Verification Conditions

Problems with HL System

- Two desirable properties for backward proof construction are missing:
  - Sub-formula property
  - Unambiguous choice of rule
- The consequence rule causes ambiguity. Its presence is however necessary to make possible the application of rules for *skip*, *assignment*, and *while*
- An alternative is to *distribute* the side conditions among the different rules

HL without Consequ. Rule

$$\frac{}{\{P\}\,\mathbf{skip}\,\{Q\}} \quad \text{if } \models P \to Q \qquad\qquad \frac{}{\{P\}\,x := e\,\{Q\}} \quad \text{if } \models P \to Q[x \mapsto e]$$

$$\frac{\{P\}\,C_1\,\{R\} \qquad \{R\}\,C_2\,\{Q\}}{\{P\}\,C_1\,;\,C_2\,\{Q\}}$$

$$\frac{\{I\ \&\&\ b\}\,C\,\{I\}}{\{P\}\,\mathbf{while}\ b\ \mathbf{do}\,\{I\}\,C\,\{Q\}} \quad \text{if } \models P \to I \ \text{ and } \models I\ \&\&\ !b \to Q$$

$$\frac{\{P\ \&\&\ b\}\,C_t\,\{Q\} \qquad \{P\ \&\&\ !b\}\,C_f\,\{Q\}}{\{P\}\,\mathbf{if}\ b\ \mathbf{then}\ C_t\ \mathbf{else}\ C_f\,\{Q\}}$$

Factorial Example

$$
\begin{aligned}
&\textbf{fact} \doteq \\
&\quad f := 1\,;\ i := 1\,; \\
&\quad \textbf{while}\ i \le n\ \textbf{do}\ \{f == \mathit{fact}(i-1)\ \&\&\ i \le n+1\} \\
&\qquad f := f * i\,; \\
&\qquad i := i + 1
\end{aligned}
$$

$\{n \ge 0\}\ \textbf{fact}\ \{f == \mathit{fact}(n)\}$

1. $\{n \ge 0\}\ f := 1\,;\ i := 1\ \{n \ge 0\ \&\&\ f == 1\ \&\&\ i == 1\}$

   1.1 $\{n \ge 0\}\ f := 1\ \{n \ge 0\ \&\&\ f == 1\}$

   1.2 $\{n \ge 0\ \&\&\ f == 1\}\ i := 1\ \{n \ge 0\ \&\&\ f == 1\ \&\&\ i == 1\}$

2. $\{n \ge 0\ \&\&\ f == 1\ \&\&\ i == 1\}\ \textbf{while}\ i \le n\ \textbf{do}\ \{f == \mathit{fact}(i-1)\ \&\&\ i \le n+1\}\ C_b\ \{f == \mathit{fact}(n)\}$

   2.1. $\{f == \mathit{fact}(i-1)\ \&\&\ i \le n\}\ C_b\ \{f == \mathit{fact}(i-1)\ \&\&\ i \le n+1\}$

   2.1.1. $\{f == \mathit{fact}(i-1)\ \&\&\ i \le n\}\ f := f * i\ \{f == \mathit{fact}(i-1) * i\ \&\&\ i \le n\}$

   2.1.2. $\{f == \mathit{fact}(i-1) * i\ \&\&\ i \le n\}\ i := i+1\ \{f == \mathit{fact}(i-1)\ \&\&\ i \le n+1\}$

with side conditions:

$$1.1 \ \models n \geq 0 \ \rightarrow \ (n \geq 0 \,\&\&\, f == 1)[f \mapsto 1]$$

$$1.2 \ \models n \geq 0 \,\&\&\, f == 1 \ \rightarrow \ (n \geq 0 \,\&\&\, f == 1 \,\&\&\, i == 1)[i \mapsto 1]$$

$$2. \ \models n \geq 0 \,\&\&\, f == 1 \,\&\&\, i == 1 \ \rightarrow \ f == \mathit{fact}(i-1) \,\&\&\, i \leq n+1 \text{ and}$$
$$\models f == \mathit{fact}(i-1) \,\&\&\, i \leq n+1 \,\&\&\, !i \leq n \ \rightarrow \ f == \mathit{fact}(n)$$

$$2.1.1. \ \models f == \mathit{fact}(i-1) \,\&\&\, i \leq n \ \rightarrow \ (f == \mathit{fact}(i-1) * i \,\&\&\, i \leq n)[f \mapsto f * i]$$

$$2.1.2. \ \models f == \mathit{fact}(i-1)*i \,\&\&\, i \leq n \ \rightarrow \ (f == \mathit{fact}(i-1) \,\&\&\, i \leq n+1)[i \mapsto i+1]$$

Exercise

- Show that a triple is provable in this system iff it is provable in the original system of Hoare logic.

A Strategy for Proofs

- Focus on the command and postcondition;
  guess an appropriate precondition
- In the sequence rule, we obtain the intermediate condition from the postcondition of the second command
- We do this by always choosing the *weakest* precondition (for the given postcondition)
- i.e., in rules for skip, assignment, and while, the precondition is determined by looking at the side condition and choosing the weakest condition that satisfies it

A Strategy for Proofs

Example:

$$\{P\}\, x := e_1 \,;\, y := e_2 \,;\, z := e_3 \,\{Q\}$$

$\boxed{1.}$ $\{P\}\, x := e_1 \,;\, y := e_2 \,\{R\}$

$\boxed{2.}$ $\{R\}\, z := e_3 \,\{Q\}$

A Strategy for Proofs

$$\{P\}\, x := e_1\, ;\, y := e_2\, ;\, z := e_3\, \{Q\}$$

$\boxed{1.}$ $\{P\}\, x := e_1\, ;\, y := e_2\, \{Q[z \mapsto e_3]\}$

2. $\{Q[z \mapsto e_3]\}\, z := e_3\, \{Q\}$

A Strategy for Proofs

$$\{P\}\, x := e_1\, ;\, y := e_2\, ;\, z := e_3\, \{Q\}$$

1. $\{P\}\, x := e_1\, ;\, y := e_2\, \{Q[z \mapsto e_3]\}$
   1.1. $\{P\}\, x := e_1\, \{Q[z \mapsto e_3][y \mapsto e_2]\}$,
   1.2. $\{Q[z \mapsto e_3][y \mapsto e_2]\}\, y := e_2\, \{Q[z \mapsto e_3]\}$
2. $\{Q[z \mapsto e_3]\}\, z := e_3\, \{Q\}$

In step 1.1 we are not free to choose the precondition and thus a *side condition* must be satisfied:

$$\models P \rightarrow Q[z \mapsto e_3][y \mapsto e_2][x \mapsto e_1]$$

Exercise

- Use the weakest precondition strategy to verify Factorial

$$\{n \geq 0\}\,\textbf{fact}\,\{f == fact(n)\}$$

$\boxed{1.}$ $\{n \geq 0\}\, f := 1\,;\, i := 1\,\{?_1\}$

$\boxed{2.}$ $\{?_1\}\,\textbf{while}\ i \leq n\ \textbf{do}\,\{f == fact(i-1)\ \&\&\, i \leq n+1\}\, C_w\,\{f == fact(n)\}$

$$\{n \geq 0\}\,\textbf{fact}\,\{f == fact(n)\}$$

$\boxed{1.}$ $\{n \geq 0\}\, f := 1\,;\, i := 1\,\{f == fact(i-1)\ \&\&\, i \leq n+1\}$

$\boxed{2.}$ $\{f == fact(i-1)\ \&\&\, i \leq n+1\}\,\textbf{while}\ i \leq n\ \textbf{do}\,\{f == fact(i-1)\ \&\&\, i \leq n+1\}\, C_w\,\{f == fact(n)\}$

$\boxed{2.1.}$ $\{f == fact(i-1)\ \&\&\, i \leq n+1\ \&\&\, i \leq n\}\, C_w\,\{f == fact(i-1)\ \&\&\, i \leq n+1\}$

side condition (OK):

$$\models f == fact(i-1)\ \&\&\, i \leq n+1\ \&\&\ !(i \leq n)\ \rightarrow\ f == fact(n)$$

$\{n \geq 0\}\,\mathbf{fact}\,\{f == fact(n)\}$

$\boxed{1.}$ $\{n \geq 0\}\, f := 1\,;\, i := 1\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

$\boxed{2.}$ $\{f == fact(i-1)\ \&\&\, i \leq n+1\}\,\mathbf{while}\ i \leq n\ \mathbf{do}\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}\, C_w\, \{f == fact(n)\}$

$\boxed{2.1.}$ $\{f == fact(i-1)\ \&\&\, i \leq n\}\, C_w\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

$\boxed{2.1.1.}$ $\{f == fact(i-1)\ \&\&\, i \leq n\}\, f := f * i\, \{?_2\}$

$\boxed{2.1.2.}$ $\{?_2\}\, i := i+1\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

$\{n \geq 0\}\, \mathbf{fact}\, \{f == fact(n)\}$

$\boxed{1.}$ $\{n \geq 0\}\, f := 1\, ;\, i := 1\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

$\boxed{2.}$ $\{f == fact(i-1)\ \&\&\, i \leq n+1\}\, \mathbf{while}\ i \leq n\ \mathbf{do}\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}\, C_w\, \{f == fact(n)\}$

$\boxed{2.1.}$ $\{f == fact(i-1)\ \&\&\, i \leq n\}\, C_w\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

$\boxed{2.1.1.}$ $\{f == fact(i-1)\ \&\&\, i \leq n\}\, f := f * i\, \{f == fact(i)\ \&\&\, i \leq n\}$

2.1.2. $\{f == fact(i)\ \&\&\, i \leq n\}\, i := i+1\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

side condition for 2.1.1 (OK):

$$\models f == fact(i-1)\ \&\&\, i \leq n \rightarrow (f == fact(i)\ \&\&\, i \leq n)[f \mapsto f * i]$$

$\{n \geq 0\}$ **fact** $\{f == fact(n)\}$

$\boxed{1.}$ $\{n \geq 0\}\, f := 1\,;\, i := 1\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

$\boxed{1.1}$ $\{n \geq 0\}\, f := 1\, \{?_3\}$

$\boxed{1.2}$ $\{?_3\}\, i := 1\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

1. $\{f == fact(i-1)\ \&\&\, i \leq n+1\}$ **while** $i \leq n$ **do** $\{f == fact(i-1)\ \&\&\, i \leq n+1\}\, C_w\, \{f == fact(n)\}$

   2.1. $\{f == fact(i-1)\ \&\&\, i \leq n\}\, C_w\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

      2.1.1. $\{f == fact(i-1)\ \&\&\, i \leq n\}\, f := f * i\, \{f == fact(i)\ \&\&\, i \leq n\}$

      2.1.2. $\{f == fact(i)\ \&\&\, i \leq n\}\, i := i+1\, \{f == fact(i-1)\ \&\&\, i \leq n+1\}$

$\{n \geq 0\}\,\mathbf{fact}\,\{f == \mathit{fact}(n)\}$

1. $\{n \geq 0\}\, f := 1\, ;\, i := 1\, \{f == \mathit{fact}(i-1)\ \&\&\, i \leq n+1\}$

   1.1 $\{n \geq 0\}\, f := 1\, \{f == \mathit{fact}(0)\ \&\&\, 1 \leq n+1\}$

   1.2 $\{f == \mathit{fact}(0)\ \&\&\, 1 \leq n+1\}\, i := 1\, \{f == \mathit{fact}(i-1)\ \&\&\, i \leq n+1\}$

2. $\{f == \mathit{fact}(i-1)\ \&\&\, i \leq n+1\}\,\mathbf{while}\ i \leq n\ \mathbf{do}\,\{f == \mathit{fact}(i-1)\ \&\&\, i \leq n+1\}\, C_w\, \{f == \mathit{fact}(n)\}$

   2.1. $\{f == \mathit{fact}(i-1)\ \&\&\, i \leq n\}\, C_w\, \{f == \mathit{fact}(i-1)\ \&\&\, i \leq n+1\}$

      2.1.1. $\{f == \mathit{fact}(i-1)\ \&\&\, i \leq n\}\, f := f * i\, \{f == \mathit{fact}(i)\ \&\&\, i \leq n\}$

      2.1.2. $\{f == \mathit{fact}(i)\ \&\&\, i \leq n\}\, i := i+1\, \{f == \mathit{fact}(i-1)\ \&\&\, i \leq n+1\}$

side condition for 1.1 (OK):

$$\models n \geq 0 \;\rightarrow\; (f == \mathit{fact}(0)\ \&\&\, 1 \leq n+1)[f \mapsto 1]$$

P. V. Architectures

How can a proof tool be used for verifying programs with Hoare Logic using the Weakest Preconditions strategy?
Two possibilities:

- Encode Hoare Logic directly in proof tool and reason about program constructs

- Two-phase architecture:
  (i) use Hoare Logic to construct a set of verification conditions
  (ii) use a general-purpose proof tool to discharge verification conditions

Second approach is much more flexible

Verification Conditions

- VCs are purely first-order, not containing program constructs.
- Can be checked / discharged using any standard proof tool (theorem prover or proof assistant) with support for the data types of the language.
- Modifications in the language are only reflected in the first component, not in the proof tool
- Moreover it is possible to use a multi-prover approach (will be exemplified with Frama-c / Why)

Two-phase Architecture

1. Given a Hoare triple {P} C {Q}, we mechanically produce a derivation with {P} C {Q} as conclusion, assuming that all its side conditions are valid.

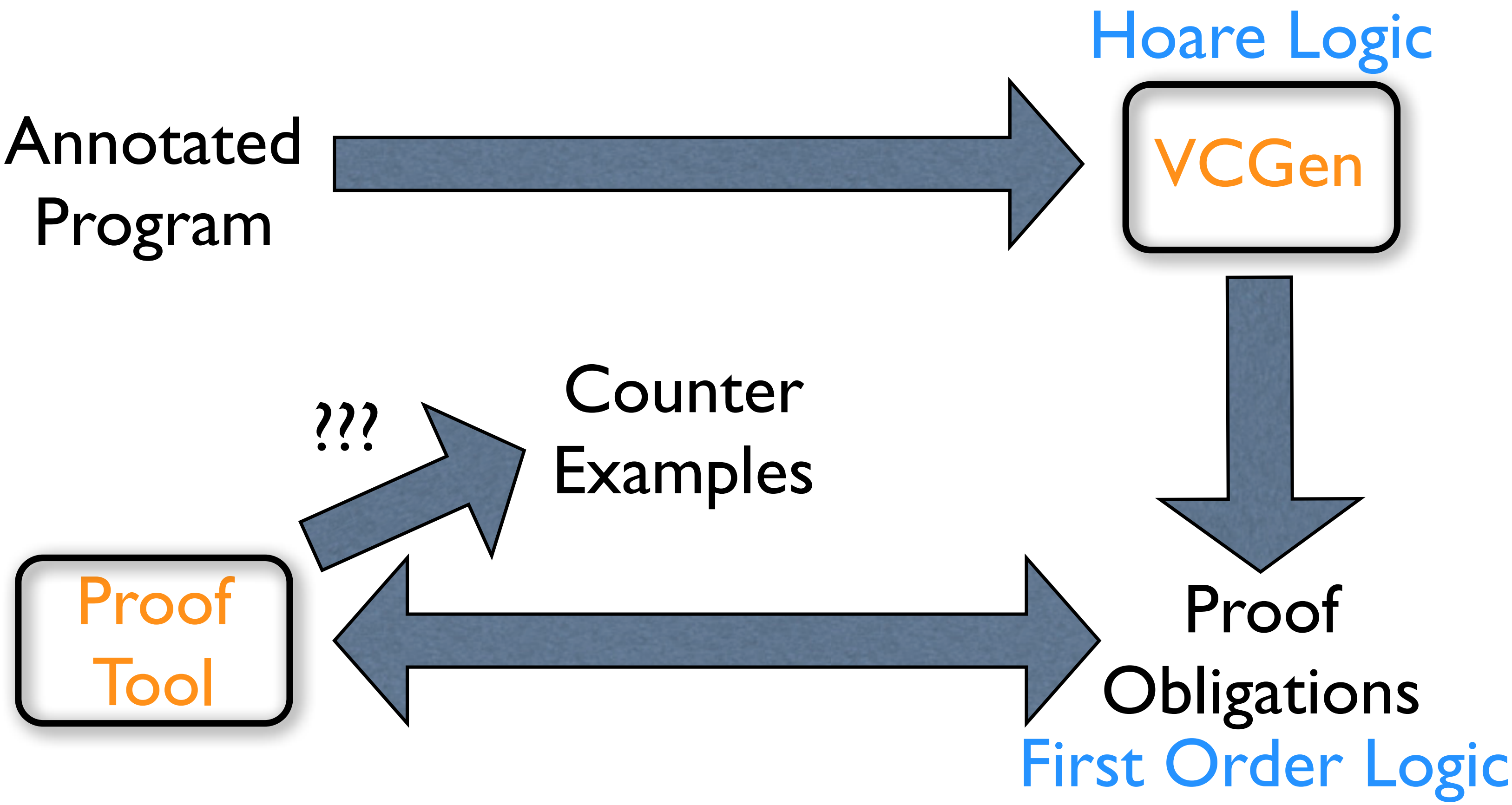
2. Each side condition generated in step 1 must now be checked. To that effect, a first-order formula [$A \rightarrow B$] is exported to a proof tool. Such a formula is called a verification condition (VC).

3. If all verification conditions can be proved valid, then {P} C {Q} is a valid Hoare triple. If at least one condition is shown not to be valid, then this is evidence that the triple is also not valid.

Question

- Note that the HL "proof tree" can always be constructed (explicitly or virtually)
- But the VCs may not all be dischargeable: automatic prover may be able to find a counter-example… or interactive proof may not suceed
- *What does it mean* when at least one VC is not valid? (the verification of the program has *failed)*
  Errors in *program*, *specification*, or *annotations*

Two-phase Architecture



An Architecture for Verification

- Our next step is then to mechanize the construction of a derivation, following the WP strategy.
- The result will be an algorithm (called a Verification Conditions Generator, VCGen) that does not even explicitly construct the proof tree; it just outputs the set of verification conditions

Weakest Preconds. Mechanized

Given program C and a postcondition Q, we can calculate an assertion wp(C,Q) such that {wp(C,Q)} C {Q} is valid

and moreover

if {P} C {Q} is valid for some P then P is stronger than wp(C,Q).

Thus wp(C,Q) is the *weakest precondition* that grants the truth of postcondition Q after execution of C.

Try guessing the definition of wp for a few language constructs…

Question

Can the weakest precondition of a loop be calculated statically?

Not really, all the reasoning depends on being able to find an appropriate *invariant*!

For this reason we *annotate* each loop with an invariant, which can be seen as the weakest precondition required to prove *any* postcondition

Weakest Precond. Algorithm

$$
\begin{array}{rcl}
\mathsf{wp}\,(\mathbf{skip}, Q) & = & Q \\
\mathsf{wp}\,(x := e, Q) & = & Q[x \mapsto e] \\
\mathsf{wp}\,(C_1; C_2, Q) & = & \mathsf{wp}\,(C_1, \mathsf{wp}\,(C_2, Q)) \\
\mathsf{wp}\,(\mathbf{if}\ b\ \mathbf{then}\ C_t\ \mathbf{else}\ C_f, Q) & = & (b\ \rightarrow\ \mathsf{wp}\,(C_t, Q))\ \&\&\ (\ !b\ \rightarrow\ \mathsf{wp}\,(C_f, Q)) \\
\mathsf{wp}\,(\mathbf{while}\ b\ \mathbf{do}\ \{I\}\ C, Q) & = & I
\end{array}
$$

VCGen Algorithm

$$
\begin{array}{rcl}
\mathsf{VC}_{\mathsf{aux}}(\mathbf{skip}, Q) & = & \emptyset \\
\mathsf{VC}_{\mathsf{aux}}(x := e, Q) & = & \emptyset \\
\mathsf{VC}_{\mathsf{aux}}(C_1; C_2, Q) & = & \mathsf{VC}_{\mathsf{aux}}(C_1, \mathsf{wp}\,(C_2, Q)) \ \cup \ \mathsf{VC}_{\mathsf{aux}}(C_2, Q) \\
\mathsf{VC}_{\mathsf{aux}}(\mathbf{if}\ b\ \mathbf{then}\ C_t\ \mathbf{else}\ C_f, Q) & = & \mathsf{VC}_{\mathsf{aux}}(C_t, Q) \ \cup \ \mathsf{VC}_{\mathsf{aux}}(C_f, Q) \\
\mathsf{VC}_{\mathsf{aux}}(\mathbf{while}\ b\ \mathbf{do}\ \{I\}\, C, Q) & = & \{[\ (I\ \&\&\ b) \rightarrow \mathsf{wp}\,(C, I)\ ], [\ (I\ \&\&\ !b) \rightarrow Q\ ]\} \\
 & & \cup\ \mathsf{VC}_{\mathsf{aux}}(C, I)
\end{array}
$$

$$
\mathsf{VCG}(\{P\}\, C\, \{Q\}) \quad = \quad \{[\ P \rightarrow \mathsf{wp}\,(C, Q)\ ]\} \ \cup \ \mathsf{VC}_{\mathsf{aux}}(C, Q)
$$

Correctness of VCGen

Let $C \in$ Comm and $P, Q \in$ Assert such that
$\models$ VCG({P} C {Q}), i.e. all verification conditions are valid.

Then {P} C {Q} is derivable in the system of (goal-directed) Hoare logic.

This is proved by showing that there exists a derivation whose side conditions are exactly those calculated by VCG({P} C {Q}).

Example: Factorial

$$
\begin{aligned}
& \mathsf{VC}_{\mathsf{aux}}(\mathbf{fact}, f == \mathit{fact}(n)) \\
= \quad & \mathsf{VC}_{\mathsf{aux}}(f := 1\,;\, i := 1, \mathsf{wp}\,(\mathbf{while}\ i \le n\ \mathbf{do}\,\{I\}\,C_w, f == \mathit{fact}(n))) \\
& \cup\ \mathsf{VC}_{\mathsf{aux}}(\mathbf{while}\ i \le n\ \mathbf{do}\,\{I\}\,C_w, f == \mathit{fact}(n)) \\
= \quad & \mathsf{VC}_{\mathsf{aux}}(f := 1\,;\, i := 1, I) \\
& \cup\ \{[\ I\ \&\&\, i \le n\ \rightarrow\ \mathsf{wp}\,(C_w, I)\ ]\} \\
& \cup\ \{[\ I\ \&\&\, i > n\ \rightarrow\ f == \mathit{fact}(n)\ ]\} \\
& \cup\ \mathsf{VC}_{\mathsf{aux}}(C_w, I) \\
= \quad & \mathsf{VC}_{\mathsf{aux}}(f := 1, \mathsf{wp}\,(i := 1, I)) \cup\ \mathsf{VC}_{\mathsf{aux}}(i := 1, I) \\
& \cup\ \{[\ f == \mathit{fact}(i-1)\ \&\&\, i \le n+1\ \&\&\, i \le n \\
& \qquad \rightarrow\ \mathsf{wp}\,(f := f * i, \mathsf{wp}\,(i := i+1, I))\ ]\} \\
& \cup\ \{[\ f == \mathit{fact}(i-1)\ \&\&\, i \le n+1\ \&\&\, i > n\ \rightarrow\ f == \mathit{fact}(n)\ ]\} \\
& \cup\ \mathsf{VC}_{\mathsf{aux}}(f := f * i, \mathsf{wp}\,(i := i+1, I)) \cup\ \mathsf{VC}_{\mathsf{aux}}(i := i+1, I)
\end{aligned}
$$

$$
\begin{array}{rl}
= & \emptyset \cup \emptyset \\
& \cup\ \{[\ f == fact(i-1)\ \&\&\, i \leq n+1\ \&\&\, i \leq n \\
& \qquad \rightarrow\ \mathsf{wp}\,(f := f * i, f == fact(i+1-1)\ \&\&\, i+1 \leq n+1)\ ]\} \\
& \cup\ \{[\ f == fact(i-1)\ \&\&\, i \leq n+1\ \&\&\, i > n\ \rightarrow\ f == fact(n)\ ]\} \\
& \cup\ \emptyset \cup \emptyset \\
& \\
= & \{[\ f == fact(i-1)\ \&\&\, i \leq n+1\ \&\&\, i \leq n \\
& \qquad \rightarrow\ f * i == fact(i+1-1)\ \&\&\, i+1 \leq n+1\ ], \\
& [\ f == fact(i-1)\ \&\&\, i \leq n+1\ \&\&\, i > n\ \rightarrow\ f == fact(n)\ ]\}
\end{array}
$$

$$\mathsf{VCG}(\{n \geq 0\}\,\mathbf{fact}\,\{f == \mathit{fact}(n)\})$$

$$\begin{aligned}
= \quad & [\, n \geq 0 \;\rightarrow\; \mathsf{wp}\,(\mathbf{fact}, f == \mathit{fact}(n))\,] \cup\; \mathsf{VC}_{\mathsf{aux}}(\mathbf{fact}, f == \mathit{fact}(n)) \\[1em]
= \quad & [\, n \geq 0 \;\rightarrow\; \mathsf{wp}\,(f := 1\,;\, i := 1, \mathsf{wp}\,(\mathbf{while}\; i \leq n\; \mathbf{do}\,\{I\}\,C_w, f == \mathit{fact}(n)))\,] \\
& \cup \{[\, f == \mathit{fact}(i-1)\;\&\&\, i \leq n+1\;\&\&\, i \leq n \\
& \qquad \rightarrow\; f * i == \mathit{fact}(i+1-1)\;\&\&\, i+1 \leq n+1\,], \\
& [\, f == \mathit{fact}(i-1)\;\&\&\, i \leq n+1\;\&\&\, i > n \;\rightarrow\; f == \mathit{fact}(n)\,]\} \\[1em]
= \quad & \{[\, n \geq 0 \;\rightarrow\; \mathsf{wp}\,(f := 1\,;\, i := 1, I)\,], \\
& [\, f == \mathit{fact}(i-1)\;\&\&\, i \leq n+1\;\&\&\, i \leq n \\
& \qquad \rightarrow\; f * i == \mathit{fact}(i+1-1)\;\&\&\, i+1 \leq n+1\,], \\
& [\, f == \mathit{fact}(i-1)\;\&\&\, i \leq n+1\;\&\&\, i > n \;\rightarrow\; f == \mathit{fact}(n)\,]\} \\[1em]
= \quad & \{[\, n \geq 0 \;\rightarrow\; 1 == \mathit{fact}(1-1)\;\&\&\, 1 \leq n+1\,], \\
& [\, f == \mathit{fact}(i-1)\;\&\&\, i \leq n+1\;\&\&\, i \leq n \\
& \qquad \rightarrow\; f * i == \mathit{fact}(i+1-1)\;\&\&\, i+1 \leq n+1\,], \\
& [\, f == \mathit{fact}(i-1)\;\&\&\, i \leq n+1\;\&\&\, i > n \;\rightarrow\; f == \mathit{fact}(n)\,]\}
\end{aligned}$$

Expanding the universal closures:

1. $\texttt{Forall}\ n.\ (n \geq 0 \ \rightarrow\ 1 == \mathit{fact}(1-1)\ \&\&\ 1 \leq n+1)$
2. $\texttt{Forall}\ i, n.\ (f == \mathit{fact}(i-1)\ \&\&\ i \leq n+1\ \&\&\ i \leq n$
   $\rightarrow\ f * i == \mathit{fact}(i+1-1)\ \&\&\ i+1 \leq n+1)$
3. $\texttt{Forall}\ i, f, n.\ (f == \mathit{fact}(i-1)\ \&\&\ i \leq n+1\ \&\&\ i > n\ \rightarrow\ f == \mathit{fact}(n))$