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**

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

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 from variables to values.
- The *set of states* is a set of functions

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

.

- 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 semantics
- 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 its meaning. We call this *axiomatic semantics.*

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:

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

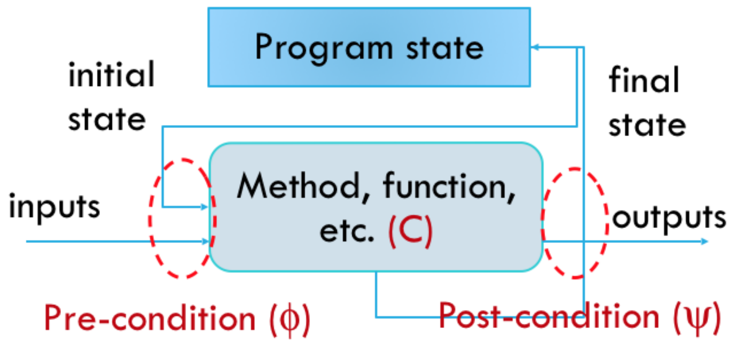
where C is a command and φ and ψ are predicates of a first order logic.

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

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

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

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

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

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 *loop invariant*:
- In the beginning of each iteration we have

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

Condition 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)
- logical symbols: operators $\wedge$, $\vee$, etc. and quantifiers (that bound only logical variables) $\forall$, $\exists$.

Semantics of Conditions

Conditions are interpreted in a model for the integers $\mathcal{Z} = (\mathbb{Z}, \cdot)$ and the states s, are assignments of values to variables.

If $\mathcal{Z} \models_s \varphi$, we say that s satisfies φ, i.e., $s \models \varphi$.

For instance, if $s(x) = -2$, $s(y) = 5$, $s(z) = -1$,

$s \models \neg(x + y < z)$ holds

$s \models y - x \times z < z$ does not hold

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 that in resulting state ψ is satisfied,

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

In this case

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

does not hold for any assertion.

Deductive 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* $\vdash_{par} \{\mathsf{true}\} z \leftarrow x; z \leftarrow z + y; u \leftarrow z\{u = x + y\}$

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

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*

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

◇