Modularity

- Modularity is important in programming;
- In verification it is useful that one can reuse correctness results;
- Let

$$\texttt{fact} = f \leftarrow 1; i \leftarrow 1; \texttt{while}\ i \leq n\ \texttt{do}\ (f \leftarrow f \times i; i \leftarrow i + 1)$$

  and $fact(n) = n!$, and we have a proof of

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

  we would like to use this result to prove a weaker specification:

$$\{n = 10\}\texttt{fact}\{f = fact(n)\}$$

  This can be achieved using the consequence rule.

- However, if we have,

$$\{n \geq 0 \wedge n = n_0\}\texttt{fact}\{f = fact(n) \wedge n = n_0\}$$

  we cannot derive the weaker triple.

Adaptation

The problem of matching a proved specification of a program with a weaker specification is called the *adaptation problem* (without the full proof of this last specification).

(Satisfiable specification) A specification (φ, ψ) is satisfiable if there is a program C such that $\models \{\varphi\}C\{\psi\}$.

(Adaptation completeness) Let (φ, ψ) satisfiable and for any program C we have $\models \{\varphi'\}C\{\psi'\}$ whenever $\models \{\varphi\}C\{\psi\}$. A deductive system of Hoare triples is *adaptation complete* iff for any program C the following rule is derivable.

$$\frac{\{\varphi\}C\{\psi\}}{\{\varphi'\}C\{\psi'\}}$$

Hoare logic is not adaptation complete, due to the presence of auxiliary variables.

- Informally, auxiliary variables are universally quantified over Hoare triples, connecting pre and post conditions. But, the side conditions in $cons_p$ rule do not take that in consideration.
- A solution was proposed by Kleymann, considering a stronger consequence rule, formalizing the difference between program and auxiliary variables.

- In the consequence rule

$$\frac{\{\varphi\}C\{\psi\}}{\{\varphi'\}C\{\psi'\}} \quad \text{if} \;\; \varphi' \to \varphi \wedge \psi \to \psi'$$

- The first side condition is interpreted in the pre-state, whereas the second is interpreted in the post-state. Both should communicate through the auxiliary variables.

- The auxiliary variables in ψ have to be interpreted in the pre-state and should be existentially quantified:in the factorial example $n = 10 \to n \geq 0 \wedge n = n_0$, does not hold, but $n = 10 \to \exists n_0.n \geq 0 \wedge n = n_0$ does.

The adequate side condition suggested by Kleymann has the form

$$\varphi' \to (\varphi \wedge (\psi \to \psi'))$$

Let $\overline{y}$ be the auxiliary variables in $\{\varphi\}C\{\psi\}$, quantification is introduced as follows:

$$\varphi' \to \exists\overline{y_f}.(\varphi[\overline{y_f}/\overline{y}] \wedge (\psi[\overline{y_f}/\overline{y}] \to \psi'))$$

We interpret the auxiliary variables in φ' and ψ' and substituted program variables in the post-state by universally quantified fresh variables

$$\frac{\{\varphi\}C\{\psi\}}{\{\varphi'\}C\{\psi'\}} \quad \text{se} \;\; \varphi' \to \forall\overline{x_f}.\exists\overline{y_f}.(\varphi[\overline{y_f}/\overline{y}] \wedge (\psi[\overline{y_f}/\overline{y}, \overline{x_f}/\overline{x}] \to \psi'[\overline{x_f}/\overline{x}]))$$

where $\overline{y}$ are the auxiliary variables in $\{\varphi\}C\{\psi\}$, $\overline{x}$ the program variables in C, and $\overline{y_f}$, $\overline{x_f}$ are fresh variables.

- The previous rule works for total correctness.

- we have a weaker condition por partial correctness

$$\varphi' \to ((\varphi \to \psi) \to \psi')$$

  The program variables are now universally quantified

$$\varphi' \to (\forall\overline{y}.(\varphi \to \psi) \to \psi')$$

  The resulting rule is

$$\frac{\{\varphi\}C\{\psi\}}{\{\varphi'\}C\{\psi'\}} \quad \text{se} \;\; \varphi' \to \forall\overline{x_f}.(\forall\overline{y_f}.(\varphi[\overline{y_f}/\overline{y}] \to \psi[\overline{y_f}/\overline{y}, \overline{x_f}/\overline{x}]) \to \psi'[\overline{x_f}/\overline{x}])$$

where $\overline{y}$ are auxiliary variables $\{\varphi\}C\{\psi\}$, $\overline{x}$ are teh program variables of C and $\overline{y_f}$ and $\overline{x_f}$ fresh.

We will only use these new consequence rules to deal with recursive procedures

Example

Given the assertion:

$$\{n \geq 0 \wedge n = n_0\}\texttt{fact}\{f = fact(n) \wedge n = n_0\}$$

To derive a weaker assertion:

$$\{n = 10\}\texttt{fact}\{f = fact(n)\}$$

we obtain the side condition

$$\begin{aligned} n = \quad & 10 \rightarrow \forall n_f, f_f.(\forall n_{0f}.n \geq 0 \wedge n = n_{0f} \rightarrow f_f = fact(n_f) \wedge n_f = n_{0f}) \\ & \rightarrow f_f = fact(10)) \end{aligned}$$

Mechanising Hoare Logic

Given a Hoare triple ($\{\varphi\}C\{\psi\}$) rules are applied from the conclusion, assuming that the side conditions hold.

- If all side conditions hold, a proof can be build;
- If some side condition does not hold, the derivation tree is not a valid deduction, but is there an alternative derivation?

There is a strategy to build the derivation trees such that we can conclude (if some side conditions does not hold) that there is no derivation for the given Hoare triple.

Tableaux

- The tableaux system allows to obtain the derivation of a Hoare triple, that is the conclusion.
- The derivation is valid if the verification conditions are satisfiable.
- But if they are not, how to ensure that there is no other derivation?
- If there is no determinism one cannot mechanise the Hoare logic.
- We will see that the tableaux ensure that if the verification conditions are not satisfiable *no other* derivation exists.
- and the tableaux can be automated.

Subformula property and Ambiguity

Most rules of Hoare logic have the *subformula property*:

> *all the assertions that occur in the premises of a rule also occur in its conclusion.*

The exceptions are:

- The rule *comp*, which requires an intermediate condition;
- The rule *cons*, where the precondition and the postcondition must be guessed.

Other property that we want is that the choice of the rules is non ambiguous, but:

- The rule *cons*, can be applied to any triple of Hoare. Thus it should be removed.

Hoare logic without the rule *cons*: system $\mathcal{H}_g$

$$\frac{}{\{\varphi\}\,\mathsf{skip}\,\{\psi\}}\ \text{if} \models \varphi \to \psi$$

$$\frac{}{\{\varphi\}\,x \leftarrow E\,\{\psi\}}\ \text{if} \models \varphi \to \psi[E/x]$$

$$\frac{\{\varphi\}\,C_1\,\{\eta\} \qquad \{\eta\}\,C_2\,\{\psi\}}{\{\varphi\}\,C_1;C_2\,\{\psi\}}$$

$$\frac{\{\varphi\ \wedge\ B\}\,C_1\,\{\psi\} \qquad \{\varphi\ \wedge\ \neg B\}\,C_2\,\{\psi\}}{\{\varphi\}\,\texttt{if}\,B\,\texttt{then}\,C_1\,\texttt{else}\,C_2\,\{\psi\}}$$

$$\frac{\{\eta\ \wedge\ B\}\,C\,\{\eta\}}{\{\varphi\}\,\texttt{while}\,B\,\texttt{do}\,\{\eta\}C\,\{\psi\}}\ \text{if} \models \varphi \to \eta \text{ and } \models \eta\ \wedge\ \neg B \to \psi$$

In the *while*$_p$ rule the loop is annotated with the invariant η, to keep the subformula property. .

We can show that the *cons* is derivable in $\mathcal{H}_g$. Let Γ be a set of assertions.

Lema 3.1. *If* $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C\{\psi\}$ *and* $\models \varphi' \to \varphi$, $\models \psi \to \psi'$, *then* $\Gamma \vdash_{\mathcal{H}_g} \{\varphi'\}C\{\psi'\}$.

Proof: By induction on the derivation of $\Gamma \vdash_{\mathcal{H}_g} \{\psi\}C\{\varphi\}$. We consider the case `skip` and sequence.

- For $C \equiv \texttt{skip}$, we have $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}\texttt{skip}\{\psi\}$, if $\models \varphi \rightarrow \psi$. We have $\models \varphi' \rightarrow \varphi$, $\models \varphi \rightarrow \psi$ and $\models \psi \rightarrow \psi'$, thus $\models \varphi' \rightarrow \psi'$, what means that $\Gamma \vdash_{\mathcal{H}_g} \{\varphi'\}\texttt{skip}\{\psi'\}$.
- For $C \equiv C_1; C_2$,we have $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C_1; C_2\{\psi\}$, if $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C_1\{\eta\}$ and $\Gamma \vdash_{\mathcal{H}_g} \{\eta\}C_2\{\psi\}$.

  By induction we have

  $$\Gamma \vdash_{\mathcal{H}_g} \{\varphi'\}C_1\{\eta\} \text{ as} \models \varphi' \rightarrow \varphi \text{ and } \models \eta \rightarrow \eta,$$
  $$\Gamma \vdash_{\mathcal{H}_g} \{\eta\}C_2\{\psi'\} \text{ as} \models \eta \rightarrow \eta \text{ and } \models \psi \rightarrow \psi',$$

  thus $\Gamma \vdash_{\mathcal{H}_g} \{\varphi'\}C_1; C_2\{\psi'\}$.

Exerc. 3.1. *Complete the previous proof.*

Equivalence between $\mathcal{H}$ and $\mathcal{H}_g$

Lema 3.2. $\Gamma \vdash_{\mathcal{H}} \{\varphi\}C\{\psi\}$ *iff* $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C\{\psi\}$

Proof:

($\Rightarrow$) By induction on the derivation of $\Gamma \vdash_{\mathcal{H}} \{\psi\}C\{\varphi\}$, using the lemma. We consider the case of assignment and consequence.

- we have $\Gamma \vdash_{\mathcal{H}} \{\varphi[E/x]\}x \leftarrow E\{\varphi\}$ and $\models \varphi[E/x] \rightarrow \varphi[E/x]$, thus $\Gamma \vdash_{\mathcal{H}_g} \{\varphi[E/x]\}x \leftarrow E\{\varphi\}$
- By the rule of consequence we have

  $$\Gamma \vdash_{\mathcal{H}} \{\varphi\}C\{\psi\},$$

  if $\Gamma \vdash_{\mathcal{H}} \{\varphi'\}C\{\psi'\}$ and $\models \varphi \rightarrow \varphi'$, $\models \psi' \rightarrow \psi$.
  By induction we have $\Gamma \vdash_{\mathcal{H}_g} \{\varphi'\}C\{\psi'\}$, thus by the previous lemma we have $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C\{\psi\}$.

($\Leftarrow$) By induction on the derivation of $\Gamma \vdash_{\mathcal{H}_g} \{\psi\}C\{\varphi\}$. We consider the case of assignment and conditional.

- we have

  $$\Gamma \vdash_{\mathcal{H}_g} \{\psi\}x \leftarrow E\{\varphi\} \text{ if } \models \psi \rightarrow \varphi[E/x].$$

  As

  $$\Gamma \vdash_{\mathcal{H}} \{\varphi[E/x]\}x \leftarrow E\{\varphi\} \text{ and } \models \psi \rightarrow \varphi[E/x]$$

  and $\models \psi \rightarrow \psi$, by $cons_p$ rule, we have $\Gamma \vdash_{\mathcal{H}} \{\psi\}x \leftarrow E\{\varphi\}$.
- we have $\Gamma \vdash_{\mathcal{H}_g} \{\psi\}\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2\, \{\varphi\}$, if

  $$\Gamma \vdash_{\mathcal{H}_g} \{\psi \wedge B\}C_1\{\varphi\} \text{ and } \Gamma \vdash_{\mathcal{H}_g} \{\psi \wedge \neg B\}C_2\{\varphi\}.$$

  By induction $\Gamma \vdash_{\mathcal{H}} \{\psi \wedge B\}C_1\{\varphi\}$ and $\Gamma \vdash_{\mathcal{H}} \{\psi \wedge \neg B\}C_2\{\varphi\}$, thus $\Gamma \vdash_{\mathcal{H}} \{\psi\}\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2\, \{\varphi\}$

Exerc. 3.2. *Complete the previous proof.*

Pro and cons

Advantages of $\mathcal{H}_g$:

- The ambiguity of rule *cons* was eliminated.

Drawbacks of $\mathcal{H}_g$:

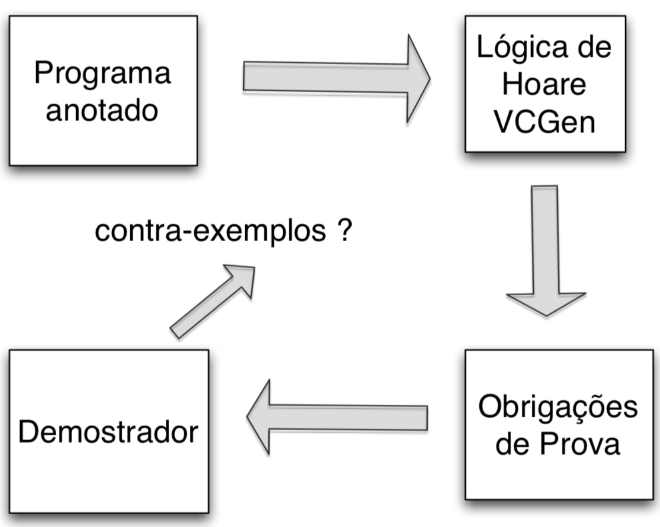
- Is still necessary to guess the intermediate preconditions in *comp*.
- Lost of modularity.

The weakest precondition strategy:tableaux

We already saw that for building a derivation for $\{\varphi\}C\{\psi\}$, where φ can or not be known (we write $\{?\}C\{\psi\}$).

1. if φ is known, we apply the unique rule of $\mathcal{H}_g$. if C is $C_1; C_2$, we build a subproof of the form $\{?\}C_2\{\psi\}$. when the proof terminates we can go on with $\{\varphi\}C_1\{\theta\}$, with θ obtained in the previous sub-derivation.
2. if φ is unknown, the construction proceeds as before, except that, in the rules for skip, assignment and loops, with a side condition $\varphi \to \theta$, we tale the precondition φ to be θ (which is exactly the $wp(C.\psi)$).

Two phases verification



Verification condition generator, VCG

Given $\{\varphi\}C\{\psi\}$ to compute $VC(C,\psi)$ we have to:

- Compute the weakest precondition $wp(C,\psi)$
- we have that $\varphi \to wp(C,\psi)$ is a verification condition (VC)
- The remaining VC are collected from the conditions introduced in the loops **while**.

Computation of the weakest preconditions (wp)

Given a program C and a postcondition ψ, we can compute $wp(C,\psi)$ such that $\{wp(C,\psi)\}C\{\psi\}$ is valid and if $\{\varphi\}C\{\psi\}$is valid for any φ then $\varphi \to wp(C,\psi)$.

$$
\begin{array}{rcl}
wp(\mathbf{skip},\psi) & = & \psi \\
wp(x \leftarrow E,\psi) & = & \psi[E/x] \\
wp(C_1;C_2,\psi) & = & wp(C_1, wp(C_2,\psi)) \\
wp(\texttt{if}\, B\, \mathbf{then}\, C_1\, \mathbf{else}\, C_2,\psi) & = & (B \to wp(C_1,\psi)) \\
 & & \wedge(\neg B \to wp(C_2,\psi)) \\
wp(\mathbf{while}\, B\, \mathbf{do}\, \{\eta\}C,\psi) & = & \eta
\end{array}
$$

Properties of wp and VCG

Given a program C and an assertion ψ if $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C\{\psi\}$, for any precondition φ, then

Lema 3.3.

1. $\Gamma \vdash_{\mathcal{H}_g} \{wp(C,\psi)\}C\{\psi\}$
2. $\Gamma \models \varphi \to wp(C,\psi)$

Proof: By induction on C. We consider the cases of `skip` and `while`.

- For $C \equiv \texttt{skip}$, we have $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}\texttt{skip}\{\psi\}$ if $\models \varphi \to \psi$. Note that $wp(\texttt{skip},\psi) = \psi$.
  1. Trivially we have $\Gamma \vdash_{\mathcal{H}_g} \{\psi\}\texttt{skip}\{\psi\}$, as $\models \psi \to \psi$.
  2. By hypothesis we have $\Gamma \models \varphi \to \psi = wp(\texttt{skip},\psi)$.

- $C \equiv \texttt{while}\, B\, \texttt{do}\, C$, we have

$$\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}\, \texttt{while}\, B\, \texttt{do}\, \{\eta\}C\, \{\psi\} \text{ if } \Gamma \vdash_{\mathcal{H}_g} \{\eta \wedge B\}C\{\eta\}$$

  and $\models \varphi \to \eta$, $\models \eta \wedge \neg B \to \psi$.

  Note that $wp(\texttt{while}\, B\, \texttt{do}\, \{\eta\}C, \psi) = \eta$

  1. As $\models \eta \to \eta$, and by hypothesis $\models \eta \wedge \neg B \to \psi$ and $\Gamma \vdash_{\mathcal{H}_g} \{\eta \wedge B\}C\{\eta\}$, then

$$\Gamma \vdash_{\mathcal{H}_g} \{\eta\}\, \texttt{while}\, B\, \texttt{do}\, \{\eta\}C\, \{\psi\}$$

  2. by hypothesis we have $\Gamma \models \varphi \to \eta = wp(\texttt{while}\, B\, \texttt{do}\, \{\eta\}C\, \psi)$.

Exerc. 3.3. *Complete the previous proof.*

Algorithm VCG

First one computes $VC(C, \psi)$ without consider the preconditions

$$\begin{aligned}
VC(\texttt{skip}, \psi) &= \emptyset \\
VC(x \leftarrow E, \psi) &= \emptyset \\
VC(C_1; C_2, \psi) &= VC(C_1, wp(C_2, \psi)) \cup VC(C_2, \psi) \\
VC(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi) &= VC(C_1, \psi) \cup VC(C_2, \psi) \\
VC(\texttt{while}\, B\, \texttt{do}\, \{\eta\}C, \psi) &= \{(\eta \wedge B) \to wp(C, \eta)\} \cup \\
&\quad\ \{(\eta \wedge \neg B) \to \psi\} \cup VC(C, \eta)
\end{aligned}$$

Next one considers the precondition:

$$VCG(\{\varphi\}C\{\psi\}) = \{\varphi \to wp(C, \psi)\} \cup VC(C, \psi)$$

Example

let fact be the program:

```
f ← 1; i ← 1;
while i ≤ n do
    {f = (i − 1)! ∧ i ≤ n + 1}                    ▷ Invariante
    f ← f ∗ i;
    i ← i + 1;
```

We compute

$$\text{VCG}(\{n \geq 0\}\mathsf{fact}\{f = n!\})$$

with

$$\begin{aligned}\theta &= f = (i-1)! \wedge i \leq n+1 \\ C_w &= f \leftarrow f * i; i \leftarrow i+1\end{aligned}$$

$$\begin{aligned}
& VC(\texttt{fact}, f = n!) \\
= \ & VC(f \leftarrow 1; i \leftarrow 1, wp(\textbf{while } i \leq n \textbf{ do}\{\theta\}C_w, f = n!)) \\
& \cup VC(\textbf{while } i \leq n \texttt{ do}\{\theta\}C_w, f = n!) \\
= \ & VC(f \leftarrow 1; i \leftarrow 1, \theta) \cup \{\theta \wedge i \leq n \rightarrow wp(C_w, \theta)\} \\
& \cup \{\theta \wedge i > n \rightarrow f = n!\} \cup VC(C_w, \theta) \\
= \ & VC(f \leftarrow 1, wp(i \leftarrow 1, \theta)) \cup VC(i \leftarrow 1, \theta) \\
& \cup \{f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow wp(f \leftarrow f * i; i \leftarrow i+1, \theta)\} \\
& \cup \{f = (i-1)! \wedge i \leq n+1 \wedge i > n \rightarrow f = n!\} \\
& \cup VC(f = f * i, wp(i \leftarrow i+1, \theta)) \cup VC(i \leftarrow i+1, \theta) \\
= \ & \emptyset \cup \emptyset \cup \{f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \\
& \qquad \rightarrow wp(f \leftarrow f * i, f = (i+1-1)! \wedge i+1 \leq n+1)\} \\
& \cup \{f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow f = n!\} \cup \emptyset \cup \emptyset \\
= \ & \{f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow f * i = (i+1-1)! \\
& \wedge i+1 \leq n+1, f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow f = n!\}
\end{aligned}$$

$$\begin{aligned}
& VCG(\{n \geq 0\}\mathsf{fact}\{f = n!\}) \\
= \ & \{n \geq 0 \rightarrow wp(\mathsf{fact}, f = n!)\} \cup VC(\mathsf{fact}, f = n!) \\
= \ & \{n \geq 0 \rightarrow wp(f \leftarrow 1; i \leftarrow 1; wp(\textbf{while } i \leq n \textbf{ do}\{\theta\}C_w, f = n!), \\
& f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow f * i = (i+1-1)! \\
& \wedge i+1 \leq n+1, f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow f = n!\} \\
= \ & \{n \geq 0 \rightarrow wp(f \leftarrow 1; i \leftarrow 1; \theta), \\
& f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow f * i = (i+1-1)! \\
& \wedge i+1 \leq n+1, f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow f = n!\}
\end{aligned}$$

We have the following proof obligations:

1. $n \geq 0 \rightarrow 1 = (1-1)! \wedge 1 \leq n+1$
2. $f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow f * i = (i+1-1)! \wedge i+1 \leq n+1)$
3. $f = (i-1)! \wedge i \leq n+1 \wedge i \leq n \rightarrow f = n!$

Teorema 3.1 (Adequacy of VCG)**.** *Let* $\{\varphi\}C\{\psi\}$ *a Hoare triple and* Γ *a set of assertions.*

$$\Gamma \models VCG(\{\varphi\}C\{\psi\}) \textit{ iff } \Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C\{\psi\}.$$

Proof:

(⇒) By induction on the derivation of C. We consider the case of assignment and sequence

- For $C \equiv x \leftarrow E$, we have

$$\begin{aligned} VCG(\{\varphi\}X \leftarrow E\{\psi\}) &= \{\varphi \to wp(X \leftarrow E, \psi)\} \cup VC(x \leftarrow E, \psi) \\ &= \{\varphi \to \psi[E/x]\}. \end{aligned}$$

If $\Gamma \models \varphi \to \psi[E/x]$, then by the assignment rule

$$\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C\{\psi\}.$$

- For $C \equiv C_1; C_2$, we have

$$\begin{aligned} VCG(\{\varphi\}C_1; C_2\{\psi\}) &= \{\varphi \to wp(C_1; C_2, \psi)\} \cup VC(C_1; C_2, \psi) \\ &= \{\varphi \to wp(C_1, wp(C_2, \psi))\} \\ &\quad \cup VC(C_1, wp(C_2, \psi)) \cup VC(C_2, \psi). \end{aligned}$$

Let $\eta = wp(C_2, \psi)$. As

$$\Gamma \models \varphi \to wp(C_1, \eta) \cup VC(C_1, \eta) = VCG(\{\varphi\}C_1\{\eta\}),$$

by induction $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C_1\{\eta\}$.
Also $\Gamma \models \eta \to \eta \cup VC(C_2, \psi) = VCG(\{\eta\}C_2\{\psi\})$, by induction $\Gamma \vdash_{\mathcal{H}_g} \{\eta\}C_2\{\psi\}$, thus $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C_1; C_2\{\psi\}$.

(⇐) By induction on the derivation of $\Gamma \vdash_{\mathcal{H}_g} \{\psi\}C\{\varphi\}$. We consider the case `skip` and conditional.

- $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}\texttt{skip}\{\psi\}$, if $\Gamma \models \varphi \to \psi = VCG(\{\varphi\}\texttt{skip}\{\psi\})$.

- $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2\, \{\psi\}$ if $\Gamma \vdash_{\mathcal{H}_g} \{\varphi \land B\}C_1\{\psi\}$ e $\Gamma \vdash_{\mathcal{H}_g} \{\varphi \land \neg B\}C_2\{\psi\}$. By induction

$$\Gamma \models VCG(\{\varphi \land B\}C_1\{\psi\}) = \{(\varphi \land B) \to wp(C_1, \psi)\} \cup VC(C_1, \psi)$$

and

$$\Gamma \models VCG(\{\varphi \land \neg B\}C_2\{\psi\}) = \{(\varphi \land \neg B) \to wp(C_2, \psi)\} \cup VC(C_2, \psi).$$

Note that,

$$wp(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi) = B \rightarrow wp(C_1, \psi) \wedge \neg B \rightarrow wp(C_2, \psi)\},$$

thus,

$$\Gamma \models \{\varphi \rightarrow wp(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi)\}.$$

Thus, $\Gamma \models \{\varphi \rightarrow wp(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi)\} \cup VC(C_1, \psi) \cup VC(C_2, \psi) = VCG(\{\varphi\}\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2\{\psi\})$.

Exerc. 3.4. *Complete the previous proof.*

References

[AFPMdS11] José Bacelar Almeida, Maria João Frade, Jorge Sousa Pinto, and Simão Melo de Sousa. *Rigorous Software Development: An Introduction to Program Verification*. Springer, 2011.