

# CS477 Formal Software Dev Methods

Elsa L Gunter  
2112 SC, UIUC  
egunter@illinois.edu

<http://courses.engr.illinois.edu/cs477>

Slides based in part on previous lectures  
by Mahesh Vishwanathan, and by Gul Agha

March 16, 2018

# Algorithm for Proving Hoare Triples?

- Have seen in Isabelle that much of proving a Hoare triple is routine
- Will this always work?
- Why not automate the whole process?
  - Can't (always) calculate needed loop invariants
  - Can't (always) prove implications (side-conditions) in Rule of Consequence application
- Can we automate all but this?
- Yes! But how?
  1. Annotate all **while** loops with needed invariants
  2. Use routine to “roll back” post-condition to **weakest precondition**, gathering side-conditions as we go
- 2 called **verification condition generation**

# Annotated Simple Imperative Language

- Give verification conditions for an annotated version of our simple imperative language
- Add a presumed invariant to each while loop

$$\begin{aligned} \langle \text{command} \rangle &::= \langle \text{variable} \rangle := \langle \text{term} \rangle \\ &| \langle \text{command} \rangle; \dots; \langle \text{command} \rangle \\ &| \text{if } \langle \text{statement} \rangle \text{ then } \langle \text{command} \rangle \text{ else } \langle \text{command} \rangle \\ &| \text{while } \langle \text{statement} \rangle \text{ inv } \langle \text{statement} \rangle \text{ do } \langle \text{command} \rangle \end{aligned}$$

Example: `while y < n inv x = y * y`  
`do`  
    `x := (2 * y) + 1;`  
    `y := y + 1`  
`od`

# Hoare Logic for Annotated Programs

Assignment Rule

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

Rule of Consequence

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

Sequencing Rule

$$\frac{\{P\} C_1 \{Q\} \quad \{Q\} C_2 \{R\}}{\{P\} C_1; C_2 \{R\}}$$

If Then Else Rule

$$\frac{\{P \wedge B\} C_1 \{Q\} \quad \{P \wedge \neg B\} C_2 \{Q\}}{\{P\} \text{ if } B \text{ then } C_1 \text{ else } C_2 \{Q\}}$$

While Rule

$$\frac{\{P \wedge B\} C \{P\}}{\{P\} \text{ while } B \text{ inv } P \text{ do } C \{P \wedge \neg B\}}$$

# Relation Between Two Languages

- Hoare Logic for Simple Imperative Programs and Hoare Logic for Annotated Programs almost the same
- What is precise relationship?
- First need precise relation between the two languages

## Definition

```
strip( $v := e$ ) =  $v := e$   
strip( $C_1 ; C_2$ ) = strip( $C_1$ ) ; strip( $C_2$ )  
strip(if  $B$  then  $C_1$  else  $C_2$  fi) =  
    if  $B$  then strip( $C_1$ ) else strip( $C_2$ ) fi  
strip(while  $B$  inv  $P$  do  $C$  od) = while  $B$  do strip( $C$ ) od
```

- We recursively remove all invariant annotations from all **while** loops

# Relation Between Two Hoare Logics

## Theorem

For all pre- and post-conditions  $P$  and  $Q$ , and annotated programs  $C$ , if  $\{P\} C \{Q\}$ , then  $\{P\} \text{strip}(C) \{Q\}$ .

## Proof.

(Sketch) Use rule induction on proof of  $\{P\} C \{Q\}$ ; in case of While Rule, erase invariant from program □

# Relation Between Two Hoare Logics

## Theorem

*For all pre- and post-conditions  $P$  and  $Q$ , and unannotated programs  $C$ , if  $\{P\} C \{Q\}$ , then there exists an annotated program  $S$  such that  $C = \text{strip}(S)$  and  $\{P\} S \{Q\}$ .*

## Proof.

(Sketch) Use rule induction on proof of  $\{P\} C \{Q\}$ ; in case of While Rule, add invariant from precondition as invariant to command. □

# Weakest Precondition

**Question:** Given post-condition  $Q$ , and annotated program  $C$ , what is the most general pre-condition  $P$  such that  $\{P\} C \{Q\}$ ?

**Answer:** Weakest Precondition

## Definition

$$\begin{aligned} \text{wp } (x := e) Q &= Q[x \leftarrow e] \\ \text{wp } (C_1; C_2) Q &= \text{wp } C_1 (\text{wp } C_2 Q) \\ \text{wp } (\text{if } B \text{ then } C_1 \text{ else } C_2 \text{ fi}) Q &= \\ &\quad (B \wedge (\text{wp } C_1 Q)) \vee ((\neg B) \wedge (\text{wp } C_2 Q)) \\ \text{wp } (\text{while } B \text{ inv } P \text{ do } C \text{ od}) Q &= P \end{aligned}$$

Assumes, without verifying, that  $P$  is the correct invariant



# Weakest Justification

**Weakest** in weakest precondition means any other valid precondition implies it:

## Theorem

*For all annotated programs  $C$ , and pre- and post-conditions  $P$  and  $Q$ , if  $\{P\} C \{Q\}$  then  $P \Rightarrow wp C Q$ .*

- Proof somewhat complicated
- Uses induction on the structure of  $C$
- In each case, want to assert triple proof must have used rule for that construct (e.g. [Sequence Rule](#) for sequences)
- Can't because of [Rule Of Consequence](#)
- Must induct on proof (rule induction) - in each case
- Uses:

## Lemma

$\forall C P Q. (P \Rightarrow Q) \Rightarrow (wp C P \Rightarrow wp C Q)$

# What About Precondition?

Question: Do we have  $\{wp \ C \ Q\} \ C \ \{Q\}$ ?

# What About Precondition?

**Question:** Do we have  $\{wp\ C\ Q\} \ C\ \{Q\}$ ?

**Answer:** Not always - need to check **while**-loop side-conditions –  
verification conditions

# What About Precondition?

**Question:** Do we have  $\{wp\ C\ Q\} \ C\ \{Q\}$ ?

**Answer:** Not always - need to check **while**-loop side-conditions – verification conditions

**Question:** How to calculate verification conditions?

# What About Precondition?

**Question:** Do we have  $\{wp \ C \ Q\} \ C \ \{Q\}$ ?

**Answer:** Not always - need to check **while**-loop side-conditions – verification conditions

**Question:** How to calculate verification conditions?

## Definition

$$vcg(x := e) \ Q = \text{true}$$

# What About Precondition?

**Question:** Do we have  $\{\text{wp } C \ Q\} \ C \ \{Q\}$ ?

**Answer:** Not always - need to check **while**-loop side-conditions – verification conditions

**Question:** How to calculate verification conditions?

## Definition

$\text{vcg } (x := e) \ Q = \text{true}$

$\text{vcg } (C_1; C_2) \ Q = (\text{vcg } C_1 \ (\text{wp } C_2 \ Q)) \wedge (\text{vcg } C_2 \ Q)$

# What About Precondition?

Question: Do we have  $\{\text{wp } C \ Q\} \ C \ \{Q\}$ ?

Answer: Not always - need to check **while**-loop side-conditions – verification conditions

Question: How to calculate verification conditions?

## Definition

$\text{vcg } (x := e) \ Q = \text{true}$

$\text{vcg } (C_1; C_2) \ Q = (\text{vcg } C_1 \ (\text{wp } C_2 \ Q)) \wedge (\text{vcg } C_2 \ Q)$

$\text{vcg } (\text{if } B \text{ then } C_1 \text{ else } C_2 \text{ fi}) \ Q = (\text{vcg } C_1 \ Q) \wedge (\text{vcg } C_2 \ Q)$

# What About Precondition?

**Question:** Do we have  $\{\text{wp } C \ Q\} \ C \ \{Q\}$ ?

**Answer:** Not always - need to check **while**-loop side-conditions – verification conditions

**Question:** How to calculate verification conditions?

## Definition

$\text{vcg } (x := e) \ Q = \text{true}$

$\text{vcg } (C_1; C_2) \ Q = (\text{vcg } C_1 \ (\text{wp } C_2 \ Q)) \wedge (\text{vcg } C_2 \ Q)$

$\text{vcg } (\text{if } B \text{ then } C_1 \text{ else } C_2 \text{ fi}) \ Q = (\text{vcg } C_1 \ Q) \wedge (\text{vcg } C_2 \ Q)$

$\text{vcg } (\text{while } B \text{ inv } P \text{ do } C \text{ od}) \ Q =$   
 $((P \wedge B) \Rightarrow (\text{wp } C \ P)) \wedge (\text{vcg } C \ P) \wedge ((P \wedge (\neg B)) \Rightarrow Q)$



# Verification Condition Guarantees $\text{wp}$ Precondition

## Theorem

$$\text{vcg } C \ Q \Rightarrow \{\text{wp } C \ Q\} \ C \ \{Q\}$$

## Proof.

(Sketch)

- Induct on structure of  $C$
- For each case, wind back as we did in specific examples:
  - Assignment:  $\text{wp } C \ Q$  exactly what is needed for Assignment Axiom
  - Sequence: Follows from inductive hypotheses, all elim, and modus ponens
  - If\_Then\_Else: Need to use Precondition Strengthening with each branch of conditional;  $\text{wp}$  and inductive hypotheses give the needed side conditions
  - While: Need to use Postcondition Weakening, While Rule and Precondition Strengthening



# Verification Condition Guarantees wp Precondition

## Corollary

$$((P \Rightarrow wp\ C\ Q) \wedge (vcg\ C\ Q)) \Rightarrow \{P\}\ C\ \{Q\}$$

This amounts to a method for proving Hoare triple  $\{P\}\ C\ \{Q\}$ :

- 1 Annotate program with loop invariants
- 2 Calculate  $wp\ C\ Q$  and  $vcg\ C\ Q$  (automated)
- 3 Prove  $P \Rightarrow wp\ C\ Q$  and  $vcg\ C\ Q$

Basic outline of interaction with Boogie: Human does 1, Boogie does 2, Z3 / Simplify / Isabelle + human / ... does 3

For more information

- <http://research.microsoft.com/en-us/projects/boogie/>
- <http://research.microsoft.com/en-us/um/people/moskal/pdf/hol-boogie.pdf>
- <http://www.cl.cam.ac.uk/research/hvg/Isabelle/dist/library/HOL/HOL-Hoare/index.html>

# Model For Hoare Logic

- Seen proof system for Hoare Logic
- What about models?
- Informally, triple modeled by
  - pairs of assignments of program variables to values
  - where executing program starting with initial assignment results in a memory that gives the final assignment
- Calls for alternate definition of execution



# Natural Semantics

---

- Aka Structural Operational Semantics, aka “Big Step Semantics”
- Provide value for a program by rules and derivations, similar to type derivations
- Rule conclusions look like

$$(C, m) \Downarrow m'$$

or

$$(E, m) \Downarrow v$$



# Simple Imperative Programming Language

---

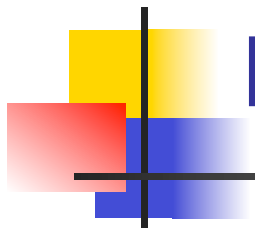
- $I \in \textit{Identifiers}$
- $N \in \textit{Numerals}$
- $B ::= \text{true} \mid \text{false} \mid B \ \& \ B \mid B \ \text{or} \ B \mid \text{not} \ B$   
 $\mid E < E \mid E = E$
- $E ::= N \mid I \mid E + E \mid E * E \mid E - E \mid - E$
- $C ::= \text{skip} \mid C; C \mid I ::= E$   
 $\mid \text{if } B \text{ then } C \text{ else } C \text{ fi} \mid \text{while } B \text{ do } C \text{ od}$



# Natural Semantics of Atomic Expressions

---

- Identifiers:  $(I, m) \Downarrow m(I)$
- Numerals are values:  $(N, m) \Downarrow N$
- Booleans:  $(\text{true}, m) \Downarrow \text{true}$   
 $(\text{false}, m) \Downarrow \text{false}$



## Booleans:

$$\frac{(B, m) \Downarrow \text{false}}{(B \ \& \ B', m) \Downarrow \text{false}}$$

$$\frac{(B, m) \Downarrow \text{true} \quad (B', m) \Downarrow b}{(B \ \& \ B', m) \Downarrow b}$$

$$\frac{(B, m) \Downarrow \text{true}}{(B \ \text{or} \ B', m) \Downarrow \text{true}}$$

$$\frac{(B, m) \Downarrow \text{false} \quad (B', m) \Downarrow b}{(B \ \text{or} \ B', m) \Downarrow b}$$

$$\frac{(B, m) \Downarrow \text{true}}{(\text{not } B, m) \Downarrow \text{false}}$$

$$\frac{(B, m) \Downarrow \text{false}}{(\text{not } B, m) \Downarrow \text{true}}$$



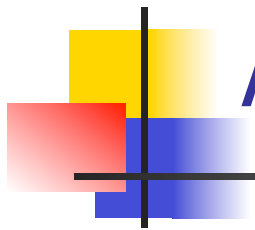
# Relations

---

$$\frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \sim V = b}{(E \sim E', m) \Downarrow b}$$

- By  $U \sim V = b$ , we mean does (the meaning of) the relation  $\sim$  hold on the meaning of  $U$  and  $V$
- May be specified by a mathematical expression/equation or rules matching  $U$  and  $V$





# Arithmetic Expressions

---

$$\frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \text{ op } V = N}{(E \text{ op } E', m) \Downarrow N}$$

where  $N$  is the specified value for  $U \text{ op } V$



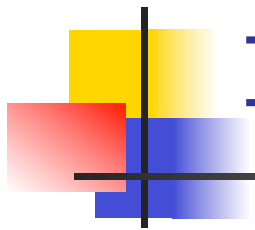
# Commands

---

Skip:  $(\text{skip}, m) \Downarrow m$

Assignment: 
$$\frac{(E, m) \Downarrow V}{(I ::= E, m) \Downarrow m[I \leftarrow V]}$$

Sequencing: 
$$\frac{(C, m) \Downarrow m' \quad (C', m') \Downarrow m''}{(C; C', m) \Downarrow m''}$$

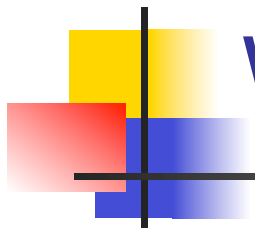


## If Then Else Command

---

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'}$$

$$\frac{(B, m) \Downarrow \text{false} \quad (C', m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'}$$

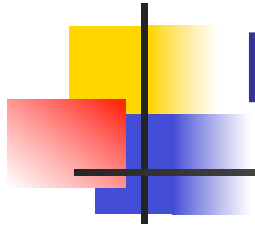


# While Command

---

$$\frac{(B, m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m}$$

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow m' \quad (\text{while } B \text{ do } C \text{ od}, m') \Downarrow m''}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m''}$$

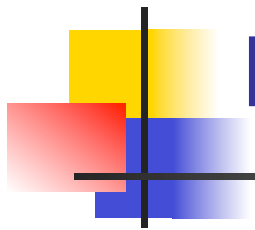


## Example: If Then Else Rule

---

---

(if  $x > 5$  then  $y := 2 + 3$  else  $y := 3 + 4$  fi,  
 $\{x \rightarrow 7\}) \Downarrow ?$



## Example: If Then Else Rule

---

---

$$(x > 5, \{x \rightarrow 7\}) \Downarrow ?$$

---

$$(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\ \{x \rightarrow 7\}) \Downarrow ?$$



## Example: Arith Relation

---

$? > ? = ?$

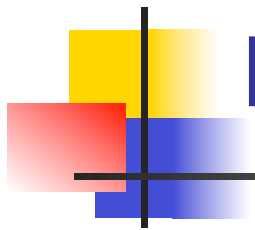
$(x, \{x \rightarrow 7\}) \Downarrow ? \quad (5, \{x \rightarrow 7\}) \Downarrow ?$

---

$(x > 5, \{x \rightarrow 7\}) \Downarrow ?$

---

$(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \rightarrow 7\}) \Downarrow ?$



## Example: Identifier(s)

---

$7 > 5 = \text{true}$

$(x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5$

$(x > 5, \{x \rightarrow 7\}) \Downarrow ?$

---

$(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \rightarrow 7\}) \Downarrow ?$





## Example: Arith Relation

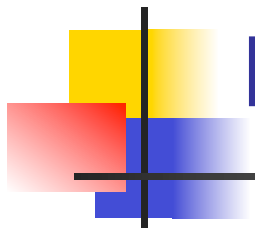
---

$$7 > 5 = \text{true}$$

$$\frac{(x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5}{(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}}$$

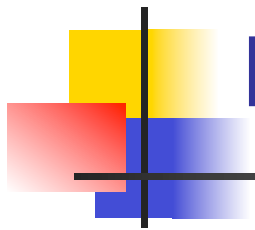
$$(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}$$

$$\frac{}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \rightarrow 7\}) \Downarrow ?}$$



## Example: If Then Else Rule

$$\begin{array}{c} 7 > 5 = \text{true} \\ \hline \frac{(x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5}{(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}} \quad \frac{}{(y := 2 + 3, \{x \rightarrow 7\}) \Downarrow ?} \\ \hline \text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\ \quad \{x \rightarrow 7\}) \Downarrow ? \end{array}$$

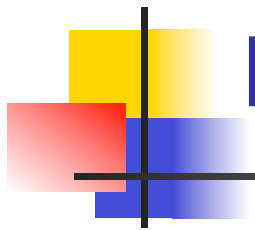


## Example: Assignment

$$\frac{\frac{7 > 5 = \text{true}}{(x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5}}{(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}} \quad \frac{(2+3, \{x \rightarrow 7\}) \Downarrow ?}{(y := 2 + 3, \{x \rightarrow 7\}) \Downarrow ?}$$

---

$$\frac{}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \rightarrow 7\}) \Downarrow ?}$$

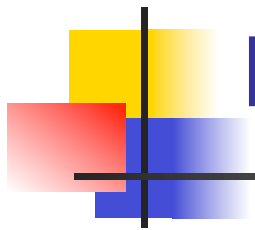


## Example: Arith Op

$$\begin{array}{c}
 \text{? + ? = ?} \\
 \hline
 (2, \{x \rightarrow 7\}) \Downarrow ? \quad (3, \{x \rightarrow 7\}) \Downarrow ? \\
 \hline
 (2+3, \{x \rightarrow 7\}) \Downarrow ? \\
 \hline
 (y := 2 + 3, \{x \rightarrow 7\}) \Downarrow ? \\
 \hline
 \cdot \\
 \hline
 \end{array}$$

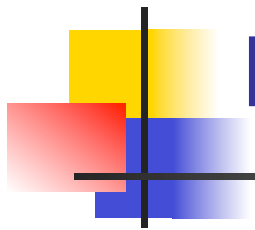
$$\begin{array}{c}
 7 > 5 = \text{true} \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} \\
 \hline
 \end{array}$$

$$\begin{array}{c}
 \text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow ?
 \end{array}$$



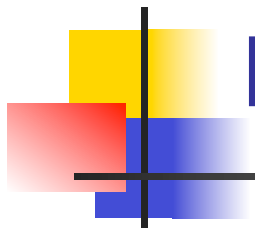
# Example: Numerals

$$\begin{array}{c}
 2 + 3 = 5 \\
 \hline
 (2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3 \\
 \hline
 (2+3, \{x \rightarrow 7\}) \Downarrow ? \\
 \hline
 (y := 2 + 3, \{x \rightarrow 7\}) \\
 \Downarrow ? \\
 \hline
 \begin{array}{c}
 7 > 5 = \text{true} \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} \\
 \hline
 \text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow ?
 \end{array}
 \end{array}$$



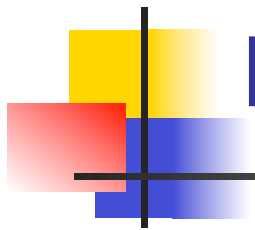
## Example: Arith Op

$$\begin{array}{c}
 2 + 3 = 5 \\
 \hline
 (2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3 \\
 \hline
 (2+3, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (y := 2 + 3, \{x \rightarrow 7\}) \Downarrow ? \\
 \hline
 \begin{array}{c}
 7 > 5 = \text{true} \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} \\
 \hline
 \text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow ?
 \end{array}
 \end{array}$$



## Example: Assignment

$$\begin{array}{c}
 2 + 3 = 5 \\
 \frac{(2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3}{(2+3, \{x \rightarrow 7\}) \Downarrow 5} \\
 \frac{7 > 5 = \text{true} \quad (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5}{(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}} \quad \frac{(2+3, \{x \rightarrow 7\}) \Downarrow 5}{(y := 2 + 3, \{x \rightarrow 7\}) \Downarrow \{x \rightarrow 7, y \rightarrow 5\}} \\
 \hline
 (\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow ?
 \end{array}$$



## Example: If Then Else Rule

$$\begin{array}{c}
 \begin{array}{c}
 7 > 5 = \text{true} \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}
 \end{array}
 \quad
 \begin{array}{c}
 2 + 3 = 5 \\
 \hline
 (2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3 \\
 \hline
 (2+3, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (y := 2 + 3, \{x \rightarrow 7\}) \\
 \Downarrow \{x \rightarrow 7, y \rightarrow 5\}
 \end{array} \\
 \hline
 (\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow \{x \rightarrow 7, y \rightarrow 5\}
 \end{array}$$



# Natural Semantics Models Hoare Logic (Soundness)

## Definition

Say a pair of states (aka assignments)  $(m_1, m_2)$  **satisfies**, or **models the Hoare triple**  $\{P\} \ C \ \{Q\}$  if  $m_1 \models P$  and  $m_2 \models Q$ . Write  $(m_1, m_2) \models \{P\} \ C \ \{Q\}$

## Theorem

Let  $\{P\} \ C \ \{Q\}$  be a valid Hoare triple (i.e., it's provable). Let  $m_1$  be a state (aka assignment) such that  $m_1 \models P$ . Let  $m_2$  be a state such that  $(C, m_1) \Downarrow m_2$ . Then  $(m_1, m_2) \models \{P\} \ C \ \{Q\}$

## Theorem

Let  $\{P\} C \{Q\}$  be a such that for all  $m_1$  and  $m_2$ , if (we can prove)  $m_1 \models P$  and  $(C, m_1) \Downarrow m_2$  then (we can prove)  $m_2 \models Q$ . Then  $\{P\} C \{Q\}$  is provable in Hoare logic.

# Simple Imperative Programming Language #2

$I \in \text{Identifiers}$

$N \in \text{Numerals}$

$E ::= N \mid I \mid E + E \mid E * E \mid E - E \mid I ::= E$

$B ::= \text{true} \mid \text{false} \mid B \& B \mid B \text{ or } B \mid \text{not } B$   
 $\mid E < E \mid E = E$

$C ::= \text{skip} \mid C; C \mid \{C\} \mid E$   
 $\mid \text{if } B \text{ then } C \text{ else } C \text{ fi}$   
 $\mid \text{while } B \text{ do } C$

# Changes for Expressions

- Need new type of *result* for expressions

$$(E, m) \Downarrow (v, m')$$

- Modify old rules for expressions:

Atomic Expressions:

$$(I, m) \Downarrow (m(I), m) \quad (N, m) \Downarrow (N, m)$$

Binary Operators:

$$\frac{(E, m) \Downarrow (U, m') \quad (E', m') \Downarrow (V, m'') \quad U \oplus V = N}{(E \oplus E', m) \Downarrow (N, m'')}$$

# New Rule for Expressions

$$\frac{(E, m) \Downarrow (V, m')}{(I ::= E, m) \Downarrow (V, m'[I \leftarrow V])}$$

# Changes for Commands

- Replace rule for Assignment by one for Expressions as Commands:

$$\frac{(E, m) \Downarrow (v, m')}{(E, m) \Downarrow m'}$$

- Unfortunately, can't stop there
  - Relations use Expressions; must be changed
  - Relations produce Booleans; all Booleans must be changed
  - `if_then_else` and `while` use Booleans; must be changed

- Must thread state through the relations:

$$\frac{(E, m) \Downarrow (U, m') \quad (E', m') \Downarrow (V, m'') \quad U \sim V = b}{(E \sim E', m) \Downarrow (b, m'')}$$

# Changes for Boolean Expressions

- Arithmetic Expressions occur in Boolean Expression; must change type of result for Booleans:

$$(B, m) \Downarrow (b, m')$$

- Modify old rules for Booleans to reflect new type:  
Atomic Booleans:

$$(\text{true}, m) \Downarrow (\text{true}, m)$$

$$(\text{false}, m) \Downarrow (\text{false}, m)$$



# Changes for Boolean Expressions

$$\frac{(B, m) \Downarrow (\text{false}, m') \quad (B, m) \Downarrow (\text{true}, m') \quad (B', m') \Downarrow (b, m'')}{(B \& B', m) \Downarrow (\text{false}, m') \quad (B \& B', m) \Downarrow (b, m'')}$$
$$\frac{(B, m) \Downarrow (\text{true}, m') \quad (B, m) \Downarrow (\text{false}, m') \quad (B', m') \Downarrow (b, m'')}{(B \text{ or } B', m) \Downarrow (\text{true}, m') \quad (B \text{ or } B', m) \Downarrow (b, m'')}$$
$$\frac{(B, m) \Downarrow (\text{true}, m')}{(\text{not } B, m) \Downarrow (\text{false}, m')}$$
$$\frac{(B, m) \Downarrow (\text{false}, m')}{(\text{not } B, m) \Downarrow (\text{true}, m')}$$

# Revised if\_then\_else Rule

$$\frac{(B, m) \Downarrow (\text{true}, m') \quad (C, m') \Downarrow m''}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m''}$$

$$\frac{(B, m) \Downarrow (\text{false}, m') \quad (C', m') \Downarrow m''}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m''}$$

# Revised while Rule

$$\frac{(B, m) \Downarrow (\text{false}, m')}{(\text{while } B \text{ do } C, m) \Downarrow m'}$$

$$\frac{(B, m) \Downarrow (\text{true}, m') \quad (C, m') \Downarrow m'' \quad (\text{while } B \text{ do } C, m'') \Downarrow m'''}{(\text{while } B \text{ do } C, m) \Downarrow m'''}$$

# Termination and Errors in SOS

- $(C,m)$ ,  $(E,m)$ ,  $(B,m)$  called **configurations**
- A configuration  $c$  **evaluates** to a result  $r$  if  $c \Downarrow r$ .
- If a configuration  $c$  evaluates to a result  $r$ , then  $c$  terminates without error
- Problem: Can not distinguish between untermination (e.g. a while loop that runs forever), versus an error (e.g. referencing an unassigned value)
- Can be (partially) remedied by adding **error** result
  - Roughly doubles number of rules

# Transition Semantics

- Aka “small step structured operational semantics”
- Defines a relation of “one step” of computation, instead of complete evaluation
  - Determines granularity of atomic computations
- Typically have two kinds of “result”: configurations and final values
- Written  $(C, m) \rightarrow (C', m')$  or  $(C, m) \rightarrow m'$

# Simple Imperative Programming Language #1 (SIMPL1)

$I \in \text{Identifiers}$

$N \in \text{Numerals}$

$E ::= N \mid I \mid E + E \mid E * E \mid E - E$

$B ::= \text{true} \mid \text{false} \mid B \& B \mid B \text{ or } B \mid \text{not } B$   
 $\mid E < E \mid E = E$

$C ::= \text{skip} \mid C; C \mid \{C\} \mid I ::= E$   
 $\mid \text{if } B \text{ then } C \text{ else } C \text{ fi}$   
 $\mid \text{while } B \text{ do } C$

# Transitions for Atomic Expressions

Identifiers:  $(l, m) \longrightarrow m(l)$

Numerals are values:  $(N, m) \longrightarrow N$

Booleans:  $(\text{true}, m) \longrightarrow \text{true}$

$(\text{false}, m) \longrightarrow \text{false}$

# Booleans:

- Values = {true, false}
- Operators: (short-circuit)

$$\begin{array}{ll} (\text{false} \& B, m) \longrightarrow \text{false} & (B, m) \longrightarrow (B'', m) \\ (\text{true} \& B, m) \longrightarrow (B, m) & \hline (B \& B', m) \longrightarrow (B'' \& B', m) \end{array}$$

$$\begin{array}{ll} (\text{true or } B, m) \longrightarrow \text{true} & (B, m) \longrightarrow (B'', m) \\ (\text{false or } B, m) \longrightarrow (B, m) & \hline (B \text{ or } B', m) \longrightarrow (B'' \text{ or } B', m) \end{array}$$

$$\begin{array}{ll} (\text{not true}, m) \longrightarrow \text{false} & (B, m) \longrightarrow (B', m) \\ (\text{not false}, m) \longrightarrow \text{true} & \hline (\text{not } B, m) \longrightarrow (\text{not } B', m) \end{array}$$



- Let  $U, V$  be arithmetic values

$$\frac{(E, m) \longrightarrow (E'', m)}{(E \sim E', m) \longrightarrow (E'' \sim E', m)}$$

$$\frac{(E, m) \longrightarrow (E', m)}{(V \sim E, m) \longrightarrow (V \sim E', m)}$$

$$(U \sim V, m) \longrightarrow b$$

where  $U \sim V = b$

# Arithmetic Expressions

$$\frac{(E, m) \longrightarrow (E'', m)}{(E \oplus E', m) \longrightarrow (E'' \oplus E', m)}$$

$$\frac{(E, m) \longrightarrow (E', m)}{(V \oplus E, m) \longrightarrow (V \oplus E', m)}$$

$$(U \oplus V, m) \longrightarrow N$$

where  $N$  is the specified value for  $U \oplus V$

# Commands - in English

- **skip** means done evaluating
- When evaluating an assignment, evaluate expression first
- If the expression being assigned is a value, update the memory with the new value for the identifier
- When evaluating a sequence, work on the first command in the sequence first
- If the first command evaluates to a new memory (ie completes), evaluate remainder with new memory

# Commands

Skip:  $(\text{skip}, m) \longrightarrow m$

Assignment: 
$$\frac{(E, m) \longrightarrow (E', m)}{(I ::= E, m) \longrightarrow (I ::= E', m)}$$

$(I ::= V, m) \longrightarrow m[I \leftarrow V]$

Sequencing:

$$\frac{(C, m) \longrightarrow (C'', m')}{(C; C', m) \longrightarrow (C''; C', m')} \quad \frac{(C, m) \longrightarrow m'}{(C; C', m) \longrightarrow (C', m')}$$

# Block Command

- Choice of level of granularity:
  - Choice 1: Open a block is a unit of work

$$(\{C\}, m) \longrightarrow (C, m)$$

- Choice 2: Blocks are syntactic sugar

$$\frac{(C, m) \longrightarrow (C', m')}{(\{C\}, m) \longrightarrow (C', m')} \quad \frac{(C, m) \longrightarrow m'}{(\{C\}, m) \longrightarrow m'}$$

# If Then Else Command - in English

- If the boolean guard in an `if_then_else` is true, then evaluate the first branch
- If it is false, evaluate the second branch
- If the boolean guard is not a value, then start by evaluating it first.

# If Then Else Command

$$(\text{if true then } C \text{ else } C' \text{ fi}, m) \longrightarrow (C, m)$$

$$(\text{if false then } C \text{ else } C' \text{ fi}, m) \longrightarrow (C', m)$$

$$\frac{(B, m) \longrightarrow (B', m)}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \longrightarrow (\text{if } B' \text{ then } C \text{ else } C' \text{ fi}, m)}$$

# While Command

$$\begin{array}{c} (\text{while } B \text{ do } C, m) \\ \longrightarrow \\ (\text{if } B \text{ then } C; \text{while } B \text{ do } C \text{ else skip fi}, m) \end{array}$$

- In English: Expand a **while** into a test of the boolean guard, with the true case being to do the body and then try the while loop again, and the false case being to stop.



# Example

$(y := i; \text{ while } i > 0 \text{ do } \{i := i - 1; y := y * i\}, \langle i \mapsto 3 \rangle)$

$\longrightarrow \underline{\quad ? \quad}$

# Alternate Semantics for SIMPL1

- Can mix Natural Semantics with Transition Semantics to get larger atomic computations
- Use  $(E, m) \Downarrow v$  and  $(B, m) \Downarrow b$  for arithmetics and boolean expressions
- Revise rules for commands

# Revised Rules for SIMPL1

Skip:  $(\text{skip}, m) \longrightarrow m$

Assignment: 
$$\frac{(E, m) \Downarrow v}{(I ::= E, m)} \longrightarrow m[I \leftarrow V]$$

Sequencing:

$$\frac{(C, m) \longrightarrow (C'', m')}{(C; C', m) \longrightarrow (C''; C', m')} \qquad \frac{(C, m) \longrightarrow m'}{(C; C', m) \longrightarrow (C', m')}$$

Blocks:

$$\frac{(C, m) \longrightarrow (C', m')}{(\{C\}, m) \longrightarrow (C', m')} \qquad \frac{(C, m) \longrightarrow m'}{(\{C\}, m) \longrightarrow m'}$$

# If Then Else Command

$$\frac{(B, m) \Downarrow \text{true}}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \longrightarrow (C, m)}$$

$$\frac{(B, m) \Downarrow \text{false}}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \longrightarrow (C', m)}$$

# Transition Semantics for SIMPL2?

- What are the choices and consequences for giving a transition semantics for the Simple Concurrent Imperative Programming Language #2, SIMP2?

# Simple Concurrent Imperative Programming Language

$I \in \text{Identifiers}$

$N \in \text{Numerals}$

$E ::= N \mid I \mid E + E \mid E * E \mid E - E$

$B ::= \text{true} \mid \text{false} \mid B \& B \mid B \text{ or } B \mid \text{not } B$   
 $\mid E < E \mid E = E$

$C ::= \text{skip} \mid C; C \mid \{C\} \mid I ::= E \mid C \parallel C'$   
 $\mid \text{if } B \text{ then } C \text{ else } C \text{ fi}$   
 $\mid \text{while } B \text{ do } C$