Validity

- **Important to establish the validity of axioms and rules**
- **Can give a *formal semantics* of our little programming language**
  - **then *prove* axioms and rules of inference of Floyd-Hoare logic are sound**
  - **this will only increase our confidence in the axioms and rules to the extent that we believe the correctness of the formal semantics!**
- **Won't go into this topic here (some discussion later)**
- **The Assignment Axiom is not valid for 'real' programming languages**
  - **In an early PhD on Hoare Logic G. Ligler showed that the assignment axiom can fail to hold in six different ways for the language Algol 60**

Expressions with Side-effects

- **The validity of the assignment axiom depends on expressions not having side effects**
- **Suppose that our language were extended so that it contained the 'block expression'**

```
BEGIN Y:=1; 2 END
```

  - **this expression has value `2`, but its evaluation also 'side effects' the variable `Y` by storing `1` in it**

- **If the assignment axiom applied to block expressions, then it could be used to deduce**

$$\vdash \{\texttt{Y=0}\}\ \texttt{X:=BEGIN Y:=1; 2 END}\ \{\texttt{Y=0}\}$$

  - **since `(Y=0)[E/X] = (Y=0)` (because `X` does not occur in `(Y=0)`)**
  - **this is clearly false; after the assignment `Y` will have the value `1`**

An Example Formal Proof

- **Here is a little formal proof**

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

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

The sequencing rule

- **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**
  - **note *side effect*: value of `R` changed to the old value of `X`**

The sequencing rule

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

Example Proof

Example: By the assignment axiom:

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

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

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

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

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

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

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

Blocks

- **Syntax:** `BEGIN VAR` V_1`;` $\cdots$ `VAR` V_n`;` C `END`
- **Semantics: command C is executed, 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`
  - **the values of `X` and `Y` are swapped using `R` as a temporary variable**
  - **this command does *not* have a side effect on the variable `R`**

The Block Rule

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

The block rule

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

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

- **Note that the block rule is regarded as including the case when there are no local variables (the '$n = 0$' case)**

The Side Condition

- **The syntactic condition that none of the variables $V_1, \ldots, V_n$ occur in P or Q is an example of a *side condition***
- **From**
  $\vdash$ {`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`
- **However from**
  $\vdash$ {`X=x` $\wedge$ `Y=y`} `R:=X; X:=Y` {`R=x` $\wedge$ `X=y`}
  one *cannot* deduce
  $\vdash$ {`X=x` $\wedge$ `Y=y`} `BEGIN VAR R; R:=X; X:=Y END` {`R=x` $\wedge$ `X=y`}
  since `R` occurs in `R=x` $\wedge$ `X=y`

Exercises

- **Consider the specification**

  $$\{X=x\}\ \texttt{BEGIN VAR X; X:=1 END}\ \{X=x\}$$

  Can this be deduced from the rules given so far?

  (i) if so, give a proof of it

  (ii) if not, explain why not and suggest additional rules and/or axioms to enable it to be deduced

- **Is the following true?**

  $$\vdash \{X=x \wedge Y=y\}\ \texttt{X:=X+Y; Y:=X-Y; X:=X-Y}\ \{Y=x \wedge X=y\}$$

  - **if so prove it**
  - **if not, give the circumstances when it fails**

- **Show**

  $$\vdash \{X=R+(Y\times Q)\}\ \texttt{BEGIN R:=R-Y; Q:=Q+1 END}\ \{X=R+(Y\times Q)\}$$

Two-armed conditionals

- **Syntax:** `IF` S `THEN` C_1 `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**
- **Example:** `IF X<Y THEN MAX:=Y ELSE MAX:=X`
  - **the value of the variable `MAX` it set to the maximum of the values of `X` and `Y`**
- **One-armed conditional is defined by:**
  - `IF` S `THEN` C $\quad =_{define} \quad$ `IF` S `THEN` C `ELSE SKIP`

The Conditional Rule

The conditional rule

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

- **From Assignment Axiom + Precondition Strengthening and**

  $\vdash\ \texttt{(X}\geq\texttt{Y}\ \Rightarrow\ \texttt{X = max(X,Y))}\ \wedge\ \texttt{(}\neg\texttt{(X}\geq\texttt{Y)}\ \Rightarrow\ \texttt{Y = max(X,Y))}$

  it follows that

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

  and

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

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

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

- **A one-armed conditional rule is derived later**

`WHILE`-commands

- **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`
  - if the value of `X` is non-zero, then its value is decreased by `2` and then the process is repeated
- **This `WHILE`-command will terminate (with `X` having value `0`) if the value of `X` is an even non-negative number**
  - in all other states it will not terminate

Invariants

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

The `WHILE`-Rule

The `WHILE`-rule

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

- **It is easy to show**

  $\vdash$ $\{$`X=R+(Y`$\times$`Q)`$\wedge$`Y`$\leq$`R`$\}$ `BEGIN R:=R-Y; Q:=Q+1 END` $\{$`X=R+(Y`$\times$`Q)`$\}$

- **Hence by the `WHILE`-rule with** $P =$ '`X=R+(Y`$\times$`Q)`' **and** $S =$ '`Y`$\leq$`R`'

  $\vdash$ $\{$`X=R+(Y`$\times$`Q)`$\}$

  `WHILE Y`$\leq$`R DO`

  `BEGIN R:=R-Y; Q:=Q+1 END`

  $\{$`X=R+(Y`$\times$`Q)` $\wedge$ $\neg$`(Y`$\leq$`R)`$\}$

Example

- **From the previous slide**

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

- **It is easy to deduce that**

```
⊢ {T} R:=X; Q:=0 {X=R+(Y×Q)}
```

- **Hence by the sequencing rule and postcondition weakening**

```
⊢ {T}
   R:=X;
   Q:=0;
   WHILE Y≤R DO
     BEGIN R:=R-Y; Q:=Q+1 END
  {R<Y ∧ X=R+(Y×Q)}
```

Summary

- **We have given:**
  - a notation for specifying what a program does
  - a way of proving that it meets its specification
- **Now we look at ways of finding proofs and organising them:**
  - finding invariants
  - derived rules
  - backwards proofs
  - annotating programs prior to proof
- **Then we see how to automate program verification**
  - the automation mechanises some of these ideas

How does one find an invariant?

The `WHILE`-rule

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

- **Look at the facts:**
  - **invariant P must hold initially**
  - **with the negated test $\neg S$ the invariant P must establish the result**
  - **when the test S holds, the body must leave the invariant P unchanged**
- **Think about how the loop works – the invariant should say that:**
  - **what has been done so far together with what remains to be done**
  - **holds at each iteration of the loop**
  - **and gives the desired result when the loop terminates**

Example

- **Consider a factorial program**

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

- **Look at the facts**
  - **initially** `X=n` **and** `Y=1`
  - **finally** `X=0` **and** `Y=n!`
  - **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!`
  - **'stuff to be done'** $\times$ **'result so far'** $=$ **'desired result'**
  - **decrease in** `X` **combines with increase in** `Y` **to make invariant**

Related example

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

- **Look at the Facts**
  - initially X=0 and Y=1
  - finally X=N and Y=N!
  - on each iteration both X an Y increase: X by 1 and Y by X
- **An invariant is** Y = X!
- **At end need** Y = N!, **but** WHILE**-rule only gives** $\neg(X<N)$
- **Ah Ha! Invariant needed:** $\boxed{Y = X! \wedge X \leq N}$
- **At end** $X \leq N \wedge \neg(X<N) \;\Rightarrow\; X=N$
- **Often need to strenthen invariants to get them to work**
  - typical to add stuff to 'carry along' like 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

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

- **These rules are useful for splitting a proof into independent bits**
  - **they enable** $\vdash \{\mathtt{P}\}\ \mathtt{C}\ \{\mathtt{Q}_1 \wedge \mathtt{Q}_2\}$ **to be proved by proving separately that both** $\vdash \{\mathtt{P}\}\ \mathtt{C}\ \{\mathtt{Q}_1\}$ **and also that** $\vdash \{\mathtt{P}\}\ \mathtt{C}\ \{\mathtt{Q}_2\}$
- **Any proof with these rules could be done without using them**
  - **i.e. they are theoretically redundant (proof omitted)**
  - **however, useful in practice**

Combining Multiple Steps

- **Proofs involve lots of tedious fiddly small steps**
  - **similar sequences are used over and over again**
- **It is tempting to take short cuts and apply several rules at once**
  - **this increases the chance of making mistakes**
- **Example:**
  - **by assignment axiom & precondition strengthening**
    - $\vdash$ $\{\mathtt{T}\}$ $\mathtt{R} := \mathtt{X}$ $\{\mathtt{R} = \mathtt{X}\}$
- **Rather than:**
  - **by the assignment axiom**
    - $\vdash$ $\{\mathtt{X} = \mathtt{X}\}$ $\mathtt{R} := \mathtt{X}$ $\{\mathtt{R} = \mathtt{X}\}$
  - **by precondition strengthening with** $\vdash$ $\mathtt{T} \Rightarrow \mathtt{X}=\mathtt{X}$
    - $\vdash$ $\{\mathtt{T}\}$ $\mathtt{R} := \mathtt{X}$ $\{\mathtt{R} = \mathtt{X}\}$