Proving Program Termination
Verifying Termination

- “Partial correctness” means that the program is correct, *provided* that it terminates.

- “Total correctness” is partial correctness and termination.
Verifying Termination

- The reason that termination is verified separately is that it requires coming up with a different sort of expression than an invariant.

- Such an expression is a “variant”. It describes a program’s inexorable movement toward a stopping point.
Variants

- Clearly the only cause for a (non-recursive) program’s non-termination could lie in while-loops.
- A variant is some expression $E$ such that:
  - $E \geq 0$ is invariant
  - The value of $E$ decreases at every iteration.
- If a loop has a variant, then the loop must terminate.
Variant Example

\{x == x_0 \land y == y_0 \land x_0 \geq 0\}
while( x > 0 )
{
  y = y + k;
  x = x-1;
}
\{y == y_0 + k*x_0\}

Here a variant for the loop would be x, since:
  x \geq 0 is invariant, and
  x decreases on each iteration.
A sufficient condition for $E$ to be a variant of

$$\text{while}(P)\;\text{Body};$$

is that we be able to derive a triple:

$$\{ E_0 = E \land E > 0 \land P \} \;\text{Body} \;\{ E_0 > E \land E \geq 0 \}$$

where $E_0$ is a free variable.
{E_0 = E \land E > 0 \land P} \text{ Block } \{E_0 > E \land E \geq 0 \}

Consider the previous while program:

```java
while( x > 0 )
{
    y = y + k;
    x = x - 1;
}
```

\{x_0 == x \land x > 0 \land x > 0\} y = y + k; x = x - 1; \{x_0 > x \land x \geq 0 \}
is the triple to be derived.
Variant as a Triple: Example

\[ \{ x_0 == x \land x > 0 \land x > 0 \} \quad y = y + k; \quad x = x - 1; \quad \{ x_0 > x \land x \geq 0 \} \]

is the triple to be derived.

Working backward from the post-condition, we need to derive:

\[ \{ x_0 == x \land x > 0 \land x > 0 \} \quad y = y + k; \quad \{ x_0 > x - 1 \land x - 1 \geq 0 \} \]

which follows from the implication rule if we can derive:

\[ \{ x_0 == x \land x > 0 \land x > 0 \} \rightarrow \{ x_0 > x - 1 \land x - 1 \geq 0 \} \]

which follows directly (assuming \( x \) integer).
Exercise

Derive a variant for the gcd program introduced earlier:

\[
\{ x \equiv x_0 \land y \equiv y_0 \land x > 0 \land y > 0 \} \\
\text{while( } x \neq y \text{ )} \\
\text{if( } x > y \text{ )} \\
\quad x = x - y; \\
\text{else} \\
\quad y = y - x; \\
\{ x \equiv \text{gcd}(x_0, y_0) \} 
\]
Exercise

Can a variant be derived for the similar triple:

\[
\{ x = x_0 \land y = y_0 \land x \geq 0 \land y \geq 0 \}
\]

\[
\text{while}( x \neq y )
\]
\[
\text{if}( x > y )
\]
\[
\quad x = x - y;
\]
\[
\text{else}
\]
\[
\quad y = y - x;
\]

\[
\{ x = \text{gcd}(x_0, y_0) \} 
\]
Weakest Preconditions (WP)
Definition of WP

For a program statement $S$ and an expectation $Q$, the **weakest-precondition** is the weakest assumption $P$ satisfying:

$$\{P\} \ S \ {Q}$$
wp obeys some fairly obvious rules:

- \( \text{wp}(x = E;, Q) = Q \ [E/X] \) as in the assignment rule
- \( \text{wp}(B_1; B_2, Q) = \)
- \( \text{wp}(\text{if}(P) B_1; \text{else } B_2, Q) = \)

wp for a loop is harder, because it generally requires an infinite formula (unwind the loop as an infinite nest of conditions).
Example: WP for a Test

- \{??\}
  if( x > y ) x = x-y; else y = y-x;
  \{gcd(x, y) == z\}

- \(wp\) is

  \(wp\)'s of the assignment statements
  \[
  ((x > y) \rightarrow gcd(x-y, y) == z) \cdot
  \wedge (\neg(x > y) \rightarrow gcd(x, y-x) == z)
  \]
Example 2: WP for a Test

- `{??}
  if( x > y ) z = x; else z = y;
  {z == max(x, y)}

- wp is

  wp’s of the assignment statements

  \((x > y) \rightarrow \max(x, y) == x\) \cdot
  \neg(x > y) \rightarrow \max(x, y) == y\)
  which simplifies to \text{true}.
When the *else* part is missing

- If the *else* part is missing, then T is effectively a “no-op”, “skip”, or trivial assignment \( x = x \);
- Since \( \text{wp}(x = x; Q) = Q \)
- The wp for
  - if(P) S
  is then
    - \( (P \rightarrow \text{wp}(S, Q)) \)
    \( \land (\neg P \rightarrow Q) \)
Example: WP for a Test without else

- `{??}
  if( x > y ) y = x;
  \{y = \text{max}(x, y)\}

- \(\text{wp is}\) wp of the assignment statement
  
  \(((x > y) \rightarrow x = \text{max}(x, x))
  \land (\neg(x > y) \rightarrow y = \text{max}(x, y))\)

- which simplifies to \text{true.}\
Alternate WP for a Test

- \( \text{wp}(\text{if}(P) \ S \ \text{else} \ T, \ Q) = \)
  
  \[ (P \land \text{wp}(S, \ Q)) \lor \]
  
  \[ (\neg P \land \text{wp}(T, \ Q)) \]

- To see that this is equivalent to the previous version, let \( \text{wp}(S, \ Q) \) be \( A \) and \( \text{wp}(T, \ Q) \) be \( B \). Then we are asking whether
  
  \[ (P \land A) \lor (\neg P \land B) \]
  
  is equivalent to
  
  \[ (P \rightarrow A) \land (\neg P \rightarrow B) \]
Alternate WP for a Test

- \((P \land A) \lor (\neg P \land B) =? (P \rightarrow A) \land (\neg P \rightarrow B)\)
- For \(P =\) true, this becomes \(A =? A\).
- For \(P =\) false, this becomes \(B =? B\).
- Therefore the two forms are equivalent.
Suppose that $S(x, y)$ stands for

$$\text{if}( x > y ) \{ x, y = y, x; \}$$

where by the assignment statement we mean *parallel assignment*: the RHS’s are evaluated then the values are *simultaneously* assigned to the LHS variables.
Sorting Example (2)

- $S(x, y)$ stands for
  \[
  \text{if}( x > y ) \{x, y = y, x;\}
  \]

- Using the WP for if, we have for any predicate $P$:
  \[
  \text{wp}(\text{if}( x > y ) \{x, y = y, x;\}, P) =
  \]
  \[
  [(x > y) \rightarrow P[x, y \leftarrow y, x]] \land [\neg(x > y) \rightarrow P]
  \]
Suppose that we wish to show:

\{true\}

\[ S(x, y); S(y, z); S(x, y) \]

\[ \{x \leq y \land y \leq z\} \]

i.e. the sequence shown sorts three numbers.
Sorting Example (4)

- Working backward from the last statement $S(x, y)$:
  
  $wp(S(x, y), \{x \leq y \land y \leq z\}) =$

- $wp(if(x>y) \{x, y = y, x;\}, \{x \leq y \land y \leq z\}) =$

- $[(x > y) \to (y \leq x \land x \leq z)]$
  
  $\land [\neg(x > y) \to (x \leq y \land y \leq z)]$

- which simplifies (using reasoning about $\leq$ and $>$) to
  
  $[(x > y) \to (x \leq z)] \land [(x \leq y) \to (y \leq z)]$

- which simplifies to
  
  $(x \leq z) \land (y \leq z)$
Sorting Example (5)

- Working backward from the middle statement $S(y, z)$: \( wp(S(y, z), (x \leq z) \land (y \leq z)) = \)

- \[
\begin{align*}
& [ (y > z) \rightarrow [(x \leq y) \land (z \leq y)] ] \\
& \land [ \neg (y > z) \rightarrow [(x \leq z) \land (y \leq z)] ]
\end{align*}
\]

- which is equivalent to
  \[
\begin{align*}
& [(y > z) \rightarrow (x \leq y)] \\
& \land [(y \leq z) \rightarrow (x \leq z)]
\end{align*}
\]
Sorting Example (6)

- Working backward from the first statement $S(x, y)$:

$$wp(S(x, y), [(y > z) \rightarrow (x \leq y)] \land [(y < z) \rightarrow (x < z)])$$

$$= \quad [\ (x > y) \rightarrow [(x > z) \rightarrow (y \leq x)] \land [(x < z) \rightarrow (y < z)]\ ]$$

$$\land [\neg (x > y) \rightarrow [(y > z) \rightarrow (x < y)] \land [(y < z) \rightarrow (x < z)]]$$

- which simplifies to

$$[\ (x > y) \rightarrow [(x < z) \rightarrow (y < z)]\ ]$$

$$\land [\ (x < y) \rightarrow [(y < z) \rightarrow (x < z)]\ ]$$

- which simplifies to

true