Formal methods

Proof techniques

Introduction

- 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

Combining multiple steps

- Example:
  - By assignment axiom & precondition strengthening
    - $\vdash \{\mathtt{T}\}\ \mathtt{R} := \mathtt{X}\ \{\mathtt{R} = \mathtt{X}\}$
- Rather than:
  - By the assignment axiom

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

    - $\vdash \{\mathtt{X} = \mathtt{X}\}\ \mathtt{R} := \mathtt{X}\ \{\mathtt{R} = \mathtt{X}\}$
  - By precondition strengthening with $\vdash\ \mathtt{T} \Rightarrow \mathtt{X=X}$

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

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

Alternative rule for Assignment

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

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

- 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**

$$\dfrac{\vdash\ P \Rightarrow Q[E/V] \qquad \dfrac{}{\vdash\ \{Q[E/V]\}\ V := E\ \{Q\}}\ ASS}{\vdash\ \{P\}\ V := E\ \{Q\}}\ PRE$$

Rules of Consequence

- As in the assignment example, the desired pre-condition and postcondition are rarely in the form required by the primitive rules

- Ideally, for each command we want a rule of the form

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

  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

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

- **Derivation Tree**

$$\dfrac{\vdash \ P \Rightarrow Q \qquad \overline{\vdash \ \{Q\}\ \texttt{SKIP}\ \{Q\}}\ SKP}{\vdash \ \{P\}\ \texttt{SKIP}\ \{Q\}}\ PRE$$

Derived While Rule

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

- **If it is possible to show that**

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

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

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

- **Then by the derived While rule**

$\vdash\ \{\mathtt{R{=}X}\ \wedge\ \mathtt{Q{=}0}\}$
$\quad\mathtt{WHILE\ Y}\leq\mathtt{R\ DO}$
$\qquad\mathtt{(\ R{:=}R{-}Y;\ Q{:=}Q{+}1\ )}$
$\{\mathtt{X{=}R{+}(Y{\times}Q)}\ \wedge\ \neg(\mathtt{Y}\leq\mathtt{R})\}$

Derived Sequencing Rule

$$\frac{\begin{array}{ll} & \vdash\ P \Rightarrow P_1 \\ \vdash\ \{P_1\}\ C_1\ \{Q_1\} & \vdash\ Q_1 \Rightarrow P_2 \\ \vdash\ \{P_2\}\ C_2\ \{Q_2\} & \vdash\ Q_2 \Rightarrow P_3 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \vdash\ \{P_n\}\ C_n\ \{Q_n\} & \vdash\ Q_n \Rightarrow Q \end{array}}{\vdash\ \{P\}\ C_1;\ \ldots\ ;\ C_n\ \{Q\}}$$

- **Example**

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

Derived Block Rule

$$\dfrac{\begin{array}{ll} & \vdash P \Rightarrow P_1 \\ \vdash \{P_1\}\ C_1\ \{Q_1\} & \vdash Q_1 \Rightarrow P_2 \\ \vdash \{P_2\}\ C_2\ \{Q_2\} & \vdash Q_2 \Rightarrow P_3 \\ . & . \\ . & . \\ . & . \\ \vdash \{P_n\}\ C_n\ \{Q_n\} & \vdash Q_n \Rightarrow Q \end{array}}{\vdash \{P\}\ \texttt{BEGIN VAR}\ V_1;\ \ldots\ \texttt{VAR}\ V_m; C_1;\ \ldots\ ;\ C_n\ \{Q\}}$$

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

Derived Sequenced Assignment Rule

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

- **Derivation tree**

$$\dfrac{\vdash \ \{P\}C\{Q[E/V]\} \qquad \dfrac{}{\vdash \ \{Q[E/V]\}\ V := E\ \{Q\}}\,ASS}{\vdash \ \{P\}\ C; V := E\ \{Q\}}\,SEQ$$

- **Example: from**

$\vdash$ {X=x∧Y=y} R:=X {R=x∧Y=y}

by the sequenced assignment rule

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

Reviev 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

Backward versus 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 Backwards Proof

- To prove $\vdash$ {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 ∧ Q=0}

(ii) $\vdash$ {R=X ∧ Q=0}

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

Example Backwards Proof

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

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

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

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

$\vdash$ $\mathtt{T} \Rightarrow \mathtt{X=X} \wedge \mathtt{0=0}$

- **This is true by pure logic.**

Example Backwards Proof

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

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

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

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

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

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

Example Backwards Proof

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

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

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

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

$$\begin{array}{ll} & \{\mathtt{X=R+(Y\times Q)} \ \wedge\ \mathtt{(Y\leq R)}\} \\ \textbf{(viii)} & \quad \mathtt{R:=R-Y} \\ & \{\mathtt{X=R+(Y\times(Q+1))}\} \end{array}$$

Example Backwards Proof

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

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

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

- **This is true by arithmetic**

Annotations

- The sequencing rule introduces a new statement R

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

- 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

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 command

```
BEGIN
   Z:=0;
   WHILE ¬(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.