3 Partial Correctness

3.1 Introduction

Overview

- The proof rules that follow constitute an axiomatic semantics of our programming language:

\[
E ::= N \mid V \mid E_1 + E_2 \mid E_1 - E_2 \mid E_1 \times E_2 \mid \ldots \\
B ::= T \mid F \mid E_1 = E_2 \mid E_1 \leq E_2 \mid \ldots \\
C ::= \text{SKIP} \mid V := E \mid C_1 ; C_2 \mid \text{IF } B \text{ THEN } C_1 \text{ ELSE } C_2 \mid \text{BEGIN VAR } V_1 ; \ldots \text{ VAR } V_n ; C \text{ END} \mid \text{WHILE } B \text{ DO } C
\]

Judgements

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

- These three kinds of things are examples of judgements
  - a logical system provides rules for establishing the truth (i.e. proving) various kinds of judgements
  - Floyd-Hoare logic provides rules for proving partial correctness specifications
  - the laws of arithmetic, which are assumed known, provide ways of proving statements about integers

- \(\vdash S\) means statement \(S\) can be proved
  - how to prove predicate calculus statements assumed known
  - this course covers axioms and rules for proving program correctness statements

Syntactic Conventions

- The symbols \(V, V_1, \ldots, V_n\) stand for arbitrary variables
  - examples of particular variables are \(X, R, Q\) etc.

- The symbols \(E, E_1, \ldots, E_n\) stand for arbitrary expressions (or terms)
  - these are things like \(X + 1, \sqrt{2}\) etc. which denote values (usually numbers)

- The symbols \(S, S_1, \ldots, S_n\) stand for arbitrary statements
  - these are conditions like \(X < Y, X^2 = 1\) etc. which are either true or false

- The symbols \(C, C_1, \ldots, C_n\) stand for arbitrary commands of our programming language
Notation for Axioms and Rules

- The axioms of Floyd-Hoare logic are specified by schemas
  - these can be instantiated to get particular partial correctness specifications
  - an example is the Skip Axiom on the next slide
- The inference rules of Floyd-Hoare logic will be specified with a notation of the form
  \[ \vdash S_1, \ldots, \vdash S_n \vdash S \]
  - this means the conclusion \( \vdash S \) may be deduced from the hypotheses \( \vdash S_1, \ldots, \vdash S_n \)
- 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

3.2 The SKIP Command

SKIP

- Syntax: SKIP
- Semantics: the state is unchanged

The SKIP Axiom

\[ \vdash \{P\} \text{SKIP} \{P\} \]

- It is an axiom schema
  - \( P \) can be instantiated with arbitrary predicate calculus formulae (statements)
- Instances of the SKIP axiom are:
  - \( \vdash \{Y = 2\} \text{SKIP} \{Y = 2\} \)
  - \( \vdash \{T\} \text{SKIP} \{T\} \)
  - \( \vdash \{R = X + (Y \times Q)\} \text{SKIP} \{R = X + (Y \times Q)\} \)

3.3 Assignment

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 \( X \)
- 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
- 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
Substitution Notation

- Define $P[E/V]$ to mean the result of replacing all occurrences of $V$ in $P$ by $E$
  - read $P[E/V]$ as ‘$P$ with $E$ for $V$’
  - for example: $(X + 1 > X)[Y + Z/X] = ((Y + Z) + 1 > Y + Z)$

- Think of this notation as the ‘cancellation law’:
  $V[E/V] = E$
  which is analogous to the cancellation property of fractions:
  $v \times (e/v) = e$

The Assignment Axiom

\[
\text{The Assignment Axiom}
\]
\[
\vdash \{P[E/V]\} \ V := E \ \{P\}
\]

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 $P$.

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

The Backwards Fallacy

- Many people feel the assignment axiom is ‘backwards’
- One common erroneous intuition is that it should be:
  $\vdash \{P\} \ V := E \ \{P[V/E]\}$
  - where $P[V/E]$ denotes the result of substituting $V$ for $E$ in $P$
  - this has the false consequence $\vdash \{X = 0\} \ X := 1 \ \{X = 0\}$, since $(X = 0)[X/1] = (X = 0)$, as $1$ does not occur in $(X = 0)$

- Another erroneous intuition is that it should be:
  $\vdash \{P\} \ V := E \ \{P[E/V]\}$
  - this has the false consequence $\vdash \{X = 0\} \ X := 1 \ \{V = 1\}$, which follows by taking $P$ to be $X = 0$, $V$ to be $X$ and $E$ to be $1$
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’:
  \[ \text{BEGIN } Y := 1; 2 \text{ END} \]
  - 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:
  \[ \vdash \{ Y = 0 \} X := \text{BEGIN } Y := 1; 2 \text{ END} \{ Y = 0 \} \]
  - 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

3.4 Rules of Consequence

Precondition Strengthening

\[ \vdash S_1, \ldots, \vdash S_n \]

- Recall that \[ \vdash S \]
  means \( \vdash S \) can be deduced from \( \vdash S_1, \ldots, \vdash S_n \)

- Using this notation, the rule of precondition strengthening is:
  \[ \frac{\vdash P \Rightarrow P', \quad \vdash \{ P' \} C \{ Q \}}{\vdash \{ P \} C \{ Q \}} \]

- Note the two hypotheses are different kinds of judgements

Example

- 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:

- \[ \vdash \{ X = n \} X := X + 1 \{ X = n + 1 \} \]
  - \( n \) is an auxiliary (or ghost) variable
Example

From

- $\vdash T \Rightarrow (E = E)$
- $\vdash \{ E = E \} X := E \{ X = E \}$

It follows that if $X$ is not in $E$ (why?):

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

Consider:

- $\{ T \} X := X + 1 \{ X = X + 1 \}$

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

  \[
  \begin{array}{c}
  \vdash \{ P \} \ C \ \{ Q' \}, \\
  \vdash Q' \Rightarrow Q \\
  \hline
  \vdash \{ P \} \ C \ \{ Q \}
  \end{array}
  \]

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

An Example Formal Proof

Here is a little formal proof:

\[
\begin{align*}
\vdash \{ R = X \} Q := 0 \{ R = X \land (Y \times Q) \} \\
= & \hspace{1em} \{ \text{postcondition weakening,} \\
  & \hspace{1em} \vdash R = X \land Q = 0 \Rightarrow R = X \land (Y \times Q) \ \} \\
\vdash \{ R = X \} Q := 0 \{ R = X \land Q = 0 \} \\
= & \hspace{1em} \{ \text{precondition strengthening,} \\
  & \hspace{1em} \vdash R = X \Rightarrow R = X \land 0 = 0 \ \} \\
\vdash \{ R = X \land 0 = 0 \} Q := 0 \{ R = X \land Q = 0 \} \\
= & \hspace{1em} \{ \text{assignment axiom} \} \\
& \hspace{1em} \text{True}
\end{align*}
\]

3.5 Sequences

Sequences

- Syntax: $C_1; \ldots; C_n$
- Semantics: the commands $C_1, \ldots, C_n$ 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 $X$
- The following 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

\[
\frac{\vdash \{ P \} \ C_1 \ \{ Q \}, \ \vdash \{ Q \} \ C_2 \ \{ R \}}{\vdash \{ P \} \ C_1 ; C_2 \ \{ R \}}
\]

Example Proof

\[
\begin{align*}
&\vdash \{ X = x \wedge Y = y \} \ R := X ; \ X := Y ; \ Y := R \ \{ Y = x \wedge X = y \} \\
&\quad = \quad \{ \text{sequencing rule} \} \\
&\vdash \{ X = x \wedge Y = y \} \ R := X ; \ X := Y \ \{ R = x \wedge X = y \} \wedge \\
&\vdash \{ X = x \wedge X = y \} \ Y := R \ \{ Y = x \wedge X = y \} \\
&\quad = \quad \{ \text{sequencing rule} \} \\
&\vdash \{ X = x \wedge X = y \} \ Y := R \ \{ Y = x \wedge X = y \} \\
&\quad = \quad \{ \text{assignment axiom} \} \\
&\text{True}
\end{align*}
\]

3.6 Blocks

Blocks

- Syntax: \texttt{BEGIN VAR V_1; \ldots \ VAR V_n; C \ END}

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

- Example: \texttt{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:

\[
\begin{align*}
&\vdash \{ P \} \ C \ \{ Q \} \\
&\vdash \{ P \} \ \text{BEGIN VAR V_1; \ldots; VAR V_n; C \ END} \ \{ Q \}
\end{align*}
\]

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

- without this condition the rule is invalid, as illustrated in the example below

From

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

it follows by the block rule that

$\vdash \{X = x \land Y = y\}$

BEGIN VAR R; \quad R := X; \quad X := Y; \quad Y := R END

$\{Y = x \land X = y\}$

since $R$ does not occur in $X = x \land Y = y$ or $X = y \land Y = x$

However from

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

one cannot deduce

$\vdash \{X = x \land Y = y\}$

BEGIN VAR R; \quad R := X; \quad X := Y END

$\{R = x \land X = y\}$

since $R$ occurs in $R = x \land X = y$

Exercises

- Consider the specification:

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

  Can this be deduced from the rules given so far?

  1. if so, give a proof of it
  2. if not, explain why not and suggest additional rules and/or axioms to enable it to be deduced

- Is the following true?

  $\vdash \{X = x \land Y = y\}$

  $X := X + Y; \quad Y := X - Y; \quad X := X - Y$

  $\{Y = x \land X = y\}$

  – if so prove it
  – if not, give the circumstances when it fails

- Show:

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

  BEGIN R := R - Y; \quad Q := Q + 1 END

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

3.7 Conditionals

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 $\text{MAX} := Y$ ELSE $\text{MAX} := X$
  
  – the value of the variable $\text{MAX}$ is set to the maximum of the values of $X$ and $Y$

• One-armed conditional is defined by: IF $S$ THEN $C$ = $\text{define}$ IF $S$ THEN $C$ ELSE $\text{SKIP}$

The Conditional Rule

$$
\begin{array}{c}
\vdash \{ P \land S \} C_1 \{ Q \}, \\
\vdash \{ P \land \neg S \} C_2 \{ Q \}, \\
\vdash \{ P \} \text{ IF } S \text{ THEN } C_1 \text{ ELSE } C_2 \{ Q \}
\end{array}
$$

• Suppose we are given: $\vdash \{ T \land X \geq Y \} \text{ MAX} := X \{ \text{MAX} = \max(X, Y) \}$
• and: $\vdash \{ T \land \neg(X \geq Y) \} \text{ MAX} := Y \{ \text{MAX} = \max(X, Y) \}$

• Then by the conditional rule it follows that: $\vdash \{ T \}$
  
  IF $X \geq Y$ THEN $\text{MAX} := X$ ELSE $\text{MAX} := Y$
  
  \{ $\text{MAX} = \max(X, Y) \}$

3.8 The WHILE Command

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 $\neg(X = 0)$ DO $X := X - 2$
  
  – if the value of $X$ is non-zero, then its value is decreased by 2 and 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 $\vdash \{ P \land S \} C \{ P \}$
• then $P$ is an invariant of $C$ whenever $S$ holds

• The WHILE rule says that:
  
  – if $P$ is an invariant of the 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 \textbf{WHILE} rule also expresses the fact that after a \textbf{WHILE} command has terminated, the test must be false
  – otherwise, it would not have terminated

\textbf{The WHILE Rule}

\[
\frac{\vdash \{P \land S \} \ C \ \{P\}}{\vdash \{P\} \ \text{WHILE S DO} \ C \ \{P \land \neg S\}}
\]

- It is easy to show:
  \[
  \vdash \{X = R + (Y \times Q) \land Y \leq R\}
  \begin{array}{l}
  \text{BEGIN}\ R := R - Y; \ Q := Q + 1 \ 
  \text{END}\ 
  \{X = R + (Y \times Q)\}
  \end{array}
  \]

- Hence by the \textbf{WHILE} rule with \( P = X = R + (Y \times Q) \):
  \[
  \vdash \{X = R + (Y \times Q)\}
  \begin{array}{l}
  \text{WHILE Y \leq R DO}\ 
  \begin{array}{l}
  \text{BEGIN}\ R := R - Y; \ Q := Q + 1 \ 
  \text{END}\ 
  \{X = R + (Y \times Q) \land \neg(Y \leq R)\}
  \end{array}
  \end{array}
  \]

\textbf{Example}

- From the previous slide:
  \[
  \vdash \{X = R + (Y \times Q)\}
  \begin{array}{l}
  \text{WHILE Y \leq R DO}\ 
  \begin{array}{l}
  \text{BEGIN}\ R := R - Y; \ Q := Q + 1 \ 
  \text{END}\ 
  \{X = R + (Y \times Q) \land \neg(Y \leq R)\}
  \end{array}
  \end{array}
  \]

- It is easy to deduce that:
  \[
  \vdash \{T\} \ R := X; \ Q := 0 \ \{X = R + (Y \times Q)\}
  \]

- Hence by the sequencing rule and postcondition weakening:
  \[
  \vdash \{T\}
  \begin{array}{l}
  R := X; \ Q := 0; \ 
  \text{WHILE Y \leq R DO}\ 
  \begin{array}{l}
  \text{BEGIN}\ R := R - Y; \ Q := Q + 1 \ 
  \text{END}\ 
  \{R < Y \land X = R + (Y \times Q)\}
  \end{array}
  \end{array}
  \]