HOARE LOGIC

Parts of the slides are taken from the lecture notes of Carl Leonadsson, Yih-Kuen Tsai, and Michael Gordon

Presented by: Yu-Fang Chen

Outline

- Prove Program Correctness
  - WHILE program
  - Hoare Triple
- Axioms and Rules
  - Assignment Axiom
  - Composition Rule
  - Conditional Rule
  - Iteration Rule

Hoare Logic

- Hoare Logic - An axiomatic basis for computer programming (1969, C.A.R. Hoare)
  - Describes a deductive system for proving program correctness.
  - A set of axioms and inference rules about asserted programs.
  - Development to the logic is still active
    - E.g., separation logic (reasoning about pointers)

WHILE Program

- Assume that we have an underlying logic **L**, e.g. Integer Arithmetic

E.g. X+5, 4-Y*Z

Define inductively

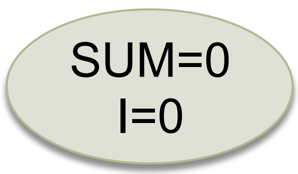
- For all integer variable **X** and term **E**, **X:=E** is a program.
- If $\mathbf{S_1}$ and $\mathbf{S_2}$ are programs, **B** is a Boolean expression, then the following are programs
  - $\mathbf{S_1;S_2}$
  - If **B** then $\mathbf{S_1}$ else $\mathbf{S_2}$ fi
  - while **B** do $\mathbf{S_1}$ od

A sample program:

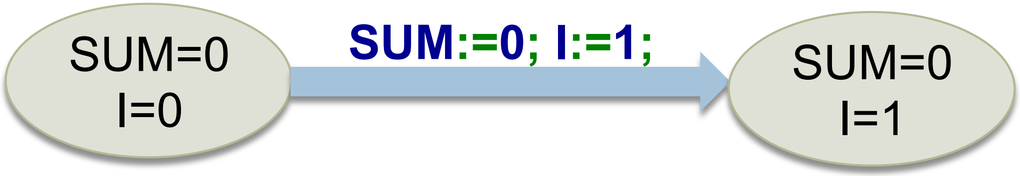
```
SUM:=0;
I:=1;
while I<100 do
   if I%2=0 then
      SUM:=SUM+I ; I:=I+1
   else
      I:=I+1
   fi
od
```

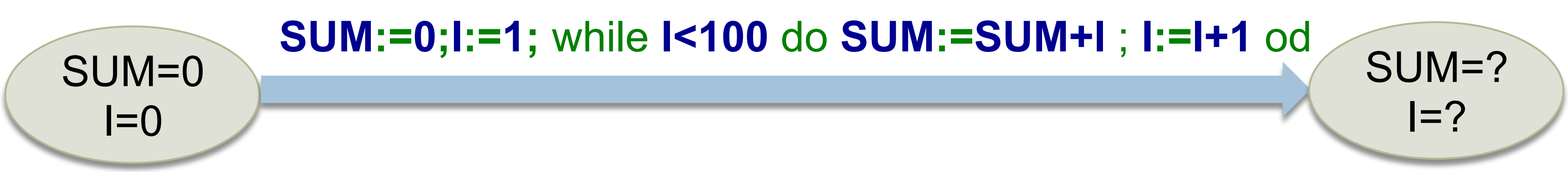
Program States and Transitions

- A **state** is a valuation of all program variables.



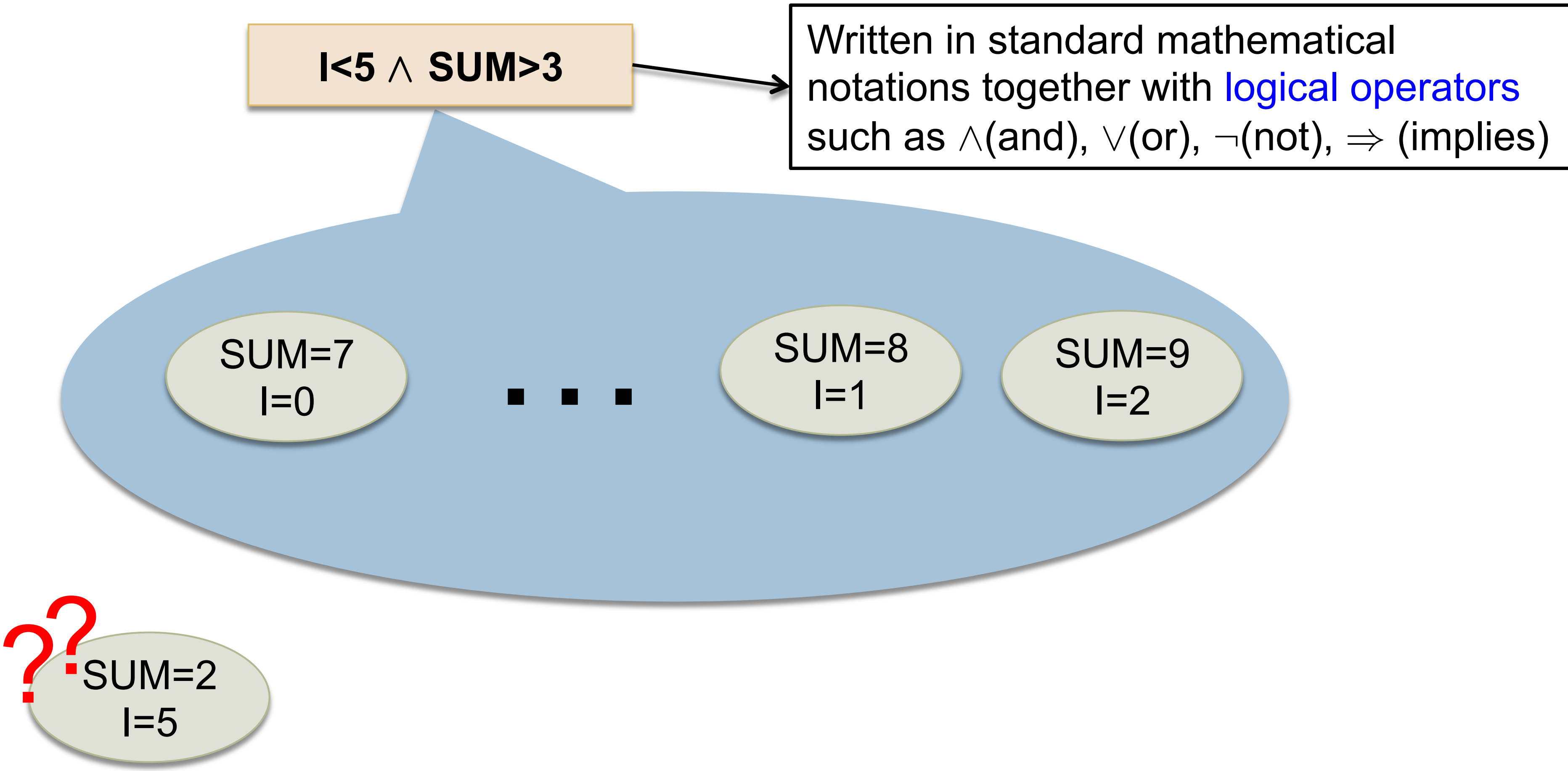
- A program statement defines **transitions** between program states.



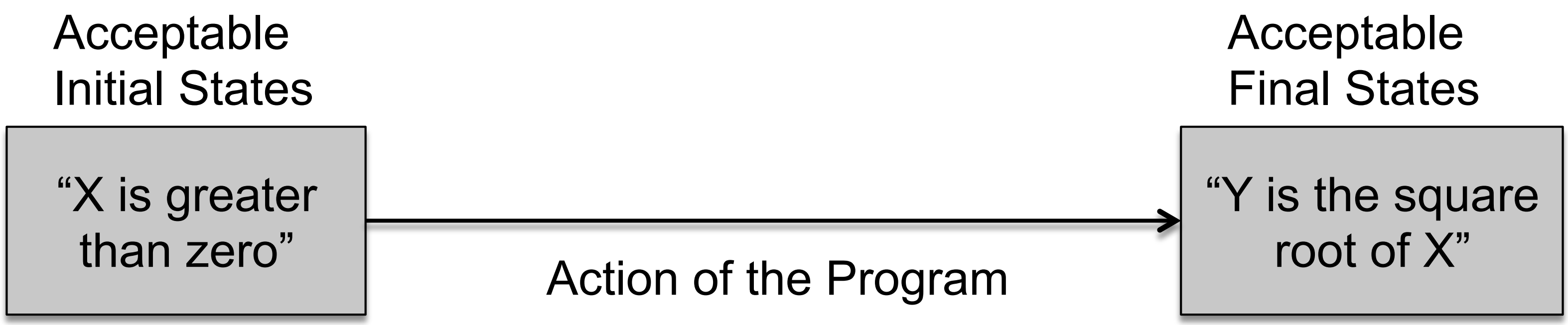


Predicates

- A **predicate** characterizes a set of program states



Specification of Imperative Programs



Hoare's notation

- C.A.R. Hoare introduced the following notation called a *partial correctness specification* for specifying what a program does:

  $$\{P\}S\{Q\}$$

  - Here S is a program,
  - **P** is a predicate describes the precondition of S
  - **Q** is a predicate describes the postcondition of S

- Note: Hoare's original notation was P{S}Q instead of {P}S{Q}, but the latter form is now more widely used

Meaning of Hoare's Notation

{P}S{Q} means

- Whenever S is executed in a state satisfying **P**
- and if the execution of S terminates
- Then the state in which S terminates satisfies **Q**.

- Example: {X = 1} X:= X+1 {X = 2}
  - **P**: the value of X is 1
  - **Q**: the value of X is 2
  - S: an assignment X:= X + 1
    - X becomes X + 1
- {X = 1} X := X + 1 {X = 2} is true
- {X = 1} X := X + 1 {X = 2} is false

Some practices

(1) Is the following formula valid?

$\{X < 1\}$ X:=X+1 ; X:= X+1 $\{X < 3\}$

(2) Is the following formula valid?

$\{X < 100\}$ while true do X:=X+1 od $\{X < 0\}$

(3) Is the following formula valid?

$\{X < 100\}$ if X=1 then S_1 else S_2 $\{X < 200\}$

$S_1 \equiv$ while true do X:=X+1 od

$S_2 \equiv$ X:=X+2

Formal versus Informal Proof

- Informal Proof:
  - Like what we used in the previous slides
- Formal verification uses formal proof
  - The rules used are described and followed very precisely
- An example: proof of $(X+1)^2 = X^2 + 2 \times X + 1$

1.	$(X+1)^2$	$= (X+1) \times (X+1)$	Definition of $()^2$.
2.	$(X+1) \times (X+1)$	$= (X+1) \times X + (X+1) \times 1$	Left distributive law of $\times$ over $+$.
3.	$(X+1)^2$	$= (X+1) \times X + (X+1) \times 1$	Substituting line 2 into line 1.
4.	$(X+1) \times 1$	$= X + 1$	Identity law for 1.
5.	$(X+1) \times X$	$= X \times X + 1 \times X$	Right distributive law of $\times$ over $+$.
6.	$(X+1)^2$	$= X \times X + 1 \times X + X + 1$	Substituting lines 4 and 5 into line 3.
7.	$1 \times X$	$= X$	Identity law for 1.
8.	$(X+1)^2$	$= X \times X + X + X + 1$	Substituting line 7 into line 6.
9.	$X \times X$	$= X^2$	Definition of $()^2$.
10.	$X + X$	$= 2 \times X$	2=1+1, distributive law.
11.	$(X+1)^2$	$= X^2 + 2 \times X + 1$	Substituting lines 9 and 10 into line 8.

The Structure of Proofs

- A proof consists of a sequence of lines
- Each line is an instance of an atom
  - E.g., the definition of ()^2
- or follows from previous lines by a rule of inference
  - E,g, the substitution of equivalent objects
- The statement on the last line of the proof is the statement proved by it
  - Thus $(X+1)^2 = X^2 + 2 \times X + 1$ is proved by the proof on the previous slides
- These are “Hibert style” formal proofs
  - can use a tree structure rather than a linear one
  - the choice is a matter of convenience

Formal proof is syntactic “symbol pushing”

- Formal system reduce verification and proof to symbol pushing.

- The rule say…
  - If you have a string of characters of this form
  - You can obtain a new string of characters of this other form

- Even if you don’t know what the strings are intended to mean, provided the rules are designed properly and you apply them correctly, you will get correct results.
  - Though not necessary the desired result

Hoare Logic

- Hoare Logic is a deductive proof system for Hoare triple {P} S {Q}

- Can be used to verify programs
  - Original proposal by Hoare
  - Tedious and error prone

- Exists tools to help its automation

Partial Correctness Specification

- An expression {P} S {Q} is called a partial correctness specification
  - P is called its *precondition*
  - Q is called its *postcondition*

- {P} S {Q} means
  - Whenever S is executed in a state satisfying P
  - and if the execution of S terminates
  - Then the state in which the execution of S terminates satisfies Q

- It is partial because for {P} S {Q} to be true, it is not necessary for the execution of S to terminate when stated in a state satisfying P

- {X = 1} while **T** do **X := X + 1** od {X = -3 } – this specification is true!

Total Correctness Specification

- A stronger kind of specification is a *total correctness specification*
  - There is no standard notation for such specifications
  - Here we use [P] S [Q]

- [P] S [Q] means
  - Whenever S is executed in a state satisfying, the execution of S terminates
  - After S terminates Q holds

- [X = 1] while **T** do **X := X + 1** od [X = -3]
  - This says the execution of the program terminates when stated in a state satisfying X = 1
  - After which Y = 1 will hold

Clearly false

Total Correctness

- Informally

  Total Correctness = Termination + Partial Correctness

- Total correctness is the ultimate goal
  - Usually easier to show partial correctness and termination separately
- Termination is usually straightforward to show, but there exists examples where it is not.

Example

```
while X > 1 do
    if X%2==1
        then X := (3*X)+1
        else X := X/2
    fi
od
```

Collatz conjecture: if the program terminates with X = 1 for all values of X

Auxiliary Variables

- $\{X=x \wedge Y=y\}$ R:=X; X:=Y; Y:=R $\{X=y \wedge Y=x\}$
  - If the program terminates, then the values of X and Y are swapped
- The variables x and y, which do not occur in the program and are used to name the initial values of program variables X and Y
- They are called auxiliary variables or ghost variables.
- Informal convention:
  - Program variables are upper case
  - Auxiliary variables are lower case

More examples

- $\{X = x \land Y = y\}$ X:=Y ; Y:=X $\{X = y \land Y = x\}$
  - It says the program can swap the values of X and Y, which is not true
- {T} S {Q}
  - Whenever S halts, Q holds
- {P} S {T}
  - This specification is true for all P and S
  - Because T is always true
- [P] S [T]
  - S terminates if initially P holds
- [T] S [Q]
  - S always terminates and ends in a state where Q holds

A More Complicated Example

{T}

R:=X;Q=0; while Y$\leq$ R do R:=R-Y; Q:=Q+1 od

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

- The specification is true if the execution of the program terminates, then Q is the quotient and R is the reminder resulting from dividing Y into X

- This is true even if X is initially negative

Some Easy Exercises

- When is [T] S [T] true?

- Write a partial correctness specification which is true iff the program S has the effect of multiplying the values of X and Y and storing the results in X

- Write a specification which is true if the execution of S always terminates when the execution is stated in a state satisfying P

Specification can be Tricky

- "The program must set Y to the maximum of X and Y"

  $$[T]\ S\ [Y=max(X,Y)]$$

- A suitable program
  - if $X \geq Y$ then Y:=X else X := X fi
- Another?
  - If $X \geq Y$ then X:=Y else X := X fi
- Or even
  - Y:=X
- Later we will be able to prove that all the programs are "correct"
- The postcondition [Y=max(X,Y)] is the maximum of X and Y in the final state

Specification can be Tricky

- The intended specification was not properly captured by

$$[T]\ S\ [Y=max(X,Y)]$$

- The correct one should be

$$[X=x \wedge Y=y]\ S\ [Y=max(x,y)]$$

- The lesson
  - It is easy to write the wrong specification
  - A proof system will not help since the incorrect program can be proved “correct”
  - Testing could be helpful in this case

Outline

- Prove Program Correctness
  - WHILE program
  - Hoare Triple
- Axioms and Rules
  - Assignment Axiom
  - Composition Rule
  - Conditional Rule
  - Iteration Rule

Formal Proof

(1) Is the following formula valid?

$\{X < 1\}\ X{:=}X+1\ ;\ X{:=}\ X+1\ \{X < 3\}$

(2) Is the following formula valid?

$\{X < 100\}$ while true do $X{:=}X+1$ od $\{X < 0\}$

(3) Is the following formula valid?

$\{X < 100\}$ if $X=1$ then S_1 else S_2 $\{X < 200\}$

$S_1 \equiv$ while true do $X{:=}X+1$ od

$S_2 \equiv X{:=}X+2$

How can we formally prove the previous examples?

Assignment Axiom

- We begin with Foyld's version of the assignment axiom

$$\{P\}\ \mathbf{X := E}\ \{?\}$$

- The term **E** might contain **X**, e.g. **E** $\equiv$ **X+1**
- An example: **X := X + 1**

The value of X **after** executing the statement

The value of X **before** executing the statement

- We need to differentiate these two values!

Assignment Axiom

- We begin with Foyld's version of the assignment axiom

$$\{P\}\ X := E\ \{?\}$$

$$\exists V.(\ X=E[V/X]\ \wedge\ P[V/X]\ )$$

Intuition: we use new variable V to denote the **old value of X**

Notations	
E[V/X] **P[V/X]**	replacing all **free occurrences** of **X** in **E** / **P** with **V**

Assignment Axiom

Foyld's Assignment Axiom

$$\frac{}{\{P\}\ X:=E\ \{\exists\ V.\ X=E[V/X] \wedge P[V/X]\}}$$

Example

$\{Y + X = 42\}\ X := X + 5\ \{\exists\ V.\ X = V + 5 \wedge Y + V = 42\}$

Example

$\{Y = 5\}\ X := X/Y + X\ \{?\}$

We do not want to have quantifiers in the reasoning path!

Assignment Axiom

Backward reasoning

Hoare's Assignment Axiom

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$$
$$\{Q[E/X]\}\ X{:=}E\ \{Q\}$$

Expressions with Side-effect

- 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 change the value of Y to 1

- If the assignment axiom applied to block expressions, then it could be used to deduce the following, which is false
  - {Y=0} X:= BEGIN Y:=1; 2 END {Y=0}
  - Notice that (Y=0)[E/X] = (Y=0)

Assignment Axiom

Backward reasoning

Hoare's Assignment Axiom

$$\frac{}{\{Q[E/X]\}\ X{:=}E\ \{Q\}}$$

Below is an informal proof of the soundness of this axiom:

Let s be the state before X := E and s′ the state after.
So, s′ = s[X $\rightarrow$ E] (assuming E has no side-effect).

Q[E/X] holds in s if and only if Q holds in s′, because
(1) Every variable, except X, has the same value in s and s′, and
(2) Q[E/X] has every X in Q replaced by E,
(3) Q has every X evaluated to E in s (s' = s[X $\rightarrow$ E]).

Assignment Axiom

Backward reasoning

Hoare's Assignment Axiom

$$\frac{}{\{Q[E/X]\}\ X{:=}E\ \{Q\}}$$

Example

{X + Y + 5 > 5} X := X + Y + 5 {X > 5}

Try it!

Example

{?} X := X + 1 {X<10}

Composition Rule

Composition Rule

$$\frac{\{P\}S_1\{R\} \quad \{R\}S_2\{Q\}}{\{P\}S_1;S_2\{Q\}}$$

Composition Rule

Example
P: {true} X:=2 ; Y:=X {X >0$\wedge$ Y=2}

(1) $2>0 \wedge 2 = 2 \Leftrightarrow$ true (Integer arithmetic)
(2) $\{2 > 0 \wedge 2 = 2\}$ X:=2 $\{X > 0 \wedge X =2\}$ (assignment axiom)
(3) $\{X > 0 \wedge X = 2\}$ Y:=X $\{X > 0 \wedge Y = 2 \}$ (assignment axiom)
(4) {true} X:=2 $\{X > 0 \wedge X =2\}$ (by (1), we can replace $2>0 \wedge 2 = 2$ in (3) with true)
(5) {true} X:=2 ; Y:=X $\{X >0\wedge Y=2\}$ (by (3), (4), and composition rule)

Composition Rule

Example

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

(1) $\{X=x \wedge Y=y\}$ R:=X $\{R=x \wedge Y=y\}$ (assignment axiom)
(2) $\{R=x \wedge Y=y\}$ X:=Y $\{R=x \wedge X=y\}$ (assignment axiom)
(3) $\{R=x \wedge X=y\}$ Y:=R $\{Y=x \wedge X=y\}$ (assignment axiom)
(4) $\{X=x \wedge Y=y\}$ R:=X; X:=Y $\{R=x \wedge X=y\}$ (by (1), (2), and composition rule)
(5) $\{X=x \wedge Y=y\}$ R:=X ; X:=Y ; Y:=R $\{Y=x \wedge X=y\}$ (by (4), (3), and composition rule)

Conditional Rule

Conditional Rule

$$\frac{\{P \wedge E\}\ S_1\ \{Q\}\quad \{P \wedge \neg E\}\ S_2\ \{Q\}}{\{P\}\ \text{if } E \text{ then } S_1 \text{ else } S_2\ \{Q\}}$$

Conditional Rule

Conditional Rule

$$\frac{\{P \wedge E\}\ S_1\ \{Q\}\quad \{P \wedge \neg E\}\ S_2\ \{Q\}}{\{P\}\ \text{if } E \text{ then } S_1 \text{ else } S_2\ \{Q\}}$$

Example

P: {true} if X < 10 then X:=10 else X:=0 fi $\{X=10 \vee X=0\}$

We can infer P if we can infer

(1) P_1: $\{true \wedge X < 10\}$ X:=10 $\{X=10 \vee X=0\}$

(2) P_2: $\{true \wedge X \geq 10\}$ X:=0 $\{X=10 \vee X=0\}$

Here we need other proof rule to prove (1) and (2)

Consequence Rule

Consequence Rule

$$\frac{P \Rightarrow P' \; \{P'\}\; S\; \{Q'\}\; Q' \Rightarrow Q}{\{P\}\; S\; \{Q\}}$$

- We can strengthen the precondition, i.e. assume more than we need
- We can weaken the postcondition, i.e. conclude less than we are allowed to

Consequence Rule

Consequence Rule

$$\frac{P \Rightarrow P' \ \{P'\}\ S\ \{Q'\}\ Q' \Rightarrow Q}{\{P\}\ S\ \{Q\}}$$

Example

P_1: {true $\wedge$ X < 10} X:=10 {X=10 $\vee$ X=0}

(1) {true} X:=10 {X=10 $\vee$ X=0} (by Assignment Rule)
(2) true$\wedge$X<10 $\Rightarrow$ true (by underlying logic)
(3) X = 10 $\vee$ X = 0 $\Rightarrow$ X = 10 $\vee$ X = 0 (by underlying logic)
(4) {true $\wedge$ X < 10} X:=10 {X=10 $\vee$ X=0} (by consequence rule, (2), and (3))

Consequence Rule

Consequence Rule

$$\frac{P \Rightarrow P' \quad \{P'\}\ S\ \{Q'\} \quad Q' \Rightarrow Q}{\{P\}\ S\ \{Q\}}$$

Example

P_2: $\{true \wedge X \geq 10\}\ X{:=}0\ \{X{=}10 \vee X{=}0\}$

Try it yourself!

Another example

Example

{T} if X $\geq$ Y then MAX :=X else MAX := Y fi {MAX = max (X,Y)}

(1) T $\wedge$ X $\geq$ Y $\Rightarrow$ X=max(X,Y) (by Underlying Logic)
(2) T $\wedge$ $\neg$(X $\geq$ Y) $\Rightarrow$ Y = max(X,Y) (by Underlying Logic)
(3) MAX=max(X,Y) $\Rightarrow$ MAX=max(X,Y) (by Underlying Logic)
(4) {X = max(X,Y) } MAX:=X {MAX=max(X,Y)} (by Assignment Axiom)
(5) {Y = max(X,Y) } MAX:=Y {MAX=max(X,Y)} (by Assignment Axiom)
(6) {T $\wedge$ X $\geq$ Y} MAX:=X {MAX=max(X,Y)} (by Consequence Rule, (1), and (3))
(7) {T $\wedge$ $\neg$(X $\geq$ Y) } MAX:=Y {MAX=max(X,Y)} (by Consequence Rule, (2), and (3))
(8) {T} if X $\geq$ Y then MAX :=X else MAX := Y fi {MAX = max (X,Y)} (by Conditional Rule, (6), and (7))

Iteration Rule

Iteration Rule

$$\frac{\{P \land B\}\ S\ \{P\}}{\{P\}\ \text{while}\ B\ \text{do}\ S\ \text{od}\ \{P \land \neg B\}}$$

Iteration Rule

Iteration Rule

$$\frac{\{P \wedge B\}\ S\ \{P\}}{\{P\}\ \text{while } B \text{ do } S \text{ od}\ \{P \wedge \neg B\}}$$

Example

$\{X \leq 10\}$ while $X < 10$ do $X := X + 1$ od $\{X = 10\}$

$$\frac{\{X+1 \leq 10\}X{:=}X+1\{X \leq 10\}\ \text{(Assignment Axiom)}}{\{X+1 \leq 10 \wedge X \leq 10\}\ X{:=}X+1\{X \leq 10\}}\ \text{by Underlying Logic}$$

$$\frac{\{X+1 \leq 10 \wedge X \leq 10\}\ X{:=}X+1\{X \leq 10\}}{\{X \leq 10\}\ \text{while } X+1 \leq 10 \text{ do } X{:=}X+1 \text{ od}\ \{X \leq 10 \wedge X+1 \not\leq 10\}\ \{X \leq 10\}}\ \text{by Iteration Rule}$$

$$\frac{\{X \leq 10\}\ \text{while } X+1 \leq 10 \text{ do } X{:=}X+1 \text{ od}\ \{X \leq 10 \wedge X+1 \not\leq 10\}\ \{X \leq 10\}}{\text{while } X+1 \leq 10 \text{ do } X{:=}X+1 \text{ od}\ \{X = 10\}}\ \text{by Underlying Logic}$$

Another Example

Example

{T}

R:=X;Q=0; while Y$\leq$ R do R:=R-Y; Q:=Q+1 od

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

Another Example

Example

$\{T\}$
$R:=X;Q=0;$ while $Y \leq R$ do $R:=R-Y;\ Q:=Q+1$ od
$\{ R< Y \wedge X = R + (Y \times Q)\}$

Is valid by underlying logic

$X=R + (Y\times Q) \wedge Y\leq R \Rightarrow X=R-Y + (Y\times Q) +Y$ $\quad$ $\{X=R-Y + (Y\times Q) +Y\}$ $R:=R-Y$ $\{X=R + (Y\times Q)+Y))\}$ (Assignment Axiom)

--- By consequence rule

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

--- By underlying logic

$\{X=R + (Y\times Q) \wedge Y\leq R\}$ $R:=R-Y$ $\{X=R + (Y\times(Q+1))\}$ $\quad$ $\{X=R + (Y\times(Q+1))\}$ $Q:=Q+1$ $\{X=R + (Y\times Q)\}$ (Assignment Axiom)

--- By composition rule

(omitted…try it yourself) $\quad$ $\{X=R + (Y\times Q) \wedge Y\leq R\}$ $R:=R-Y;\ Q:=Q+1$ $\{X=R + (Y\times Q)\}$

--- By Iteration rule

$\{T\}$ $R:=X;\ Q=0$ $\{X=R+(Y*Q)\}$ $\quad$ $\{X=R+(Y*Q)\}$ while $Y\leq R$ do $R:=R-Y;\ Q:=Q+1$ od $\{ R< Y \wedge X = R + (Y \times Q)\}$

--- By composition rule

$\{T\}$ $R:=X;Q=0;$ while $Y\leq R$ do $R:=R-Y;\ Q:=Q+1$ od $\{ R< Y \wedge X = R + (Y \times Q)\}$

Iteration Rule and Invariants

- An invariant at some point of a program is an assertion that holds whenever execution of the program reaches that point.

Iteration Rule

$$\frac{\{P \wedge B\}\ S\ \{P\}}{\{P\}\ \text{while } B \text{ do } S \text{ od } \{P \wedge \neg B\}}$$

- Assertion P in the iteration rule for a while loop is called a loop invariant of the while loop.

How Does One Find an Invariant?

Iteration Rule

$$\frac{\{P \wedge B\}\ S\ \{P\}}{\{P\}\ \text{while}\ B\ \text{do}\ S\ \text{od}\ \{P \wedge \neg B\}}$$

- Look at the facts
  - Invariant P must hold initially
  - With negated test \neg B the invariant must establish the result
  - When the test B 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
  - Gives **the desired result** when the loop terminates

Example

Example

$\{X=n \wedge Y=1\}$
while $X \neq 0$ do $Y:=Y \times X$; $X:=X-1$ od
$\{X=0 \wedge Y=n!\}$

- 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 results so far
  - X! is what remains to be computed
  - n! is the desired results
- The invariant here is $X! \times Y = n!$
  - "Stuff to be done" $\times$ "result so far" = "desired result"
  - Decrease in X combines with increase in Y to make invariant
- Try to prove the specification using the given invariant.

Example

Example

$\{X=0 \wedge Y=1\}$
while $X < N$ do $X:=X+1$; $Y:=Y \times X$ od
$\{Y=N!\}$

- Look at the facts
  - Initially X=0 and Y=1
  - Finally X=N and Y=N!
  - On each loop both X and Y are increased: X by 1 and Y by X
- An invariant should be Y = X!
- Try to prove the specification using the given invariant

Example

Example

$\{X=0 \wedge Y=1\}$
while $X < N$ do $X:=X+1$; $Y:=Y \times X$ od
$\{Y=N!\}$

- Look at the facts
  - Initially X=0 and Y=1
  - Finally X=N and Y=N!
  - On each loop both X and Y are increased: X by 1 and Y by X

- An invariant is $Y = X!$, but not sufficient to prove the results
- At the end need $Y = N!$, but the Iteration rule only gives $\neg (X<N)$

- The invariant needed is $Y = X! \wedge X \leq N$
- At the end, $X \leq N \wedge \neg (X < N) \Rightarrow X=N$
- Often need to strengthen invariants to get them to work.
  - Typical to add thing to “carry along” such as $X \leq N$

Conjunction/Disjunction Rule

Conjunction Rule

$$\frac{\{P_1\}\ S\ \{Q_1\}\quad \{P_2\}\ S\ \{Q_2\}}{\{P_1 \wedge Q_1\}\ S\ \{P_2 \wedge Q_2\}}$$

Disjunction Rule

$$\frac{\{P_1\}\ S\ \{Q_1\}\quad \{P_2\}\ S\ \{Q_2\}}{\{P_1 \vee Q_1\}\ S\ \{P_2 \vee Q_2\}}$$

Some Quick Review

- Which of the following is correct?

Hoare's Assignment Axiom

$$\frac{}{\{P[E/X]\}\ X:=E\ \{P\}}$$

Hoare's Assignment Axiom

$$\frac{}{\{P\}\ X:=E\ \{P[E/X]\}}$$

Some Quick Review

Composition Rule

$$\frac{\{P\}S_1\{R\} \quad \{R\}S_2\{Q\}}{\{P\}S_1;S_2\{Q\}}$$

Iteration Rule

$$\frac{\{P \wedge B\}\ S\ \{P\}}{\{P\}\ \text{while } B \text{ do } S \text{ od}\ \{P \wedge \neg B\}}$$

Conditional Rule

$$\frac{\{P \wedge E\}\ S_1\ \{Q\} \quad \{P \wedge \neg E\}\ S_2\ \{Q\}}{\{P\}\ \text{if } E \text{ then } S_1 \text{ else } S_2\ \{Q\}}$$

Consequence Rule

$$\frac{P \Rightarrow P' \quad \{P'\}\ S\ \{Q'\} \quad Q' \Rightarrow Q}{\{P\}\ S\ \{Q\}}$$

Further Studies

- Soundness and completeness proof for the axioms and inference rules.
- Richer program constructs: pointers, procedure call, arrays, code block
- Automation. E.g., finding loop invariants