

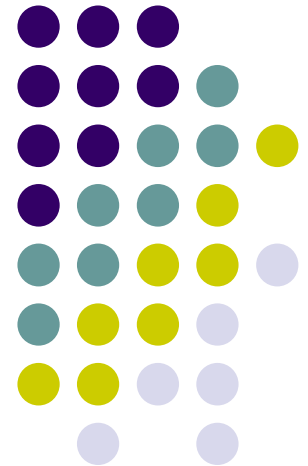
# Proof techniques

---

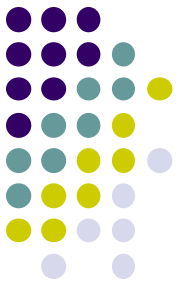
Lecture #7

J.Vain

28.03.2018

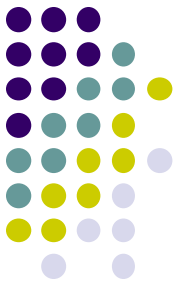


Slides adapted from  
Mike Gordon's course



# Lecture plan

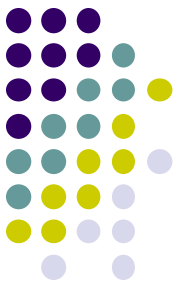
- We have given:
  - a notation for specifying what a program does
  - a way of proving that it meets its specification
- We will now look at ways of organising proofs to make them easier:
  - Derived rules
  - Backwards proofs
  - Annotating programs prior to proof



# 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

# How to combine multiple proof steps?



- **Example:**

- By assignment axiom & precondition strengthening

- $\vdash \{T\} R := X \{R = X\}$

- **Rather than:**

- By the assignment axiom

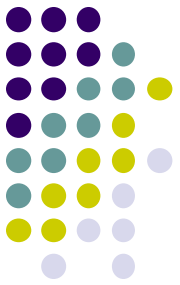
- $\vdash \{X = X\} R := X \{R = X\}$

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

- By precondition strengthening with  $\vdash T \Rightarrow X=X$

- $\vdash \{T\} R := X \{R = X\}$

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

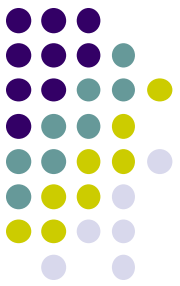


# A rule for assignment

- Rather than having the assignment axiom, we could have defined assignment by the following assignment rule

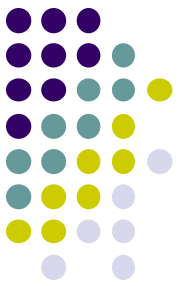
$$\boxed{\frac{\vdash P \Rightarrow Q[E/V]}{\vdash \{P\} V := E \{Q\}}}$$

- If we have both rules, they may be inconsistent
- The more complex the rule, the more likely we are to make a mistake formulating it
- We may not be able to prove everything we could with the smaller step rules



# Solution

- We have a small set of simple primitive rules
- We derive the other (possibly more complex) rules from the primitives
- We do the proof just once to derive the rule
- Rules for new commands defined in terms of existing commands can be built in a similar way
  - Core set of commands; the rest built on top



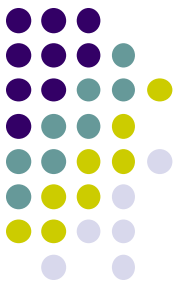
# Derived Assignment Rule

## Derived Assignment Rule

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

- Derivation tree

$$\frac{\vdash P \Rightarrow Q[E/V] \quad \overline{\vdash \{Q[E/V]\} V := E \{Q\}} \begin{matrix} ASS \\ PRE \end{matrix}}{\vdash \{P\} V := E \{Q\}}$$



# Rules of Consequence

- As in the assignment example, the desired precondition and postcondition are rarely in the form required by the primitive rules
- Ideally, for each command we want a rule of the form

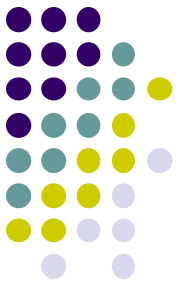
$$\frac{\dots}{\vdash \{P\} C \{Q\}}$$

where  $P$  and  $Q$  are distinct meta-variables.

- Some of the rules are already in this form eg the sequencing rule

We can derive rules of this form for the other commands using the rules of consequence





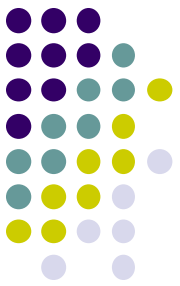
# Derived Skip Rule

## Derived Skip Rule

$$\frac{\vdash P \Rightarrow Q}{\vdash \{P\} \text{ SKIP } \{Q\}}$$

- Derivation Tree

$$\frac{\vdash P \Rightarrow Q \quad \overline{\vdash \{Q\} \text{ SKIP } \{Q\}} \begin{matrix} SKP \\ PRE \end{matrix}}{\vdash \{P\} \text{ SKIP } \{Q\}}$$



# Derived While Rule

$$\frac{\vdash P \Rightarrow R \quad \vdash \{R \wedge S\} C \{R\} \quad \vdash R \wedge \neg S \Rightarrow Q}{\vdash \{P\} \text{ WHILE } S \text{ DO } C \{Q\}}$$

- If it is possible to show that

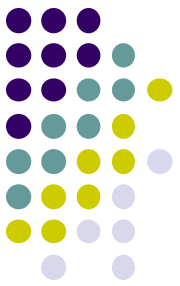
$$\vdash R=X \wedge Q=0 \Rightarrow X=R+(Y \times Q)$$

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

$$\vdash X=R+(Y \times Q) \wedge \neg(Y \leq R) \Rightarrow X=R+(Y \times Q) \wedge \neg(Y \leq R)$$

- then by the derived While rule

$$\begin{array}{l} \vdash \{R=X \wedge Q=0\} \\ \quad \text{WHILE } Y \leq R \text{ DO} \\ \quad \quad ( R:=R-Y; Q:=Q+1 ) \\ \quad \{X=R+(Y \times Q) \wedge \neg(Y \leq R)\} \end{array}$$



# Derived Sequencing Rule

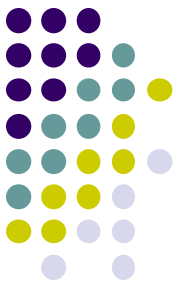
$$\begin{array}{c}
 \vdash P \Rightarrow P_1 \\
 \vdash \{P_1\} C_1 \{Q_1\} \quad \vdash Q_1 \Rightarrow P_2 \\
 \vdash \{P_2\} C_2 \{Q_2\} \quad \vdash Q_2 \Rightarrow P_3 \\
 \cdot \\
 \cdot \\
 \cdot \\
 \vdash \{P_n\} C_n \{Q_n\} \quad \vdash Q_n \Rightarrow Q \\
 \hline
 \vdash \{P\} C_1; \dots ; C_n \{Q\}
 \end{array}$$

- **Example**

$$\begin{array}{l}
 \vdash \{X=x \wedge Y=y\} R := X \{R=x \wedge Y=y\} \\
 \vdash \{R=x \wedge Y=y\} X := Y \{R=x \wedge X=y\} \\
 \vdash \{R=x \wedge X=y\} Y := R \{Y=x \wedge X=y\}
 \end{array}$$

---

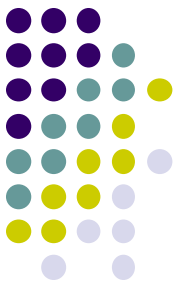

$$\vdash \{X=x \wedge Y=y\} R := X; X := Y; Y := R \{Y=x \wedge X=y\}$$



# Derived Block Rule

$$\frac{\begin{array}{l} \vdash P \Rightarrow P_1 \\ \vdash \{P_1\} C_1 \{Q_1\} \quad \vdash Q_1 \Rightarrow P_2 \\ \vdash \{P_2\} C_2 \{Q_2\} \quad \vdash Q_2 \Rightarrow P_3 \\ \vdots \\ \vdots \\ \vdots \\ \vdash \{P_n\} C_n \{Q_n\} \quad \vdash Q_n \Rightarrow Q \end{array}}{\vdash \{P\} \text{ BEGIN VAR } V_1; \dots \text{ VAR } V_m; C_1; \dots ; C_n \{Q\}}$$

where none of the variables  $V_1, \dots, V_m$  occur in  $P$  or  $Q$ .



# Derived Sequenced Assignment Rule

$$\frac{\vdash \{P\} C \{Q[E/V]\}}{\vdash \{P\} C; V := E \{Q\}}$$

- Derivation tree

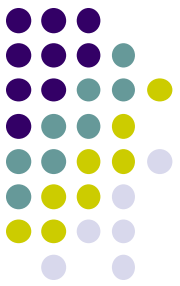
$$\frac{\vdash \{P\} C \{Q[E/V]\} \quad \frac{}{\vdash \{Q[E/V]\} V := E \{Q\}} \text{ASS}}{\vdash \{P\} C; V := E \{Q\}} \text{SEQ}$$

- Example: from

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

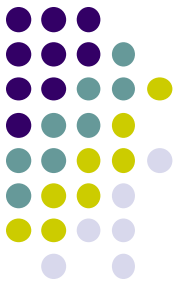
by the sequenced assignment rule

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



# Review of proving

- Previously it was shown how to prove  $\{P\}C\{Q\}$  by
  - proving properties of the components of  $C$
  - and then putting these together, with the appropriate proof rule, to get the desired property of  $C$
- For example, to prove  $\vdash \{P\}C_1;C_2\{Q\}$
- First prove  $\vdash \{P\}C_1\{R\}$  and  $\vdash \{R\}C_2\{Q\}$
- then deduce  $\vdash \{P\}C_1;C_2\{Q\}$  by sequencing rule



# Forward and Backward Proof

- This method is called *forward proof*
  - Move forward from axioms via rules to conclusion
- The problem with forwards proof is that it is not always easy to see what you need to prove to get where you want to be
- It is more natural to work backwards
  - Starting from the goal of showing  $\{P\}C\{Q\}$
  - Generate subgoals until problem solved



# Backwards vs Forward Proof

- Backwards proof just involves using the rules backwards

- Given the rule

$$\frac{\vdash S_1}{\vdash S_2}$$

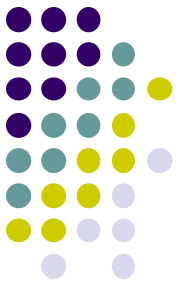
- Forwards proof says:

- If we have proved  $\vdash S_1$  we can deduce  $\vdash S_2$

- Backwards proof says:

- To prove  $\vdash S_2$  it is sufficient to prove  $\vdash S_1$





# Example Backward Proof

- To prove

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

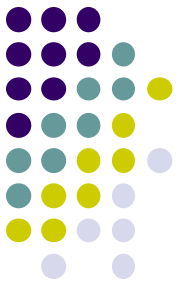
- By the sequencing rule, it is sufficient to prove

(i)  $\vdash \{T\} R:=X; Q:=0 \{R=X \wedge Q=0\}$

$\vdash \{R=X \wedge Q=0\}$

(ii) 

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



# Example Backward Proof

$$(i) \quad \vdash \{T\} R:=X; Q:=0 \{R=X \wedge Q=0\}$$

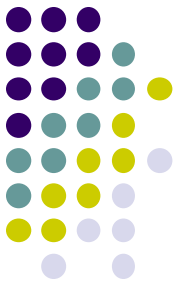
- To prove (i), by the sequenced assignment axiom, we must prove:

$$(iii) \quad \vdash \{T\} R:=X \{R=X \wedge 0=0\}$$

- To prove (iii), by the derived assignment rule, we must prove:

$$\vdash T \Rightarrow X=X \wedge 0=0$$

- This is true by pure logic.



# Example Backward Proof

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

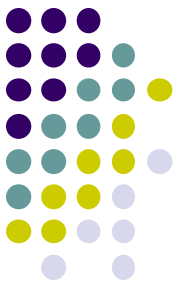
- To prove (ii), by the derived while rule, we must prove:

(iv)  $R=X \wedge Q=0 \Rightarrow (X = R+(Y \times Q))$

$$\frac{\vdash P \Rightarrow R \quad \vdash \{R \wedge S\} C \{R\} \quad \vdash R \wedge \neg S \Rightarrow Q}{\vdash \{P\} \text{ WHILE } S \text{ DO } C \{Q\}}$$

(v)  $X = R+Y \times Q \wedge \neg(Y \leq R) \Rightarrow (X = R+(Y \times Q) \wedge R < Y)$

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



# Example Backward Proof

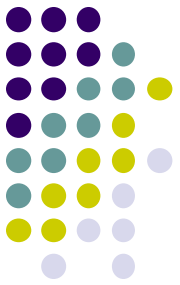
- To prove (vi), by the block rule, we must prove

$$\begin{array}{l} \{X = R + (Y \times Q) \wedge (Y \leq R)\} \\ \text{(vii)} \quad R := R - Y; \quad Q := Q + 1 \\ \{X = R + (Y \times Q)\} \end{array}$$

- To prove (vii), by the sequenced assignment rule, we must prove

$$\frac{\vdash \{P\} C \{Q[E/V]\}}{\vdash \{P\} C; V := E \{Q\}}$$

$$\begin{array}{l} \{X = R + (Y \times Q) \wedge (Y \leq R)\} \\ \text{(viii)} \quad R := R - Y \\ \{X = R + (Y \times (Q + 1))\} \end{array}$$



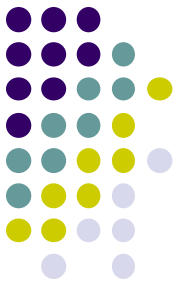
# Example Backward Proof

$$\begin{array}{l} \{X=R+(Y \times Q) \wedge (Y \leq R)\} \\ \text{(viii)} \quad R := R - Y \\ \{X=R+(Y \times (Q+1))\} \end{array}$$

- To prove (viii), by the derived assignment rule, we must prove

$$\text{(ix)} \quad X=R+(Y \times Q) \wedge Y \leq R \Rightarrow (X = (R-Y)+(Y \times (Q+1)))$$

- This is true by arithmetic



# Annotations

- The sequencing rule introduces a new statement  $R$

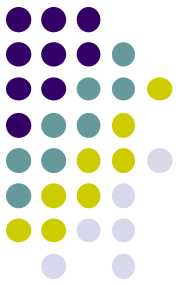
$$\frac{\vdash \{P\} C_1 \{R\}, \quad \vdash \{R\} C_2 \{Q\}}{\vdash \{P\} C_1; C_2 \{Q\}}$$

- To apply this rule, you must come up with a suitable statement for  $R$
- If the second command is an assignment, the sequenced assignment rule can be used
  - It then effectively fills in the value

# Annotate First



- It is helpful to think up these statements, before you start the proof and annotate the program with them
  - The information is then available when you need it in the proof
  - This can help avoid you being bogged down in details
  - The annotation should be true whenever control reaches that point in program!



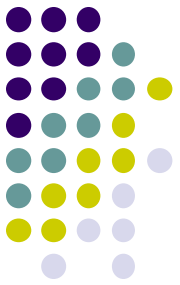
# Annotation example

- Example, the following program could be annotated at the points indicated.

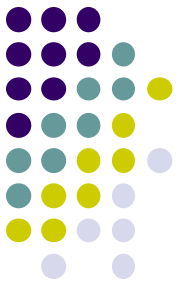
```
{T}
BEGIN
  R:=X;
  Q:=0; {R=X ∧ Q=0} ← P1
  WHILE Y ≤ R DO {X = R+Y×Q} ← P2
    BEGIN R:=R-Y; Q:=Q+1 END
  END
  {X = R+Y×Q ∧ R < Y}
```



# Summary



- We have looked at three ways of organizing proofs that make it easier for humans to apply them:
  - deriving “bigger step” rules
  - backwards proof
  - annotating programs



# Home Assignment

Prove that the program

```
BEGIN
  Z := 0 ;
  WHILE  $\neg(X=0)$  DO BEGIN
    IF ODD(X) THEN Z := Z+Y ELSE SKIP ;
    Y := Y*2 ; X := X/2 ;
  END
END
```

computes the product of the initial values of X and Y and leaves the result in Z.