COL:750

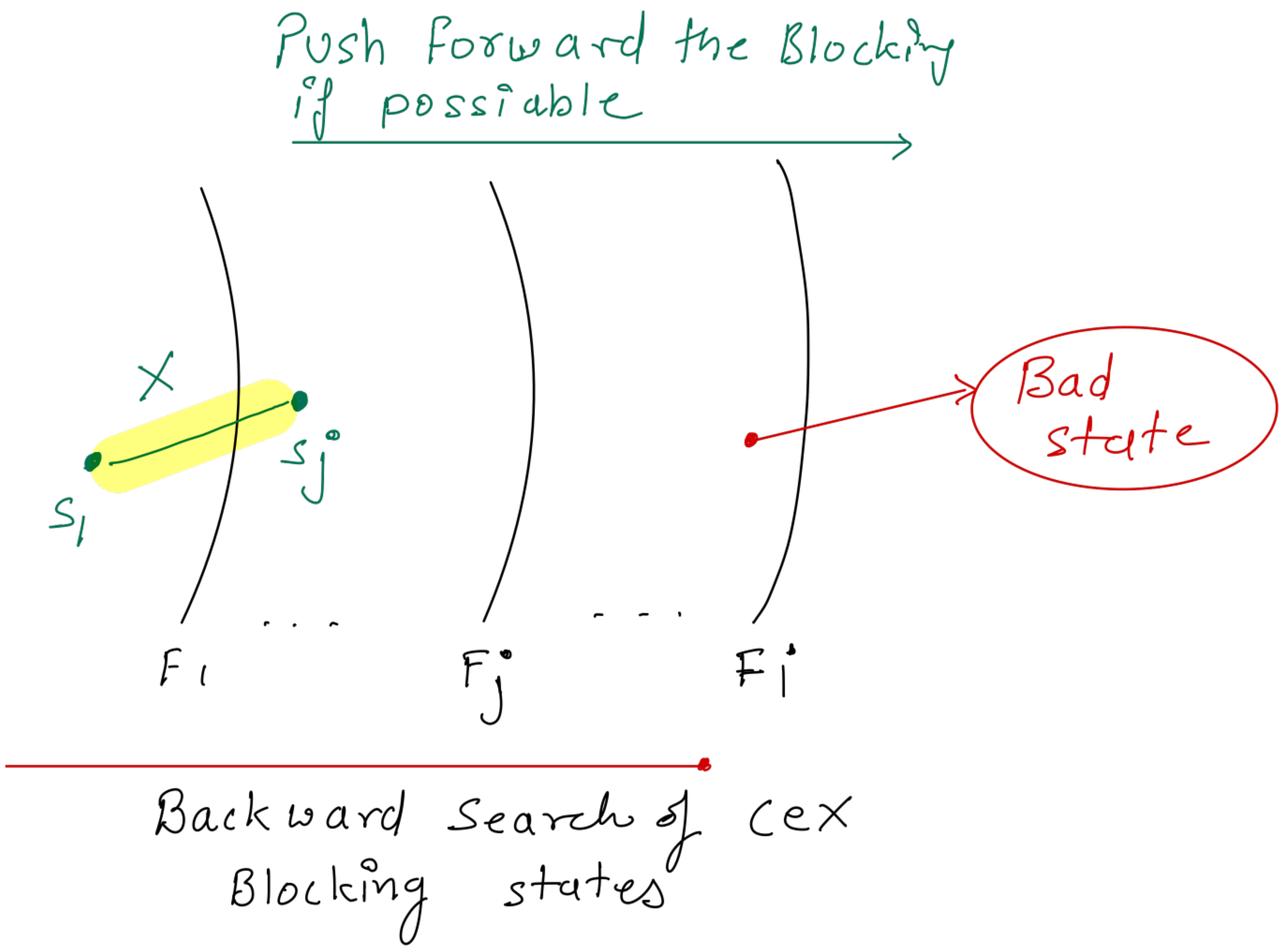
Foundations of Automatic Verification

Instructor: Priyanka Golia

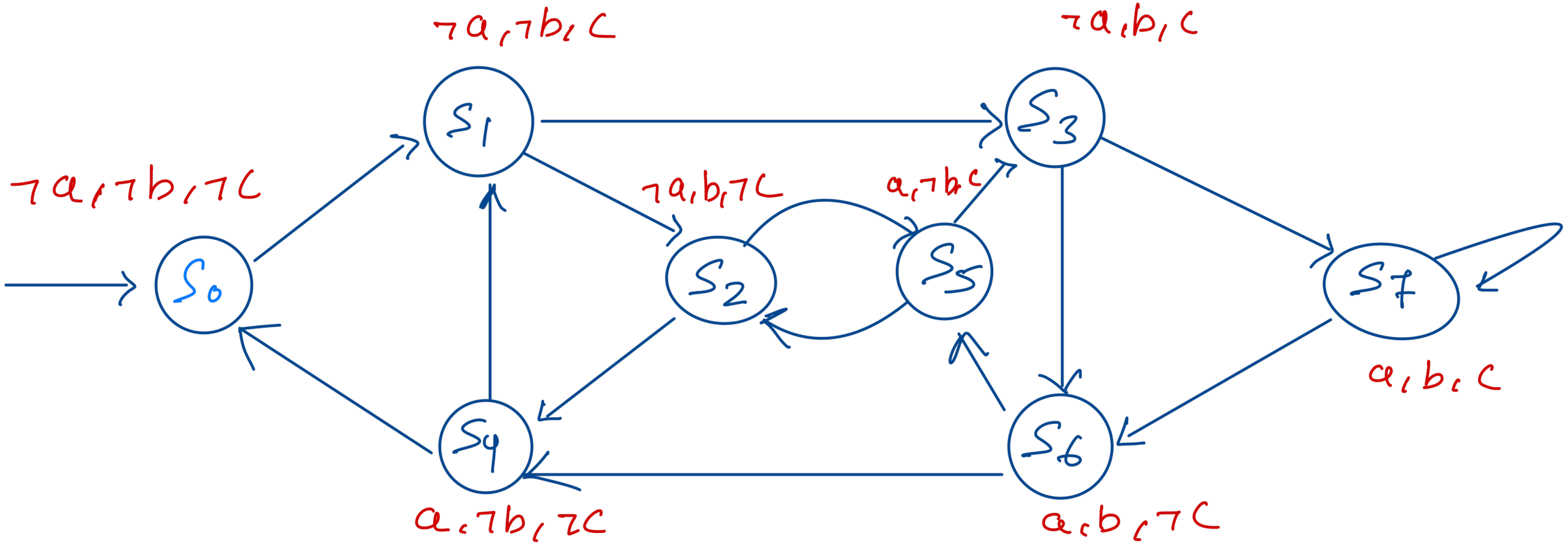
Course Webpage

https://priyanka-golia.github.io/teaching/COL-750/index.html

IC3 : Incremental Construction of Inductive Clauses for Indubitable Correctness.



IC3 : Incremental Construction of Inductive Clauses for Indubitable Correctness.



$$T\,(a, b, c, a', b', c') = (a' \leftrightarrow b) \wedge (b' \leftrightarrow c)$$

$$\forall \Box\, \neg a \vee \neg b \vee \neg c$$

Bounded vs Unbounded Model Checking

Bounded:

We unroll the transition system up to a fixed depth k.

Is there a counterexample of length ≤ k?

If no counterexample is found, we increase k and repeat.

It only checks for violations up to length k

Output— counterexample

Tools: CBMC, NuSMV

Unbounded:

We still unroll transitions, but with a different purpose: to construct an inductive invariant (proof that holds for all k).

We generalize beyond specific length and reason about all reachable states.

The property is proven inductively.

Output— proof/ counterexample

Tools: NuSMV, IC3, PDR

Reasoning About Code!

Our programming language (little but cute!)

Expressions:

E:= Z | V | $E_1 + E_2 | E_1 - E_2 | E_1 \times E_2 | \ldots$

Z = intergers, V = variables

Boolean Expressions:

B := T | F | $E_1 = E_2 | E_1 \leq E_2 | E_1 < E_2 | \ldots$

Commands:

$C := V = E \,|$

$C_1; C_2 \,|$

$IF\ B\ Then\ C_1\ Else\ C_2 \,|$

While $B\ Do\ C$

$x = 17;$

$y = 42;$

$z = x + y;$

$y = 1;$

$z = 0;$

$While(z! = x)do$

$\quad z = z + 1;$

$\quad y = y * z;$

Reasoning About Code!

$x = 17;$

$y = 42;$

$z = x + y;$

$\{True\}$

$x = 17$

$\{x = 17\}$

$y = 42$

$\{(x = 17) \land (y = 42)\}$

$z = x + y$

$\{(x = 17) \land (y = 42) \land (z = 59)\}$

Forward reasoning

Reasoning About Code!

$x = y$

$x = x + 1$

$\{y \leq 0\}$

$x = y$

$\{x \leq 0\}$

$x = x + 1$

$\{x \leq 1\}$

Forward reasoning

$\{y + 1 > 0\}$

$x = y$

$\{x + 1 > 0\}$

$x = x + 1$

$\{(x > 0)\}$

Backward reasoning

Accepted initial state

$y \leq 0$

Action of the program

Accepted final state

$x \leq 1$

Forward Reasoning

Hoare Triples

Partial correctness specification for specifying what a program does:

$$\{P\}\ C\ \{Q\}$$

P is called "Pre-condition"
C is code
Q is called "Post-condition"

C is a command
P, Q are conditions on the program variables used in C

If P holds Trues, and C is executed and terminated, then Q is guaranteed to be True afterwards — If this holds, then **{P} C {Q} is a valid Hoare Triple.**

$\{x \neq 0\}\ y = x \times x\ \{y > 0\}$ Valid Hoare Triple

$\{x \geq 0\}\ y = 2 \times x\ \{y > 0\}$ Not a Valid Hoare Triple

Hoare Triples — Partial and Total Correctness

What if the code doesn't terminate!

$\{P\}\ C\ \{Q\}$ is valid under partial correctness if from a instance in P, when C is executed, and if C is terminated, then Q will hold.

$\{P\}\ C\ \{Q\}$ is valid under total correctness if from instances in P, when C is executed, and C is guaranteed to terminate, and Q will hold.

$\{x = 1\}$ *While True Do* $x = x$ $\{y = 2\}$ *Valid*

In partial correctness!
the postcondition is trivially satisfied in all terminating executions — of which there are none.

Hoare Triples — Partial and Total Correctness

What if the code doesn't terminate!

$\{P\}\ C\ \{Q\}$ is valid under partial correctness if from a instance in P, when C is executed, and if C is terminated, then Q will hold.

$\{P\}\ C\ \{Q\}$ is valid under total correctness if from instances in P, when C is executed, and C is guaranteed to terminate, and Q will hold.

Total Correctness = Partial Correctness + Termination!

We will restrict to Partial Correctness

One can show partial correctness and termination separately!

Termination

$$While\ x > 1\ Do$$
$$If(x\,\%\,2 == 1)\ Then\ x = 3x + 1\ Else\ x = x/2$$

Collatz conjecture! No proof that it will terminate

Our Task! Is to prove the correctness of the code, given their specification.

We had a specification, we wrote a code.

Input, Specification, output {P} C {Q}

From the code, we can construct Pre and Post condition to have a valid Hoare triples!
This should represent input, output of the specification.

Want strongest Post condition!

$\{x \geq 0\} y = 1; z = 2y + x; \{z \geq 0\}$

$\{x \geq 0\} y = 1; z = 2y + x; \{z \geq 2\}$

$\{x \geq 0\} y = 1; z = 2y + x; \{z \geq x\}$

$\{x \geq 0\} y = 1; z = 2y + x; \{z == x + 2\}$

In some sense (informally), solution space of Q should be as small as possible!

What exactly can happen after running C, assuming the input satisfies P?

Our Task! Is to prove the correctness of the code, given their specification.

We had a specification, we wrote a code.

Input, Specification, output $\{P\}\ C\ \{Q\}$

From the code, we can construct Pre and Post condition to have a valid Hoare triples!
This should represent input, output of the specification.

Want weakest pre condition!

least restrictive condition that still guarantees that after executing C, the postcondition Q holds.

$\{x \geq 6\} x = x + 1 \{x > 5\}$

$\{x == 10\} x = x + 1 \{x > 5\}$

$\{x \geq 5\} x = x + 1 \{x > 5\}$

It's the most general condition that guarantees Q holds after executing the statement.

In some sense (informally), solution space of P should be as big as possible!

Floyd - Hoare Logic

A deductive proof system for Hoare triples

Proof system by Hoare and “some” underlying ideas by Floyd

Deductive proof system — derive conclusion from premises using a set of rules and axioms

To prove/disprove that a Hoare Triple is valid

A proof in Floyd-Hoare logic is a sequence of lines, each of which is either an axiom of the logic or follows from earlier lines by a rule of inference of the logic

Floyd - Hoare Logic Assignment Axiom

Assignment Axiom:

$$\{Q[x = E]\}\ x = E;\ \{Q\}$$

With respect to Post condition!

$\{y + 1 > 0\}$ $Q[x = y]$

$x = y$

$\{x + 1 > 0\}$ $Q[x = x + 1]$

$x = x + 1$

$\{(x > 0)\}$

Backward reasoning

$\{x = 2\}$ $Q[x = x + 1]$

$x = x + 1$

$\{(x = 3)\}$

Floyd - Hoare Logic

Sequence Rule

$$\frac{\{P\}C_1\{R\}, \{R\}C_2\{Q\}}{\{P\}C_1, C_2\{Q\}}$$

Premises

Conclusion

$$\frac{\{y+1 \geq 0\}x = y\{x+1 \geq 0\}, \{x+1 \geq 0\}x = x+1; \{x > 0\}}{\{y+1 \geq 0\}x = y; x = x+1; \{x > 0\}}$$

$$\frac{\{True\}x = 5\{x == 5\}, \{x == 5\}y = 2 \times x; \{y = 10\}}{\{True\}x = 5; y = 2 \times ; \{y = 10\}}$$

Floyd - Hoare Logic

Conditional Rule

$$\frac{\{P \land B\}C_1\{Q\}, \{P \land \neg B\}C_2\{Q\}}{\{P\}if\ B\ then\ C_1\ else\ C_2\ \{Q\}}$$

$$\frac{\{True \land (x > 0)\}y = 1\{y = 1 \lor y = 2\}, \{True \land (x \leq 0)\}y = 2\{y = 2\}}{\{True\}if\ x > 0\ then\ y = 1\ else\ y = 2\ \{y = 1 \lor y = 2\}}$$

Floyd - Hoare Logic

While Rule

- Loop Invariants (I) —
    - It should hold before the loop starts.
    - If it holds before the an iterations, and loop guard (B) is True, it must hold after executing the body C.
    - If the loop terminates, $I \land \neg B$ holds

      And, should be strong to imply the post condition

while $x > 0$ do

$x = x - 1$

$I = x \geq 0$

$sum = 0;\ i = 1;$

while $i \leq n$ do

$sum = sum + i; i = i + 1$

$I = sum \geq 0$

$I = (1 \leq i \leq n) \land (sum = \sum_{j=1}^{i-1} j)$

Floyd - Hoare Logic

While Rule

$$\frac{\{I \wedge B\}C\{I\}}{\{I\}\ While\ B\ do\ C\{I \wedge \neg B\}}$$

If executing C once preserve the truth of I, then executing C a number of times also preserve the truth of I

while $x > 0$ do

$x = x - 1$

$$\frac{\{x \geq 0) \wedge (x > 0)\}x = x - 1\{x \geq 0\}}{\{x \geq 0\}\ While\ x > 0\ do\ x = x - 1\ \{x \geq 0 \wedge \neg B\}}$$

Floyd - Hoare Logic

Consequence Rule

$$\frac{P' \rightarrow P \quad \{P\}C\{Q\} \quad Q \rightarrow Q'}{\{P'\}C\{Q'\}}$$

$$\frac{\{x = 2\}x = x + 1\{x = 3\} \ (x = 3) \rightarrow (x \geq 3)}{\{x = 2\}x = x + 1\{x \geq 3\}}$$

$$\frac{(x = 0) \rightarrow (x \geq 0) \quad \{x \geq 0\}x = x + 1\{x > 1\}}{\{x = 0\}x = x + 1\{x \geq 1\}}$$

Toy Examples:

$$z = x\ ;\ z = z + y$$

To prove/disprove $\{True\}\ z = x\ ;\ z = z + y\ \{z = x + y\}$

$\{True\}$

$z = x$

$\{z = x\}$

$z = z + y$

$\{z = x + y\}$

Compute precondition using Assignment Axiom:

$$\{Q[x = E]\}\ x = E;\ \{Q\}$$

$$x + y = z + y \quad x = x$$

Sequence Rule $\dfrac{\{P\}C_1\{R\}, \{R\}C_2\{Q\}}{\{P\}C_1, C_2\{Q\}}$

$$\{True\}\ z = x\ ;\ z = z + y\ \{z = x + y\}$$

Toy Examples:

$$\{True\} \;\; Prog \;\; \{y = x + 1\}$$

$a = x + 1;$

$if(a - 1 == 0)$

$\quad y = 1;$

$else$

$\quad y = a;$

Toy Examples:

$$\{True\} \quad Prog \quad \{y = x + 1\}$$

$\{True\}$

$(x == 0) \rightarrow x = 0) \wedge (\neg(x == 0) \rightarrow x + 1 = x + 1)$

$a = x + 1;$

$((a - 1 == 0) \rightarrow x = 0) \wedge (\neg(a - 1 == 0) \rightarrow a = x + 1)$

$if(a - 1 == 0)$

$\{1 = x + 1\}$

$y = 1;$

$\{y = x + 1\}$

$else$

$\{a = x + 1\}$

$y = a;$

$\{y = x + 1\}$

$\{y = x + 1\}$

Conditional Rule $\dfrac{\{P \wedge B\}C_1\{Q\}, \{P \wedge \neg B\}C_2\{Q\}}{\{P\}if\ B\ then\ C_1\ else\ C_2\ \{Q\}}$

Assignment Axiom

Toy Examples:

$$\{n \geq 0\} \;\; prog \;\; \{fact = n!\}$$

$i = 1$

$fact = 1$

$while \; i \leq n \; do$

$\quad fact = fact \times i$

$\quad i = i + 1$

Toy Examples:

$$\{n \geq 0\} \ \ prog \ \{fact = n!\}$$

$\{(0 \leq n)\}$

$\{(1 \leq n+1) \wedge (1 = (1-1)!)\}$

$i = 1$

$\{(1 \leq i \leq n+1) \wedge (1 = (i-1)!)\}$

$fact = 1$

$\{(1 \leq i \leq n+1) \wedge (fact = (i-1)!)\}$

$while \ i \leq n \ do$

$\{(1 \leq i \leq n+1) \wedge (fact = (i-1)!)\} \wedge \{i \leq n\}$

$\{(1 \leq i \leq n) \wedge (fact \times i = i \times (i-1)!)\}$

$fact = fact \times i$

$\{(1 \leq i \leq n) \wedge (fact = (i)!)\}$

$i = i + 1$

$\{(1 \leq i \leq n+1) \wedge (fact = (i-1)!)\}$

$\{(1 \leq i \leq n+1) \wedge (fact = (i-1)!)\} \wedge \{ i > n\}$

$\{fact = n!\}$

While Rule $$\frac{\{I \wedge B\}C\{I\}}{\{I\} \ While \ B \ do \ C\{I \wedge \neg B\}}$$

$$I = (1 \leq i \leq n+1) \wedge (fact = (i-1)!)$$

We want $\{I \wedge B\}C\{I\}$

Using Consequence Rule $$\frac{P' \rightarrow P \quad \{P\}C\{Q\}}{\{P'\}C\{Q\}}$$

Toy Examples:

$$\{True\}\ \ prog\ \{y = x!\}$$

$y = 1$

$z = 0$

$while\ z != x\ do$

$\quad z = z + 1$

$\quad y = y \times z$

Assuming input to this program is x.

Toy Examples:

$$\{n \geq 0\} \;\; prog \; \{sum = \sum_{i=1}^{n} i\}$$

$sum = 0$

$i = 1$

$while \; i \leq n \; do$

$\quad sum = sum + i$

$\quad i = i + 1$

Assuming input to this program is n.