

# HOARE LOGIC



# HOARE LOGIC

## INTRODUCTION

- Since finding the exact  $wp$  or  $sp$  for while-loops is difficult, we will use an over-approximation in the form of an **inductive invariant** which preserves soundness.
- Much of the rest of the course (and majority of research in verification) deals with how to handle the verification problem for loops/loop-like constructs.
- Hoare Logic is a program logic/verification strategy which can be directly used to prove the validity of Hoare Triples.
- Also provides a framework for specifying and verifying Inductive Loop Invariants.



# DEFINITION

- Given sets of states  $P$  and  $Q$ , a program  $c$  satisfies the specification  $\{P\}c\{Q\}$  if:
  - $\forall \sigma, \sigma'. \sigma \in P \wedge (\sigma, c) \hookrightarrow^* (\sigma', \text{skip}) \Rightarrow \sigma' \in Q$
- Using FOL formulae  $P$  and  $Q$  to express sets of states, we can now use the symbolic semantics  $\rho(c)$ :
  - $\forall V, V'. P \wedge \rho(c) \rightarrow Q[V'/V]$
- Hoare Logic is a program logic/proof system to directly prove the validity of Hoare Triples.
- We will study it in two forms:
  - A set of inference rules
  - A procedure to generate verification conditions (VCs) in FOL



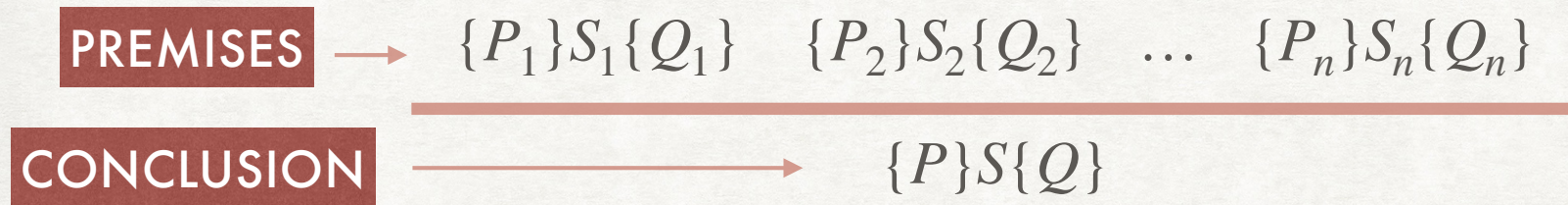
## RELATION WITH WP AND SP

- How are Hoare Triples, Weakest Pre-condition and Strongest Post-condition related with each other?
  - $\{wp(P, c)\} \subseteq \{P\}$
  - $\{P\} \subseteq \{sp(P, c)\}$
- **Homework:** Prove this formally using the definitions!



# INFERENCE RULES

## FORMAT



Key Idea: Use the validity of Hoare triples for smaller statements to establish validity for compound statements



# INFERENCE RULES

## PRIMITIVE STATEMENTS

---

$$\{P[e/x]\} x := e \{P\}$$

[R-ASSIGN]

---

$$\{\forall x. P\} x := \text{havoc} \{P\}$$

[R-HAVOC]

---

$$\{Q \rightarrow P\} \text{assume}(Q) \{P\}$$

[R-ASSUME]

---

$$\{Q \wedge P\} \text{assert}(Q) \{P\}$$

[R-ASSERT]



# EXAMPLES

- Which of the following are true?
  - $\{y = 10\} \ x := 10 \ \{y = x\}$
  - $\{x = n - 1\} \ x := x + 1 \ \{x = n\}$
  - $\{y = x\} \ y := 2 \ \{y = x\}$
  - $\{z = 10\} \ y := 2 \ \{z = 10\}$
  - $\{y = 10\} \ y := x \ \{y = x\}$
- The last Hoare triple is valid, but we cannot prove it using [R-ASSIGN].
  - According to [R-ASSIGN], we have  $\{y = x[x/y]\} \ y := x \ \{y = x\}$ . Hence,  $\{x = x\} \ y := x \ \{y = x\}$ , which simplifies to  $\{ \top \} \ y := x \ \{y = x\}$ .
  - Notice that  $y = 10 \Rightarrow \top$ .



# PRE-CONDITION STRENGTHENING

$$\{P'\} \text{ c } \{Q\} \quad P \Rightarrow P'$$

---

$$\{P\} \text{ c } \{Q\}$$

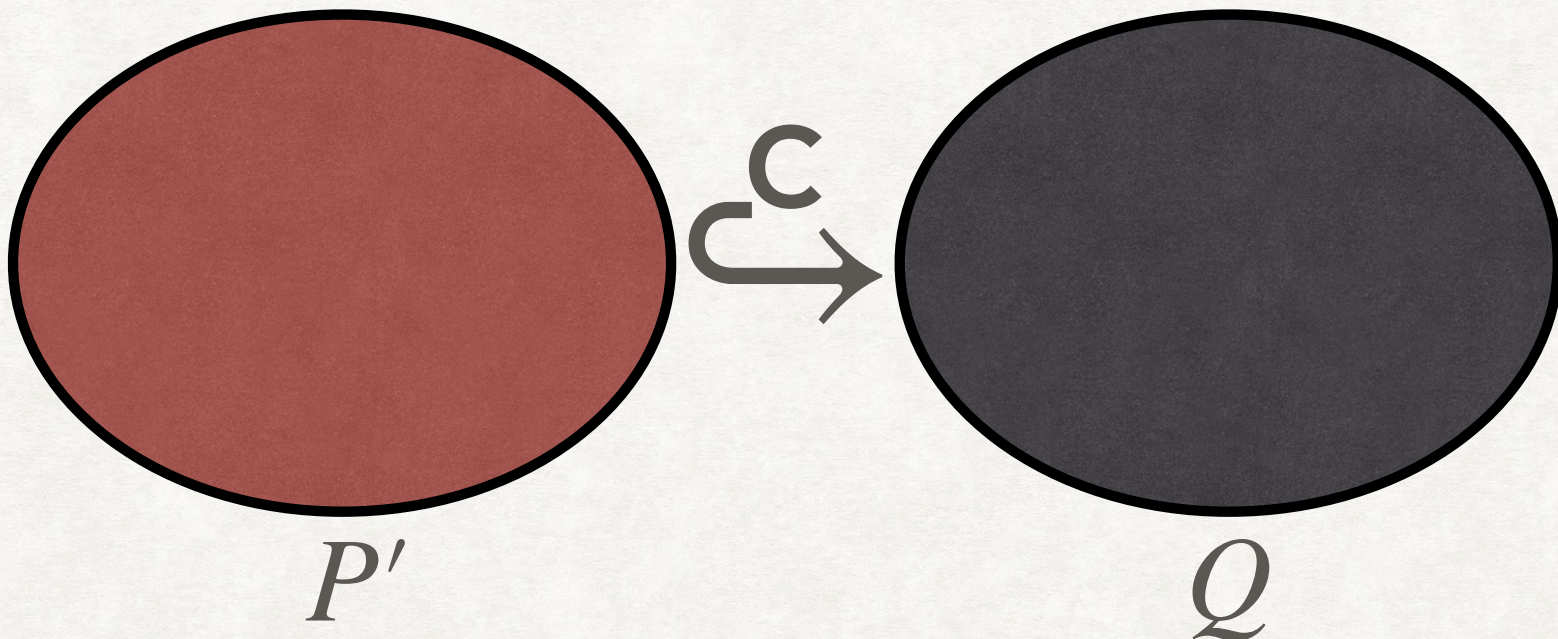
[R-STRENGTHEN-PRE]



# PRE-CONDITION STRENGTHENING

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

[R-STRENGTHEN-PRE]

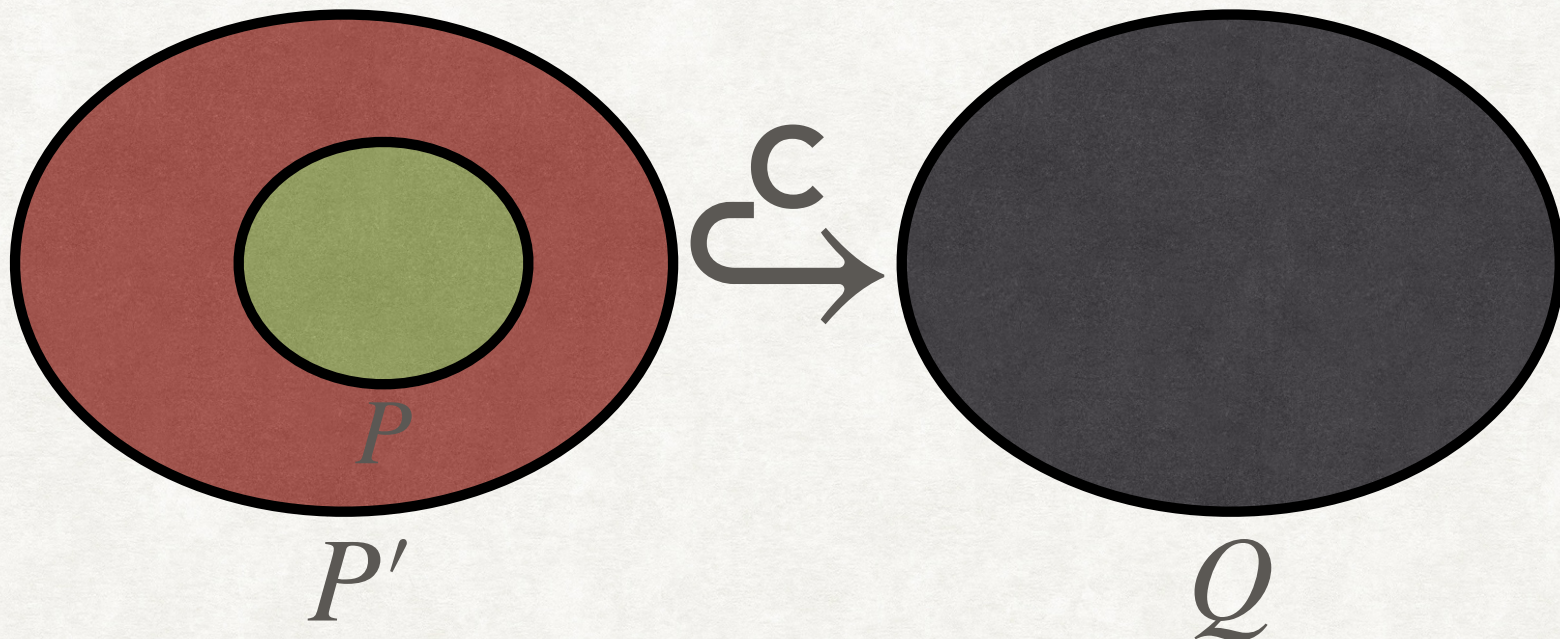




# PRE-CONDITION STRENGTHENING

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

[R-STRENGTHEN-PRE]



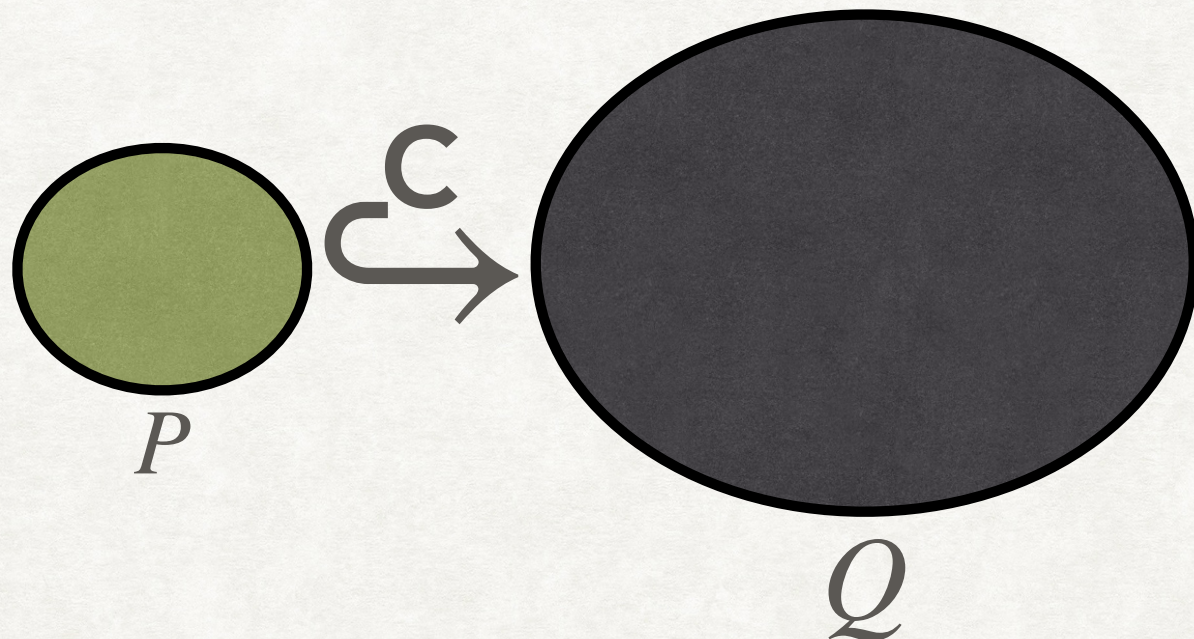


# PRE-CONDITION STRENGTHENING

$$\{P'\} \subset \{Q\} \quad P \Rightarrow P'$$

[R-STRENGTHEN-PRE]

$$\{P\} \subset \{Q\}$$





# PRE-CONDITION STRENGTHENING

$$\{P'\} \text{ c } \{Q\} \quad P \Rightarrow P'$$

[R-STRENGTHEN-PRE]

---

$$\{P\} \text{ c } \{Q\}$$

---

$$\{true\} \ y := x \ \{y = x\} \quad y = 10 \Rightarrow true$$

---

$$\{y = 10\} \ y := x \ \{y = x\}$$



# POST-CONDITION WEAKENING

$$\{P\} \text{ c } \{Q'\} \quad Q' \Rightarrow Q$$

---

$$\{P\} \text{ c } \{Q\}$$

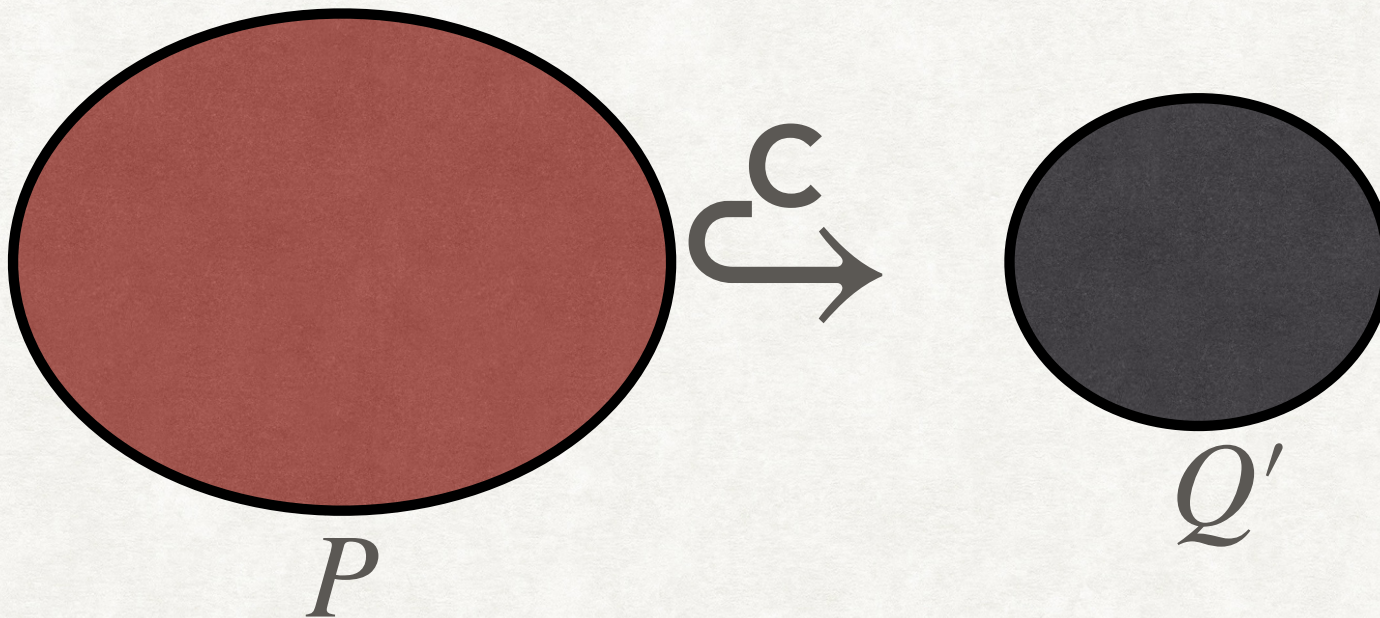
[R-WEAKEN-POST]



# POST-CONDITION WEAKENING

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

[R-WEAKEN-POST]

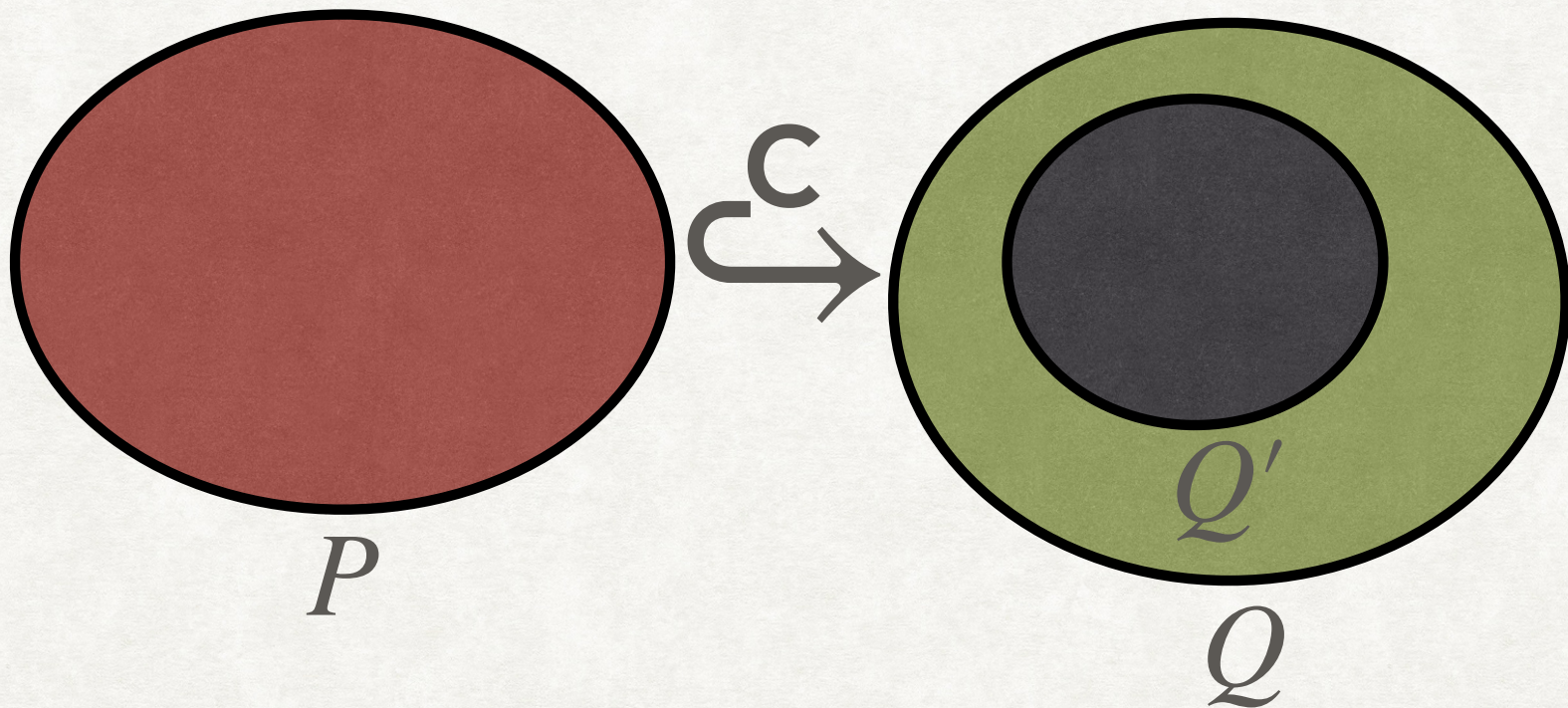




# POST-CONDITION WEAKENING

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

[R-WEAKEN-POST]





# INFERENCE RULES

## COMPOUND STATEMENTS

$$\{P\} \text{ c}_1 \{R\} \quad \{R\} \text{ c}_2 \{Q\}$$

---

$$\{P\} \text{ c}_1; \text{ c}_2 \{Q\}$$

[R-SEQ]



# INFERENCE RULES

## COMPOUND STATEMENTS

$$\{P\} c_1 \{R\} \quad \{R\} c_2 \{Q\}$$

---

$$\{P\} c_1; c_2 \{Q\}$$

[R-SEQ]

$$\{P \wedge F\} c_1 \{Q\} \quad \{P \wedge \neg F\} c_2 \{Q\}$$

---

$$\{P\} \text{ if } (F) \text{ then } c_1 \text{ else } c_2 \{Q\}$$

[R-IF-THEN-ELSE]



# INFERENCE RULES

## COMPOUND STATEMENTS

$$\{P\} c_1 \{R\} \quad \{R\} c_2 \{Q\}$$

[R-SEQ]

---

$$\{P\} c_1; c_2 \{Q\}$$

$$\{P \wedge F\} c_1 \{Q\} \quad \{P \wedge \neg F\} c_2 \{Q\}$$

[R-IF-THEN-ELSE]

---

$$\{P\} \text{ if } (F) \text{ then } c_1 \text{ else } c_2 \{Q\}$$

Prove This!



# SEQUENCING

## EXAMPLE

$$\{P\} \text{ c}_1 \{R\} \quad \{R\} \text{ c}_2 \{Q\}$$

[R-SEQ]

---

$$\{P\} \text{ c}_1; \text{ c}_2 \{Q\}$$

---

$$\{true\} \text{ x} := 2; \text{ y} := \text{x} \{y = 2 \wedge \text{x} = 2\}$$



# SEQUENCING

## EXAMPLE

$$\{P\} c_1 \{R\} \quad \{R\} c_2 \{Q\}$$

[R-SEQ]

$$\{P\} c_1; c_2 \{Q\}$$

$$\{true\} x := 2 \{x = 2\}$$

$$\{x = 2\} y := x \{y = 2 \wedge x = 2\}$$

$$\{true\} x := 2; y := x \{y = 2 \wedge x = 2\}$$



# IF-THEN-ELSE

## EXAMPLE

$$\{P \wedge F\} \text{ c}_1 \{Q\} \quad \{P \wedge \neg F\} \text{ c}_2 \{Q\}$$

[R-IF-THEN-ELSE]

---

$$\{P\} \text{ if } (F) \text{ then c}_1 \text{ else c}_2 \{Q\}$$

---

$$\{true\} \text{ if } (x > 0) \text{ then } y := x \text{ else } y := -x \{y \geq 0\}$$



# IF-THEN-ELSE

## EXAMPLE

$$\{P \wedge F\} c_1 \{Q\} \quad \{P \wedge \neg F\} c_2 \{Q\}$$

[R-IF-THEN-ELSE]

$$\{P\} \text{ if } (F) \text{ then } c_1 \text{ else } c_2 \{Q\}$$

$$\{x \geq 0\} y := x \{y \geq 0\} \quad x > 0 \Rightarrow x \geq 0$$

$$\{x > 0\} y := x \{y \geq 0\}$$

$$\{x \leq 0\} y := -x \{y \geq 0\}$$

$$\{true\} \text{ if } (x > 0) \text{ then } y := x \text{ else } y := -x \{y \geq 0\}$$



# WHILE LOOPS

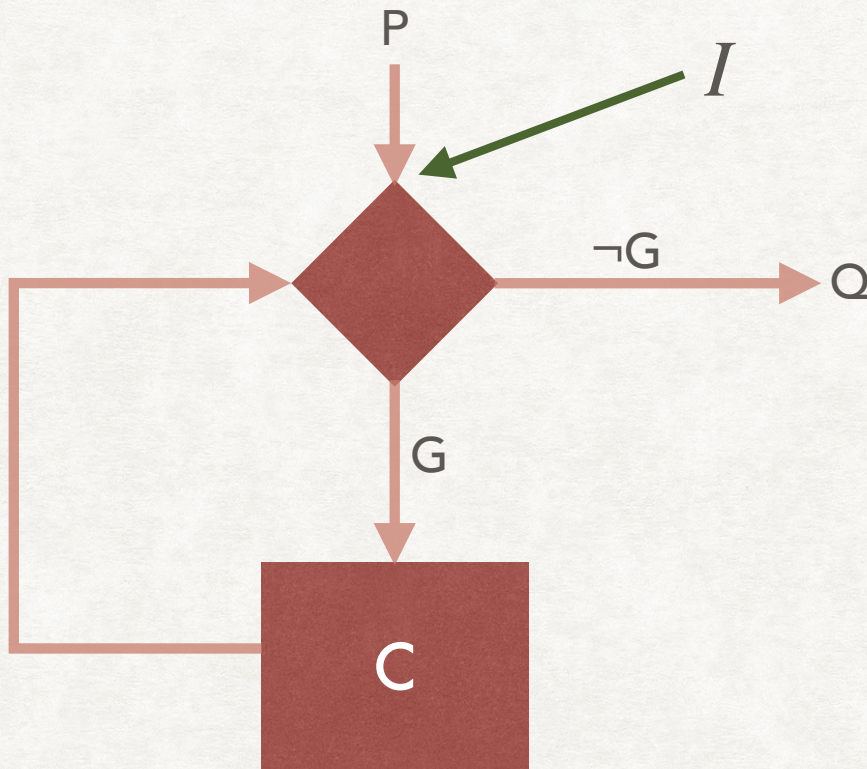
## LOOP INVARIANTS

- Our goal is to prove the validity of  $\{P\} \text{ while}(G) \text{ do } c \{Q\}$ 
  - Both  $sp$  and  $wp$  lead to non-terminating procedures.
- We will instead assume a loop invariant  $I$  which is an over-approximation of the states possible during execution of the loop, but is sufficient enough to prove the Hoare Triple.
  - $I$  is assumed to be provided by the programmer.
- Hoare Logic provides inference rules to prove that  $I$  is indeed a loop invariant.



# WHILE LOOPS

## LOOP INVARIANTS



- $I$  needs to satisfy three properties:
- $I$  must hold initially at the start of the loop.
- $I$  must hold at the end of every iteration of the loop.
- After exiting from the loop,  $I$  must imply the post-condition.



# WHILE LOOPS

## LOOP INVARIANTS

- Consider the code
  - `i:=0; j:=0; while(i<n) do i:=i+1; j:=i+j;`
- Which of the following are loop invariants?
  - $i \leq n$
  - $i < n$
  - $i \geq 0$



# WHILE LOOP INFERENCE RULE

$$\{I \wedge F\} \text{ c } \{I\}$$

---

$$\{I\} \text{ while}(F) \text{ do c; } \{I \wedge \neg F\}$$

[R-WHILE-1]



# WHILE LOOP INFERENCE RULE

$$\{I \wedge F\} \text{ c } \{I\}$$

---

[R-WHILE-1]

$$\{I\} \text{ while}(F) \text{ do c; } \{I \wedge \neg F\}$$

$$P \Rightarrow I \quad \{I \wedge F\} \text{ c } \{I\} \quad I \wedge \neg F \Rightarrow Q$$

---

[R-WHILE-2]

$$\{P\} \text{ while}(F) \text{ do c; } \{Q\}$$



# WHILE LOOP

## INFERENCE RULE - EXAMPLE

Prove  $\{i = 0 \wedge n > 0\}$  while( $i < n$ ) do  $i := i+1$ ;  $\{i = n\}$

Loop Invariants:  $i \geq 0$ ,  $i \leq n$ ,  $n > 0 \dots$

Which loop invariant is useful for proving the Hoare Triple?

$$I \triangleq i \leq n$$



# WHILE LOOP

## INFERENCE RULE - EXAMPLE

Prove  $\{i = 0 \wedge n > 0\}$  while( $i < n$ ) do  $i := i+1$ ;  $\{i = n\}$

Loop Invariants:  $i \geq 0$ ,  $i \leq n$ ,  $n > 0 \dots$

Which loop invariant is useful for proving the Hoare Triple?

$$I \triangleq i \leq n$$

---

$\{i = 0 \wedge n > 0\}$  while( $i < n$ ) do  $i := i+1$ ;  $\{i = n\}$



# WHILE LOOP

## INFERENCE RULE - EXAMPLE

Prove  $\{i = 0 \wedge n > 0\} \text{ while}(i < n) \text{ do } i := i+1; \{i = n\}$

Loop Invariants:  $i \geq 0, i \leq n, n > 0 \dots$

Which loop invariant is useful for proving the Hoare Triple?

$$I \triangleq i \leq n$$

---

$$\{i \leq n \wedge i < n\} i := i+1; \{i \leq n\}$$

---

$$\{i = 0 \wedge n > 0\} \Rightarrow i \leq n \quad \{i \leq n \wedge i < n\} i := i+1; \{i \leq n\} \quad i \leq n \wedge i \geq n \Rightarrow i = n$$

---

$$\{i = 0 \wedge n > 0\} \text{ while}(i < n) \text{ do } i := i+1; \{i = n\}$$



# LOOP INVARIANT VS INDUCTIVE LOOP INVARIANT

- Consider again the code
  - $i := 0; j := 0; \text{ while}(i < n) \text{ do } i := i + 1; j := i + j;$
- Is  $j \geq 0$  a loop invariant?
  - Yes, it does hold at the beginning and at the end of every iteration.
- Does  $\{j \geq 0 \wedge i < n\} i := i + 1; j := i + j; \{j \geq 0\}$  hold?
  - NO!  $j \geq 0$  is not an inductive loop invariant.
  - The inference rule admits only inductive loop invariants.
- How to strengthen the invariant to make it inductive?
  - $j \geq 0 \wedge i \geq 0$  is an inductive loop invariant.