Formal verification of programs
Slides block 3

ADA 2024/25

Dep. de Matemática Universidade de Aveiro
Alexandre Madeira
(madeira@ua.pt)

November 25, 2024

References

The main reference for this part of the course is the text:

- Mike Gordon: Specification and Verification I, lecture notes

Back to our initial plan

For the "algorithms development" we mathematically formulate:

- what is a **programming language**
- what is a **program**
- how to **interpret programs**

Formal Semantics of programs

Back to our initial plan

For the "algorithms development" we mathematically formulate:

- what is a **programming language**
- what is a **program**
- how to **interpret programs**

Formal Semantics of programs

To make its "analysis", we mathematically formalise:

- the notions of **property** and **behaviour**
- the notions of **specification** and **algorithm correctness**
- the notion of **correctness proof**

Formal Verification of programs

Formal development of programs/algorithms

- **Formal Specification:** precise (mathematical) description of what a program should do
- **Formal Verification:** (mathematical) proof that a program satisfies a given specification
- **Formal Development:** development of programs/algorithms following a systematic procedure that (mathematically) assures the satisfaction of specification

Formal development of programs/algorithms

- **Formal Specification:** precise (mathematical) description of what a program should do
- **Formal Verification:** (mathematical) proof that a program satisfies a given specification
- **Formal Development:** development of programs/algorithms following a systematic procedure that (mathematically) assures the satisfaction of specification

$$Correctness - by - construction$$

Outline

Program Specification

$$\text{Pre-condition} \xRightarrow{\text{Program Execution}} \text{Post-condition}$$

Program Specification

$$\text{Pre-condition} \xrightarrow{\text{Program Execution}} \text{Post-condition}$$

$$\text{“ } x \text{ greater that } y\text{”} \xrightarrow{\text{Program Execution}} \text{“ } z \text{ is the diference between } x \text{ and } y\text{”}$$

$$\text{“ } x \text{ greater 0”} \xrightarrow{\text{Program Execution}} \text{“ } z \text{ is the square root of } x\text{”}$$

Partial correctness Specification

Hoare Triples

are expressions

$$\{P\}\ C\ \{Q\}$$

where

- C is a **program**
- P and Q are **conditions** on program variables used in C

Partial Correctness

$\{P\}\ C\ \{Q\}$ is true if

whenever C **is executed in a state satisfying** P

and **if** C **terminates**

then the **state in which** C **terminates satisfies** Q

Partial Correctness

$\{P\}\ C\ \{Q\}$ is true if

whenever C **is executed in a state satisfying** P

and **if** C **terminates**

then the **state in which** C **terminates satisfies** Q

Examples

$\{x = 1\}\ x\ := x + 1\ \{x = 2\}$

from any state where $x = 1$,

whenever the program $x\ := x + 1$ terminates

it achieves at a state satisfying $x = 2$

Partial Correctness

$\{P\}\ C\ \{Q\}$ is true if

whenever C **is executed in a state satisfying** P

and **if** C **terminates**

then the **state in which** C **terminates satisfies** Q

Examples

$\{x = 1\}\ x := x + 1\ \{x = 2\}$

from any state where $x = 1$,

whenever the program $x := x + 1$ terminates

it achieves at a state satisfying $x = 2$

- $\{x = 1\}\ x := x + 1\ \{x = 2\}$ holds
- $\{x = 1\}\ x := x + 1\ \{x = 1\}$ does not holds

Hoare Triples

Exercise 1

Discuss the validity of the following Hoare triples:

1. $\{x = a \wedge y = b\}\ \ x\ := y;\ y\ := x\ \{x = b \wedge y = a\}$

Hoare Triples

Exercise 1

Discuss the validity of the following Hoare triples:

1. $\{x = a \wedge y = b\}\ \ x\ := y;\ y\ := x\ \{x = b \wedge y = a\}$
2. $\{x = a \wedge y = b\}\ \ r\ := x;\ x\ := y;\ y\ := r\ \{x = b \wedge y = a\}$

Hoare Triples

Exercise 1

Discuss the validity of the following Hoare triples:

1. $\{x = a \wedge y = b\}\ x := y;\ y := x\ \{x = b \wedge y = a\}$
2. $\{x = a \wedge y = b\}\ r := x;\ x := y;\ y := r\ \{x = b \wedge y = a\}$
3. $\{\mathit{true}\}\ C\ \{Q\}$

Hoare Triples

Exercise 1

Discuss the validity of the following Hoare triples:

1. $\{x = a \wedge y = b\}\ x := y;\ y := x\ \{x = b \wedge y = a\}$
2. $\{x = a \wedge y = b\}\ r := x;\ x := y;\ y := r\ \{x = b \wedge y = a\}$
3. $\{\mathit{true}\}\ C\ \{Q\}$
4. $\{P\}\ C\ \{\mathit{true}\}$

Hoare Triples

Exercise 1

Discuss the validity of the following Hoare triples:

1. $\{x = a \wedge y = b\}\ x := y;\ y := x\ \{x = b \wedge y = a\}$
2. $\{x = a \wedge y = b\}\ r := x;\ x := y;\ y := r\ \{x = b \wedge y = a\}$
3. $\{\mathit{true}\}\ C\ \{Q\}$
4. $\{P\}\ C\ \{\mathit{true}\}$
5. $\{\mathit{true}\}\ C\ \{\mathit{true}\}$

Hoare Triples

Exercise 1

Discuss the validity of the following Hoare triples:

1. $\{x = a \wedge y = b\}\ x\ := y;\ y\ := x\ \{x = b \wedge y = a\}$
2. $\{x = a \wedge y = b\}\ r\ := x;\ x\ := y;\ y\ := r\ \{x = b \wedge y = a\}$
3. $\{\mathtt{true}\}\ C\ \{Q\}$
4. $\{P\}\ C\ \{\mathtt{true}\}$
5. $\{\mathtt{true}\}\ C\ \{\mathtt{true}\}$
6. $\{\mathtt{true}\}\ C\ \{\mathtt{false}\}$

Hoare Triples

Exercise 1

Discuss the validity of the following Hoare triples:

1. $\{x = a \wedge y = b\}\ x := y;\ y := x\ \{x = b \wedge y = a\}$
2. $\{x = a \wedge y = b\}\ r := x;\ x := y;\ y := r\ \{x = b \wedge y = a\}$
3. $\{\mathit{true}\}\ C\ \{Q\}$
4. $\{P\}\ C\ \{\mathit{true}\}$
5. $\{\mathit{true}\}\ C\ \{\mathit{true}\}$
6. $\{\mathit{true}\}\ C\ \{\mathit{false}\}$
7. $\{x = 1\}\ \mathit{while\ true\ do\ skip}\ \{x = 1\}$

Hoare Triples

Exercise 1

Discuss the validity of the following Hoare triples:

1. $\{x = a \land y = b\}\ x := y;\ y := x\ \{x = b \land y = a\}$
2. $\{x = a \land y = b\}\ r := x;\ x := y;\ y := r\ \{x = b \land y = a\}$
3. $\{\mathit{true}\}\ C\ \{Q\}$
4. $\{P\}\ C\ \{\mathit{true}\}$
5. $\{\mathit{true}\}\ C\ \{\mathit{true}\}$
6. $\{\mathit{true}\}\ C\ \{\mathit{false}\}$
7. $\{x = 1\}\ \mathit{while\ true\ do\ skip}\ \{x = 1\}$
8. $\{x = 1\}\ \mathit{while\ true\ do\ skip}\ \{\mathit{false}\}$

Foundations for a design-by-contract methodology?

A program to swap the values of variables x and y

$$\{x = a \land y = b\}\ C\ \{x = b \land y = a\}$$

Development process: to determine a program C that satisfies the triple.

Foundations for a design-by-contract methodology?

A program to swap the values of variables x and y

$$\{x = a \wedge y = b\}\ C\ \{x = b \wedge y = a\}$$

Development process: to determine a program C that satisfies the triple.
The program

$$C \equiv (\ r\ := x;\ x\ := y;\ y\ := r)$$

is a possible implementation, since

$$\{x = a \wedge y = b\}\ \ r\ := x;\ x\ := y;\ y\ := r\ \{x = b \wedge y = a\}$$

holds!

Total correctness

$[P]\ C\ [Q]$ is true if

whenever C **is executed in a state satisfying** P

then, the **execution** C **terminates** and

state in which C **terminates, satisfies** Q

Total correctness

$[P]\ C\ [Q]$ is true if

whenever C **is executed in a state satisfying** P

then, the **execution** C **terminates** and

state in which C **terminates, satisfies** Q

Total correctness

Total correctness = termination + partial correctness

Total correctness

$[P]\ C\ [Q]$ is true if

whenever C **is executed in a state satisfying** P

then, the **execution** C **terminates** and

state in which C **terminates, satisfies** Q

Total correctness

Total correctness = termination + partial correctness

To prove that $[P]\ C\ [Q]$ **is true**:

- prove that C terminates
- prove $\{P\}\ C\ \{Q\}$

Total correctness

Exercise 2

Discuss the validity of the following Hoare triples:

1. $[x = a \wedge y = b]\ \ x\ := y;\ y\ := x\ [x = b \wedge y = a]$

Total correctness

Exercise 2

Discuss the validity of the following Hoare triples:

1. $[x = a \wedge y = b]\ \ x\ := y;\ y\ := x\ [x = b \wedge y = a]$
2. $[x = a \wedge y = b]\ \ r\ := x;\ x\ := y;\ y\ := r\ [x = b \wedge y = a]$

Total correctness

Exercise 2

Discuss the validity of the following Hoare triples:

1. $[x = a \wedge y = b]\ x := y;\ y := x\ [x = b \wedge y = a]$
2. $[x = a \wedge y = b]\ r := x;\ x := y;\ y := r\ [x = b \wedge y = a]$
3. $[\mathit{true}]\ C\ [Q]$

Total correctness

Exercise 2

Discuss the validity of the following Hoare triples:

1. $[x = a \wedge y = b]\ \ x\ := y;\ y\ := x\ [x = b \wedge y = a]$
2. $[x = a \wedge y = b]\ \ r\ := x;\ x\ := y;\ y\ := r\ [x = b \wedge y = a]$
3. $[\mathit{true}]\ C\ [Q]$
4. $[P]\ C\ [\mathit{true}]$

Total correctness

Exercise 2

Discuss the validity of the following Hoare triples:

1. $[x = a \wedge y = b]\ x := y;\ y := x\ [x = b \wedge y = a]$
2. $[x = a \wedge y = b]\ r := x;\ x := y;\ y := r\ [x = b \wedge y = a]$
3. $[\mathit{true}]\ C\ [Q]$
4. $[P]\ C\ [\mathit{true}]$
5. $[P]\ C\ [\mathit{false}]$

Total correctness

Exercise 2

Discuss the validity of the following Hoare triples:

1. $[x = a \wedge y = b]\ \ x\ := y;\ y\ := x\ [x = b \wedge y = a]$
2. $[x = a \wedge y = b]\ \ r\ := x;\ x\ := y;\ y\ := r\ [x = b \wedge y = a]$
3. $[\mathit{true}]\ C\ [Q]$
4. $[P]\ C\ [\mathit{true}]$
5. $[P]\ C\ [\mathit{false}]$
6. $[\mathit{true}]\ C\ [\mathit{true}]$

Total correctness

Exercise 2

Discuss the validity of the following Hoare triples:

1. $[x = a \land y = b]\ x := y;\ y := x\ [x = b \land y = a]$
2. $[x = a \land y = b]\ r := x;\ x := y;\ y := r\ [x = b \land y = a]$
3. $[\mathit{true}]\ C\ [Q]$
4. $[P]\ C\ [\mathit{true}]$
5. $[P]\ C\ [\mathit{false}]$
6. $[\mathit{true}]\ C\ [\mathit{true}]$
7. $[\mathit{false}]\ C\ [\mathit{false}]$

Total correctness

Exercise 2

Discuss the validity of the following Hoare triples:

1. $[x = a \wedge y = b]\ \ x\ := y;\ y\ := x\ [x = b \wedge y = a]$
2. $[x = a \wedge y = b]\ \ r\ := x;\ x\ := y;\ y\ := r\ [x = b \wedge y = a]$
3. $[\mathit{true}]\ C\ [Q]$
4. $[P]\ C\ [\mathit{true}]$
5. $[P]\ C\ [\mathit{false}]$
6. $[\mathit{true}]\ C\ [\mathit{true}]$
7. $[\mathit{false}]\ C\ [\mathit{false}]$
8. $[x = 1]\ \mathit{while\ true\ do\ skip}\ \ [x = 1]$

Total correctness

Exercise 2

Discuss the validity of the following Hoare triples:

1. $[x = a \wedge y = b]\ \ x\ := y;\ y\ := x\ [x = b \wedge y = a]$
2. $[x = a \wedge y = b]\ \ r\ := x;\ x\ := y;\ y\ := r\ [x = b \wedge y = a]$
3. $[\mathit{true}]\ C\ [Q]$
4. $[P]\ C\ [\mathit{true}]$
5. $[P]\ C\ [\mathit{false}]$
6. $[\mathit{true}]\ C\ [\mathit{true}]$
7. $[\mathit{false}]\ C\ [\mathit{false}]$
8. $[x = 1]\ \mathit{while\ true\ do\ skip}\ \ [x = 1]$
9. $[x = 1]\ \mathit{while\ true\ do\ skip}\ \ [\mathit{false}]$

Example

A concrete example

$$\{\texttt{true}\}$$

$$r := x;$$

$$q := 0;$$

$$\texttt{while } y \leq r \texttt{ do } r := r - y;\ q := q + 1$$

$$\{r < y \wedge x = r + (y \times q)\}$$

This triple is true if whenever the execution of the program halts, then q is the quotient and r is the reminder of the division of y by x

Exercises

Exercise 3

1. *Write a partial correctness specification which is true if and only if the program C has the effect of multiplying the values of x and y and storing the result in x.*
2. *Write a specification which is true if the execution of C always halts when execution is started in a state satisfying P.*

The relevance to properly specify

"The program C must set y to the maximum of x and y"

First attempt:

$$[\texttt{true}]\ C\ [y = max(x, y)]$$

The relevance to properly specify

"The program C must set y to the maximum of x and y"

First attempt:

$$[\texttt{true}]\ C\ [y = max(x, y)]$$

Possible implementation(s) for C:

- `if` $x \geq y$ `then` $y := x$ `else skip`

The relevance to properly specify

"The program C must set y to the maximum of x and y"

First attempt:

$$[\texttt{true}]\ C\ [y = max(x, y)]$$

Possible implementation(s) for C:

- $\texttt{if}\ x \geq y\ \texttt{then}\ y := x\ \texttt{else skip}$
- $\texttt{if}\ x \geq y\ \texttt{then}\ x := y\ \texttt{else skip}$

The relevance to properly specify

"The program C must set y to the maximum of x and y"

First attempt:

$$[\texttt{true}]\ C\ [y = max(x, y)]$$

Possible implementation(s) for C:

- $\texttt{if}\ x \geq y\ \texttt{then}\ y := x\ \texttt{else skip}$
- $\texttt{if}\ x \geq y\ \texttt{then}\ x := y\ \texttt{else skip}$
- $y := x$

The relevance to properly specify

"The program C must set y to the maximum of x and y"

First attempt:

$$[\mathtt{true}]\ C\ [y = max(x, y)]$$

Possible implementation(s) for C:

- $\mathtt{if}\ x \geq y\ \mathtt{then}\ \ y\ := x\ \mathtt{else\ skip}$
- $\mathtt{if}\ x \geq y\ \mathtt{then}\ \ x\ := y\ \mathtt{else\ skip}$
- $y := x$
- ...

Therefore the specification is wrong for our purposes!

The relevance to properly specify

"The program C must set y to the maximum of x and y"

First attempt:

$$[\mathtt{true}]\ C\ [y = max(x, y)]$$

Possible implementation(s) for C:

- $\mathtt{if}\ x \geq y\ \mathtt{then}\ \ y\ := x\ \mathtt{else\ skip}$
- $\mathtt{if}\ x \geq y\ \mathtt{then}\ \ x\ := y\ \mathtt{else\ skip}$
- $y := x$
- ...

Therefore the specification is wrong for our purposes!

A **proper specification**:

$$[x = a \wedge y = b]\ C\ [y = max(a, b)]$$

Language to express specification

The predicate statements

$$\texttt{Pred} \ni P ::= t \mid a = a \mid a > a \mid \neg P \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \forall x.P \mid \exists x.P$$

for $t \in \mathbb{B}, a \in \texttt{AExp}, x \in \texttt{Var}$

Usually, we use the usual notations (e.g. $Odd(x)$, $Even(x)$, $Prime(x)$) for the well known predicates

Outline

Floyd-Hoare Rules: Assignment

Substitution

$$P[a/x]$$

denotes the result of substituting the expression a by all the occurrences of the variable x

Floyd-Hoare Rules: Assignment

Substitution

$$P[a/x]$$

denotes the result of substituting the expression a by all the occurrences of the variable x

- $(x = y)[z/x] \Leftrightarrow z = y$

Floyd-Hoare Rules: Assignment

Substitution

$$P[a/x]$$

denotes the result of substituting the expression a by all the occurrences of the variable x

- $(x = y)[z/x] \Leftrightarrow z = y$
- $(x = y)[1/y] \Leftrightarrow x = 1$

Floyd-Hoare Rules: Assignment

Substitution

$$P[a/x]$$

denotes the result of substituting the expression a by all the occurrences of the variable x

- $(x = y)[z/x] \Leftrightarrow z = y$
- $(x = y)[1/y] \Leftrightarrow x = 1$
- $(x = y)[x + 1/x] \Leftrightarrow x + 1 = y$

Floyd-Hoare Rules: Assignment

Skip rule

$$\frac{}{\{P\}\ skip\ \{P\}}(sk)$$

Floyd-Hoare Rules: Assignment

Assignment rule

$$\frac{}{\{P[a/x]\}\ \ x\ := a\ \{P\}}(Assign)$$

Floyd-Hoare Rules: Assignment

Assignment rule

$$\frac{}{\{P[a/x]\}\ \ x\ := a\ \{P\}}(Assign)$$

Example 1

$\{y = 2\}\ x := 2\ \{y = x\}$ holds, since $(y = 2)[x/y] \Leftrightarrow (y = 2)$

Exercise 4

Verify:

- $\{x + 1 = n + 1\}\ \ x\ := x + 1\ \{x = n + 1\}$
- $\{a = a\}\ \ x\ := a\ \{x = a\}$

Floyd-Hoare Rules: Assignment

Exercise 5

Other **WRONG** *attempts on defining a rule for assignments are:*

- $\overline{\{P\}\ \ x := a\ \{P[a/x]\}}$ *and*
- $\overline{\{P\}\ \ x := a\ \{P[x/a]\}}$

Show that these rules are not sound.

Floyd-Hoare Rules: Assignment

Alternative assignment rule (by Floyd)

$$\frac{}{\{P\}\ \ x\ := a\ \{\exists v. \big(x = (a[v/x]) \wedge P[v/x]\big)\}}(Ass')$$

Actually we can prove that (Ass) and (Ass').

Exercise 6

Use it to show that $\{x = 1\}\ x := x + 1\ \{x = 2\}$

Floyd-Hoare Rules: Strengthening and weakening

Precondition Strengthening

$$\frac{P \;\Rightarrow\; P' \qquad \{P'\}\ C\ \{Q\}}{\{P\}\ C\ \{Q\}}$$

Post-condition Weakening

$$\frac{\{P\}\ C\ \{Q'\} \qquad Q' \;\Rightarrow\; Q}{\{P\}\ C\ \{Q\}}$$

Floyd-Hoare Rules: Strengthening and weakening

Exercise 7

Show that:

- $\{x = n\}\ x := x + 1\ \{x = n + 1\}$

Floyd-Hoare Rules: Strengthening and weakening

Exercise 7

Show that:

- $\{x = n\}\ \ x\ := x + 1\ \{x = n + 1\}$
- $\{\mathit{true}\}\ \ x\ := a\ \{x = a\}$
- $\{r = a\}\ \ q\ := 0\ \{r = a + (y \times q)\}$

Floyd-Hoare Rules: Sequential composition

Sequential composition

$$\frac{\{P\}\ C_1\ \{Q'\} \quad \{Q'\}\ C_2\ \{Q\}}{\{P\}\ C_1; C_2\ \{Q\}}$$

Exercise 8

1. *Write a program to swap the values of the variables x and y and verify its correctness.*

Floyd-Hoare Rules: Sequential composition

Sequential composition

$$\frac{\{P\}\ C_1\ \{Q'\} \quad \{Q'\}\ C_2\ \{Q\}}{\{P\}\ C_1; C_2\ \{Q\}}$$

Exercise 8

1. *Write a program to swap the values of the variables x and y and verify its correctness.*
2. *Prove that the program*

$$x := x + y;\ y := x - y;\ x := x - y$$

does the same.

Floyd-Hoare Rules: Sequential composition

Sequential composition

$$\frac{\{P\}\ C_1\ \{Q'\} \quad \{Q'\}\ C_2\ \{Q\}}{\{P\}\ C_1; C_2\ \{Q\}}$$

Exercise 8

1. *Write a program to swap the values of the variables x and y and verify its correctness.*
2. *Prove that the program*

$$x \ := x + y;\ y \ := x - y;\ x \ := x - y$$

   does the same.
3. *Prove that*
   $\{x = r + (y \times q)\}\ \ r \ := r - y;\ q \ := q + 1\ \{x = r + (y \times q)\}$

Floyd-Hoare Rules: conditionals

Conditional

$$\frac{\{P \wedge b\}\ C_1\ \{Q\} \quad \{P \wedge \neg b\}\ C_2\ \{Q\}}{\{P\}\ \texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \{Q\}}$$

Exercise 9

Prove that

$\{\mathit{true}\}$

$\quad \mathit{if}\ x \geq y\ \mathit{then}\ MAX := x\ \mathit{else}\ MAX := y$

$\{MAX = max(x, y)\}$

Floyd-Hoare Rules: conditionals

Exercise 10

Suppose that `While` language is now enriched with the command

$$\mathtt{if}\ b\ \mathtt{then}\ c$$

Introduce a suitable rule for the Floyd-Hoare Calculus and comment the sentence: "this command is just an abbreviation of a `While` command.

Exercise 11

Prove that if $\{P \wedge b\}\ c_1\ \{Q\}$ *and* $\{P \wedge \neg b\}\ c_2\ \{Q\}$ *then*

$$\{P\}\ \mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ (\mathtt{if}\ \neg b\ \mathtt{then}\ c_2)\ \{Q\}$$

Floyd-Hoare Rules: Conjunction and disjunction

Specification conjunction

$$\frac{\{P_1\}\ C\ \{Q_1\} \quad \{P_2\}\ C\ \{Q_2\}}{\{P_1 \wedge P_2\}\ C\ \{Q_1 \wedge Q_2\}}$$

Specification disjunction

$$\frac{\{P_1\}\ C\ \{Q_1\} \quad \{P_2\}\ C\ \{Q_2\}}{\{P_1 \vee P_2\}\ C\ \{Q_1 \vee Q_2\}}$$

Floyd-Hoare Rules: while rule

Invariants

P is said to be invariant of C whenever b holds

$$\{P \wedge b\}\ C\ \{P\}$$

Observe:

- if executing C once preserves the truth of P
- then, executing C any number of times also preserves the truth of P

Floyd-Hoare Rules: while rule

Invariants

P is said to be invariant of C whenever b holds

$$\{P \land b\}\ C\ \{P\}$$

Observe:

- if executing C once preserves the truth of P
- then, executing C any number of times also preserves the truth of P

While rule

$$\frac{\{P \land b\}\ C\ \{P\}}{\{P\}\ \texttt{while}\ b\ \texttt{do}\ C\ \ \{P \land \neg b\}}$$

Floyd-Hoare Rules: while rule

Exercise 12

$$\{x \leq n\}\ \mathit{while}\ x < n\ \mathit{do}\ \ x\ := x + 1\ \ \{x \geq n\}$$

Floyd-Hoare Rules: while rule

Exercise 12

$$\{x \leq n\}\ \mathtt{while}\ x < n\ \mathtt{do}\ \ x\ := x + 1\ \ \{x \geq n\}$$

Exercise 13

For $x, y \in \mathbb{N}$, show that

$\{\mathtt{true}\}$
$\quad r\ := x;$
$\quad q\ := 0;$
$\quad \mathtt{while}\ y \leq r\ \mathtt{do}\ (\ r\ := r - y;\ q\ := q + 1)$
$\{x = r + (y \times q) \land \neg(y \leq r)\}$

Hint: invariant suggestion $x = r + (y \times q)$

Non termination programs

As observed before, `while` b `do` c is the unique command of `While` that potentially causes non-termination

Exercise 14

Comment the statement:

$$\{\mathit{true}\}\ \mathit{while}\ \mathit{true}\ \mathit{do}\ x := 0\ \ \{\mathit{false}\}$$

Finding invariants

invariants shall reflect

- what has been done and what remains to be done
- hold at each iteration of the cycle
- shall give the intended result when the cycle terminates

Finding invariants

invariants shall reflect

- what has been done and what remains to be done
- hold at each iteration of the cycle
- shall give the intended result when the cycle terminates

Example

$\{x = n \wedge y = 1\}$
  $\texttt{while}\ x \neq 0\ \texttt{do}\ \ y\ := y \times x;\ x\ := x - 1$
$\{x = 0 \wedge y = n!\}$

- "what was already calculated": y
- "what remains to be done": $x!$
- "final result": $n!$

Finding invariants

invariants shall reflect

- what has been done and what remains to be done
- hold at each iteration of the cycle
- shall give the intended result when the cycle terminates

Example

$\{x = n \wedge y = 1\}$

$\quad \texttt{while}\ x \neq 0\ \texttt{do}\ \ y\ := y \times x;\ x\ := x - 1$

$\{x = 0 \wedge y = n!\}$

- "what was already calculated": y
- "what remains to be done": $x!$
- "final result": $n!$

$$x! \times y = n!$$

Exercises

Exercise 15

Prove

$\{x = n \wedge y = 1\}$

$\quad \textit{while}\; x \neq 0 \;\textit{do}\;\; y := y \times x;\; x := x - 1$

$\{x = 0 \wedge y = n!\}$

Exercises

Exercise 15

Prove

$\{x = n \wedge y = 1\}$

$\quad \mathit{while}\; x \neq 0 \; \mathit{do} \;\; y := y \times x; \; x := x - 1$

$\{x = 0 \wedge y = n!\}$

Exercise 16

Prove

$\{x = 0 \wedge y = 1\}$

$\quad \mathit{while}\; x < N \; \mathit{do} \;\; x := x + 1; \; y := y \times x$

$\{y = n!\}$

Exercises

Exercise 17

Determine a program c such that, for any $a, b \in \mathbb{N}$,

$$\{x = a \wedge y = b\}\ c\ \{z = a^b\}$$

Outline

The verification of programs

Three ways to verify the vailidy of $\{P\}\ c\ \{Q\}$

- Using the **Floyd-Hoare calculus** (as did in the previous section)
- By calculating and prooving the **generated verification conditions**
- By using **weakest pre-conditions/strongest post-conditions**

The verification of programs

Verification Conditions generation

Use an algorithm to extract the **"proof obligations" of $\{P\}\ c\ \{Q\}$**, i.e. a set of logic statements $V = VC(\{P\}\ c\ \{Q\})$ such that,

if V is true, then $\{P\}\ c\ \{Q\}$ holds

The verification of programs

Verification Conditions generation

Use an algorithm to extract the **"proof obligations" of $\{P\}\ c\ \{Q\}$**, i.e. a set of logic statements $V = VC(\{P\}\ c\ \{Q\})$ such that,

if V is true, then $\{P\}\ c\ \{Q\}$ holds

Weakest pre-conditions (Dijsktra predicate transformers)

Use an algorithm to extract the **"weakest precondition of c wrt Q"**, i.e. the weakest condition $wpc(c, Q)$ that makes the triple $\{wpc(c, Q)\}\ c\ \{Q\}$ valid. Hence,

$P \rightarrow wpc(c, Q)$ is true iff $\{P\}\ c\ \{Q\}$ holds

Verification Conditions Generation

To verify the triple $\{P\}\ c\ \{Q\}$

1. annotate programs
2. calculate $VC(\{P\}\ c\ \{Q\})$ (recursive process)
3. if the (first-order) formulas of $VC(\{P\}\ c\ \{Q\})$ are proved, the triple $\{P\}\ c\ \{Q\}$ is valid

More operationally:

1. Input: a Hoare triple annotated with mathematical statements
2. An algorithm generates the set of verification conditions
3. The verification conditions are passed to a theorem prover which attempts to prove them automatically. Often it requires human aid.
4. if the verification conditions are proved the triple $\{P\}\ c\ \{Q\}$ is valid

Program Annotations

a) Program $C_{\texttt{Fib}}$	b) Annotated program $C^{\mathrm{A}}_{\texttt{Fib}}$
$x := 1\,;$	$x := 1\,;$
$y := 0\,;$	$\{x == 1\}$
$i := 1\,;$	$y := 0\,;$
$\textbf{while}\ i < n\ \textbf{do}$	$\{x == 1\ \&\&\ y == 0\}$
$\{$	$i := 1\,;$
$\quad aux := y\,;$	$\{x == 1\ \&\&\ y == 0\ \&\&\ i == 1\}$
$\quad y := x\,;$	$\textbf{while}\ i < n\ \textbf{do}\,\{i \leq n\ \&\&\ x == \texttt{Fib}(i)\ \&\&\ y == \texttt{Fib}(i-1)\}$
$\quad x := x + aux\,;$	$\{$
$\quad i := i + 1$	$\quad aux := y\,;$
$\}$	$\{i \leq n\ \&\&\ x == \texttt{Fib}(i)\ \&\&\ aux == \texttt{Fib}(i-1)\}$
	$\quad y := x\,;$
	$\{i < n\ \&\&\ x == \texttt{Fib}(i)\ \&\&\ y == \texttt{Fib}(i)\ \&\&\ aux == \texttt{Fib}(i-1)\}$
	$\quad x := x + aux\,;$
	$\{i \leq n\ \&\&\ x == \texttt{Fib}(i) + \texttt{Fib}(i-1)\ \&\&\ y == \texttt{Fib}(i)\}$
	$\quad i := i + 1\,;$
	$\}$

Program Annotations

A program is properly annotated (to the verification conditions generation) if:

statements have been inserted at the following places:

- before c_2 in $c_1; c_2$ and c_2 is not an assignment
- after the "do" in whiles cycle `while` b `do` $\{I\}c_1$

Verification Conditions Generation

Assignments

$$VC(\{P\}\ \ x\ := a\ \{Q\}) = \{P \to Q[a/x]\}$$

Example

$$\begin{aligned} VC(\{x = 0\}\ \ x\ := x + 1\ \{x = 1\}) &= \{x = 0 \to (x = 1)[x + 1/x]\} \\ &= \{x = 0 \to x = 0\} \end{aligned}$$

Verification Conditions Generation

Skip

$$VC(\{P\}\ \texttt{skip}\ \{Q\}) = \{P \to Q\}$$

Example

$+++$

Verification Conditions Generation

Conditionals

$$VC(\{P\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Q\}) \;=\; VC(\{P \wedge b\}\ c_1\ \{Q\}) \cup VC(\{P \wedge \neg b\}\ c_2\ \{Q\})$$

Exercise 18

1. *Suggest a rule for the command* $\mathit{if}\ b\ \mathit{then}\ c$ *with the expected semantics* ($\mathit{if}\ b\ \mathit{then}\ c \equiv \mathit{if}\ b\ \mathit{then}\ c\ \mathit{else}\ \mathit{skip}$)*. Calculate* $VC(\{\mathit{true}\}\ \mathit{if}\ x < 0\ \mathit{then}\ x := -x\ \{x \geq 0\})$
2. *Calculate*

   $$VC(\{\mathit{true}\}\ \mathit{if}\ x \geq y\ \mathit{then}\ M := x\ \mathit{else}\ M := y\ \{M = \max(x, y)\})$$

Verification Conditions Generation

Sequential Compositions

1. If c_n is not an assignment,
$$VC(\{P\}\ c_1; \ldots; c_{n-1}\{R\}c_n\ \{Q\}) \ = \ VC(\{P\}\ c_1; \ldots; c_{n-1}\ \{R\}) \cup VC(\{R\}\ c_n\ \{Q\})$$
2. and
$VC(\{P\}\ c_1; \ldots; c_{n-1};\ x\ := a\ \{Q\}) =$
$VC(\{P\}\ c_1; \ldots; c_{n-1}\ \{Q[a/x]\})$

Exercise 19

Calculate:

$$VC(\{x = a \wedge y = b\}\ \ r\ := x;\ x\ := y;\ y\ := r\ \{x = b \wedge y = a\})$$

Verification Conditions Generation

While

$$
\begin{aligned}
VC(\{P\}\ \texttt{while}\ b\ \texttt{do}\ \{I\}c\ \ \{Q\}) \quad = \quad & \{P \to I\} \\
& \cup \\
& \{(I \wedge \neg b) \to Q\} \\
& \cup \\
& VC(\{I \wedge b\}\ c\ \{I\})
\end{aligned}
$$

Verification Conditions Generation

While

$$VC(\{P\}\ \texttt{while}\ b\ \texttt{do}\ \{I\}c\ \ \{Q\}) \ = \ \{P \rightarrow I\} \cup \{(I \wedge \neg b) \rightarrow Q\} \cup VC(\{I \wedge b\}\ c\ \{I\})$$

Exercise 20

1. *Calculate* $VC(\{s = 2^i\}\ \mathit{while}\ i < n\ \mathit{do}\ \ i\ := i + 1;\ s\ := s \times 2\ \ \{s = 2^i\})$

Verification Conditions Generation

While

$$VC(\{P\}\ \texttt{while}\ b\ \texttt{do}\ \{I\}c\ \ \{Q\}) = \{P \to I\} \cup \{(I \wedge \neg b) \to Q\} \cup VC(\{I \wedge b\}\ c\ \{I\})$$

Exercise 20

1. *Calculate* $VC(\{s = 2^i\}\ \textit{while}\ i < n\ \textit{do}\ \ i\ := i + 1;\ s\ := s \times 2\ \ \{s = 2^i\})$
2. *Verify*
   $\{\textit{true}\}$
   $r\ := x;$
   $q\ := 0;$
   $\textit{while}\ y \leq r\ \textit{do}\ \ r\ := r - y;\ q\ := q + 1$
   $\{x = r + (y \times q) \wedge \neg(y \leq r)\}$ *using verification conditions. Hint: invariant suggestion* $x = r + (y \times q)$

Verification Conditions Generation

Exercise 21

Using verification conditions generator, prove that, for x, y naturals

$\{x = a \wedge y = b\}$

$\quad z := 1;$

$\quad \textit{while}\ y \geq 1\ \textit{do}\ (\ z := x \times z;\ y := y - 1)$

$\{z = a^b\}$

Verification Conditions Generation

Theorem 1

If the statements $VC(\{P\}\ c\ \{Q\})$ *are true, then* $\{P\}\ c\ \{Q\}$ *holds*

Verification Conditions Generation

Theorem 1

If the statements $VC(\{P\}\ c\ \{Q\})$ are true, then $\{P\}\ c\ \{Q\}$ holds

And the converse implication?

Verification Conditions Generation

Theorem 1

If the statements $VC(\{P\}\ c\ \{Q\})$ are true, then $\{P\}\ c\ \{Q\}$ holds

And the converse implication?

Consider the annotated triple

$$\{\texttt{true}\}\ \texttt{while false do}\ \{\texttt{false}\}\ x := 0\ \{\texttt{true}\}$$

Try to to prove it

- using Floyd-Hoare Logic (forget the annotations)
- using the verification conditions generations

what do you conclude?

Verification Conditions Generation

Theorem 1

If the statements $VC(\{P\}\ c\ \{Q\})$ are true, then $\{P\}\ c\ \{Q\}$ holds

And the converse implication?

Consider the annotated triple

$$\{\texttt{true}\}\ \texttt{while false do}\ \{\texttt{false}\}\ x := 0\ \{\texttt{true}\}$$

Try to to prove it

- using Floyd-Hoare Logic (forget the annotations)
- using the verification conditions generations

what do you conclude?

the converse implication does not hold!

Predicate transformer

Goal: find the weakest pre-condition $wpc(c, Q,)$ for which the triple

$$\{wpc(c, Q)\}\ c\ \{Q\}$$

Then

$$\{P\}\ c\ \{Q\} \textbf{ iff } P \to wpc(c, Q)$$

Predicate transformer

Goal: find the weakest pre-condition $wpc(c, Q,)$ for which the triple

$$\{wpc(c, Q)\}\ c\ \{Q\}$$

Then

$$\{P\}\ c\ \{Q\}\ \textbf{iff}\ P \rightarrow wpc(c, Q)$$

A program is understood as a "predicates transformer", that transforms post-conditions and programs into "weakest pre-conditions"

Predicate transformers

Assignment

$$wpc(\ x\ := a, Q) = Q[a/x]$$

Exercise 22

Calculate

- $wpc(\ x\ := x + y, y > x)$
- $wpc(\ y\ := 2 \times y, y < 5)$
- $wpc(\ y\ := 2 \times y, even(y))$

Predicate transformers

Sequential Composition

$$wpc(c_1; c_2, Q) = wpc(c_1, wpc(c_2, Q))$$

Exercise 23

Calculate

- $wpc(\ x\ := z + 1;\ y\ := x + y, y > 5)$
- $wpc(\ r\ := x;\ x\ := y;\ y\ := r, x = b \wedge y = a)$

Predicate transformers

Conditionals

$$wpc(\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, Q) = (b \to wpc(c_1, Q)) \wedge (\neg b \to wpc(c_2, Q))$$

Exercise 24

Calculate

- $wpc(\mathit{if}\; x \leq y\; \mathit{then}\;\; m \; := x \; \mathit{else}\;\; m \; := y, m = \min(x, y))$

Exercise

Exercise 25

Prove

$$\{x \leq 7\} \;\; x := 5; \textit{if}\; x \leq 7 \;\textit{then}\; x := x + 2 \;\textit{else}\; \textit{skip}\; \{x = 7\}$$

using:

1. *Weakest pre-conditions*
2. *Verification conditions*
3. *Hoare Logic*

Predicate transformers

While

can not in general compute a finite formula, other techniques shall be considered (not in this course). A sound rule:

$$wpc(\texttt{while } b \texttt{ do } c\ , Q) = \texttt{if } b \texttt{ then } wpc(c, wpc(\texttt{while } b \texttt{ do } c\ , Q)) \texttt{ else } Q$$

Dijkstra's Guarded Command Language (GCL)

Guarded Command Language (GCL)

$$c := \texttt{skip} \mid x := a \mid c_1; c_2 \mid \texttt{assert}\ \alpha \mid \texttt{assume}\ \alpha \mid havoc\ x \mid c || c$$

Hoare Logic Rules for GCL

semantics of skip and $x := a$ define as for While language

(assert) $\dfrac{P \rightarrow \alpha}{\{P\}\ \texttt{assert}\ \alpha\ \{P \wedge \alpha\}}$

(assume) $\dfrac{}{\{P\}\ \texttt{assume}\ \alpha\ \{P \wedge \alpha\}}$

(choice) $\dfrac{\{P\}\ c_1\ \{Q\} \quad \{P\}\ c_2\ \{Q\}}{\{P\}\ c_1 || c_2\ \{Q\}}$

Dijkstra's Guarded Command Language (GCL)

Exercise 26

1. *What is*
   - $wpc(\mathtt{assert}\ P, C)$
   - $wpc(\mathtt{assume}\ P, C)$
2. *Given conditions P and Q and a program C, determine a program C' in the Dijstra's guarded language such that $\{P\}\ C\ \{Q\}$ iff $\{\mathtt{true}\}\ C'\ \{\mathtt{true}\}$. Prove it!*

Dijkstra's Guarded Command Language (GCL)

GCL captures Conditionals:

For instance,

$\{P\}$ `if` b `then` c_1 `else` c_2 $\{Q\}$ iff $\{P\}$ `assert` b ; $c_1 || $`assert` $\neg b$; c_2 $\{Q\}$

We can integrate specification in the code:

$$\{P\}\ c\ \{Q\}$$

iff

$$\{\texttt{true}\}\ \texttt{assume}\ P\ ; c; \texttt{assert}\ Q\ \ \{\texttt{true}\}$$

Exercise 27

Prove the previous observations!

This was just an appetizer...