Calculate wlp for Loop-Free Programs

- We start with the calculations of wlp in loop-free programs because: 1) if a program is loop-free and runtime-error-free, then $wlp \Leftrightarrow wp$ 2) if a program can create runtime errors, then we can add "error-avoiding information" to convert wlp to wp. 3) We will handle loops in the future (we actually cannot find wp or wlp for loops).

- The following algorithm takes S and q and calculates a predicate for $wlp(S, q)$. Since this calculation procedure is textual or syntactical, so we use $\equiv$ instead of $=$ or $\Leftrightarrow$ here.
  - $wlp\ (\mathbf{skip}, q) \equiv q$.
  - [Backward Assignment Rule] $wlp\ \big(v \coloneqq e, Q(v)\big) \equiv Q(e)$, where Q is a predicate function.
    - this operation that takes us from $Q(v)$ to $Q(e)$ is called **syntactic substitution**; we will study this carefully in the future.
    - $wlp(x \coloneqq x + 1, x \geq 2) \equiv x + 1 \geq 2$
  - $wlp(S_1; S_2, q) \equiv wlp\big(S_1, wlp(S_2, q)\big)$.
  - $wlp\ (\mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{fi}, q) \equiv \big(B \rightarrow wlp(S_1, q)\big) \wedge \big(\neg B \rightarrow wlp(S_2, q)\big)$
    $\Leftrightarrow \big(B \wedge wlp(S_1, q)\big) \vee \big(\neg B \wedge wlp(S_2, q)\big)$.
  - $wlp\ (\mathbf{if}\ B_1 \rightarrow S_1 \ \square\ B_2 \rightarrow S_2\ \mathbf{fi}, q) \equiv \big(B_1 \rightarrow wlp(S_1, q)\big) \wedge \big(B_2 \rightarrow wlp(S_2, q)\big)$.

1. Calculate the following weakest liberal preconditions.
   a. $wlp(x \coloneqq x + 1,\ x \geq 0) \equiv x + 1 \geq 0 \Leftrightarrow x \geq -1$

   b. $wlp\ (y \coloneqq y + x;\ x \coloneqq x + 1,\ x \geq 0) \equiv wlp\ (y \coloneqq y + x,\ x + 1 \geq 0) \equiv x + 1 \geq 0$

   c. $wlp\ (y \coloneqq y + x;\ x \coloneqq x + 1,\ x \geq y) \equiv wlp\ (y \coloneqq y + x,\ x + 1 \geq y) \equiv x + 1 \geq y + x \Leftrightarrow y \leq 1$

   d. $wlp\ (x \coloneqq x + 1;\ y \coloneqq y + x,\ x \geq y) \equiv wlp\ (x \coloneqq x + 1,\ x \geq y + x) \equiv x + 1 \geq y + x + 1 \Leftrightarrow y \leq 0$

   - 7.c and 7.d show that $wlp(S_1; S_2, q)$ and $wlp(S_2; S_1, q)$ do not have to be semantically equal.

   e. $wlp\ (\mathbf{if}\ y \geq 0\ \mathbf{then}\ x \coloneqq y\ \mathbf{fi}, x \geq 0)$
   $\equiv wlp\ (\mathbf{if}\ y \geq 0\ \mathbf{then}\ x \coloneqq y\ \mathbf{else\ skip\ fi}, x \geq 0)$
   $\equiv (y \geq 0 \rightarrow y \geq 0) \wedge (y < 0 \rightarrow x \geq 0)$
   $\Leftrightarrow T \wedge (y < 0 \rightarrow x \geq 0)$
   $\Leftrightarrow y < 0 \rightarrow x \geq 0$
   $\Leftrightarrow y \geq 0 \vee x \geq 0$

   f. $wlp\ (\mathbf{if}\ y \geq 0 \rightarrow x \coloneqq y \ \square\ x < 0 \rightarrow x \coloneqq y + 1\ \mathbf{fi}, x \geq 0)$
   $\equiv (y \geq 0 \rightarrow y \geq 0) \wedge (x < 0 \rightarrow y + 1 \geq 0)$
   $\Leftrightarrow x < 0 \rightarrow y \geq -1$
   $\Leftrightarrow x \geq 0 \vee y \geq -1$

Avoid Runtime Error in Expressions

- Runtime errors can appear while evaluating expressions. To avoid such errors while calculating $\sigma(e)$, we define **domain predicate** $D(e)$ such that "if $\sigma \vDash D(e)$ then $\sigma(e) \neq \perp_e$" or "$\sigma \vDash D(e)$ logically implies $\sigma(e) \neq \perp_e$".
  - For example, to avoid runtime error, we can define $D(b[b[k]]) \equiv 0 \leq k < size(b) \land 0 \leq b[k] < size(b)$.
  - Here is another example, we can define $D(x/y + u/v) \equiv y \neq 0 \land v \neq 0$.

- From the above examples, we can see that a domain predicate will be a conjunction of several "requirements" on the variables / expressions. Remind that, we say $\sigma \vDash p \land q$ iff $\sigma \vDash p$ and $\sigma \vDash q$.

- The calculation of $D(e)$ can be pure textual. We define $D(e)$ as follows:
  - If e contains no array selection, no "/", no "%", no $sqrt()$, then $D(e) \equiv T$
  - $D(b[e]) \equiv D(e) \land 0 \leq e < size(b)$.
  - $D(e_1/e_2) \equiv D(e_1 \,\%\, e_2) \equiv D(e_1) \land D(e_2) \land e_2 \neq 0$.
  - $D(sqrt(e)) \equiv D(e) \land e \geq 0$.
  - $D(op\ e) \equiv D(e)$.
  - $D(e_1\ op\ e_2) \equiv D(e_1) \land D(e_2)$ for all binary operator op other than "/" and "%".
  - $D(f(e_1, e_2, \ldots, e_n)) \equiv D(e_1) \land D(e_2) \land \ldots \land D(e_n)$ for $f()$ other than $sqrt()$.
  - $D\ (\textbf{if } B \textbf{ then } e_1 \textbf{ else } e_2 \textbf{ fi}) \equiv D(B) \land (B \to D(e_1)) \land (\neg B \to D(e_2))$.

2. Calculate domain predicate $D(e)$ for the following expressions.

   a. $D(x > 0 \to sqrt(x) > 0)$

   $$\begin{aligned} &\equiv D(x > 0) \land D(sqrt(x) > 0) \\ &\equiv T \land \big(D(sqrt(x)) \land D(0)\big) \\ &\Leftrightarrow x \geq 0 \end{aligned}$$

   b. $D(b[b[k]])$

   $$\begin{aligned} &\equiv D(b[k]) \land 0 \leq b[k] < size(b) \\ &\equiv (D(k) \land 0 \leq k < size(b)) \land 0 \leq b[k] < size(b) \\ &\equiv (T \land 0 \leq k < size(b)) \land 0 \leq b[k] < size(b) \\ &\Leftrightarrow 0 \leq k < size(b) \land 0 \leq b[k] < size(b) \end{aligned}$$

   c. Let $B \equiv 0 \leq k < size(b)$.

   $D(\textbf{if } B \textbf{ then } b[k] \textbf{ else } -1 \textbf{ fi})$

   $$\begin{aligned} &\equiv D(B) \land (B \to D(b[k])) \land (\neg B \to D(-1)) \\ &\equiv T \land (B \to D(b[k])) \land (\neg B \to T) \\ &\Leftrightarrow B \to D(b[k]) \\ &\equiv 0 \leq k < size(b) \to T \land 0 \leq k < size(b) \\ &\Leftrightarrow T \end{aligned}$$

Avoid Runtime Error in Statements

- To avoid runtime errors in the execution of S, we can define domain predicate $D(S)$ that gives a sufficient condition that avoids runtime errors.

- The calculation of $D(S)$ is textual as well. Let us define $D(S)$ as follows:

- $D(\mathbf{skip}) \equiv T$
- $D(v := e) \equiv D(e)$
- $D(b[e_1] := e_2) \equiv D(b[e_1]) \wedge D(e_2)$
- $D(S_1; S_2) \equiv D(S_1) \wedge wp(S_1, D(S_2))$
  - $D(S_1)$ guarantees the execution of S_1 is error-free.
  - $D(S_2)$ guarantees the execution of S_2 is error-free, so $wp(S_1, D(S_2))$ guarantees the of S_2 is error-free before S_1 is executed.

- $D(\mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{fi}) \equiv D(B) \wedge (B \to D(S_1)) \wedge (\neg B \to D(S_2))$.
- $D(\mathbf{if}\ B_1 \to S_1 \,\square\, B_2 \to S_2\ \mathbf{fi}) \equiv D(B_1 \vee B_2) \wedge (B_1 \vee B_2) \wedge (B_1 \to D(S_1)) \wedge (B_2 \to D(S_2))$.
- $D(\mathbf{while}\ B\ \mathbf{do}\ S_1\ \mathbf{od}) \equiv D(B) \wedge (B \to D(S_1))$.
- $D(\mathbf{do}\ B_1 \to S_1 \,\square\, B_2 \to S_2\ \mathbf{od}) \equiv D(B_1 \vee B_2) \wedge (B_1 \to D(S_1)) \wedge (B_2 \to D(S_2))$.
  - Although we cannot calculate wp or wlp for a loop, but we can calculate its domain predicate. We will discuss how to avoid divergence in the future.

<u>Calculate wp for Loop-Free Programs</u>

- $wp(S, q) \equiv D(S) \wedge wlp(S, q) \wedge D(wlp(S, q))$

3. Let $w \Leftrightarrow wp(x := b[k],\ sqrt(x) \geq 1)$, calculate w.

   - $D(x := b[k]) \equiv D(b[k]) \equiv T \wedge 0 \leq k < size(b) \Leftrightarrow 0 \leq k < size(b)$
   - $wlp(x := b[k],\ sqrt(x) \geq 1) \equiv sqrt(b[k]) \geq 1$
   - $D(wlp(x := b[k],\ sqrt(x) \geq 1)) \equiv D(sqrt(b[k]) \geq 1) \Leftrightarrow b[k] \geq 0 \wedge 0 \leq k < size(b)$
   - $$\begin{aligned} w &\equiv (0 \leq k < size(b)) \ \wedge\ (sqrt(b[k]) \geq 1) \ \wedge\ (b[k] \geq 0 \wedge 0 \leq k < size(b)) \\ &\Leftrightarrow sqrt(b[k]) \geq 1 \wedge b[k] \geq 0 \wedge 0 \leq k < size(b) \\ &\Leftrightarrow b[k] \geq 1 \wedge 0 \leq k < size(b) \end{aligned}$$

4. Let $w \Leftrightarrow wp(x := y; z := v/x\ , z > x + 2)$, calculate w.

   - $wlp(x := y; z := v/x\ , z > x + 2) \equiv wlp(x := y,\ v/x > x + 2) \equiv v/y > y + 2$
   - $D(wlp(x := y; z := v/x, z > x + 2)) \equiv D(v/y > y + 2) \Leftrightarrow y \neq 0$
   - $$\begin{aligned} D(x := y; z := v/x) &\equiv D(x := y) \wedge wp(x := y, D(z := v/x)) \\ &\equiv T \wedge wp(x := y, x \neq 0) \\ &\equiv T \wedge D(x := y) \wedge wlp(x := y, x \neq 0) \wedge D(wlp(x := y, x \neq 0)) \\ &\equiv T \wedge T \wedge y \neq 0 \wedge D(y \neq 0) \Leftrightarrow y \neq 0 \end{aligned}$$
   - $w \equiv y \neq 0\ \wedge\ v/y > y + 2\ \wedge\ \ y \neq 0\ \Leftrightarrow\ y \neq 0 \wedge v/y > y + 2$