Program verification

Nelma Moreira

Program verification
Lecture 4

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

$\{size \geq 1\}$`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

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

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

Extension of VCG for arrays

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

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

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:

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

Exerc. 4.2. *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 approach 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

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

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

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*)

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

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

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

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

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

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

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

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

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

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

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

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 $\mathcal{H}_s$

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

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

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

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

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

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

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

VCG algorithm: Compute VC without preconditions

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

We define VCG^s as:

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

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

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

We have

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

Exerc. 4.3. *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. 4.4. *Prove the previous result.* ◇

Exemp. 4.1.

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

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

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

- The notion of array introduced before is unrealistic since arrays are virtually infinite.
- we will consider expressions of the for $\mathtt{array}[N]$, 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.
- Introduzimos a expressão $\mathtt{len}(A)$ que dado um array A retorna o seu tamanho.

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

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

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

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

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

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

e

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

Extension of VCG for array[N]

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

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

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

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

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

Examples:

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

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

[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.