models \{ p \} S \{ q \} \) then \( \models p \rightarrow w \). Hint: Make the argument analogous to the one for \( wp \).

For Problems 11 and 12, be sure to include the domain predicates. If you want, logically simplify as you go.

11. Use \( wp \) to calculate a total correctness precondition \( p \) for \( \{ p \} x := y/b[i] \{ x > 0 \} \).

12. Use \( wp \) to calculate total correctness conditions \( p_1 \) and \( p_2 \) for \( \{ p_1 \} y := \sqrt{c[t]}(b[j]) \{ z < y \} \) and \( \{ p_2 \} j := x/j \{ p_1 \} \)

13. Which of the following statements are correct?
   a. For all \( \sigma \in \Sigma \), \( \sigma \models wp(S, q) \) iff \( M(S, \sigma) \models q \)
   b. For all \( \sigma \in \Sigma \), \( \sigma \models wp(S, q) \) iff \( M(S, \sigma) \cup \Sigma \models q \)
   c. \( \models_{\text{tot}} \{ wp(S, q) \} S \{ q \} \)
   d. \( \models_{\text{tot}} \{ wp(S, q) \} S \{ q \} \)
   e. \( \models_{\text{tot}} \{ p \} S \{ q \} \) iff \( \models p \rightarrow wp(S, q) \)
   f. \( \models \{ p \} S \{ q \} \) iff \( \models p \rightarrow wp(S, q) \)
   g. \( \models \{ \neg wp(S, q) \} S \{ \neg q \} \)
   h. \( \models_{\text{tot}} \{ \neg wp(S, q) \} S \{ \neg q \} \)
   i. \( wp(S, q) \land wp(S, \neg q) \) is not satisfiable
   j. \( \not\models p \rightarrow wp(S, q) \) iff \( \not\models_{\text{tot}} \{ p \} S \{ q \} \)
   k. \( \not\models p \rightarrow wp(S, q) \) iff \( \not\models \{ p \} S \{ q \} \)
Solution to Activity 11 (Weakest Preconditions, pt. 2)

1. (wp involving sums) Recall $Q(i, s) \equiv 0 \leq i \leq n \land s = \text{sum}(0, i)$.
   
a. $wp(i := i+1; s := s+i, Q(i, s))$
      $\equiv wp(i := i+1, wp(s := s+i, Q(i, s)))$
      $\equiv wp(i := i+1, Q(i, s+i))$
      $\equiv wp(i := i+1, 0 \leq i \leq n \land s+i = \text{sum}(0, i))$
      $\equiv 0 \leq i+1 \leq n \land s+i = \text{sum}(0, i+1)$
   
b. $wp(s := s+i+1; i := i+1, Q(i, s))$
      $\equiv wp(s := s+i+1, wp(i := i+1, Q(i, s)))$
      $\equiv wp(s := s+i+1, Q(i+1, s))$
      $\equiv wp(s := s+i+1, 0 \leq i+1 \leq n \land s = \text{sum}(0, i+1))$
      $\equiv 0 \leq i+1 \leq n \land s+i+1 = \text{sum}(0, i+1)$
   
c. $wp(s := s+i; i := i+1, Q(i, s))$
      $\equiv wp(s := s+i, wp(i := i+1, Q(i, s)))$
      $\equiv wp(s := s+i, Q(i+1, s))$
      $\equiv wp(s := s+i, 0 \leq i+1 \leq n \land s = \text{sum}(0, i+1))$
      $\equiv 0 \leq i+1 \leq n \land s+i = \text{sum}(0, i+1)$ [which isn't compatible with $s = \text{sum}(0, i)]$

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

9. Assume $\vdash \{p\} S \{q\}$; we want to show that $\vdash p \rightarrow w$ where $w \equiv wlp(S, q)$. The definition of $wlp$ tells us that for any $\sigma \vdash \neg w$, we have $M(S, \sigma) \subseteq \Sigma$ and $M(S, \sigma) \not\models q$. The contrapositive of this is $((M(S, \sigma) \not\subseteq \Sigma \lor M(S, \sigma) \not\models q) \Rightarrow \sigma \not\models \neg w)$. Since $\sigma \in \Sigma, \sigma \not\models \neg w$ implies $\sigma \models w$. Turning back to the triple, $\sigma \vdash \{p\} S \{q\}$ implies $(\sigma \models p \Rightarrow (M(S, \sigma) \not\subseteq \Sigma \lor M(S, \sigma) \not\models q))$. Applying the contrapositive tells us $(\sigma \models p \Rightarrow \sigma \models w)$, so $\sigma \models p \rightarrow w$. Since this holds for any $\sigma \in \Sigma$, we know $\vdash p \rightarrow w$. 

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

12. For \( \{ p_1 \} \) \( y := \sqrt{b[j]} \{ z < y \} \), let \( p_1 \Leftrightarrow wp(y := \sqrt{b[j]}, z < y) \)
\[ \Leftrightarrow D(\sqrt{b[j]}) \land wlp(y := \sqrt{b[j]}, z < y) \]
\[ \Leftrightarrow 0 \leq j < \text{size}(b) \land b[j] \geq 0 \land z < \sqrt{b[j]} \]
For \( \{ p_2 \} \) \( j := x / j; \{ p_1 \} \), let \( p_2 \Leftrightarrow wp(j := x / j, p_1) \)
\[ \Leftrightarrow D(x / j) \land wlp(j := x / j, p_1) \]
\[ \Leftrightarrow j \neq 0 \land (0 \leq j < \text{size}(b) \land b[j] \geq 0 \land z < \sqrt{b[j]})(x / j / j] \]
\[ \Leftrightarrow j \neq 0 \land 0 \leq x / j < \text{size}(b) \land b[x / j] \geq 0 \land z < \sqrt{b[x / j]}. \]

13. (Properties of \( wp \) and \( wlp \)) The following properties are correct:
(a) and (b) are the basic definitions of \( wp \) and \( wlp \)
(c) and (d) say that \( wp \) and \( wlp \) are preconditions
(e) and (f) say that \( wp \) and \( wlp \) are weakest preconditions
(g) and (h) also say that \( wp \) and \( wlp \) are weakest
(j) and (k) are the contrapositives of (e) and (f).

However, (i) is incorrect: It claims that \( wlp(S, q) \land wlp(S, \neg q) \) is never satisfiable, but if \( M(S, \sigma) \subseteq \{ \bot \} \), then \( \sigma \) satisfies both \( wlp(S, q) \) and \( wlp(S, \neg q) \).