Deductive verification

1. Partial and total correctness calculus (Hoare logics).
2. Weak-preconditions and Verification condition generators.
3. Tools for the specification , verification and certification programs: Dafny
4. Correction of imperative and object orient programs with Dafny

Origines

Hoare logics are the base of deductive verification of programs(1969, *An Axiomatic base for Computer Programming*)

Tony Hoare

Inventor also of the quick Sort and has a Turing award from 1980.

Robert Floyd

Some ideas from the 1967 paper *Assigning Meaning to Programs*.

Automatic program verification

Consider the following program to compute $\sum_{m=1}^{100} m$:

$x \leftarrow 0;$
$y \leftarrow 1;$
while $y! = 101$ **do**
  $x \leftarrow x + y;$
  $y \leftarrow y + 1;$

- How can we prove that when the program stops we have $x = \sum_{m=1}^{100} m$?.
- We could execute the program using an operational semantics.
- But if we change the `while`,condition to`y!=c`, for any `c` ?
- To execute for several values of `c|` is not an option

Verification using deductive systems

- Given a program and specification, we want to verify that the program satisfies the specification .
- We considere Hoare logics based on pre and post conditions:

  > A formula is an assertion that if the pre-condition holds before the execution of the program, the post-condition must hold after the program execution.

Example

$x \leftarrow 0;$
$y \leftarrow 1;$
Require: $\{x = 0 \wedge y = 1\}$
while $y! = 101$ **do**
$x \leftarrow x + y;$
$y \leftarrow y + 1;$
Ensure: $\{x = \sum_{n=0}^{100} n\}$

Simple imperative language - While

Syntactic categories

- **Num** integers, n
- **Bool** truth values, `true` and `false`
- **Var** variables, x
- **Aexp** arithmetic expressions, E
- **Bexp** Boolean expressions, B
- **Com** statements/commands, C

BNFs

For n in **Num** and x in **Var**

$$\begin{aligned} E &::= n \mid x \mid E + E \mid E - E \mid E \times E \\ B &::= \mathsf{true} \mid \mathsf{false} \mid E = E \mid E < E \mid !B \mid B \ \wedge \ B \\ C &::= \texttt{skip} \mid x \leftarrow E \mid C \ ; \ C \mid \texttt{if } B \texttt{ then } C \texttt{ else } C \mid \texttt{while } B \texttt{ do } C \end{aligned}$$

Semantics

Expressions denote integers or Booleans.

To evaluate an expression it is needed to know the values of the variables that occur in it

A state s is a function rom variables to values.

The set of states is a set of functions

$$\textbf{State} = \textbf{Var} \to \mathbb{Z}$$

.

The commands are evaluated in a state and can modify the state.

The semantics of a program is the state in which it stops.

The semantics (or meaning) of each command and expression can be defined by a transition system - operational semanticspten ou por funções em domínios – semântica denotacional –or by domain functions – denotational semantics.

Partial and total correctness

We aim to verify that the program has a given property and not necessarily to determine the meaning of it.

In particular, we will consider properties of partial correctness given by logical formulae (φ, ψ) :

> *If the program C is run in a state that satisfies φ, then the state resulting from C's execution will satisfy ψ*

partial correctness+ termination=total correctness

Given the undecidability of the halting problem, the properties of partial correctness are specially important in formal software verification.

Assertions–Hoare Triples

The properties of partial correctness of programs are assertions as:

$$\{\varphi\}\, C\, \{\psi\}$$

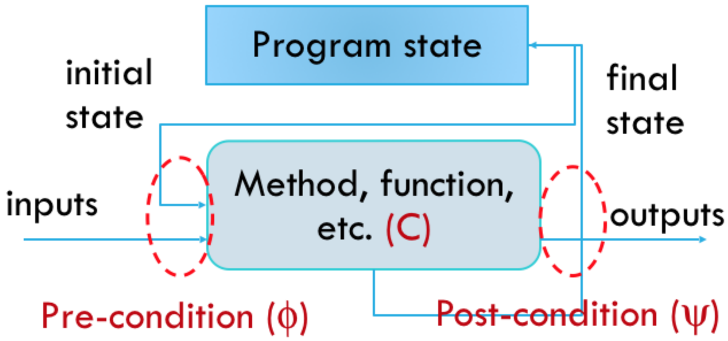
where C is a command and φ and ψ are predicates of a first order logic.

The predicate φ is a *precondition* and ψ is a *postcondition*.

An assertion is valid if:

- if φ is true in the initial state
- If the execution of C terminates in the state s'
- then ψ is true in the state s'

Pre and post conditions



Examples

$\{x = 1\}\texttt{x} \leftarrow \texttt{x} + \texttt{1}\{x = 2\}$ the assertion is true

$\{x = 1\}\texttt{y} \leftarrow \texttt{x}\{y = 1\}$ the assertion is true

$\{x = 1\}\texttt{y} \leftarrow \texttt{x}\{y = 2\}$ the assertion is false

$\{x = x_0 \;\wedge\; y = y_0\}\texttt{r} \leftarrow \texttt{x}\,;\texttt{x} \leftarrow \texttt{y}\,;\texttt{y} \leftarrow \texttt{r}\{x = y_0 \;\wedge\; y = x_0\}$

The variables x_0 and y_0 are called logic variables as they occur only in the conditions.

$\{\mathsf{true}\}C\{\psi\}$ if C stops ψ holds

$\{\varphi\}C\{\mathsf{true}\}$ is always true for any C and φ.

Example

  $x \leftarrow 0;$
  $y \leftarrow 1;$
Require: $\{x = 0 \wedge y = 1\}$
  while $y! = 101$ **do**
    $x \leftarrow x + y;$
    $y \leftarrow y + 1;$
Ensure: $\{x = \sum_{n=0}^{100} n\}$

- We want to infere that $x = \sum_{m=1}^{100} m$ given that before the `while` we had $y = 0$ and $x = 1$.
- It is easy to see that in the end of the loop $y = 101$,but we want the value of x!
- We have to know an *invariante do ciclo* loop invariant:
- In the beginning of each iteration we have

$$x = 1 + 2 + 3 + \cdots + (y - 1)$$

Conditions language

In an assertion, $\{\varphi\}C\{\psi\}$, φ, ψ are formulae $\varphi, \psi, \ldots$ of a first-order language for arithmetics:

- constants 0 and 1 (decimal integers can be seen as abbreviations)
- functional symbols $-$,$+$, $-$ and $\times$ (to form terms)
- Predicate symbols $<$, $=$ (to build predicates)
- logic symbols: operators $\wedge$, $\vee$, etc. and quantifiers (that bound only logical variables) $\forall$, $\exists$.

São interpretadas nos naturais numa estrutura $\mathcal{N} = (\mathbb{N}, \cdot)$ e os estados s, correspondem a atribuições de valores às variáveis.

Se $\mathcal{N} \models_s \varphi$, dizemos que s satifaz φ, i.e., $s \models \varphi$.

Por exemplo, se $s(x) = -2$, $s(y) = 5$, $s(z) = -1$,

$s \models \neg(x + y < z)$ verifica-se

$s \models y - x \times z < z$ não se verifica

Partial correctness

A (Hoare) triple $\{\varphi\}C\{\psi\}$ is satisfied for *partial correctness* if for all states tha satisfy φ, the state that results from running C satisfy ψ, if C stops,

$$\models_{par} \{\varphi\}C\{\psi\}.$$

Note that

while true **do**
  $x \leftarrow 0$;

satisfies all assertions

Total correctness

A triple $\{\varphi\}C\{\psi\}$ is satisfied for total correctness if for all states that satisfy φ, is *ensured that C stops* and the in the resulting state $\psi$is satisfied,

$$\models_{tot} \{\varphi\}C\{\psi\}$$

In this case

while true **do**
  $x \leftarrow 0$;

does not hold for any assertion.

Deduction system for partial correctness/Hoare Logic

- A deduction system is a set of axioms and a set of inference rules.
- A derivation (or proof) is a finite sequence of rule applications and axioms.
- If an assertion $\{\varphi\}C\{\psi\}$ is derived from the partial correctness calculus we say that

$$\vdash_{par} \{\varphi\}C\{\psi\}$$

  is *valid*.
- The calculus is*sound* if:

$$\vdash_{par} \{\varphi\}C\{\psi\} \text{ implies } \models_{par} \{\varphi\}C\{\psi\}.$$

Deduction system for partial correctness/Hoare Logic

[$skip_p$]

$$\{\varphi\}\,\texttt{skip}\,\{\varphi\}$$

[ass_p]

$$\{\varphi[E/x]\}\,x \leftarrow E\,\{\varphi\}$$

[$comp_p$]

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

where $\varphi[E/x]$ is the formula that is obtained substituting x by E.

[if_p]

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

[$while_p$]

$$\frac{\{\psi \ \wedge \ B\}\,C\,\{\psi\}}{\{\psi\}\,\texttt{while}\,B\,\texttt{do}\,C\,\{\psi \ \wedge \ \neg B\}}$$

where ψ is the invariant

[$cons_p$]

$$\frac{\vdash \varphi' \to \varphi \qquad \{\varphi\}\, C\, \{\psi\} \qquad \vdash \psi \to \psi'}{\{\varphi'\}\, C\, \{\psi'\}}$$

Exemp. 2.1. *Show that* $\vdash_{par} \{\mathsf{true}\}z \leftarrow x; z \leftarrow z + y; u \leftarrow z\{u = x + y\}$

$$\dfrac{\dfrac{\{x+y=x+y\}z \leftarrow x\{z+y=x+y\} \qquad \dfrac{\{z+y=x+y\}z \leftarrow z+y\{z=x+y\} \qquad \{z=x+y\}u \leftarrow z\{u=x+y\}}{\{z+y=x+y\}z \leftarrow z+y; u \leftarrow z\{u=x+y\}}\, comp_p}{\{x+y=x+y\}z \leftarrow x; z \leftarrow z+y; u \leftarrow z\{u=x+y\}}\, comp_p}{\{\mathsf{true}\}z \leftarrow x; z \leftarrow z+y; u \leftarrow z\{u=x+y\}}\, cons_p$$

Exerc. 2.1. *Deduce the following assertions*

- $\{x = 1\}\mathtt{x} \leftarrow \mathtt{x} + \mathtt{1}\{x = 2\}$
- $\{x = 1\}\mathtt{y} \leftarrow \mathtt{x}\{y = 1\}$
- $\{x = x_0 \wedge y = y_0\}\mathtt{r} \leftarrow \mathtt{x}\,;\mathtt{x} \leftarrow \mathtt{y}\,;\mathtt{y} \leftarrow \mathtt{r}\{x = y_0 \wedge y = x_0\}$

◇

Exerc. 2.2. *Show that*

$$\vdash_p \{x = r + (y \times q)\}r \leftarrow r - y; q \leftarrow q + 1\, \{x = r + (y \times q)\}$$

◇

Exerc. 2.3. *Show that*

$$\vdash_p \{\mathsf{true}\}z \leftarrow x + 1; \,\mathtt{if}\, z - 1 = 0 \,\mathtt{then}\, y \leftarrow 1 \,\mathtt{else}\, y \leftarrow z\{y = x + 1\}$$

◇

`tableaux` fot partial correctness

Let $C = C_1; C_2; \ldots; C_n$ and we want $\vdash_p \{\varphi\}C\{\psi\}$. We can consider several problems of the form $\vdash_p \{\varphi_i\}C_i\{\varphi_{i+1}\}$. For that we annotate the commands that compose C with formulae φ_i and consider a proof `tableaux` :

$\{\varphi_0\}$
$C_1;$
$\{\varphi_1\}$ justification
$C_2;$
$\vdots$
$\{\varphi_{n-1}\}$ justification
$C_n;$
$\{\varphi_n\}$

Then we need to show

$$\vdash_p \{\varphi_i\}C_{i+1}; \{\varphi_{i+1}\},$$

starting with φ_n. But how to obtain φ_i?

Weakest preconditions (wp)

For each command C and postcondition ψ a formula $wp(C, \psi)$ is the weakest precondition that being true in state s, ensures that in the state s' obtained after the execution of C and if C stops, the postcondition ψ holds.

- $\models_p \{wp(C, \psi)\}C\{\psi\}$
- $\models_p \{\varphi\}C\{\psi\}$ implies $\varphi \to wp(C, \psi)$ (called verification condition)

`tableaux` **for partial correctness**

- a formula φ_i obtained from C_{i+1} and φ_{i+1} is the *weakest precondition* of C_{i+1}
- given the postcondition φ_{i+1}, we can write

$$wp(C_{i+1}, \varphi_{i+1}) = \varphi_i.$$

- From $wp()$and using the consequence rule $(cons_p)$ we can automatically generate the verification conditions ,
- that can be proved automatically or assisted by a solver.
- In general if $\{\varphi\}C\{\psi\}$ the verification condition is:

$$\varphi \to wp(C, \psi)$$

Weakest preconditions - ass_p

Assignment

$$\begin{array}{ll} \{\psi[E/x]\} & \\ x \leftarrow E & \\ \{\psi\} & \qquad ass_p \end{array}$$

A verification condition for $\{\varphi\}x \leftarrow E\{\psi\}$, is

$$\varphi \to \psi[E/x]$$

and $wp(x \leftarrow E, \psi) = \psi[E/x]$.

Exemp. 2.2. *Compute*

1. $wp(x \leftarrow 0, x = 0)$ *is* $0 = 0$.
2. $wp(x \leftarrow x + 1, x > 0)$ *is* $x + 1 > 0$.

Weakest preconditions - $cons_p$

Consequence

The rule $cons_p$ can be applied when $\varphi' \to \varphi$ and we have $\{\varphi\}\,C\,\{\psi\}$. In this case the *tableaux* can have two formulas in a row: φ' and below φ.

$$
\begin{array}{ll}
\{\varphi'\} & \\
\{\varphi\} & \qquad\qquad cons_p
\end{array}
$$

Exerc. 2.4. *Show with a* tableaux $\vdash_p \{y = 5\}x \leftarrow y + 1\{x = 6\}$. $\diamond$

Weakest preconditions if_p

Conditional

We want φ such that $wp(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi) = \varphi$.

$$
\begin{array}{ll}
\{(B \to \varphi_1)\ \wedge\ (\neg B \to \varphi_2)\} & \\
\texttt{if}\, B\, \texttt{then} & \\
\quad \{\varphi_1\} & \\
\quad C_1 & \\
\quad \{\psi\} & \qquad\qquad if_p \\
\texttt{else} & \\
\quad \{\varphi_2\} & \\
\quad C_2 & \\
\quad \{\psi\} & \\
\{\psi\} & \qquad\qquad if_p
\end{array}
$$

We can compute $\{\varphi_1\}C_1\{\psi\}$ e $\{\varphi_2\}C_2\{\psi\}$, and then $\varphi \equiv (B \to \varphi_1)\ \wedge\ (\neg B \to \varphi_2)$, i.e.,

$$wp(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi) = (B \to \varphi_1)\ \wedge\ (\neg B \to \varphi_2)$$

and the verification conditions generated by φ_1 and φ_2.

Exemp. 2.3. *Show with* $a$tableaux

$$\vdash_p \{\mathsf{true}\}$$
$$a \leftarrow x + 1;$$
$$\texttt{if}\, a - 1 = 0\, \texttt{then}$$
$$y \leftarrow 1$$
$$\texttt{else}$$
$$y \leftarrow a$$
$$\{y = x + 1\}$$

$\{\mathsf{true}\}$	
$\{(x = 0 \rightarrow 1 = 1) \ \wedge \ (\neg(x = 0) \rightarrow x + 1 = x + 1)\}$	$cons_p$
$\{(x + 1 - 1 = 0 \rightarrow 1 = x + 1) \ \wedge \ (\neg(x + 1 - 1 = 0) \rightarrow x + 1 = x + 1)\}$	$cons_p$
$a \leftarrow x + 1$	
$\{(a - 1 = 0 \rightarrow 1 = x + 1) \ \wedge \ (\neg(a - 1 = 0) \rightarrow a = x + 1)\}$	ass_p
$\texttt{if}\, a - 1 = 0\, \texttt{then}$	
$\{1 = x + 1\}$	if'_p
$y \leftarrow 1$	
$\{y = x + 1\}$	ass_p
$\texttt{else}$	
$\{a = x + 1\}$	if'_p
$y \leftarrow a$	
$\{y = x + 1\}$	ass_p

Weakest preconditions if_p

We use the following inference rule:

$[if'_p\ ]$

$$\frac{\{\varphi_1\}\, C_1\, \{\psi\} \qquad \{\varphi_2\}\, C_2\, \{\psi\}}{\{(B \rightarrow \varphi_1) \ \wedge \ (\neg B \rightarrow \varphi_2)\}\, \texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2\, \{\psi\}}$$

Exerc. 2.5. *Show that this rule can be deduce from the inference system H* $\diamond$

Weakest preconditions - $while_p$

We want $\vdash_p \{\varphi\}\texttt{while}\, B\, \texttt{do}\, C\, \{\psi\}$.

To use $while_p$ rule we need a formula η such that:

- $\varphi \rightarrow \eta$
- $\eta \ \wedge \ \neg B \rightarrow \psi$ e

- $\vdash_p \{\eta\}\texttt{while}\, B \,\texttt{do}\, C\{\eta \wedge \neg B\}$

Invariant

One cycle invariant $\texttt{while}\, B \,\texttt{do}\, C$ is a formula η such that

$$\models_p \{\eta \wedge B\}C\{\eta\}.$$

Weakest preconditions - *$while_p$*

$$\begin{array}{lll}
\{\varphi\} & & \\
\{\eta\} & & \\
\texttt{while}\, B \,\texttt{do} & & \\
\quad \{\eta \wedge B\} & & \\
\quad C & & \\
\quad \{\eta\} & & \\
\{\eta \wedge \neg B\} & & while_p \\
\{\psi\} & cons_p &
\end{array}$$

We have that $wp(\texttt{while}\, B \,\texttt{do}\, C, \psi) = \eta$, the verification conditions are $\varphi \to \eta$, $\eta \wedge \neg B \to \psi$ and the verification conditions of $\{\eta \wedge B\}C\{\eta\}$.

Weakest preconditions - *$while_p$*

Exemp. 2.4. *Show that*

$$\vdash_p \{\mathsf{true}\}y \leftarrow 1; z \leftarrow 0; \texttt{while}\, \neg z = x\; do\; (z \leftarrow z+1; y \leftarrow y \times z)\{y = x!\}$$

The invariant I is : $y = z!$ and verifies the conditions:

1. Is implied by the precondition of `while` which is $y = 1 \wedge z = 0$:
   $y = 1 \wedge z = 0 \to y = z!$
2. $y = z! \wedge z = x \to y = x!$

We start with I inside the cycle until we obtain I' and show that $I \wedge B \to I'$.

weakest preconditions - $while_p$

$y \leftarrow 1$
$z \leftarrow 0$
$\{y = z!\}$ $\quad$?
$\texttt{while}\, \neg z = x\, \texttt{do}$
$\quad \{y = z! \;\wedge\; \neg z = x\}$
$\quad \{y \times (z+1) = (z+1)!\}$ $\quad$ $cons_p$
$\quad z = z + 1$
$\quad \{y \times z = z!\}$ $\quad$ ass_p
$\quad y = y \times z$
$\quad \{y = z!\}$ $\quad$ ass_p
$\{y = x!\}$ $\quad$?

because $(y = z! \;\wedge\; \neg z = x) \rightarrow y = z! \rightarrow y \times (z+1) = (z+1)!$.

$\{\mathsf{true}\}$
$\{1 = 0!\}$ $\quad$ $cons_p$
$y \leftarrow 1$
$\{y = 0!\}$ $\quad$ ass_p
$z \leftarrow 0$
$\{y = z!\}$ $\quad$ ass_p
$\texttt{while}\, \neg z = x\, \texttt{do}$
$\quad \{y = z! \;\wedge\; \neg z = x\}$
$\quad \{y \times (z+1) = (z+1)!\}$ $\quad$ $cons_p$
$\quad z \leftarrow z + 1$
$\quad \{y \times z = z!\}$ $\quad$ ass_p
$\quad y \leftarrow y \times z$
$\quad \{y = z!\}$ $\quad$ ass_p
$\{y = z! \;\wedge\; z = x\}$ $\quad$ $while_p$
$\{y = x!\}$ $\quad$ $cons_p$

Exerc. 2.6. *Show that*

$$
\begin{aligned}
\vdash_p \{\mathsf{true}\} \\
r \leftarrow x; q \leftarrow 0; \\
\texttt{while}\, y \leq r\, \texttt{do} \\
r \leftarrow r - y; \\
q \leftarrow q + 1 \\
\{r < y \ \wedge \ x = r + (y \times q)\}
\end{aligned}
$$

◇

The condition $x = r + (y \times q)$ is the invariant.

Exerc. 2.7. *Show that*

$\{x \geq 0\} z \leftarrow x; y \leftarrow 0;$ $\texttt{while}\ \neg z = 0\ \texttt{do}\ (y \leftarrow y + 1; z \leftarrow z - 1)\{x = y\}.$

◇