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 Hoare triple. Thus it should be removed.

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

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

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

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

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

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

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 7.1. *If* $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C\{\psi\}$ *and* $\models \varphi' \implies \varphi$, $\models \psi \implies \psi'$, *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 $\models \varphi \implies \psi$. We have $\models \varphi' \implies \varphi$, $\models \varphi \implies \psi$ and $\models \psi \implies \psi'$, thus $\models \varphi' \implies \psi'$, 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

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

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

Exerc. 7.1. *Complete the previous proof.*

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

Lema 7.2. $\Gamma \vdash_{\mathcal{H}} \{\varphi\}C\{\psi\}$ *iff* $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C\{\psi\}$

Proof:

($\Rightarrow$) By induction on the derivation of $\Gamma \vdash_{\mathcal{H}} \{\varphi\}C\{\psi\}$, 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 $\models \varphi[E/x] \implies \varphi[E/x]$, 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 $\models \varphi \implies \varphi'$, $\models \psi' \implies \psi$.
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 $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}C\{\psi\}$. We consider the case of assignment and conditional.

- we have

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

As

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

and $\models \psi \implies \psi$, by $cons_p$ rule, we have $\Gamma \vdash_{\mathcal{H}} \{\varphi\}x \leftarrow E\{\psi\}$.

- we have $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2\, \{\psi\}$, if

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

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

Exerc. 7.2. *Complete the previous proof.*

Pro and Cons

Advantages of $\mathcal{H}_g$:

- The ambiguity of rule *cons* was eliminated.

Drawbacks of $\mathcal{H}_g$:

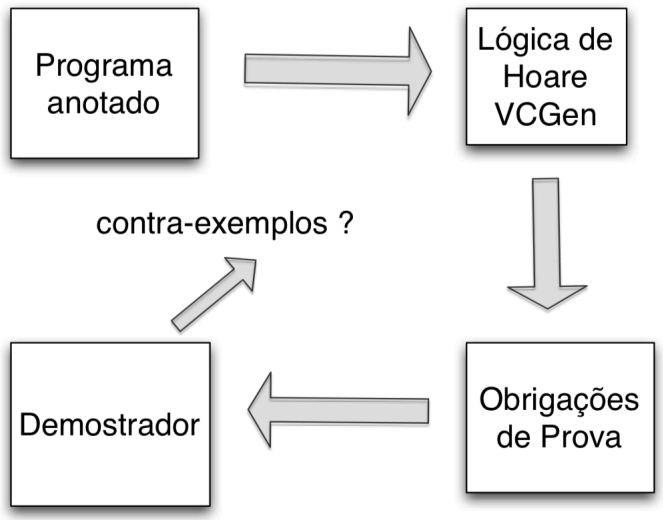
- Is still necessary to guess the intermediate preconditions in *comp*.

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 $\varphi \implies wp(C, \psi)$ 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 $\varphi \implies wp(C,\psi)$.

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

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 7.3.

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 $C \equiv$ `skip`, we have $\Gamma \vdash_{\mathcal{H}_g} \{\varphi\}\texttt{skip}\{\psi\}$ if $\models \varphi \implies \psi$. Note that $wp(\texttt{skip},\psi) = \psi$.
  1. Trivially we have $\Gamma \vdash_{\mathcal{H}_g} \{\psi\}\texttt{skip}\{\psi\}$, as $\models \psi \implies \psi$.
  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 $\models \varphi \implies \eta$, $\models \eta \wedge \neg B \implies \psi$.

  Note that $wp(\texttt{while}\, B\, \texttt{do}\, \{\eta\}C,\psi) = \eta$
  1. As $\models \eta \implies \eta$, and by hypothesis $\models \eta \wedge \neg B \implies \psi$ 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. 7.3. *Complete the previous proof.*

Algorithm VCG

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

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

Next one considers the precondition:

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

Example

let fact be the program:

$f \leftarrow 1; i \leftarrow 1;$
while $i \leq n$ **do**
  $\{f = (i-1)! \wedge i \leq n+1\}$ ▷ Invariante
  $f \leftarrow f * i;$
  $i \leftarrow i + 1;$

We compute

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

with

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

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

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

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

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

Proof:

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

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

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

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

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

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

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

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

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

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

($\Leftarrow$) 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 $\Gamma \models \varphi \implies \psi = VCG(\{\varphi\}\texttt{skip}\{\psi\})$.

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

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

  and

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

  Note that,

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

  thus,

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

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

Exerc. 7.4. *Complete the previous proof.*

Verification Conditions for programs with arrays

Let maxarray be the following program:

$max \leftarrow 0;$
$i \leftarrow 1;$
while $i < size$ **do**
  if $u[i] > u[max]$ **then**
    $max \leftarrow i$
  else
    skip;
  $i \leftarrow i + 1$

We want to check that

$\{size \geq 1\}$ maxarray $\{0 \leq max < size \wedge \forall a.0 \leq a < size \rightarrow u[a] \leq u[max]\}$

Which is the invariant?

The annotated program is:

Require: $\{size \geq 1\}$
$max \leftarrow 0;$
$i \leftarrow 1;$
while $i < size$ **do** $\{\theta\}$
  if $u[i] > u[max]$ **then**
    $max \leftarrow i$
  else
    skip;
  $i \leftarrow i + 1$
Ensure: $\{0 \leq max < size \wedge \forall a.0 \leq a < size \rightarrow u[a] \leq u[max]\}$

where the invariant is

$$\theta \quad = \quad 1 \leq i \leq size \wedge 0 \leq max < i \wedge \forall a.0 \leq a < i \rightarrow u[a] \leq u[max]$$

Exerc. 7.5. *Using the system* $\mathcal{H}_g$ *build a tableaux for*

$$\{size \geq 1\}\text{maxarray}\{0 \leq max < size \wedge \forall a.0 \leq a < size \rightarrow u[a] \leq u[max]\}$$

$\diamond$

The verification conditions can be calculated by applying the VCG for

$\{size \geq 1\}$ maxarray $\{0 \leq max < size \wedge \forall a.0 \leq a < size \rightarrow u[a] \leq u[max]\}$

We assume

$$\begin{array}{rcl} \theta & = & 1 \leq i \leq size \wedge 0 \leq max < i \wedge \forall a.0 \leq a < i \rightarrow u[a] \leq u[max] \\ C & = & \texttt{if } u[i] > u[max] \texttt{ then } max \leftarrow i \texttt{ else skip}; i \leftarrow i + 1; \\ \psi & = & 0 \leq max < size \wedge \forall a.0 \leq a < size \rightarrow u[a] \leq u[max] \end{array}$$

We have

$$\begin{array}{rcl} VCG(\{size \geq 1\}\mathsf{maxarray}\{\psi\}) & = & \{size \geq 1 \implies wp(\mathsf{maxarray}, \psi)\} \\ & \cup & VC(\mathsf{maxarray}, \psi). \end{array}$$

$$\begin{array}{rcl} wp(C, \theta) & = & (u[i] > u[max] \implies (1 \leq i+1 \leq size \\ & \wedge & 0 \leq i < i+1 \wedge \forall a.0 \leq a < i+1 \rightarrow u[a] \leq u[i]) \\ & \wedge & (u[i] \leq u[max] \implies (1 \leq i+1 \leq size \\ & \wedge & 0 \leq max < i+1 \\ & \wedge & \forall a.0 \leq a < i+1 \rightarrow u[a] \leq u[max])) \\ wp(\mathsf{maxarray}, \theta) & = & (1 \leq 1 \leq size \\ & \wedge & 0 \leq 0 < 1 \wedge \forall a.0 \leq a < 1 \rightarrow u[a] \leq u[0]) \\ VC(\mathsf{maxarray}, \psi) & = & \{\theta \wedge i < size \implies wp(C, \theta), \theta \wedge i \geq size \implies \psi\} \\ & = & \{(1 \leq i < size \wedge 0 \leq max < i \wedge \\ & & \forall a.0 \leq a < i \rightarrow u[a] \leq u[max]) \implies wp(C, \theta), \\ & & 1 \leq i = size \wedge 0 \leq max < i \wedge \\ & & \forall a.0 \leq a < i \rightarrow u[a] \leq u[max]) \implies \psi\} \end{array}$$

Extension of VCG for arrays

We add the following rule to $\mathcal{H}_g$:

$$\frac{}{\{\varphi\}\, u[E] \leftarrow E' \,\{\psi\}} \text{ if } \models \varphi \implies \psi[u[E \triangleright E']/u]$$

we expand wp and VC in the following way:

$$\begin{array}{rcl} wp(u[E] \leftarrow E', \psi) & = & \psi[u[E \triangleright E']/u] \\ VC(u[E] \leftarrow E', \psi) & = & \emptyset \end{array}$$

For instance:

$$wp(u[i] \leftarrow 10, u[j] > 100) = u[i \triangleright 10][j] > 100$$
$$VC(u[i] \leftarrow 10, u[j] > 100) = \emptyset$$

Exerc. 7.6. *Using the VCG algorithm calculate:*

1. $VCG(\{u[j] > 100\}u[i] \leftarrow 10\{u[j] > 100\})$
2. $VCG(\{i \neq j \wedge u[j] > 100\}u[i] \leftarrow 10\{u[j] > 100\})$
3. $VCG(\{i = 70\}u[i] \leftarrow 10\{u[i] = 10\})$

$\diamond$

Safety Properties

In the operational semantics we considered, every expression evaluates to a value and command execution would not produce any error.

We now consider some modifications that approximate the language to a real programming language:

- incorporating in the language semantics a special error value;
- modifying the evaluation relation to admit evaluation of commands to a special error state;

Error semantics for arithmetic expressions

$\mathcal{A} : \mathbf{Aexp} \to (\mathbf{State} \to (\mathbb{Z} \cup \{\mathsf{error}\}))$

$$
\begin{array}{rcl}
\mathcal{A}[\![n]\!]s & = & n \\
\mathcal{A}[\![x]\!]s & = & s(x) \\
\mathcal{A}[\![E_1 \odot E_2]\!]s & = & \begin{cases} \mathcal{A}[\![E_1]\!]s \odot \mathcal{A}[\![E_2]\!]s & \text{if } \mathcal{A}[\![E_1]\!]s \neq \mathsf{error} \neq \mathcal{A}[\![E_2]\!]s \\ \mathsf{error} & \text{otherwise} \end{cases} \\
 & & \text{para } \odot \in \{+, -, \times\} \\
\mathcal{A}[\![E_1 \div E_2]\!]s & = & \begin{cases} \mathcal{A}[\![E_1]\!]s \div \mathcal{A}[\![E_2]\!]s & \text{if } \mathcal{A}[\![E_1]\!]s \neq \mathsf{error} \neq \mathcal{A}[\![E_2]\!]s \\ & \text{and } \mathcal{A}[\![E_2]\!]s \neq 0 \\ \mathsf{error} & \text{otherwise} \end{cases}
\end{array}
$$

Error semantics for Boolean expressions

$\mathbf{T} = \{\mathsf{true}, \mathsf{false}\}$, $\mathcal{B} : \mathbf{Bexp} \to (\mathbf{State} \to (\mathbf{T} \cup \{\mathsf{error}\}))$

$$
\begin{array}{rcl}
\mathcal{B}[\![\mathtt{true}]\!]s & = & \mathsf{true} \\
\mathcal{B}[\![\mathtt{false}]\!]s & = & \mathsf{false} \\
\mathcal{B}[\![\neg b]\!]s & = & \begin{cases} \mathsf{true} & \text{if } \mathcal{B}[\![b]\!]s = \mathsf{false} \\ \mathsf{false} & \text{if } \mathcal{B}[\![b]\!]s = \mathsf{true} \\ \mathsf{error} & \text{if } \mathcal{B}[\![b]\!]s = \mathsf{error} \end{cases} \\
\mathcal{B}[\![E_1 \odot E_2]\!]s & = & \begin{cases} \mathcal{A}[\![E_1]\!]s \odot \mathcal{A}[\![E_2]\!]s & \text{if } \mathcal{A}[\![E_1]\!]s \neq \mathsf{error} \neq \mathcal{A}[\![E_2]\!]s \\ \mathsf{error} & \text{otherwise} \end{cases} \\
 & & \text{for } \odot \in \{=, <, \leq\}. \\
\mathcal{B}[\![b_1 \wedge b_2]\!]s & = & \begin{cases} \mathsf{false} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathsf{false} \\ \mathsf{error} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathsf{error} \\ \mathcal{B}[\![b_1]\!]s & \text{otherwise} \end{cases}
\end{array}
$$

Natural semantics with errors (*big-step*)

$$\langle \texttt{skip}, s\rangle \longrightarrow s$$

$$\langle x \leftarrow E, s\rangle \longrightarrow \begin{cases} s[\mathcal{A}[\![E]\!]s/x] & \text{if } \mathcal{A}[\![E]\!]s \neq \mathsf{error} \\ \mathsf{error} & \text{otherwise} \end{cases}$$

$$\frac{\langle C_1, s\rangle \longrightarrow \mathsf{error}}{\langle C_1; C_2, s\rangle \longrightarrow \mathsf{error}}$$

$$\frac{\langle C_1, s\rangle \longrightarrow s' \ , \ \langle C_2, s'\rangle \longrightarrow s''}{\langle C_1; C_2, s\rangle \longrightarrow s''} \text{ if } s' \neq \mathsf{error}$$

$$\langle \texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2, s\rangle \longrightarrow \mathsf{error} \text{ if } \mathcal{B}[\![B]\!]s = \mathsf{error}$$

$$\frac{\langle C_1, s\rangle \longrightarrow s'}{\langle \texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2, s\rangle \longrightarrow s'} \text{ if } \mathcal{B}[\![B]\!]s = \mathsf{true}$$

$$\frac{\langle C_2, s\rangle \longrightarrow s'}{\langle \texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2, s\rangle \longrightarrow s'} \text{ if } \mathcal{B}[\![B]\!]s = \mathsf{false}$$

$$\langle \texttt{while } B \texttt{ do } C, s\rangle \longrightarrow \mathsf{error} \text{ if } \mathcal{B}[\![B]\!]s = \mathsf{error}$$

$$\frac{\langle C, s\rangle \longrightarrow \mathsf{error}}{\langle \texttt{while } B \texttt{ do } C, s\rangle \longrightarrow \mathsf{error}} \text{ if } \mathcal{B}[\![B]\!]s = \mathsf{true}$$

$$\frac{\langle C, s\rangle \longrightarrow s', \langle \texttt{while } B \texttt{ do } C, s'\rangle \longrightarrow s''}{\langle \texttt{while } B \texttt{ do } C, s\rangle \longrightarrow s''} \text{ if } \mathcal{B}[\![B]\!]s = \mathsf{true}, s' \neq \mathsf{error}$$

$$\langle \texttt{while } B \texttt{ do } C, s\rangle \longrightarrow s \text{ if } \mathcal{B}[\![B]\!]s = \mathrm{F}$$

Hoare logic safety-sensitive

To extend the deductive systems of Hoare logic

- consider the structure of each command
- consider the possible values of the expressions that occur
- associate *safety side conditions* to each expression E which we denote by $\mathsf{safe}(E)$ (which is an assertion).

Hoare logic with safety conditions: system $\mathcal{H}_s$

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

$$\frac{}{\{\varphi\}\, x \leftarrow E\, \{\psi\}} \text{ if } \varphi \implies \mathsf{safe}(E) \text{ and } \varphi \implies \psi[E/x]$$

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

$$\frac{\{\varphi \wedge B\}\, C_1\, \{\psi\} \qquad \{\varphi \wedge \neg B\}\, C_2\, \{\psi\}}{\{\varphi\}\, \texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2\, \{\psi\}} \text{ if } \varphi \implies \mathsf{safe}(B)$$

$$\frac{\{\eta \wedge B\}\, C\, \{\eta\}}{\{\psi\}\, \texttt{while}\, B\, \texttt{do}\, \{\eta\} C\, \{\varphi\}} \text{ if } \psi \implies \eta,\ \eta \implies \mathsf{safe}(B) \text{ and } \eta \wedge \neg B \implies \varphi$$

VCG algorithm: calculation of the weakest preconditions ($\mathbf{wp}^s$)

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

VCG algorithm: Compute VC without preconditions

$$\begin{array}{rcl}
VC^s(\texttt{skip}, \psi) & = & \emptyset \\
VC^s(x \leftarrow E, \psi) & = & \emptyset \\
VC^s(C_1; C_2, \psi) & = & VC^s(C_1, wp^s(C_2, \psi)) \cup \\
 & & VC^s(C_2, \psi) \\
VC^s(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi) & = & VC^s(C_1, \psi) \cup VC^s(C_2, \psi) \\
VC^s(\texttt{while}\, B\, \texttt{do}\, \{\eta\} C, \psi) & = & \{\eta \implies \mathsf{safe}(B)\} \cup \\
 & & \{(\eta \wedge B) \implies wp^s(C, \eta)\} \cup \\
 & & \{(\eta \wedge \neg B) \implies \psi\} \cup VC^s(C, \eta)
\end{array}$$

We define VCG^s as:

$$VCG^s(\{\varphi\} C \{\psi\}) = \{\varphi \implies wp^s(C, \psi)\} \cup VC^s(C, \psi)$$

The function safe for the While$^{\texttt{int}}$ language

$$
\begin{array}{rcl}
\mathsf{safe}(n) & = & \mathsf{true} \\
\mathsf{safe}(x) & = & \mathsf{true} \\
\mathsf{safe}(-E) & = & \mathsf{safe}(E) \\
\mathsf{safe}(E_1 \odot E_2) & = & \mathsf{safe}(E_1) \wedge \mathsf{safe}(E_2) \\
 & & \text{with } \odot \in \{+, -, \times, =, <, \leq\} \\
\mathsf{safe}(E_1 \div E_2) & = & \mathsf{safe}(E_1) \wedge \mathsf{safe}(E_2) \wedge E_2 \neq 0 \\
\mathsf{safe}(\neg B) & = & \mathsf{safe}(B) \\
\mathsf{safe}(B_1 \wedge B_2) & = & \mathsf{safe}(B_1) \wedge (B_1 \implies \mathsf{safe}(B_2)) \\
\mathsf{safe}(B_1 \vee B_2) & = & \mathsf{safe}(B_1) \wedge (\neg B_1 \implies \mathsf{safe}(B_2))
\end{array}
$$

We have

$$\mathcal{A}[\![E]\!]s \neq \mathsf{error} \text{ iff } [\![\mathsf{safe}(E)]\!]s = \mathsf{true}.$$

Exerc. 7.7. *Prove the previous proposition.* $\diamond$

Adequacy of VCG^s

Let $\{\varphi\}C\{\psi\}$ be a Hoare triple and Γ a set of assertions. Then

$$\Gamma \models VCG^s(\{\varphi\}C\{\psi\}) \text{ iff } \Gamma \vdash_{\mathcal{H}_s} \{\varphi\}C\{\psi\}.$$

Proof. ($\Rightarrow$) By induction on the structure of C.

($\Leftarrow$) By induction on the derivation of $\Gamma \vdash_{\mathcal{H}_s} \{\varphi\}C\{\psi\}$.

□

Exerc. 7.8. *Prove the previous result.* $\diamond$

Ex. 7.1.

$$
\begin{array}{rcl}
\mathsf{safe}((x \div y) > 2) & = & \mathsf{safe}(x) \wedge \mathsf{safe}(y) \wedge y \neq 0 \wedge \mathsf{safe}(2) \\
 & = & \mathsf{true} \wedge \mathsf{true} \wedge y \neq 0 \wedge \mathsf{true} \\
 & \equiv & y \neq 0
\end{array}
$$

$$
\begin{array}{rcl}
\mathsf{safe}(7 > x \wedge (x \div y) > 2) & = & \mathsf{safe}(7 > x) \wedge \\
 & & ((7 > x) \rightarrow \mathsf{safe}((x \div y) > 2) \\
 & = & \mathsf{true} \wedge \mathsf{true} \wedge (7 > x \rightarrow (y \neq 0)) \\
 & \equiv & 7 > x \rightarrow y \neq 0
\end{array}
$$

Bounded Arrays: While$^{\texttt{array}[N]}$

- The notion of array introduced before is unrealistic since arrays are virtually infinite.
- we will consider expressions of the for $\texttt{array}[N]$, representing arrays of size N, that admit as valid indexes nonnegative integers below N.
- What to do if the operations refer indexes out of the array limits?
- We consider as error situations.
- We introduce $\texttt{len}(A)$ that given a array A returns its length.

Syntax of language While$^{\texttt{array}[N]}$

For $n \in \mathbf{Num}, x \in \mathbf{Var}, u \in \mathbf{Array}$

$$
\begin{array}{rrll}
Exp_{\texttt{array}[N]} & A ::= & u \mid A[E \triangleright E] \\
\\
Exp_{\texttt{int}} & E ::= & n \mid x \mid -E \mid E + E \mid E - E \\
& & \mid E \times E \mid E \div E \\
& & \mid A[E] \mid \texttt{len}(A) \\
\\
Exp_{\texttt{bool}} & B ::= & \texttt{true} \mid \texttt{false} \mid \neg B \mid E = E \\
& & \mid B < E \mid B \leq B \mid B \wedge B \mid B \ \vee \ B
\end{array}
$$

Semantics of the arithmetic expressions for While$^{\texttt{array}[N]}$

We only need to define the semantics for $Exp_{\texttt{array}[N]}$.

$$
\begin{array}{rcl}
\mathcal{A}[\![u]\!]s & = & s(u) \\
\\
\mathcal{A}[\![A[E \triangleright E']]\!]s & = & \begin{cases}
\mathcal{A}[\![A]\!]s[\mathcal{A}[\![E']\!]s/\mathcal{A}[\![E]\!]s] & \text{if} \\
\qquad \mathcal{A}[\![A]\!]s \neq \mathsf{error} & \\
\qquad \mathcal{A}[\![E]\!]s \neq \mathsf{error} & \\
\quad 0 \leq \mathcal{A}[\![E]\!]s < \mathcal{A}[\![\texttt{len}(A)]\!]s & \\
\qquad \mathcal{A}[\![E']\!]s \neq \mathsf{error} & \\
\mathsf{error} & \text{otherwise.}
\end{cases}
\end{array}
$$

e

$$\begin{aligned}
\mathcal{A}[\![\texttt{len}(A)]\!]s &= N \\
\mathcal{A}[\![A[E]]\!]s &= \begin{cases} \mathcal{A}[\![A]\!]s(\mathcal{A}[\![E]\!]s) & \text{if } \begin{array}{r} \mathcal{A}[\![A]\!]s \neq \mathsf{error} \\ \mathcal{A}[\![E]\!]s \neq \mathsf{error} \\ 0 \leq \mathcal{A}[\![E]\!]s < \mathcal{A}[\![\texttt{len}(A)]\!]s \end{array} \\ \mathsf{error} & \text{otherwise.} \end{cases}
\end{aligned}$$

Extension of VCG for array[N]

We add the following rule to $\mathcal{H}_g$:

$$\frac{}{\{\varphi\}\, u[E] \leftarrow E' \,\{\psi\}} \text{ if } \varphi \implies \mathsf{safe}(u[E \triangleright E']) \text{ and } \varphi \implies \psi[u[E \triangleright E']/u]$$

we extend wp^s and VC^s and safe as follows:

$$\begin{aligned}
wp^s(u[E] \leftarrow E', \psi) &= \mathsf{safe}(u[E \triangleright E']) \wedge \psi[u[E \triangleright E']/u] \\
VC^s(u[E] \leftarrow E', \psi) &= \emptyset
\end{aligned}$$

$$\begin{aligned}
\mathsf{safe}(u) &= \mathsf{true} \\
\mathsf{safe}(\texttt{len}(A)) &= \mathsf{true} \\
\mathsf{safe}(A[E]) &= \mathsf{safe}(A) \wedge \mathsf{safe}(E) \wedge 0 \leq E \leq \texttt{len}(A) \\
\mathsf{safe}(A[E \triangleright E']) &= \mathsf{safe}(A) \wedge \mathsf{safe}(E) \wedge \\
&\quad 0 \leq E \leq \texttt{len}(A) \wedge \mathsf{safe}(E')
\end{aligned}$$

Ex. 7.2.

$$\begin{aligned}
\mathsf{safe}(u[x \div 2]) &= \mathsf{safe}(u) \wedge \mathsf{safe}(x \div 2) \wedge 0 \leq x \div 2 < \texttt{len}(u) \\
&= \mathsf{true} \wedge \mathsf{safe}(x) \wedge \mathsf{safe}(2) \wedge 2 \neq 0 \\
&\quad \wedge 0 \leq x \div 2 < \texttt{len}(u) \\
&= \mathsf{true} \wedge \mathsf{true} \wedge \mathsf{true} \wedge 2 \neq 0 \wedge 0 \leq x \div 2 < \texttt{len}(u) \\
&\equiv 2 \neq 0 \wedge 0 \leq x \div 2 < \texttt{len}(u)
\end{aligned}$$

$$\begin{aligned}
\mathsf{safe}(u[3 \triangleright 10]) &= \mathsf{safe}(u) \wedge \mathsf{safe}(3) \wedge 0 \leq 3 < \texttt{len}(u) \wedge \mathsf{safe}(10) \\
&= \mathsf{true} \wedge \mathsf{true} \wedge 0 \leq 3 < \texttt{len}(u) \wedge \mathsf{true} \\
&\equiv 0 \leq 3 < \texttt{len}(u)
\end{aligned}$$

maxarray

Now one needs consider the range of valid indexes of an array u

$$valid\_range(u, i, j) = 0 \leq i \leq j < \texttt{len}(u) \vee i > j$$

Require: $\{size \geq 1 \wedge valid\_range(u, 0, size - 1)\}$
  $max \leftarrow 0$;
  $i \leftarrow 0$;
  while $i < size$ **do** $\{\theta\}$
    if $u[i] > u[max]$ **then**
      $max \leftarrow i$
    else
      skip;
    $i \leftarrow i + 1$
Ensure: $\{0 \leq max < size \wedge \forall a.0 \leq a < size \rightarrow u[a] \leq u[max]\}$

$$
\begin{aligned}
wp^s(C, \theta) = & \mathsf{safe}(u[i] \ > \ u[max]) \\
& \wedge (u[i] \ > \ u[max] \implies (\mathsf{safe}(i) \wedge \mathsf{safe}(i+1) \\
& \wedge 1 \leq i + 1 \leq size \wedge 0 \leq i < i + 1 \\
& \wedge \forall a.0 \leq a < i + 1 \rightarrow u[a] \leq u[i])) \\
& \wedge (\neg(u[i] \ \leq \ u[max]) \implies (\mathsf{safe}(i+1) \\
& \wedge 1 \leq i + 1 \leq size \wedge 0 \leq max < i + 1 \\
& \wedge \forall a.0 \leq a < i + 1 \rightarrow u[a] \leq u[max])) \\
wp^s(\mathsf{maxarray}, \theta) = & (\mathsf{safe}(0) \wedge \mathsf{safe}(1) \wedge 1 \leq 1 \leq size \\
& \wedge 0 \leq 0 < 1 \wedge \forall a.0 \leq a < 1 \rightarrow u[a] \leq u[0]) \\
VC^s(\mathsf{maxarray}, \psi) = & \{\theta \implies \mathsf{safe}(i < size), \\
& (1 \leq i < size \wedge 0 \leq max < i \wedge \\
& \forall a.0 \leq a < i \rightarrow u[a] \leq u[max]) \implies wp^s(C, \theta), \\
& (1 \leq i = size \wedge 0 \leq max < i \wedge \\
& \forall a.0 \leq a < i \rightarrow u[a] \leq u[max]) \implies \psi\}
\end{aligned}
$$

See [AFPMdS11] Chap. 6.5, 7.

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.