Weakest Precondition Calculus

Andrei Arusoaie

October 17, 2025

Hoare Logic - quick recap

- C. A. R. Hoare, 1969: `https://dl.acm.org/doi/pdf/10.1145/363235.363259`
- Hoare triples:
$$\{P\}\, S\, \{Q\}$$
- *Partial correctness*: If S is executed in a state satisfying P and the execution of S terminates then the resulted program state satisfies Q: $\models \{P\}S\{Q\}$.

Hoare Logic - quick recap

- C. A. R. Hoare, 1969:
`https://dl.acm.org/doi/pdf/10.1145/363235.363259`
- Hoare triples:
$$\{P\}\, S\, \{Q\}$$
- *Partial correctness*: If S is executed in a state satisfying P and the execution of S terminates then the resulted program state satisfies Q: $\models \{P\}S\{Q\}$.
- *Total correctness*: If S is executed in a state satisfying P then the execution of S terminates and the resulted program state satisfies Q: $\models [P]S[Q]$.

Summary of Hoare Logic Proof Rules

Assignment	$\dfrac{\cdot}{\vdash \{Q[e/x]\}\ \texttt{x := e}\ \{Q\}}$
Precondition Strengthening	$\dfrac{\vdash \{P'\}\ S\ \{Q\} \qquad P \rightarrow P'}{\vdash \{P\}\ S\ \{Q\}}$
Postcondition Weakening	$\dfrac{\vdash \{P\}\ S\ \{Q'\} \qquad Q' \rightarrow Q}{\vdash \{P\}\ S\ \{Q\}}$
Composition	$\dfrac{\vdash \{P\}\ S_1\ \{Q\} \qquad \vdash \{Q\}\ S_2\ \{R\}}{\vdash \{P\}\ S_1 ; S_2\ \{R\}}$
If	$\dfrac{\vdash \{P \wedge C\}\ S_1\ \{Q\} \qquad \vdash \{P \wedge \neg C\}\ S_2\ \{Q\}}{\vdash \{P\}\ \texttt{if}\ C\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{Q\}}$
While	$\dfrac{\vdash \{I \wedge C\}\ S\ \{I\}}{\vdash \{I\}\ \texttt{while}\ C\ \texttt{do}\ S\ \texttt{end}\ \{I \wedge \neg C\}}$

Soundness

- It can be shown that the proof rules for Hoare logic are sound:

$$\text{If } \vdash \{P\}S\{Q\}, \text{ then } \models \{P\}S\{Q\}.$$

- Completeness does not hold:

$$\text{If } \models \{P\}S\{Q\}, \text{ then } \vdash \{P\}S\{Q\}.$$

Automation

- Manual proofs of sequents are tedious.
- What can be automated?
  - The computation of the preconditions for assignments? Yes!
  - The invariants for loops? No, these are not always easy to find.
  - However, if someone else provides the invariants, many parts can be automated.
- *Weakest precondition calculus* is step towards automation.

IMP with Invariants for Loops

- **Arithmetic Expressions**

$$\text{AExp} ::= \text{Var} \mid \text{Int} \mid \text{AExp} + \text{AExp} \mid \text{AExp} \; / \; \text{AExp}$$

- **Conditionals:**

$$\text{BExp} ::= \textit{true} \mid \textit{false} \mid \text{AExp} < \text{AExp} \mid \textit{not}\ \text{BExp} \mid \text{BExp}\ \textit{and}\ \text{BExp}$$

- **Statements:**

$$\begin{aligned}\text{Stmt} ::= {} & \text{Var} := \text{AExp} \\ & \mid \textit{if}\ \text{BExp}\ \textit{then}\ \text{Stmt}\ \textit{else}\ \text{Stmt} \\ & \mid \textit{while}\ \text{BExp}\ \boxed{\text{inv: } \varphi}\ \textit{do}\ \text{Stmt}\ \textit{end} \\ & \mid \text{Stmt}\ ;\ \text{Stmt} \\ & \mid \textit{skip}\end{aligned}$$

Example Program in IMP with Invariants

The `sumPgm` program with invariant:

```
sum := 0;
i := 0;
while (i < n)
  inv: i ≤ n ∧ 2 * sum = i * (i + 1)
do
    i := i + 1
    sum := sum + i;
end
```

Proof Rule for While with Loop Invariants

Proof Rule for While with invariants is updated:

$$\frac{\vdash \{I \wedge C\} S \{I\}}{\vdash \{I\}\, \texttt{while}\, C\, \texttt{inv:}\, I\, \texttt{do}\, S\, \{I \wedge \neg C\}}$$

- Beware that completeness is completely lost: if the invariant is not strong enough, there is a high chance that some valid triples might not be derived.

Automation using Verification Conditions

- Automating Hoare logic is based on generating **verification conditions**.
- A **verification condition** (VC) is a formula, ψ, such that the program is correct if and only if ψ is valid.
- Two steps are needed:
  1. First, generate VCs from the source code.
  2. Second, use an automated tool to check the validity of the VCs.

Forwards vs. Backwards methods for generating VCs

- Two main approaches to VCs:
  1. **Forward:** Start from the precondition and generate formulas to prove the postcondition.
     - This computes the **strongest postconditions (sp)**.
  2. **Backward:** Start from the postcondition and works backwards to find the precondition.
     - This computes the **weakest preconditions (wp)**.
- Here we focus on the **weakest (liberal) precondition**.

Weakest Preconditions

- The *weakest precondition (wp)* for a statement S and postcondition Q is a formula $wp(S, Q)$ such that:
  - If $wp(S, Q)$ holds before executing S, then Q will hold after S finishes.
  - $wp(S, Q)$ is the *weakest* formula satisfying this, meaning any weaker precondition would fail to ensure Q after S.
- Calculating $wp(S, Q)$ depends on the type of statement S.
- We start with Q and going *backwards* we compute $wp(S, Q)$.
- The sequent $\vdash \{P\}S\{Q\}$ is valid iff $\models P \rightarrow wp(S, Q)$.

Weakest (Liberal) Precondition Calculus

- However, it si very difficult to compute the weakest precondition of loops, due to termination.
- Recall: If $wp(S, Q)$ holds before executing S, then Q will hold after S finishes.
- This is why we need some sort of function that does not rely on the fact that S terminates.
- We call this function: Weakest Liberal Precondition, shorthanded as *wlp*.

wlp for Assignment

For assignment, the weakest liberal precondition is defined by substituting the assigned variable with the expression:

$$\mathrm{wlp}(x := e, Q) = Q[e/x]$$

If $Q[e/x]$ is the precondition then Q holds after assigning e to x.

Example

Let the postcondition Q be $i \leq n \land 2 \cdot s = i \cdot (i+1)$.

- Example 1: $s := s + i$

  $\text{wlp}(s := s + i, Q) = (i \leq n) \land (2 \cdot \boxed{(\mathrm{s} + \mathrm{i})} = i \cdot (i+1))$

A quick note to avoid confusion:
$\vdash \text{wlp}(s := s + i, Q)\{s := s + i\}Q$ is valid because the precondition

$$(i \leq n) \land (2 \cdot (s + i) = i \cdot (i+1))$$

holds **before** executing the assignment; after the assignment, the value of s changes (i.e., it becomes $s + i$), so s holds the value of $s + i$. Therefore, $i \leq n \land 2 \cdot s = i \cdot (i+1)$ is now true for this new value of s.

Another Example

Let the postcondition Q' be $(i \leq n) \land (2 \cdot (s + i) = i \cdot (i + 1))$.

- Example 2: $i := i + 1$
  $$\text{wlp}(i := i + 1, Q') = (\mathrm{i} + 1 \leq n) \land$$
  $$(2 \cdot (s + (\mathrm{i} + 1)) = (\mathrm{i} + 1) \cdot ((\mathrm{i} + 1) + 1))$$

wlp for Sequential Composition

For sequential composition, where S_1 is followed by S_2:

$$\mathrm{wlp}(S_1; S_2, Q) = \mathrm{wlp}(S_1, \mathrm{wlp}(S_2, Q))$$

This calculates the weakest precondition by chaining the conditions backward through each statement.

Example

We want to compute $\mathrm{wlp}(i := i + 1; s := s + i, Q)$ with the postcondition $Q : i \leq n \wedge 2 \cdot s = i \cdot (i + 1)$.

- Step 1: Calculate $\mathrm{wlp}(s := s + i, Q)$

$$\mathrm{wlp}(s := s + i, Q) = (i \leq n) \wedge (2 \cdot (s + i) = i \cdot (i + 1)) = Q'$$

- Step 2: Compute $\mathrm{wlp}(i := i + 1, \mathrm{wlp}(s := s + i, Q)) = \mathrm{wlp}(i := i + 1, Q')$

$$\begin{aligned}\mathrm{wlp}(i := i + 1, Q') = {} & (i + 1 \leq n) \wedge \\ & (2 \cdot (s + (i + 1)) = (i + 1) \cdot ((i + 1) + 1))\end{aligned}$$

wlp for the Skip Statement

For the skip statement, the weakest liberal precondition is simply:

$$\text{wlp}(\texttt{skip}, Q) = Q$$

This means that executing ‘skip‘ does not alter Q.

wlp for Conditional Statements

For conditional statements, the wlp is defined as follows:

$$\text{wlp}(\texttt{if } C \texttt{ then } S_1 \texttt{ else } S_2, Q) =$$

$$(C \rightarrow \text{wlp}(S_1, Q)) \land (\neg C \rightarrow \text{wlp}(S_2, Q))$$

This considers both branches based on the truth value of C.

Example

We compute wlp(if $x < 0$ then $m := -x$ else $m := x$, Q) where Q is $m \geq 0$.

- For the ‘then‘ branch ($x < 0$):

$$\text{wlp}(m := -x, Q) = (m \geq 0)[-x/m] = -x \geq 0$$

- For the ‘else‘ branch ($x \geq 0$):

$$\text{wlp}(m := x, Q) = (m \geq 0)[x/m] = x \geq 0$$

- wlp(if $x < 0$ then $m := -x$ else $m := x$, Q) =
$$(x < 0 \rightarrow -x \geq 0) \wedge (x \geq 0 \rightarrow x \geq 0)$$

Example

We compute wlp(if $x < 0$ then $m := -x$ else $m := x$, Q) where Q is $m \geq 0$.

- For the 'then' branch ($x < 0$):

$$\text{wlp}(m := -x, Q) = (m \geq 0)[-x/m] = -x \geq 0$$

- For the 'else' branch ($x \geq 0$):

$$\text{wlp}(m := x, Q) = (m \geq 0)[x/m] = x \geq 0$$

- wlp(if $x < 0$ then $m := -x$ else $m := x$, Q) = $(x < 0 \to -x \geq 0) \land (x \geq 0 \to x \geq 0)$
- wlp(if $x < 0$ then $m := -x$ else $m := x$, Q) = true

So Q (i.e., $m \geq 0$) holds for both branches.

wlp for While Loops with Invariants

For while loops with an invariant I, the wlp is defined as:

$$\text{wlp}(\texttt{while } C \texttt{ inv:} I \texttt{ do } S, Q) =$$

$$I \wedge \forall x_1, \ldots, x_k . \Big(\big((C \wedge I) \rightarrow \text{wlp}(S, I) \big) \wedge \big((\neg C \wedge I) \rightarrow Q \big) \Big) [x_i / w_i]$$

where $w_1, \ldots, w_k$ are variables modified in S, and $x_1, \ldots, x_k$ are fresh variables.

Example

$$\mathrm{wlp}(\texttt{while}\ (\mathsf{i} < \mathsf{n})\ \texttt{inv:}\ (\mathsf{i} \leq \mathsf{n})\ \mathrm{do}\ i := i+1, \mathsf{i} = \mathsf{n}) = ?$$

$$\mathrm{wlp}(\texttt{while}\ (\mathsf{i} < \mathsf{n})\ \texttt{inv:}\ \mathsf{i} \leq \mathsf{n}\ \mathrm{do}\ i := i+1,\ \mathsf{i} = \mathsf{n}) =$$

$$\begin{aligned}
&= (\mathsf{i} \leq \mathsf{n}) \wedge \forall x. \Big(\big((\mathsf{i} < \mathsf{n} \wedge \mathsf{i} \leq \mathsf{n}) \to wlp(i := i+1, (\mathsf{i} \leq \mathsf{n}) \big) \big) \\
&\qquad\qquad \wedge \big((\neg \mathsf{i} < \mathsf{n} \wedge \mathsf{i} \leq \mathsf{n}) \to \mathsf{i} = \mathsf{n} \big) \Big) [x/i] \\
&= (\mathsf{i} \leq \mathsf{n}) \wedge \forall x. \Big(\big((\mathsf{x} < \mathsf{n} \wedge \mathsf{x} \leq \mathsf{n}) \to wlp(x := x+1, \mathsf{x} \leq \mathsf{n} \big) \big) \\
&\qquad\qquad \wedge \big((\neg \mathsf{x} < \mathsf{n} \wedge \mathsf{x} \leq \mathsf{n}) \to \mathsf{x} = \mathsf{n} \big) \Big) \\
&= (\mathsf{i} \leq \mathsf{n}) \wedge \forall x. \Big(\big((\mathsf{x} < \mathsf{n} \wedge \mathsf{x} \leq \mathsf{n}) \to (x+1 \leq n) \big) \big) \\
&\qquad\qquad \wedge \big((\neg \mathsf{x} < \mathsf{n} \wedge \mathsf{x} \leq \mathsf{n}) \to \mathsf{x} = \mathsf{n} \big) \Big) \\
&\equiv (\mathsf{i} \leq \mathsf{n}) \wedge \forall x. \Big(\big(x < n \to (x+1 \leq n) \big) \wedge \big(x = n \to \mathsf{x} = \mathsf{n} \big) \Big) \\
&\equiv (\mathsf{i} \leq \mathsf{n})
\end{aligned}$$

Summary

- $\mathrm{wlp}(x := e, Q) = Q[e/x]$
- $\mathrm{wlp}(S_1; S_2, Q) = \mathrm{wlp}(S_1, \mathrm{wlp}(S_2, Q))$
- $\mathrm{wlp}(\texttt{skip}, Q) = Q$
- $\mathrm{wlp}(\texttt{if}\ C\ \texttt{then}\ S_1\ \texttt{else}\ S_2, Q) =$
  $$(C \rightarrow \mathrm{wlp}(S_1, Q)) \wedge (\neg C \rightarrow \mathrm{wlp}(S_2, Q))$$
- $\mathrm{wlp}(\texttt{while}\ C\ \texttt{inv:}I\ \texttt{do}\ S, Q) =$

  $$I \wedge \forall x_1, \ldots, x_k. \Big(((C \wedge I) \rightarrow \mathrm{wlp}(S, I)) \wedge ((\neg C \wedge I) \rightarrow Q) \Big)[x_i/w_i]$$

  where $w_1, \ldots, w_k$ are variables modified in S, and $x_1, \ldots, x_k$ are fresh variables.

Important results

Theorem (Soundness)

For all statements S and postconditions Q, $\vdash \{wlp(S, Q)\}\, S\, \{Q\}$.

Important results

Theorem (Soundness)

For all statements S and postconditions Q, $\vdash \{wlp(S, Q)\}\, S\, \{Q\}$.

Key ideas:

- The proof is on structural induction on S.

Important results

Theorem (Soundness)

For all statements S and postconditions Q, $\vdash \{wlp(S, Q)\}\, S\, \{Q\}$.

Key ideas:

- The proof is on structural induction on S.
- For the case of the loop, an induction on the length of its execution is needed.

Important results

Theorem (Soundness)

For all statements S and postconditions Q, $\vdash \{wlp(S, Q)\}\, S\, \{Q\}$.

Key ideas:

- The proof is on structural induction on S.
- For the case of the loop, an induction on the length of its execution is needed.
- Also important, the interpretation of the universally quantified formula $\forall x_1, \ldots, x_k. \Big(\big((C \wedge I) \rightarrow \text{wlp}(S, I) \big) \wedge \big((\neg C \wedge I) \rightarrow Q \big) \Big)[x_i / w_i]$ does not depend on the values of the variables w_i, so the formula still holds for two consecutive program states.

Important results

Theorem (WLP property)

For any triple $\{P\}S\{Q\}$ *that is derivable using the proof rules (including the modified one for the loop), we have* $P \to wlp(S, Q)$.

Important results

Theorem (WLP property)

For any triple $\{P\}S\{Q\}$ that is derivable using the proof rules (including the modified one for the loop), we have $P \to wlp(S, Q)$.

Key ideas:

- The proof is on structural induction on S.

Important results

Theorem (WLP property)

For any triple $\{P\}S\{Q\}$ that is derivable using the proof rules (including the modified one for the loop), we have $P \rightarrow wlp(S, Q)$.

Key ideas:

- The proof is on structural induction on S.
- An important consequence of this theorem is the fact that, without loss of generality, we can look for a proof of $P \rightarrow \text{wlp}(S, Q)$ instead of finding a proof derivation for $\{P\}S\{Q\}$.

Important results

Theorem (WLP property)

For any triple $\{P\}S\{Q\}$ that is derivable using the proof rules (including the modified one for the loop), we have $P \rightarrow wlp(S, Q)$.

Key ideas:

- The proof is on structural induction on S.
- An important consequence of this theorem is the fact that, without loss of generality, we can look for a proof of $P \rightarrow \mathrm{wlp}(S, Q)$ instead of finding a proof derivation for $\{P\}S\{Q\}$.
- In order to prove $P \rightarrow \mathrm{wlp}(S, Q)$, various automatic proving tools can be employed.

Total correctness

- So far, we considered only partial correctness.

Total correctness

- So far, we considered only partial correctness.
- Total correctness needs, besides partial correctness, a proof for termination.
- *Total correctness*: If S is executed in a state satisfying P then the execution of S terminates and the resulted program state satisfies Q: $\models [P]S[Q]$.

Total correctness

- So far, we considered only partial correctness.
- Total correctness needs, besides partial correctness, a proof for termination.
- *Total correctness*: If S is executed in a state satisfying P then the execution of S terminates and the resulted program state satisfies Q: $\models [P]S[Q]$.
- Obviously, termination is hard to prove in the case of loops.

Total correctness

- So far, we considered only partial correctness.
- Total correctness needs, besides partial correctness, a proof for termination.
- *Total correctness*: If S is executed in a state satisfying P then the execution of S terminates and the resulted program state satisfies Q: $\models [P]S[Q]$.
- Obviously, termination is hard to prove in the case of loops.
- In fact, most of the Hoare logic rules remain unchanged when dealing with total correctness, except the rule for loops.

The Rule for Loops in the context of Total Correctness

$$\frac{\vdash \{I \wedge C \wedge v = \xi\}\, S\, \{I \wedge v \prec \xi\} \qquad wf(\prec)}{\vdash \{I\}\, \texttt{while}\, C\, \texttt{inv:}\, I\, \texttt{variant:}\, v\, \texttt{do}\, S\, \texttt{end}\, \{I \wedge \neg C\}}$$

Here, v is an expression called *variant*, and ξ is a fresh logical variable.
The meaning of $wf(\prec)$ is: $\prec$ is a *well-founded relation*, that is, there is no infinite sequence $\xi_1 \succ \xi_2 \succ \xi_3 \succ \cdots$.

An example of a well-founded relation on unbounded integers is:

$$x \prec y = x < y \wedge 0 \leq y.$$

IMP with Variants for Loops

- **Arithmetic Expressions**

$$\text{AExp} ::= \text{Var} \mid \text{Int} \mid \text{AExp} + \text{AExp} \mid \text{AExp} \,/\, \text{AExp}$$

- **Conditionals:**

$$\text{BExp} ::= \textit{true} \mid \textit{false} \mid \text{AExp} < \text{AExp} \mid \textit{not}\ \text{BExp} \mid \text{BExp}\ \textit{and}\ \text{BExp}$$

- **Statements:**

$$\begin{aligned}\text{Stmt} ::= {} & \text{Var} := \text{AExp} \\ & \mid \textit{if}\ \text{BExp}\ \textit{then}\ \text{Stmt}\ \textit{else}\ \text{Stmt} \\ & \mid \textit{while}\ \text{BExp}\ \boxed{\text{inv: } \varphi}\ \boxed{\text{variant: } \psi}\ \textit{do}\ \text{Stmt}\ \textit{end} \\ & \mid \text{Stmt}\ ;\ \text{Stmt} \\ & \mid \textit{skip}\end{aligned}$$

Once again the updated loop rule

$$\frac{\vdash \{I \wedge C \wedge v = \xi\}\, S\, \{I \wedge v \prec \xi\} \qquad wf(\prec)}{\vdash \{I\}\, \texttt{while}\ C\ \texttt{inv:}\, I\ \texttt{variant:}\, v\ \texttt{do}\ S\ \texttt{end}\, \{I \wedge \neg C\}}$$

Rules for Total Correctness (only While is changed)

Assignment	$\dfrac{\cdot}{\vdash \{Q[e/x]\}\ \texttt{x := e}\ \{Q\}}$
Precondition Strengthening	$\dfrac{\vdash \{P'\}\, S\, \{Q\} \qquad P \rightarrow P'}{\vdash \{P\}\, S\, \{Q\}}$
Postcondition Weakening	$\dfrac{\vdash \{P\}\, S\, \{Q'\} \qquad Q' \rightarrow Q}{\vdash \{P\}\, S\, \{Q\}}$
Composition	$\dfrac{\vdash \{P\}\, S_1\, \{Q\} \qquad \vdash \{Q\}\, S_2\, \{R\}}{\vdash \{P\}\, S_1 ; S_2\, \{R\}}$
If	$\dfrac{\vdash \{P \wedge C\}\, S_1\, \{Q\} \qquad \vdash \{P \wedge \neg C\}\, S_2\, \{Q\}}{\vdash \{P\}\ \texttt{if}\ C\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{Q\}}$
While	$\dfrac{\vdash \{I \wedge C \wedge v = \xi\}\, S\, \{I \wedge v \prec \xi\} \qquad wf(\prec)}{\vdash \{I\}\ \texttt{while}\ C\ \texttt{inv:}\, I\ \texttt{variant:}\, v\ \texttt{do}\ S\ \texttt{end}\ \{I \wedge \neg C\}}$

Weakest (Strict) Precondition

- $\mathrm{wp}(x := e, Q) = Q[e/x]$
- $\mathrm{wp}(S_1; S_2, Q) = \mathrm{wp}(S_1, \mathrm{wp}(S_2, Q))$
- $\mathrm{wp}(\texttt{skip}, Q) = Q$
- $\mathrm{wp}(\texttt{if}\ \ C\ \ \texttt{then}\ \ S_1\ \ \texttt{else}\ \ S_2, Q) =$
  $$(C \to \mathrm{wp}(S_1, Q)) \land (\neg C \to \mathrm{wp}(S_2, Q))$$
- $\mathrm{wp}(\texttt{while}\ \ C\ \ \texttt{inv:}I\,\texttt{variant}: v\ \ \texttt{do}\ \ S, Q) =$
  $$I \land \forall x_1, \ldots, x_k, \xi. \Big(((C \land I \land \xi = v) \to \mathrm{wp}(S, I \land v \prec \xi)) \land ((\neg C \land I) \to Q) \Big)[x_i/w_i],$$
  where $w_1, \ldots, w_k$ are variables modified in s, and $x_1, \ldots, x_k, \xi$ are fresh variables.

Example

Let `incToN` be:

```
while i < n
  inv: i <= n
  variant: n - i
do
  i := i + 1
```

What is $wp(\texttt{incToN}, \mathsf{i} = \mathsf{n})$?

Example

```
while i < n
  inv: i ≤ n
  variant: n - i
do
  i := i + 1
```

$$wp(\texttt{incToN}, \mathsf{i} = \mathsf{n}) =$$
$$(\mathsf{i} \leq \mathsf{n}) \wedge$$
$$\forall x, \xi. \Big(((\mathsf{i} < \mathsf{n} \wedge \mathsf{i} \leq \mathsf{n} \wedge \xi = \mathsf{n - i}) \rightarrow wp(i := i{+}1, \mathsf{i} \leq \mathsf{n} \wedge \mathsf{n - i} \prec \xi))$$
$$\wedge (\neg(\mathsf{i} < \mathsf{n}) \wedge \mathsf{i} \leq \mathsf{n} \rightarrow \mathsf{i} = \mathsf{n})\Big)[x/i]$$

Example - continued

$$
\begin{aligned}
& wp(\texttt{incToN}, \text{i} = \text{n}) = \\
& (\text{i} \leq \text{n}) \wedge \\
& \forall x, \xi. \Big(((\text{i} < \text{n} \wedge \text{i} \leq \text{n} \wedge \xi = \text{n - i}) \to wp(i := i{+}1, \text{i} \leq \text{n} \wedge \text{n - i} \prec \xi)) \\
& \qquad \wedge (\neg(\text{i} < \text{n}) \wedge \text{i} \leq \text{n} \to \text{i} = \text{n}) \Big) [x/i]
\end{aligned}
$$

Recall that $x \prec y = x < y \wedge 0 \leq y$. So, we compute:

$$
\begin{aligned}
& wp(i := i + 1, \text{i} \leq \text{n} \wedge \text{n - i} \prec \xi) = \\
& wp(i := i + 1, \text{i} \leq \text{n} \wedge \text{n - i} < \xi \wedge 0 \leq \xi) =
\end{aligned}
$$

$$= i + 1 \leq n \wedge n - (i + 1) < \xi \wedge 0 \leq \xi$$

Example - continued

$wp(\texttt{incToN}, \text{i = n}) =$

$(\text{i} \leq \text{n}) \wedge$

$\forall x, \xi . \Big(((\text{i < n} \wedge \text{i} \leq \text{n} \wedge \xi = \text{n - i}) \rightarrow wp(i := i + 1, \text{i} \leq \text{n} \wedge \text{n - i} \prec \xi))$

$\wedge (\neg(\text{i < n}) \wedge \text{i} \leq \text{n} \rightarrow \text{i = n})\Big)[x/i] =$

$(\text{i} \leq \text{n}) \wedge$

$\forall x, \xi . \Big(((\text{i < n} \wedge \text{i} \leq \text{n} \wedge \xi = \text{n - i}) \rightarrow (i + 1 \leq n \wedge n - (i + 1) < \xi \wedge 0 \leq \xi))$

$\wedge (\neg(\text{i < n}) \wedge \text{i} \leq \text{n} \rightarrow \text{i = n})\Big)[x/i] =$

$(\text{i} \leq \text{n}) \wedge$

$\forall x, \xi . \Big(((\text{x < n} \wedge \text{x} \leq \text{n} \wedge \xi = \text{n - x}) \rightarrow (x + 1 \leq n \wedge n - (x + 1) < \xi \wedge 0 \leq \xi))$

$\wedge (\neg(\text{x < n}) \wedge \text{x} \leq \text{n} \rightarrow \text{x = n})\Big) \equiv$

$(\text{i} \leq \text{n}) \wedge \forall x, \xi . \Big(((\text{x < n} \wedge \xi = \text{n - x}) \rightarrow (x + 1 \leq n \wedge n - (x + 1) < \xi \wedge 0 \leq \xi)) \wedge (x = n \rightarrow \text{x = n})\Big)$

Which can be simplified to:

$(\text{i} \leq \text{n}) \wedge \forall x, \xi . \Big(((\text{x < n} \wedge \xi = \text{n - x}) \rightarrow (x + 1 \leq n \wedge n - (x + 1) < \xi \wedge 0 \leq \xi))\Big)$

Example - continued

$wp(\texttt{incToN}, \text{i = n}) =$
$(\text{i} \leq \text{n}) \wedge \forall x, \xi. \Big(((\text{x < n} \wedge \xi = \text{n - x}) \rightarrow (x + 1 \leq n \wedge n - (x + 1) < \xi \wedge 0 \leq \xi))\Big)$

But:

- $x < n$ implies $x + 1 \leq n$.
- Since $\xi = n - x$, we have $n - (x + 1) < \xi \equiv n - (x + 1) < n - x \equiv (n - x) - 1 < (n - x) \equiv -1 < 0$.
- Also, $x < n \equiv 0 < n - x$. Since $\xi = n - x$ we have $0 < \xi$ which implies $0 \leq \xi$.

Note that here we used the variant to prove termination!

Therefore, the universally quantified formula is true, so we have:

$$wp(\texttt{incToN}, \text{i = n}) = \text{i} \leq \text{n}.$$

Properties

Theorem (Soundness)

For all statements S and postconditions Q, $\vdash \{wp(S, Q)\}S\{Q\}$.

Properties

Theorem (Soundness)

For all statements S and postconditions Q, $\vdash \{wp(S, Q)\}S\{Q\}$.

Theorem

For any triple $\{P\}S\{Q\}$ *that is derivable using the proof rules (including the modified one for the loop) we have* $P \to wp(S, Q)$.

Properties

Theorem (Soundness)

For all statements S and postconditions Q, $\vdash \{wp(S, Q)\}S\{Q\}$.

Theorem

For any triple $\{P\}S\{Q\}$ that is derivable using the proof rules (including the modified one for the loop) we have $P \rightarrow wp(S, Q)$.

As a consequence, for proving that a triple $\{P\}S\{Q\}$ is valid (for total correctness), we can without loss of generality prove the formula $\models P \rightarrow \mathrm{wp}(S, Q)$.

Summary – Weakest Precondition Calculus

- *Weakest Liberal Precondition (WLP)*:
  - Definition: The $\text{wlp}(S, Q)$ provides the *weakest condition* P such that if P holds before executing statement S, Q will hold after execution, provided S terminates.
  - Usage: Focuses on *partial correctness*, ensuring postcondition Q is true if S terminates.
- *Weakest Precondition (WP)*:
  - Definition: The $\text{wp}(S, Q)$ provides the *weakest condition* P ensuring that Q holds after S, accounting for both *termination and correctness*.
  - Usage: Ensures *total correctness*, as it demands both termination and the truth of Q.