Weakest Preconditions pt. 2

CS 536: Science of Programming, Fall 2019

9/20; 9/24 misc cleanup; 10/9 p.3

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. *Some Examples of Calculating wp/wlp:*

- The programs in these examples don't end in "state" ⊥, so the *wp* and *wlp* are equivalent.
- **Example 2**: $wp(\mathtt{x := x+1}, \mathtt{x} \geq 0) \equiv \mathtt{x+1} \geq 0$
- **Example 3**: $wp(\mathtt{y := y+x;\ x := x+1}, \mathtt{x} \geq 0)$
  $\equiv wp(\mathtt{y := y+x}, wp(\mathtt{x := x+1}, \mathtt{x} \geq 0))$
  $\equiv wp(\mathtt{y := y+x}, \mathtt{x+1} \geq 0) \equiv \mathtt{x+1} \geq 0$
- **Example 4**: $wp(\mathtt{y := y+x;\ x := x+1}, \mathtt{x} \geq \mathtt{y})$
  $\equiv wp(\mathtt{y := y+x}, wp(\mathtt{x := x+1}, \mathtt{x} \geq \mathtt{y}))$
  $\equiv wp(\mathtt{y := y+x}, \mathtt{x+1} \geq \mathtt{y})$
  $\equiv \mathtt{x+1} \geq \mathtt{y+x}$
  - If we were asked only to calculate the *wp*, we'd stop here. If we also wanted to logically simplify the *wp* then $\mathtt{x+1} \geq \mathtt{y+x} \Leftrightarrow \mathtt{y} \leq 1$.
- **Example 5**: (Swap the two assignments in Example 4)
  $wp(\mathtt{x := x+1;\ y := y+x}, \mathtt{x} \geq \mathtt{y})$
  $\equiv wp(\mathtt{x := x+1}, wp(\mathtt{y := y+x}, \mathtt{x} \geq \mathtt{y}))$
  $\equiv wp(\mathtt{x := x+1}, \mathtt{x} \geq \mathtt{y+x}))$
  $\equiv \mathtt{x+1} \geq \mathtt{y+x+1}$ [$\Leftrightarrow \mathtt{y} \leq 0$ if you want to logically simplify]
- **Example 6**: $wp(\mathbf{if}\ \mathtt{y} \geq 0\ \mathbf{then}\ \mathtt{x := y}\ \mathbf{fi}, \mathtt{x} \geq 0)$
  $\equiv wp(\mathbf{if}\ \mathtt{y} \geq 0\ \mathbf{then}\ \mathtt{x := y}\ \mathbf{else\ skip\ fi}, \mathtt{x} \geq 0)$
  $\equiv (\mathtt{y} \geq 0 \wedge wp(\mathtt{x := y}, \mathtt{x} \geq 0)) \vee (\mathtt{y} < 0 \wedge wp(\mathbf{skip}, \mathtt{x} \geq 0))$
  $\equiv (\mathtt{y} \geq 0 \wedge \mathtt{y} \geq 0) \vee (\mathtt{y} < 0 \wedge \mathtt{x} \geq 0)$.
  If we want to simplify logically, we can continue with
  $\Leftrightarrow \mathtt{y} \geq 0 \vee (\mathtt{y} < 0 \wedge \mathtt{x} \geq 0)$
  $\Leftrightarrow (\mathtt{y} \geq 0 \vee \mathtt{y} < 0) \wedge (\mathtt{y} \geq 0 \vee \mathtt{x} \geq 0)$
  $\Leftrightarrow \mathtt{y} \geq 0 \vee \mathtt{x} \geq 0$ (which is also $\Leftrightarrow \mathtt{y} < 0 \rightarrow \mathtt{x} \geq 0$, if you prefer)

D. 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 $\sigma \vDash D(e)$ implies $\sigma(e) \neq \bot_e$.
  - This predicate has to be defined recursively, since we need to handle complex expressions like $D(\mathtt{b[b[i]]}) \equiv 0 \leq \mathtt{i} < \mathtt{size(b)} \wedge 0 \leq \mathtt{b[i]} < \mathtt{size(b)}$.
  - As with *wp* and *sp*, the domain predicate for an expression is unique only up to logical equivalence. For example, $D(\mathtt{x/y + u/v}) \equiv \mathtt{y} \neq 0 \wedge \mathtt{v} \neq 0 \Leftrightarrow \mathtt{v} \neq 0 \wedge \mathtt{y} \neq 0$.
- **Definition** (Domain predicate $D(e)$ for expression e) We must define D for each kind of expression that can cause a runtime error:
  - $D(c) \equiv D(v) \equiv \mathtt{T}$ if where c is a constant and v is a variable.
    - Evaluation of a variable or constant doesn't cause failure.
  - $D(\mathtt{b}[e]) \equiv D(e) \wedge 0 \leq e < \mathtt{size(b)}$
  - $D(e_1 / e_2) \equiv D(e_1 \,\%\, e_2) \Leftrightarrow D(e_1) \wedge D(e_2) \wedge e_2 \neq 0$
  - $D(\mathtt{sqrt}(e)) \equiv D(e) \wedge e \geq 0$
    - 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:
    - $D(e_1\ op\ e_2) \equiv D(e_1) \wedge D(e_2)$, except for $op \equiv /$ or $\%$
    - $D(op\ e) \equiv D(e)$, unless you add an operator that can cause runtime failure.
  - $D(\mathbf{if}\ B\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2\ \mathbf{fi}) \equiv D(B) \wedge (B \rightarrow D(e_1)) \wedge (\neg B \rightarrow D(e_2))$
    - (For a conditional expression, we only need safety of the one brach we execute.)
- **Example 7**: $D(\mathtt{b[b[i]]}) \equiv D(\mathtt{b[i]}) \wedge 0 \leq \mathtt{b[i]} < \mathtt{size(b)}$

  $\equiv D(\mathtt{i}) \wedge 0 \leq \mathtt{i} < \mathtt{size(b)} \wedge 0 \leq \mathtt{b[i]} < \mathtt{size(b)}$

  $\Leftrightarrow 0 \leq \mathtt{i} < \mathtt{size(b)} \wedge 0 \leq \mathtt{b[i]} < \mathtt{size(b)}$
- **Example 8**: $D((\mathtt{-b + sqrt(b*b - 4*a*c))/(2*a)})$

  $\equiv D(e) \wedge D(\mathtt{2*a}) \wedge \mathtt{2*a} \neq 0$ where $e \equiv \mathtt{-b + sqrt(b*b - 4*a*c)}$

  $\equiv D(\mathtt{-b}) \wedge D(\mathtt{sqrt(b*b - 4*a*c)}) \wedge D(\mathtt{2*a}) \wedge \mathtt{2*a} \neq 0$

  $\Leftrightarrow D(\mathtt{sqrt(b*b - 4*a*c)}) \wedge \mathtt{2*a} \neq 0$ Since $D(\mathtt{-b}) \equiv D(\mathtt{2*a}) \equiv \mathtt{T}$

  $\equiv D(\mathtt{b*b - 4*a*c}) \wedge (\mathtt{b*b - 4*a*c} \geq 0) \wedge \mathtt{2*a} \neq 0$

  $\Leftrightarrow \mathtt{b*b - 4*a*c} \geq 0 \wedge \mathtt{2*a} \neq 0$
- **Example 9**: $D(\mathbf{if}\ 0 \leq \mathtt{i} < \mathtt{size(b)}\ \mathbf{then}\ \mathtt{b[i]}\ \mathbf{else}\ 0\ \mathbf{fi})$

  $\equiv D(B) \wedge (B \rightarrow D(\mathtt{b[i]})) \wedge (\neg B \rightarrow D(0))$ where $B \equiv 0 \leq \mathtt{i} < \mathtt{size(b)}$

  $\equiv (B \rightarrow D(\mathtt{b[i]})) \wedge (\neg B \rightarrow \mathtt{T})$ since $D(B)$ and $D(0) \equiv \mathtt{T}$

  $\Leftrightarrow B \rightarrow D(\mathtt{b[i]})$ since everything implies true

  $\equiv B \rightarrow D(\mathtt{i}) \wedge 0 \leq \mathtt{i} < \mathtt{size(b)}$ expanding $D(\mathtt{b[i]})$

$\Leftrightarrow B \rightarrow B$ since $B \equiv 0 \le \mathtt{i} < \mathtt{size}(\mathtt{b}))$

$\Leftrightarrow \mathtt{T}$

E. Avoiding Runtime Errors in Programs

- Recall that we extended our notion of operational semantics to include $\langle S, \sigma\rangle \rightarrow^* \langle E, \bot_e\rangle$ 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.
  - $D(\mathbf{skip}) \equiv \mathtt{T}$
  - $D(v := e) \equiv D(e)$
  - $D(\mathtt{b}[e_1] := e_2) \equiv D(\mathtt{b}[e_1]) \wedge D(e_2)$
  - $D(S_1\mathbf{;}\ S_2, q) \equiv D(S_1) \wedge D(S_2)$
  - $D(\mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{fi}, q) \equiv D(B) \wedge (B \rightarrow D(S_1)) \wedge (\neg B \rightarrow D(S_2))$
  - $D(\mathbf{if}\ B_1 \rightarrow S_1 \ \square\ B_2 \rightarrow S_2\ \mathbf{fi}, q) \equiv D(B_1 \vee B_2) \wedge (B_1 \vee B_2) \wedge (B_1 \rightarrow D(S_1)) \wedge (B_2 \rightarrow D(S_2))$
    - The condition $(B_1 \vee B_2)$ avoids failure of the nondeterministic **if-fi** due to none of the guards holding. This definition extends easily to **if-fi** with more than two guarded commands.
  - $D(\mathbf{while}\ B\ \mathbf{do}\ S_1\ \mathbf{od}) \equiv D(B) \wedge (B \rightarrow D(S_1))$
  - $D(\mathbf{do}\ B_1 \rightarrow S_1 \ \square\ B_2 \rightarrow S_2\ \mathbf{od}) \equiv D(B_1 \vee B_2) \wedge (B_1 \rightarrow D(S_1)) \wedge (B_2 \rightarrow D(S_2))$
    - The domain predicate for nondeterministic **do-od** is like that for **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 $wp(S, q) \Leftrightarrow wlp\,(S, D(S) \wedge q)$ [9/25] check me~~
- ***Definition***: $wp(S, q) \equiv w \wedge D(w) \wedge D(S)$ where $w \equiv wlp(S, q)$ [10/9]
- (Definition of $D(\varphi)$ – interesting case is $D(Q\ \mathtt{x}.\delta) \equiv \forall\, \mathtt{x}.D(\delta)$)
- **Example 10**: If a program does a division, then the *wp* and *wlp* can differ:
  - $D(\mathtt{x} := \mathtt{y};\ \mathtt{z} := \mathtt{v/x})$
    $\equiv D(\mathtt{x} := \mathtt{y}) \wedge D(\mathtt{z} := \mathtt{v/x})$
    $\Leftrightarrow D(\mathtt{v/x})$ // since $D(\mathtt{x} := \mathtt{y}) \equiv D(\mathtt{y}) \equiv \mathtt{T}$
    $\Leftrightarrow \mathtt{x} \neq 0$ // Technically, $D(\mathtt{v/x}) \equiv D(\mathtt{v}) \wedge D(\mathtt{x}) \wedge \mathtt{x} \neq 0$
  - $wp\,(\mathtt{x} := \mathtt{y};\ \mathtt{z} := \mathtt{v/x}, \mathtt{z} > \mathtt{x}+2)$
    $\equiv wlp\,(\mathtt{x} := \mathtt{y}, wlp\,(\mathtt{z} := \mathtt{v/x}, \mathtt{z} > \mathtt{x}+2 \wedge D(\mathtt{x} := \mathtt{y};\ \mathtt{z} := \mathtt{v/x})))$
    $\Leftrightarrow wlp\,(\mathtt{x} := \mathtt{y}, wlp\,(\mathtt{z} := \mathtt{v/x}, \mathtt{z} > \mathtt{x}+2 \wedge \mathtt{x} \neq 0))$ // Substituting from above
    $\equiv wlp\,(\mathtt{x} := \mathtt{y}, \mathtt{v/x} > \mathtt{x}+2 \wedge \mathtt{x} \neq 0)$
    $\equiv \mathtt{v/y} > \mathtt{y}+2 \wedge \mathtt{y} \neq 0$

Weakest Preconditions, pt. 1

CS 536: Science of Programming

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 $wp(S, q_1 \vee q_2)$, $wp(S, q_1)$, 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 $wp(S, q_1 \vee q_2)$ and $wp(S, q_1)$ and $wp(S, q_2)$, related if S is deterministic? If S is nondeterministic?

2. Calculate the *wlp* in each of the following cases. Just syntactically calculate the *wlp*; don't also logically simplify the result.)
   a. $wlp(\texttt{k} := \texttt{k} - \texttt{s}, \texttt{n} = 3 \wedge \texttt{k} = 4 \wedge \texttt{s} = -7)$.
   b. $wlp(\texttt{n} := \texttt{n*(n-k)}, \texttt{n} = 3 \wedge \texttt{k} = 4 \wedge \texttt{s} = -7)$.
   c. $wlp(\texttt{n} := \texttt{n*(n-k)};\ \texttt{k} := \texttt{k} - \texttt{s}, \texttt{n} > \texttt{k} + \texttt{s})$

3. Let $Q(i, s) \equiv 0 \le i \le \texttt{n} \wedge s = \texttt{sum}(0, i)$ where $\texttt{sum}(u, v)$ is the sum of u, $u+1$, …, v (when $u \le v$) or 0 (when $u > v$).
   a. Calculate $wp(\texttt{i} := \texttt{i+1};\ \texttt{s} := \texttt{s+i}, Q(\texttt{i}, \texttt{s}))$.
   b. Calculate $wp(\texttt{s} := \texttt{s+i+1};\ \texttt{i} := \texttt{i+1}, Q(\texttt{i}, \texttt{s}))$.
   c. Calculate $wp(\texttt{s} := \texttt{s+i};\ \texttt{i} := \texttt{i+1}, Q(\texttt{i}, \texttt{s}))$. (This one isn't compatible with $\texttt{s = sum(0, i)}$.)

4. Calculate the *wp* below. (Again, just calculate the syntactically *wp* without logically simplifying the result.)
   a. $wp(\textbf{if } B \textbf{ then } \texttt{x} := \texttt{x/2} \textbf{ fi};\ \texttt{y} := \texttt{x}, \texttt{x} = 5 \wedge \texttt{y} = \texttt{z})$.
   b. $wp(\textbf{if } \texttt{x} \ge 0 \textbf{ then } \texttt{x} := \texttt{x*2} \textbf{ else } \texttt{x} := \texttt{y} \textbf{ fi};\ \texttt{x} := \texttt{c*x}, \texttt{a} \le \texttt{x} < \texttt{y})$

For Problems 5 and 6, be sure to include the domain predicates. If you want, logically simplify as you go.

5. Calculate p to be the *wp* of $\{p\}\ \texttt{x} := \texttt{y/b[i]}\ \{\texttt{x} > 0\}$.

6. Calculate p_1 and p_2 to be the *wp* of $\{p_1\}\ \texttt{y} := \texttt{sqrt(b[j])}\ \{\texttt{z} < \texttt{y}\}$ and $\{p_2\}\ \texttt{j} := \texttt{x/j}\ \{p_1\}$.

Solution to Activity 11 (Weakest Preconditions, pt. 2)

1. For deterministic S, $wp(S, q_1 \vee q_2) \Leftrightarrow wp(S, q_1) \vee wp(S, q_2)$. For nondeterministic S, we have $\Rightarrow$ instead of $\Leftrightarrow$.

2. (Calculate *wlp*)

   a. $wlp(\texttt{k} := \texttt{k - s}, \texttt{n} = 3 \wedge \texttt{k} = 4 \wedge \texttt{s} = -7) \equiv \texttt{n} = 3 \wedge \texttt{k-s} = 4 \wedge \texttt{s} = -7$

   b. $wlp(\texttt{n} := \texttt{n*(n-k)}, \texttt{n} = 3 \wedge \texttt{k} = 4 \wedge \texttt{s} = -7) \equiv \texttt{n*(n-k)} = 3 \wedge \texttt{k} = 4 \wedge \texttt{s} = -7$

   c. $wlp(\texttt{n} := \texttt{n*(n-k)}\texttt{;}\ \texttt{k} := \texttt{k-s}, \texttt{n} > \texttt{k+s})$
   $\equiv wlp(\texttt{n} := \texttt{n*(n-k)}, wlp(\texttt{k} := \texttt{k-s}, \texttt{n} > \texttt{k+s}))$
   $\equiv wlp(\texttt{n} := \texttt{n*(n-k)}, \texttt{n} > \texttt{k-s+s})$
   $\equiv \texttt{n*(n-k)} > \texttt{k-s+s}$

3. (*wp* involving sums) We have $Q(i, s) \equiv 0 \le \texttt{i} \le \texttt{n} \wedge \texttt{s} = \texttt{sum(0, i)}$.

   a. $wp(\texttt{i} := \texttt{i+1}\texttt{;}\ \texttt{s} := \texttt{s+i}, Q(\texttt{i}, \texttt{s}))$
   $\equiv wp(\texttt{i} := \texttt{i+1}, wp(\texttt{s} := \texttt{s+i}, Q(\texttt{i}, \texttt{s})))$
   $\equiv wp(\texttt{i} := \texttt{i+1}, Q(\texttt{i}, \texttt{s+i}))$
   $\equiv wp(\texttt{i} := \texttt{i+1}, 0 \le \texttt{i} \le \texttt{n} \wedge \texttt{s+i} = \texttt{sum(0, i)})$
   $\equiv 0 \le \texttt{i+1} \le \texttt{n} \wedge \texttt{s+i} = \texttt{sum(0, i+1)}$

   b. $wp(\texttt{s} := \texttt{s+i+1}\texttt{;}\ \texttt{i} := \texttt{i+1}, Q(\texttt{i}, \texttt{s}))$
   $\equiv wp(\texttt{s} := \texttt{s+i+1}, wp(\texttt{i} := \texttt{i+1}, Q(\texttt{i}, \texttt{s})))$
   $\equiv wp(\texttt{s} := \texttt{s+i+1}, Q(\texttt{i+1}, \texttt{s}))$
   $\equiv wp(\texttt{s} := \texttt{s+i+1}, 0 \le \texttt{i+1} \le \texttt{n} \wedge \texttt{s} = \texttt{sum(0, i+1)})$
   $\equiv 0 \le \texttt{i+1} \le \texttt{n} \wedge \texttt{s+i+1} = \texttt{sum(0, i+1)}$

   c. $wp(\texttt{s} := \texttt{s+i}\texttt{;}\ \texttt{i} := \texttt{i+1}, Q(\texttt{i}, \texttt{s}))$
   $\equiv wp(\texttt{s} := \texttt{s+i}, wp(\texttt{i} := \texttt{i+1}, Q(\texttt{i}, \texttt{s})))$
   $\equiv wp(\texttt{s} := \texttt{s+i}, Q(\texttt{i+1}, \texttt{s}))$
   $\equiv wp(\texttt{s} := \texttt{s+i}, 0 \le \texttt{i+1} \le \texttt{n} \wedge \texttt{s} = \texttt{sum(0, i+1)})$
   $\equiv 0 \le \texttt{i+1} \le \texttt{n} \wedge \texttt{s+i} = \texttt{sum(0, i+1)}$ [which isn't compatible with $\texttt{s} = \texttt{sum(0, i)}$]

4. (*wp* of `if-then`)

   a. $wp(\textbf{if } B \textbf{ then } \texttt{x} := \texttt{x/2} \textbf{ fi;}\ \texttt{y} := \texttt{x}, \texttt{x} = 5 \wedge \texttt{y} = \texttt{z})$
   $\equiv wp(\textbf{if } B \textbf{ then } \texttt{x} := \texttt{x/2} \textbf{ fi}, wp(\texttt{y} := \texttt{x}, \texttt{x} = 5 \wedge \texttt{y} = \texttt{z}))$
   $\equiv wp(\textbf{if } B \textbf{ then } \texttt{x} := \texttt{x/2} \textbf{ fi}, \texttt{x} = 5 \wedge \texttt{x} = \texttt{z})$
   $\equiv (B \rightarrow wp(\texttt{x} := \texttt{x/2}, \texttt{x} = 5 \wedge \texttt{x} = \texttt{z})) \wedge (\neg B \rightarrow wp(\textbf{skip}, \texttt{x} = 5 \wedge \texttt{x} = \texttt{z}))$
   $\equiv (B \rightarrow \texttt{x/2} = 5 \wedge \texttt{x/2} = \texttt{z}) \wedge (\neg B \rightarrow \texttt{x} = 5 \wedge \texttt{x} = \texttt{z})$

   b. $wp(\textbf{if } \texttt{x} \ge 0 \textbf{ then } \texttt{x} := \texttt{x*2} \textbf{ else } \texttt{x} := \texttt{y} \textbf{ fi};\ \texttt{x} := \texttt{c*x}, \texttt{a} \le \texttt{x} < \texttt{y})$
   $\equiv wp(S, wp(\texttt{x} := \texttt{c*x}, \texttt{a} \le \texttt{x} < \texttt{y}))$ where S is the **if** statement
   $\equiv wp(S, \texttt{a} \le \texttt{c*x} < \texttt{y})$
   $\equiv wp(\textbf{if } \texttt{x} \ge 0 \textbf{ then } \texttt{x} := \texttt{x*2} \textbf{ else } \texttt{x} := \texttt{y} \textbf{ fi}, \texttt{a} \le \texttt{c*x} < \texttt{y})$

$\equiv (\texttt{x} \geq 0 \rightarrow wp(\texttt{x} := \texttt{x*2}, \texttt{a} \leq \texttt{c*x} < \texttt{y})) \wedge (\texttt{x} < 0 \rightarrow wp(\texttt{x} := \texttt{y}, \texttt{a} \leq \texttt{c*x} < \texttt{y}))$

$\equiv (\texttt{x} \geq 0 \rightarrow \texttt{a} \leq \texttt{c*(x*2)} < \texttt{y}) \wedge (\texttt{x} < 0 \rightarrow \texttt{a} \leq \texttt{c*y} < \texttt{y})$

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

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