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**
    - $\vdash \{\mathtt{T}\}\ \mathtt{R} := \mathtt{X}\ \{\mathtt{R} = \mathtt{X}\}$
- **Rather than:**
  - **By the assignment axiom**
    - $\vdash \{\mathtt{X} = \mathtt{X}\}\ \mathtt{R} := \mathtt{X}\ \{\mathtt{R} = \mathtt{X}\}$

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

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

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

A rule for assignment

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

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

- 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**

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

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**

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

Derived While Rule

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

- If it is possible to show that

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

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

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

- then by the derived While rule

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

Derived Sequencing Rule

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

- Example

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

Derived Block Rule

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

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

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

- 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**

$$\vdash \{\mathtt{T}\}$$

```
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 Backward Proof

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

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

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

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

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

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

Example Backward Proof

(ii)
$$\vdash \{R=X \wedge Q=0\}$$
```
WHILE Y≤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:**

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

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

(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**

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

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

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

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

Example Backward Proof

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

- 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 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 ¬(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.