Formal methods

Proving partial correctness of programs

Judgements

- Three kinds of things that could be true or false have been introduced
  - Statements of mathematics, e.g. $(\mathtt{X}+\mathtt{1})^2 = \mathtt{X}^2+\mathtt{2}\times\mathtt{X}+\mathtt{1}$
  - Partial correctness specifications $\{P\}\ C\ \{Q\}$
  - Total correctness specifications $[P]\ C\ [Q]$

Terms from formal logic

- **Floyd-Hoare logic (FHL)** gives rules for proving the partial and total correctness of programs, i.e. terms $\vdash \{P\}\ C\ \{Q\}$ and $\vdash [P]\ C\ [Q]$
- **Predicate calculus** gives rules for proving theorems of logic
- **Arithmetics** gives decision rules for proving statements about integers
- **Theorems** are statements, which can be proved to be true.
- **Axioms** are statements which are assumed to be true.
- $\vdash S$ means that S can be proved (unconditionally) using proof rules
- $\Gamma \vdash S$ means that S can be deduced from the assumptions (from axioms) $\Gamma = \{A_1, A_2, \ldots, A_n\}$

Terms from proof theory

- ***Deduction (proof)*** - sequence (tree) of *statements* where every statement is either
  - an *axiom* *or*
  - deduced from true statements by proof rules
- **Properties of the proof rules:**
  - *Correctness (soundness)* - it is **not possible** to deduce something that is **not correct** from correct assumptions.
  - *Completeness* - **all** statements that are **correct are deducible** from axioms using the proof rules.
- ***Deduction system*** ≅ ***set of axioms (or axiom schemas) + set of deduction rules***

FHL deduction systems

Let us have some programing language *PL* then in FHL for this *PL*

- there is an axiom or inference rule for each command of the *PL*
- axioms are given as axiom schemas which can be instantiated for particular specification (Hoare triple)
- application of rules in the proof is determined by the syntactical structure of the program

FHL deduction systems

- **The inference rules of Floyd-Hoare logic will be specified with a notation of the form**

$$\frac{\vdash\ S_1,\ \ldots\ ,\ \vdash\ S_n}{\vdash\ S}$$

  - **This means the** *conclusion* $\vdash S$ **may be deduced from the** *hypotheses* $\vdash S_1$, ... , $\vdash S_n$
  - **The hypotheses can either all be theorems of Floyd-Hoare logic**
  - **or a mixture of theorems of Floyd-Hoare logic and theorems of predicate calculus**

SKIP

- **Syntax:** `SKIP`
- **Semantics: the state is unchanged**

The Skip Axiom

$$\vdash \ \{P\}\ \texttt{SKIP}\ \{P\}$$

SKIP

- It is a simple axiom schema
  - P can be instantiated with different values
- Instances of the skip axiom are:
  - $\vdash\ \{\mathtt{Y} = \mathtt{2}\}\ \mathtt{SKIP}\ \{\mathtt{Y} = \mathtt{2}\}$
  - $\vdash\ \{\mathtt{T}\}\ \mathtt{SKIP}\ \{\mathtt{T}\}$
  - $\vdash\ \{\mathtt{R=X+(Y} \times \mathtt{Q)}\}\ \mathtt{SKIP}\ \{\mathtt{R=X+(Y} \times \mathtt{Q)}\}$

Assignment

- **Syntax:** V := E
- **Semantics: the state is changed by assigning the value of the term E to the variable V**
- **Example:** `X:=X+1`
  - **This adds one to the value of the variable** `X`

Substitution Notation

- Define $P\texttt{[}E/V\texttt{]}$ to mean the result of replacing all occurrences of V in P by E
  - Read $P\texttt{[}E/V\texttt{]}$ as 'P with E for V'
  - For example,

$$(\texttt{X}+1 > \texttt{X})\texttt{[Y+Z/X]} \;=\; ((\texttt{Y}+\texttt{Z})+1 > \texttt{Y}+\texttt{Z})$$

Assignment Axiom

The Assignment Axiom

$$\vdash \ \{P[E/V]\}\ V := E\ \{P\}$$

Where V is any variable, E is any expression, P is any statement and the notation $P[E/V]$ denotes the result of substituting the term E for all occurrences of the variable V in the statement P.

Assignment Axiom

$$\vdash \ \{P[E/V]\}\ V := E\ \{P\}$$

- **The assignment axiom says that**
  - **the value of a variable** V *after* **executing an assignment command** $V := E$
  - **equals the value of the expression** E **in the state** *before* **executing it**

Assignment Axiom

- **If a statement P is to be true *after* the assignment**
- **Then the statement obtained by substituting E for V in P must be true *before* executing it**
- **Every statement about V in the postcondition, must correspond to a statement about E in the precondition**
  - **In the initial state V has a value which is about to be lost**

Assignment Axiom

$$\vdash \ \{P[E/V]\}\ V := E\ \{P\}$$

- **Instances of the assignment axiom are**
  - $\vdash \ \{\mathtt{Y} = 2\}\ \mathtt{X} := 2\ \{\mathtt{Y} = \mathtt{X}\}$
  - $\vdash \ \{\mathtt{X} + 1 = \mathtt{n} + 1\}\ \mathtt{X} := \mathtt{X} + 1\ \{\mathtt{X} = \mathtt{n} + 1\}$
  - $\vdash \ \{E = E\}\ \mathtt{X} := E\ \{\mathtt{X} = E\}$ **(if** $\mathtt{X}$ **does not occur in** E**)**

Precondition strengthening

$$\frac{\vdash\ S_1,\ \ldots\ ,\ \vdash\ S_n}{\vdash\ S}$$

means $\vdash\ S$ **can be deduced from** $\vdash\ S_1, \ldots, \vdash\ S_n$

- **Using this notation, the rule of precondition strengthening is**

Precondition strengthening

$$\frac{\vdash\ P \Rightarrow P', \qquad \vdash\ \{P'\}\ C\ \{Q\}}{\vdash\ \{P\}\ C\ \{Q\}}$$

Precondition strengthening

- From
  - $\vdash$ X=n $\Rightarrow$ X+1=n+1
    - trivial arithmetical fact
  - $\vdash \{X + 1 = n + 1\}\ X := X + 1\ \{X = n + 1\}$
    - instance of the assignment axiom
- It follows by precondition strengthening that

$$\vdash \{X = n\}\ X := X + 1\ \{X = n + 1\}$$

Postcondition weakening

- **Just as the previous rule allows the precondition of a partial correctness specification to be strengthened, the following one allows us to weaken the postcondition**

Postcondition weakening

$$\frac{\vdash \ \{P\}\ C\ \{Q'\}, \qquad \vdash \ Q' \Rightarrow Q}{\vdash \ \{P\}\ C\ \{Q\}}$$

Example

- **Here is a little formal proof**

1. $\vdash$ $\{\texttt{R=X} \wedge \texttt{0=0}\}$ $\texttt{Q:=0}$ $\{\texttt{R=X} \wedge \texttt{Q=0}\}$ By the assignment axiom
2. $\vdash$ $\texttt{R=X} \Rightarrow \texttt{R=X} \wedge \texttt{0=0}$ By pure logic
3. $\vdash$ $\{\texttt{R=X}\}$ $\texttt{Q:=0}$ $\{\texttt{R=X} \wedge \texttt{Q=0}\}$ By precondition strengthening
4. $\vdash$ $\texttt{R=X} \wedge \texttt{Q=0} \Rightarrow \texttt{R=X+(Y} \times \texttt{Q)}$ By laws of arithmetic
5. $\vdash$ $\{\texttt{R=X}\}$ $\texttt{Q:=0}$ $\{\texttt{R=X+(Y} \times \texttt{Q)}\}$ By postcondition weakening

- **The rules precondition strengthening and post-condition weakening are sometimes called the** *rules of consequence*

Sequences

- **Syntax:** $C_1; \cdots ;C_n$
- **Semantics: the commands** $C_1, \cdots , C_n$ **are executed in that order**
- **Example:** `R:=X; X:=Y; Y:=R`
  - **The values of** `X` **and** `Y` **are swapped using** `R` **as a temporary variable**
  - **This command has the** *side effect* **of changing the value of variable** `R` **to the old value of variable** `X`

Sequencing rule

- **The next rule enables a partial correctness specification for a sequence $C_1;C_2$ to be derived from specifications for C_1 and C_2**

The sequencing rule

$$\frac{\vdash \ \{P\}\ C_1\ \{Q\}, \qquad \vdash \ \{Q\}\ C_2\ \{R\}}{\vdash \ \{P\}\ C_1;C_2\ \{R\}}$$

Example

$$\text{(i)} \quad \vdash \{\mathtt{X=x} \wedge \mathtt{Y=y}\}\ \mathtt{R:=X}\ \{\mathtt{R=x} \wedge \mathtt{Y=y}\}$$

$$\text{(ii)} \quad \vdash \{\mathtt{R=x} \wedge \mathtt{Y=y}\}\ \mathtt{X:=Y}\ \{\mathtt{R=x} \wedge \mathtt{X=y}\}$$

$$\text{(iii)} \quad \vdash \{\mathtt{R=x} \wedge \mathtt{X=y}\}\ \mathtt{Y:=R}\ \{\mathtt{Y=x} \wedge \mathtt{X=y}\}$$

Hence by (i), (ii) and the sequencing rule

$$\text{(iv)} \quad \vdash \{\mathtt{X=x} \wedge \mathtt{Y=y}\}\ \mathtt{R:=X;\ X:=Y}\ \{\mathtt{R=x} \wedge \mathtt{X=y}\}$$

Hence by (iv) and (iii) and the sequencing rule

$$\text{(v)} \quad \vdash \{\mathtt{X=x} \wedge \mathtt{Y=y}\}\ \mathtt{R:=X;\ X:=Y;\ Y:=R}\ \{\mathtt{Y=x} \wedge \mathtt{X=y}\}$$

Blocks

- **Syntax:** `BEGIN VAR` V_1`;` $\cdots$ `VAR` V_n`;` C `END`

- **Semantics: the command C is executed, and then the values of $V_1, \cdots, V_n$ are restored to the values they had before the block was entered**

  - **The initial values of $V_1, \cdots, V_n$ inside the block are unspecified**

- **Example:** `BEGIN VAR R; R:=X; X:=Y; Y:=R END`

- **This command does *not* have a side effect on the variable** `R`

Block rule

- **The block rule takes care of local variables**

The block rule

$$\frac{\vdash\ \{P\}\ C\ \{Q\}}{\vdash\ \{P\}\ \texttt{BEGIN VAR}\ V_1;\ \ldots;\ \texttt{VAR}\ V_n;\ C\ \texttt{END}\ \{Q\}}$$

where none of the variables $V_1, \ldots, V_n$ **occur in** P **or** Q**.**

Example

- $\vdash$ `{X=x` $\wedge$ `Y=y} R:=X; X:=Y; Y:=R {Y=x` $\wedge$ `X=y}`
- **it follows by the block rule that**

$\vdash$ `{X=x` $\wedge$ `Y=y}`

```
    BEGIN VAR R; R:=X; X:=Y; Y:=R END
```

`{Y=x` $\wedge$ `X=y}`

- **since** `R` **does not occur in** `X=x` $\wedge$ `Y=y` **or** `X=y` $\wedge$ `Y=x`

Conditionals

- **Syntax:** $\texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2$
- **Semantics:**
  - **If the statement S is true in the current state, then C_1 is executed**
  - **If S is false, then C_2 is executed**

The conditional rule

$$\frac{\vdash\ \{P \wedge S\}\ C_1\ \{Q\}, \qquad \vdash\ \{P \wedge \neg S\}\ C_2\ \{Q\}}{\vdash\ \{P\}\ \texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2\ \{Q\}}$$

Conditionals

- **Suppose we are given**

$$\vdash \{\mathtt{T} \wedge \mathtt{X}{\geq}\mathtt{Y}\}\ \mathtt{MAX:=X}\ \{\mathtt{MAX=max(X,Y)}\}$$

$$\vdash \{\mathtt{T} \wedge \neg(\mathtt{X}{\geq}\mathtt{Y})\}\ \mathtt{MAX:=Y}\ \{\mathtt{MAX=max(X,Y)}\}$$

- **Then by the conditional rule it follows that**

$$\vdash \{\mathtt{T}\}\ \mathtt{IF}\ \mathtt{X}{\geq}\mathtt{Y}\ \mathtt{THEN}\ \mathtt{MAX:=X}\ \mathtt{ELSE}\ \mathtt{MAX:=Y}\ \{\mathtt{MAX=max(X,Y)}\}$$

WHILE command

- **Syntax:** `WHILE` S `DO` C
- **Semantics:**
  - **If the statement S is true in the current state, then C is executed and the `WHILE`-command is repeated**
  - **If S is false, then nothing is done**
  - **Thus C is repeatedly executed until the value of S becomes false**
  - **If S never becomes false, then the execution of the command never terminates**
- **Example:** `WHILE ¬(X=0) DO X:= X-2`

Invariants

- Suppose $\vdash \{P \wedge S\}\ C\ \{P\}$
- then P is an *invariant* of C whenever S holds
- The `WHILE`-rule says that
  - if P is an invariant of the $\boxed{\text{body}}$ of a `WHILE`-command whenever the test condition holds
  - then P is an invariant of the $\boxed{\text{whole}}$ `WHILE`-command

Invariants

- **In other words**
  - **if executing C *once* preserves the truth of P**
  - **then executing C *any number of times* also preserves the truth of P**
- **The `WHILE`-rule also expresses the fact that after a `WHILE`-command has terminated, the test must be false**
  - **Otherwise, it wouldn't have terminated**

WHILE-rule

The $\texttt{WHILE}$**-rule**

$$\frac{\vdash\ \{P \wedge S\}\ C\ \{P\}}{\vdash\ \{P\}\ \texttt{WHILE}\ S\ \texttt{DO}\ C\ \{P \wedge \neg S\}}$$

P *S*

$\vdash$ {X=R+(Y×Q) ∧ Y≤R} BEGIN R:=R-Y; Q:=Q+1 END {X=R+(Y×Q)}

Hence by the WHILE**-rule with** P = 'X=R+(Y×Q)'

$\vdash$ {X=R+(Y×Q)}
WHILE Y≤R DO
BEGIN R:=R-Y; Q:=Q+1 END
{X=R+(Y×Q) ∧ ¬(Y≤R)}

Example: sequential composition

From

$$\vdash \{\mathtt{X{=}R{+}(Y{\times}Q)}\}$$
$$\mathtt{WHILE\ Y{\le}R\ DO}$$
$$\mathtt{BEGIN\ R{:=}R{-}Y;\ Q{:=}Q{+}1\ END}$$
$$\{\mathtt{X{=}R{+}(Y{\times}Q)} \wedge \neg(\mathtt{Y{\le}R})\}$$

$$\vdash \{\mathtt{T}\}\ \mathtt{R{:=}X;\ Q{:=}0}\ \{\mathtt{X{=}R{+}(Y{\times}Q)}\}$$

deduce

$$\vdash \{\mathtt{T}\}$$
$$\mathtt{R{:=}X;}$$
$$\mathtt{Q{:=}0;}$$
$$\mathtt{WHILE\ Y{\le}R\ DO}$$
$$\mathtt{BEGIN\ R{:=}R{-}Y;\ Q{:=}Q{+}1\ END}$$
$$\{\mathtt{R{<}Y} \wedge \mathtt{X{=}R{+}(Y{\times}Q)}\}$$

How to find an invariant

- **Look at the facts:**
  - **It must hold initially**
  - **With the negated test it must establish the result**
  - **The body must leave it unchanged**
- **Think about how the loop works**
  - **The invariant says that what** *has been done so far* **together with what** *remains to be done* **gives** *the desired result*

Example

- **Consider a factorial program**

$$
\begin{array}{l}
\{\texttt{X=n} \wedge \texttt{Y=1}\} \\
\ \texttt{WHILE X}\neq\texttt{0 DO} \\
\quad\ \texttt{BEGIN Y:=Y}\times\texttt{X; X:=X-1 END} \\
\{\texttt{X=0} \wedge \texttt{Y=n!}\}
\end{array}
$$

Example

```
{X=n ∧ Y=1}
 WHILE X≠0 DO
   BEGIN Y:=Y×X; X:=X-1 END
{X=0 ∧ Y=n!}
```

- **Look at the Facts**
  - **Finally** `X=0` **and** `Y=n!`
  - **Initially** `X=n` **and** `Y=1`
  - **On each loop** `Y` **is increased and,** `X` **is decreased**
- **Think how the loop works**
  - `Y` **holds the result so far**
  - `X!` **is what remains to be computed**
  - `n!` **is the desired result**
- **The invariant is** `X!×Y = n!`

Example 2

```
{X=0 ∧ Y=1}
 WHILE X<N DO
   BEGIN X:=X+1; Y:=Y×X END
{Y=N!}
```

- **Look at the Facts**
  - **Finally** `X=N` **and** `Y=N!`
  - **Initially** `X=0` **and** `Y=1`
  - **On each iteration both** `X` **an** `Y` **increase**

Example 2

```
{X=0 ∧ Y=1}
 WHILE X<N DO
   BEGIN X:=X+1; Y:=Y×X END
{Y=N!}
```

- **An invariant is** `Y = X!`
- **At end need** `Y = N!`
- **Ah Ha!: invariant needed:** $\boxed{\mathtt{Y = X!} \land \mathtt{X} \leq \mathtt{N}}$
- **At end** $\neg(\mathtt{X<N}) \Rightarrow \mathtt{X=N}$

Conjunction and disjunction

Specification conjunction

$$\frac{\vdash\ \{P_1\}\ C\ \{Q_1\}, \qquad \vdash\ \{P_2\}\ C\ \{Q_2\}}{\vdash\ \{P_1 \wedge P_2\}\ C\ \{Q_1 \wedge Q_2\}}$$

Specification disjunction

$$\frac{\vdash\ \{P_1\}\ C\ \{Q_1\}, \qquad \vdash\ \{P_2\}\ C\ \{Q_2\}}{\vdash\ \{P_1 \vee P_2\}\ C\ \{Q_1 \vee Q_2\}}$$

Summary

- We have shown how rules can be devised that allow us to make judgements about partial correctness statements
- It can be hard to get the rules right in the first place
- We can use the rules to prove that programs meet their specifications

Summary

- The rules reduce the proof to symbol pushing
  - With practice this is routine
  - The hard part is in formulating invariants