CS228 : Logic for Computer Science

Module 4: Hoare Logic and Program Verification

Lecture 20: Proof Rules for Hoare Logic

Instructor: S. Akshay

IIT Bombay, India

Logistics

Recall: Quiz 2

- **Friday, Oct 23rd, 6pm**
- Syllabus:
  - Module 3 (Predicate Logic/FOL): all of Lectures 12–18
  - Module 4 (Hoare logic): from Lecture 19 up to and including Thursday's lecture (Oct 15th)

Logistics

Recall: Quiz 2

- **Friday, Oct 23rd, 6pm**
- Syllabus:
  - Module 3 (Predicate Logic/FOL): all of Lectures 12–18
  - Module 4 (Hoare logic): from Lecture 19 up to and including Thursday's lecture (Oct 15th)

Optional SMT solver demo

- **Wednesday, Oct 14th, 3:30pm**
- Venue: CC 103

Topics Covered till date

Module 1: Propositional Logic

1. Syntax
2. Semantics
3. Formal proof systems
4. Normal Forms

Module 2: The Problem of Satisfiability

1. Propositional Resolution
2. DPLL
3. CDCL
4. SAT tool demo

Topics Covered till date

Module 1: Propositional Logic

1. Syntax
2. Semantics
3. Formal proof systems
4. Normal Forms

Module 2: The Problem of Satisfiability

1. Propositional Resolution
2. DPLL
3. CDCL
4. SAT tool demo

Module 3: Predicate Logic

1. Syntax of predicate logic
2. Semantics of predicate logic
3. Natural deduction for FOL
4. Structures, power and limits of FOL
5. Normal forms, DPLL(T) and SMT

Topics Covered till date

Module 1: Propositional Logic

1. Syntax
2. Semantics
3. Formal proof systems
4. Normal Forms

Module 2: The Problem of Satisfiability

1. Propositional Resolution
2. DPLL
3. CDCL
4. SAT tool demo

Module 3: Predicate Logic

1. Syntax of predicate logic
2. Semantics of predicate logic
3. Natural deduction for FOL
4. Structures, power and limits of FOL
5. Normal forms, DPLL(T) and SMT

Module 4: Hoare Logic and Program Verification

1. Introduction to Hoare logic
2. Today: Proof rules for Hoare logic

Recap: Hoare triples and correctness

A Hoare triple $\{\varphi\}\ P\ \{\psi\}$: precondition, program, postcondition

- Partial correctness ($\models_{par}$): *if* P terminates when started in a state satisfying φ, then ψ holds in the final state.
- Total correctness ($\models_{tot}$): started in any state satisfying φ, P *does terminate*, and ψ holds in the final state.

Recap: Hoare triples and correctness

A Hoare triple $\{\varphi\}\ P\ \{\psi\}$: precondition, program, postcondition

- Partial correctness ($\models_{par}$): *if* P terminates when started in a state satisfying φ, then ψ holds in the final state.
- Total correctness ($\models_{tot}$): started in any state satisfying φ, P *does terminate*, and ψ holds in the final state.

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Qn: Recall **Fac**$_1$. Which of these hold?

	partial	total
$\{\top\}$ **Fac**$_1$ $\{y = x!\}$		
$\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$		

Recap: Hoare triples and correctness

A Hoare triple $\{\varphi\}\ P\ \{\psi\}$: precondition, program, postcondition

- Partial correctness ($\models_{par}$): *if* P terminates when started in a state satisfying φ, then ψ holds in the final state.
- Total correctness ($\models_{tot}$): started in any state satisfying φ, P *does terminate*, and ψ holds in the final state.

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Qn: Recall **Fac**$_1$. Which of these hold?

	partial	total
$\{\top\}$ **Fac**$_1$ $\{y = x!\}$	✓	
$\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$	✓	

Recap: Hoare triples and correctness

A Hoare triple $\{\varphi\}\ P\ \{\psi\}$: precondition, program, postcondition

- Partial correctness ($\models_{par}$): *if* P terminates when started in a state satisfying φ, then ψ holds in the final state.
- Total correctness ($\models_{tot}$): started in any state satisfying φ, P *does terminate*, and ψ holds in the final state.

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Qn: Recall **Fac**$_1$. Which of these hold?

	partial	total
$\{\top\}$ **Fac**$_1$ $\{y = x!\}$	✓	×
$\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$	✓	✓

- The two statements from Lecture 19: statement 1 (terminates with $y = x!$ if $x \geq 0$) is the *total* entry for $x \geq 0$; statement 2 (whenever it terminates, $y = x!$) is the *partial* entry for $\top$.

Recap: Hoare triples and correctness

A Hoare triple $\{\varphi\}\ P\ \{\psi\}$: precondition, program, postcondition

- Partial correctness ($\models_{par}$): *if* P terminates when started in a state satisfying φ, then ψ holds in the final state.
- Total correctness ($\models_{tot}$): started in any state satisfying φ, P *does terminate*, and ψ holds in the final state.

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Qn: Recall **Fac**$_1$. Which of these hold?

	partial	total
$\{\top\}$ **Fac**$_1$ $\{y = x!\}$	✓	×
$\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$	✓	✓

- The two statements from Lecture 19: statement 1 (terminates with $y = x!$ if $x \geq 0$) is the *total* entry for $x \geq 0$; statement 2 (whenever it terminates, $y = x!$) is the *partial* entry for $\top$.

total correctness = partial correctness + termination

Recap: Partial correctness and non-termination

Exercise 20.1 (a): $\mathbf{Fac}_1$ never stops when $x = -2$.

- Why does $\{\top\}\ \mathbf{Fac}_1\ \{y = x!\}$ still hold under partial correctness?

Recap: Partial correctness and non-termination

Exercise 20.1 (a): $\mathbf{Fac}_1$ never stops when $x = -2$.

- Why does $\{\top\}\ \mathbf{Fac}_1\ \{y = x!\}$ still hold under partial correctness?
- Vacuous! No terminating run starts with $x < 0$, so no run can violate $y = x!$.

Recap: Partial correctness and non-termination

Exercise 20.1 (a): $\mathbf{Fac}_1$ never stops when $x = -2$.

- Why does $\{\top\}\ \mathbf{Fac}_1\ \{y = x!\}$ still hold under partial correctness?
- Vacuous! No terminating run starts with $x < 0$, so no run can violate $y = x!$.

Exercise 20.1 (b): Let P be **while** $\top$ $\{x := 0\}$.

- Does $\{\top\}\ P\ \{\bot\}$ hold under partial correctness? Under total correctness?

Recap: Partial correctness and non-termination

Exercise 20.1 (a): $\mathbf{Fac}_1$ never stops when $x = -2$.

- Why does $\{\top\}\ \mathbf{Fac}_1\ \{y = x!\}$ still hold under partial correctness?
- Vacuous! No terminating run starts with $x < 0$, so no run can violate $y = x!$.

Exercise 20.1 (b): Let P be **while** $\top$ $\{x := 0\}$.

- Does $\{\top\}\ P\ \{\bot\}$ hold under partial correctness? Under total correctness?
  - Note: the $\top$ in the guard is part of the *program* (a Boolean expression, like **while**); the plain $\top$ and $\bot$ in the braces are *assertions*, i.e., formulas of the logic (you can think of them as abbreviations of, e.g., $0 \leq 0$ and $\neg(0 \leq 0)$).

Recap: Partial correctness and non-termination

Exercise 20.1 (a): $\mathbf{Fac}_1$ never stops when $x = -2$.

- Why does $\{\top\}\ \mathbf{Fac}_1\ \{y = x!\}$ still hold under partial correctness?
- Vacuous! No terminating run starts with $x < 0$, so no run can violate $y = x!$.

Exercise 20.1 (b): Let P be **while** $\top$ $\{x := 0\}$.

- Does $\{\top\}\ P\ \{\bot\}$ hold under partial correctness? Under total correctness?
  - Note: the $\top$ in the guard is part of the *program* (a Boolean expression, like **while**); the plain $\top$ and $\bot$ in the braces are *assertions*, i.e., formulas of the logic (you can think of them as abbreviations of, e.g., $0 \leq 0$ and $\neg(0 \leq 0)$).
- Partial: yes (it never terminates, so again vacuous). Total: no.

Recap: Partial correctness and non-termination

Exercise 20.1 (a): $\mathbf{Fac}_1$ never stops when $x = -2$.

- Why does $\{\top\}\ \mathbf{Fac}_1\ \{y = x!\}$ still hold under partial correctness?
- Vacuous! No terminating run starts with $x < 0$, so no run can violate $y = x!$.

Exercise 20.1 (b): Let P be **while** $\top$ $\{x := 0\}$.

- Does $\{\top\}\ P\ \{\bot\}$ hold under partial correctness? Under total correctness?
  - Note: the $\top$ in the guard is part of the *program* (a Boolean expression, like **while**); the plain $\top$ and $\bot$ in the braces are *assertions*, i.e., formulas of the logic (you can think of them as abbreviations of, e.g., $0 \leq 0$ and $\neg(0 \leq 0)$).
- Partial: yes (it never terminates, so again vacuous). Total: no.

A partially correct program may do nothing useful: partial correctness says nothing about termination!

Program variables and logical variables

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Program variables and logical variables

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Fac$_2$ *inp* x *out* y	$y := 1;$ **while** $(x \neq 0)\{$ $\quad y := y * x;$ $\quad x := x - 1;$ $\}$

Program variables and logical variables

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Fac$_2$ *inp* x *out* y	$y := 1;$ **while** $(x \neq 0)\{$ $\quad y := y * x;$ $\quad x := x - 1;$ $\}$

- **Fac**$_2$ is another program that computes the factorial.
- $\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$ satisfies partial and total correctness.
- Qn: What about $\{x \geq 0\}$ **Fac**$_2$ $\{y = x!\}$?

Program variables and logical variables

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Fac$_2$ *inp* x *out* y	$y := 1;$ **while** $(x \neq 0)\{$ $\quad y := y * x;$ $\quad x := x - 1;$ $\}$

- **Fac**$_2$ is another program that computes the factorial.
- $\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$ satisfies partial and total correctness.
- Qn: What about $\{x \geq 0\}$ **Fac**$_2$ $\{y = x!\}$?
- How do we correct this?

Program variables and logical variables

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Fac$_2$ *inp* x *out* y	$y := 1;$ **while** $(x \neq 0)\{$ $\quad y := y * x;$ $\quad x := x - 1;$ $\}$

- **Fac**$_2$ is another program that computes the factorial.
- $\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$ satisfies partial and total correctness.
- Qn: What about $\{x \geq 0\}$ **Fac**$_2$ $\{y = x!\}$?
- How do we correct this? What is the correct specification? Need to remember initial value!

Program variables and logical variables

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Fac$_2$ *inp* x *out* y	$y := 1;$ **while** $(x \neq 0)\{$ $\quad y := y * x;$ $\quad x := x - 1;$ $\}$

- **Fac**$_2$ is another program that computes the factorial.
- $\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$ satisfies partial and total correctness.
- Qn: What about $\{x \geq 0\}$ **Fac**$_2$ $\{y = x!\}$?
- How do we correct this? What is the correct specification? Need to remember initial value!
- We obtain $\{x = x_0 \wedge x \geq 0\}$ **Fac**$_2$ $\{y = x_0!\}$

Program variables and logical variables

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Fac$_2$ *inp* x *out* y	$y := 1;$ **while** $(x \neq 0)\{$ $\quad y := y * x;$ $\quad x := x - 1;$ $\}$

- **Fac**$_2$ is another program that computes the factorial.
- $\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$ satisfies partial and total correctness.
- Qn: What about $\{x \geq 0\}$ **Fac**$_2$ $\{y = x!\}$?
- How do we correct this? What is the correct specification? Need to remember initial value!
- We obtain $\{x = x_0 \wedge x \geq 0\}$ **Fac**$_2$ $\{y = x_0!\}$ x_0 is called a logical variable.

Program variables and logical variables

Fac$_1$ *inp* x *out* y	$y := 1;$ $z := 0;$ **while** $(z \neq x)\{$ $\quad z := z + 1;$ $\quad y := y * z;$ $\}$

Fac$_2$ *inp* x *out* y	$y := 1;$ **while** $(x \neq 0)\{$ $\quad y := y * x;$ $\quad x := x - 1;$ $\}$

- **Fac**$_2$ is another program that computes the factorial.
- $\{x \geq 0\}$ **Fac**$_1$ $\{y = x!\}$ satisfies partial and total correctness.
- Qn: What about $\{x \geq 0\}$ **Fac**$_2$ $\{y = x!\}$?
- How do we correct this? What is the correct specification? Need to remember initial value!
- We obtain $\{x = x_0 \wedge x \geq 0\}$ **Fac**$_2$ $\{y = x_0!\}$ x_0 is called a logical variable.

For a Hoare triple $\{\varphi\}P\{\psi\}$, its logical variables are those that are free in φ or ψ and do not occur in P.

Topic 20.1

Proving Programs: Proof Rules

A proof calculus for program correctness

- For propositional logic, we developed the natural deduction proof system to show validity.
- We will now come up with proof rules for Hoare logic!

A proof calculus for program correctness

- For propositional logic, we developed the natural deduction proof system to show validity.
- We will now come up with proof rules for Hoare logic!
- Note: Proof rules for partial and total correctness will differ slightly.

A proof calculus for program correctness

Rule1: Sequential Composition

$$\frac{\{\varphi\}\ C_1\ \{\eta\} \qquad \{\eta\}\ C_2\ \{\psi\}}{\{\varphi\}\ C_1; C_2\ \{\psi\}}\ \text{Seq. Composition}$$

- Easier to read the rules bottom-up!
- To show that ψ holds on running $C_1; C_2$ starting from φ, we have to find η and show that η holds after running C_1 from φ and from η running C_2 results in ψ holding.
- η is called a midcondition.

A proof calculus for program correctness

Rule1: Sequential Composition

$$\frac{\{\varphi\}\ C_1\ \{\eta\} \qquad \{\eta\}\ C_2\ \{\psi\}}{\{\varphi\}\ C_1; C_2\ \{\psi\}}\ \text{Seq. Composition}$$

- Easier to read the rules bottom-up!
- To show that ψ holds on running $C_1; C_2$ starting from φ, we have to find η and show that η holds after running C_1 from φ and from η running C_2 results in ψ holding.
- η is called a midcondition.

Example:

$$\frac{\{y+z>4\}\ y := y+z-1\ \{y>3\} \qquad \{y>3\}\ x := y+4\ \{x>7\}}{\{y+z>4\}\ y := y+z-1; x := y+4\ \{x>7\}}\ \text{Seq. Composition}$$

Hoare Logic: Proof rule for Assignment

Rule 2: Assignment

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

Hoare Logic: Proof rule for Assignment

Rule 2: Assignment

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

- This rule is an axiom.
- $\psi[E/x]$ refers to replacing every *free* occurrence of x in ψ by E.
- Example: $\{x + 1 + 5 = y\}\ x := x + 1\ \{x + 5 = y\}$

Hoare Logic: Proof rule for Assignment

Rule 2: Assignment

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

- This rule is an axiom.
- $\psi[E/x]$ refers to replacing every *free* occurrence of x in ψ by E.
- Example: $\{x + 1 + 5 = y\}\ x := x + 1\ \{x + 5 = y\}$
- Why are we doing this "backward"? Would forward work?
- No! $\{\varphi\}\ x := E\ \{\varphi[E/x]\}$ does not hold in general. **Exercise 20.2**: Show this by giving a counter-example.

Hoare Logic: Proof rule for Assignment

Rule 2: Assignment

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

- This rule is an axiom.
- $\psi[E/x]$ refers to replacing every *free* occurrence of x in ψ by E.
- Example: $\{x + 1 + 5 = y\}\ x := x + 1\ \{x + 5 = y\}$
- Why are we doing this "backward"? Would forward work?
- No! $\{\varphi\}\ x := E\ \{\varphi[E/x]\}$ does not hold in general. **Exercise 20.2**: Show this by giving a counter-example. Soln: let φ be $x = 6$ and E be 5.

Hoare Logic: Proof rule for Assignment

Rule 2: Assignment

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

- This rule is an axiom.
- $\psi[E/x]$ refers to replacing every *free* occurrence of x in ψ by E.
- Example: $\{x + 1 + 5 = y\}\ x := x + 1\ \{x + 5 = y\}$
- Why are we doing this "backward"? Would forward work?
- No! $\{\varphi\}\ x := E\ \{\varphi[E/x]\}$ does not hold in general. **Exercise 20.2**: Show this by giving a counter-example. Soln: let φ be $x = 6$ and E be 5. Then $\varphi[E/x]$ gives $5 = 6$, and $\{x = 6\}\ x := 5\ \{5 = 6\}$ fails.

Hoare Logic: Proof rule for Assignment

Rule 2: Assignment

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

- This rule is an axiom.
- $\psi[E/x]$ refers to replacing every *free* occurrence of x in ψ by E.
- Example: $\{x + 1 + 5 = y\}\ x := x + 1\ \{x + 5 = y\}$
- Why are we doing this "backward"? Would forward work?
- No! $\{\varphi\}\ x := E\ \{\varphi[E/x]\}$ does not hold in general. **Exercise 20.2**: Show this by giving a counter-example.

Exercise 20.3: Let P be $x := 2$. Are these true? (i) $\{2 = 2\}\ P\ \{x = 2\}$, (ii) $\{x = 2\}\ P\ \{2 = 2\}$, (iii) $\{x = 3\}\ P\ \{x = 3\}$, (iv) $\{2 = 4\}\ P\ \{x = 4\}$.

Hoare Logic: Proof rule for Assignment

Rule 2: Assignment

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

- This rule is an axiom.
- $\psi[E/x]$ refers to replacing every *free* occurrence of x in ψ by E.
- Example: $\{x + 1 + 5 = y\}\ x := x + 1\ \{x + 5 = y\}$
- Why are we doing this "backward"? Would forward work?
- No! $\{\varphi\}\ x := E\ \{\varphi[E/x]\}$ does not hold in general. **Exercise 20.2**: Show this by giving a counter-example.

Exercise 20.3: Let P be $x := 2$. Are these true? (i) $\{2 = 2\}\ P\ \{x = 2\}$, (ii) $\{x = 2\}\ P\ \{2 = 2\}$, (iii) $\{x = 3\}\ P\ \{x = 3\}$, (iv) $\{2 = 4\}\ P\ \{x = 4\}$. Sol: Only (i), (ii), (iv) are true.

Hoare Logic: Proof rule for If-then-Else

Rule 3: If-then-Else

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

Hoare Logic: Proof rule for If-then-Else

Rule 3: If-then-Else

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

- In the *then*-branch we know B held; in the *else*-branch we know B failed. We use this information in the sub-proofs!

Hoare Logic: Proof rule for If-then-Else

Rule 3: If-then-Else

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

- In the *then*-branch we know B held; in the *else*-branch we know B failed. We use this information in the sub-proofs!

Example

$$\frac{\{x > 4 \wedge z > 0\}\ x := x + z\ \{x > 3\} \qquad \{x > 4 \wedge \neg(z > 0)\}\ x := x - 1\ \{x > 3\}}{\{x > 4\}\ \textbf{if}\ (z > 0)\ \{x := x + z\}\ \textbf{else}\ \{x := x - 1\}\ \{x > 3\}}\ \text{If-then-Else}$$

Hoare Logic: Proof rule for If-then-Else

Rule 3: If-then-Else

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

- In the *then*-branch we know B held; in the *else*-branch we know B failed. We use this information in the sub-proofs!

Example

$$\frac{\{x > 4 \wedge z > 0\}\ x := x + z\ \{x > 3\} \qquad \{x > 4 \wedge \neg(z > 0)\}\ x := x - 1\ \{x > 3\}}{\{x > 4\}\ \textbf{if}\ (z > 0)\ \{x := x + z\}\ \textbf{else}\ \{x := x - 1\}\ \{x > 3\}}\ \text{If-then-Else}$$

(Each premise needs to be proved (using other rules).)

Hoare Logic: Strengthening Preconditions, Weakening Postconditions

Rule 4: Strengthen the precondition, weaken the postcondition

$$\frac{\vdash_{AR} \varphi \to \varphi' \qquad \{\varphi'\}\ C\ \{\psi'\} \qquad \vdash_{AR} \psi' \to \psi}{\{\varphi\}\ C\ \{\psi\}}\ \text{Implied}$$

Hoare Logic: Strengthening Preconditions, Weakening Postconditions

Rule 4: Strengthen the precondition, weaken the postcondition

$$\frac{\vdash_{AR} \varphi \to \varphi' \qquad \{\varphi'\}\ C\ \{\psi'\} \qquad \vdash_{AR} \psi' \to \psi}{\{\varphi\}\ C\ \{\psi\}}\ \text{Implied}$$

- The rule links Hoare logic to the base logic: $\vdash_{AR} \varphi \to \varphi'$ means the implication is provable in natural deduction for first order logic, taking facts about arithmetic (e.g., $\forall x\,(x = x + 0)$) as premises.

Hoare Logic: Strengthening Preconditions, Weakening Postconditions

Rule 4: Strengthen the precondition, weaken the postcondition

$$\frac{\vdash_{AR} \varphi \to \varphi' \qquad \{\varphi'\}\ C\ \{\psi'\} \qquad \vdash_{AR} \psi' \to \psi}{\{\varphi\}\ C\ \{\psi\}}\ \text{Implied}$$

- The rule links Hoare logic to the base logic: $\vdash_{AR} \varphi \to \varphi'$ means the implication is provable in natural deduction for first order logic, taking facts about arithmetic (e.g., $\forall x\,(x = x + 0)$) as premises.
- Strengthen the precondition = assume more than needed; weaken the postcondition = promise less.

Hoare Logic: Strengthening Preconditions, Weakening Postconditions

Rule 4: Strengthen the precondition, weaken the postcondition

$$\frac{\vdash_{AR} \varphi \to \varphi' \qquad \{\varphi'\}\ C\ \{\psi'\} \qquad \vdash_{AR} \psi' \to \psi}{\{\varphi\}\ C\ \{\psi\}}\ \text{Implied}$$

- The rule links Hoare logic to the base logic: $\vdash_{AR} \varphi \to \varphi'$ means the implication is provable in natural deduction for first order logic, taking facts about arithmetic (e.g., $\forall x\,(x = x + 0)$) as premises.
- Strengthen the precondition = assume more than needed; weaken the postcondition = promise less.
- Note that this works regardless of C!

Hoare Logic: Strengthening Preconditions, Weakening Postconditions

Rule 4: Strengthen the precondition, weaken the postcondition

$$\frac{\vdash_{AR} \varphi \to \varphi' \qquad \{\varphi'\}\ C\ \{\psi'\} \qquad \vdash_{AR} \psi' \to \psi}{\{\varphi\}\ C\ \{\psi\}}\ \text{Implied}$$

- The rule links Hoare logic to the base logic: $\vdash_{AR} \varphi \to \varphi'$ means the implication is provable in natural deduction for first order logic, taking facts about arithmetic (e.g., $\forall x\,(x = x + 0)$) as premises.
- Strengthen the precondition = assume more than needed; weaken the postcondition = promise less.
- Note that this works regardless of C!

Example

$$\frac{\vdash_{AR} x > 9 \to x > 7 \qquad \{x > 7\}\ x := x - 4\ \{x > 3\} \qquad \vdash_{AR} x > 3 \to x > 2}{\{x > 9\}\ x := x - 4\ \{x > 2\}}\ \text{Implied}$$

Hoare Logic: Proof rule for partial correctness of While Loop

Rule 5: partial-while

$$\frac{\{\eta \wedge B\}\ C\ \{\eta\}}{\{\eta\}\ \textbf{while}\ B\ \{C\}\ \{\eta \wedge \neg B\}}\ \text{Partial-while}$$

Hoare Logic: Proof rule for partial correctness of While Loop

Rule 5: partial-while

$$\frac{\{\eta \wedge B\}\ C\ \{\eta\}}{\{\eta\}\ \textbf{while}\ B\ \{C\}\ \{\eta \wedge \neg B\}}\ \text{Partial-while}$$

- η is a loop-invariant: it holds before the loop and is preserved by every iteration (that is what the premise says).

Hoare Logic: Proof rule for partial correctness of While Loop

Rule 5: partial-while

$$\frac{\{\eta \wedge B\}\ C\ \{\eta\}}{\{\eta\}\ \textbf{while}\ B\ \{C\}\ \{\eta \wedge \neg B\}}\ \text{Partial-while}$$

- η is a loop-invariant: it holds before the loop and is preserved by every iteration (that is what the premise says).
- Partial correctness!
  - If loop does not terminate, Hoare triple is vacuously satisfied.
  - If it does terminate, then $\eta \wedge \neg B$ must be satisfied at termination.

Hoare Logic: Proof rule for partial correctness of While Loop

Rule 5: partial-while

$$\frac{\{\eta \wedge B\}\ C\ \{\eta\}}{\{\eta\}\ \textbf{while}\ B\ \{C\}\ \{\eta \wedge \neg B\}}\ \text{Partial-while}$$

- η is a loop-invariant: it holds before the loop and is preserved by every iteration (that is what the premise says).
- Partial correctness!
  - If loop does not terminate, Hoare triple is vacuously satisfied.
  - If it does terminate, then $\eta \wedge \neg B$ must be satisfied at termination.
- Example

$$\frac{\{y = x + z \wedge (z \neq 0)\}\ x := x + 1; z := z - 1\ \{y = x + z\}}{\{y = x + z\}\ \textbf{while}\ (z \neq 0)\ \{x := x + 1; z := z - 1\}\ \{y = x + z \wedge (z = 0)\}}$$

Topic 20.2

Using the Proof Rules: Proof Outlines and Loop Invariants

A proof calculus for program correctness

$$\frac{\{\varphi\}\ C_1\ \{\eta\} \qquad \{\eta\}\ C_2\ \{\psi\}}{\{\varphi\}\ C_1; C_2\ \{\psi\}}\ \text{Seq. Comp.}$$

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

A proof calculus for program correctness

$$\frac{\{\varphi\}\ C_1\ \{\eta\} \qquad \{\eta\}\ C_2\ \{\psi\}}{\{\varphi\}\ C_1; C_2\ \{\psi\}}\ \text{Seq. Comp.}$$

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

$$\frac{\vdash_{AR} \varphi \rightarrow \varphi' \qquad \{\varphi'\}\ C\ \{\psi'\} \qquad \vdash_{AR} \psi' \rightarrow \psi}{\{\varphi\}\ C\ \{\psi\}}\ \text{Implied}$$

A proof calculus for program correctness

$$\frac{\{\varphi\}\ C_1\ \{\eta\} \qquad \{\eta\}\ C_2\ \{\psi\}}{\{\varphi\}\ C_1; C_2\ \{\psi\}}\ \text{Seq. Comp.}$$

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

$$\frac{\vdash_{AR} \varphi \to \varphi' \qquad \{\varphi'\}\ C\ \{\psi'\} \qquad \vdash_{AR} \psi' \to \psi}{\{\varphi\}\ C\ \{\psi\}}\ \text{Implied}$$

$$\frac{\{\eta \wedge B\}\ C\ \{\eta\}}{\{\eta\}\ \textbf{while}\ B\ \{C\}\ \{\eta \wedge \neg B\}}\ \text{Partial-while}$$

A proof calculus for program correctness

$$\frac{\{\varphi\}\ C_1\ \{\eta\} \qquad \{\eta\}\ C_2\ \{\psi\}}{\{\varphi\}\ C_1; C_2\ \{\psi\}}\ \text{Seq. Comp.}$$

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

$$\frac{\vdash_{AR} \varphi \to \varphi' \qquad \{\varphi'\}\ C\ \{\psi'\} \qquad \vdash_{AR} \psi' \to \psi}{\{\varphi\}\ C\ \{\psi\}}\ \text{Implied}$$

$$\frac{\{\eta \wedge B\}\ C\ \{\eta\}}{\{\eta\}\ \textbf{while}\ B\ \{C\}\ \{\eta \wedge \neg B\}}\ \text{Partial-while}$$

We say sequent $\vdash_{par} \{\varphi\}\ P\ \{\psi\}$ is valid if ψ can be derived from φ by applying these rules!

A proof calculus for program correctness

$$\frac{\{\varphi\}\ C_1\ \{\eta\} \qquad \{\eta\}\ C_2\ \{\psi\}}{\{\varphi\}\ C_1; C_2\ \{\psi\}}\ \text{Seq. Comp.}$$

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

$$\frac{}{\{\psi[E/x]\}\ x := E\ \{\psi\}}\ \text{Assignment}$$

$$\frac{\vdash_{AR} \varphi \to \varphi' \qquad \{\varphi'\}\ C\ \{\psi'\} \qquad \vdash_{AR} \psi' \to \psi}{\{\varphi\}\ C\ \{\psi\}}\ \text{Implied}$$

$$\frac{\{\eta \wedge B\}\ C\ \{\eta\}}{\{\eta\}\ \textbf{while}\ B\ \{C\}\ \{\eta \wedge \neg B\}}\ \text{Partial-while}$$

We say sequent $\vdash_{par} \{\varphi\}\ P\ \{\psi\}$ is valid if ψ can be derived from φ by applying these rules!

This proof system is sound so we can use this to infer partial correctness, i.e., $\models_{par}$.

Using the proof rules in practice!

Writing trees is unwieldy (and difficult in latex!). So we use a different convention...

Using the proof rules in practice!

Writing trees is unwieldy (and difficult in latex!). So we use a different convention...

Example 1: Show that $\vdash_{par} \{y < 5\}\ x := y + 1\ \{x < 6\}$

Using the proof rules in practice!

Writing trees is unwieldy (and difficult in latex!). So we use a different convention...

Example 1: Show that $\vdash_{par} \{y < 5\}\ x := y + 1\ \{x < 6\}$

$$x := y + 1$$

1. Start by writing down the program

Using the proof rules in practice!

Writing trees is unwieldy (and difficult in latex!). So we use a different convention...

Example 1: Show that $\vdash_{par} \{y < 5\}\ x := y + 1\ \{x < 6\}$

$\{y < 5\}$

$x := y + 1$
$\{x < 6\}$

1. Start by writing down the program
2. Write down the pre- and post-conditions before and after

Using the proof rules in practice!

Writing trees is unwieldy (and difficult in latex!). So we use a different convention...

Example 1: Show that $\vdash_{par} \{y < 5\}\ x := y + 1\ \{x < 6\}$

$\{y < 5\}$

$x := y + 1$
$\{x < 6\}$ Assignment

1. Start by writing down the program
2. Write down the pre- and post-conditions before and after
3. Work backwards using rule corresponding to prog. statement to get a mid condition

Using the proof rules in practice!

Writing trees is unwieldy (and difficult in latex!). So we use a different convention...

Example 1: Show that $\vdash_{par} \{y < 5\}\ x := y + 1\ \{x < 6\}$

$\{y < 5\}$
$\{y + 1 < 6\}$
$x := y + 1$
$\{x < 6\}$ Assignment

1. Start by writing down the program
2. Write down the pre- and post-conditions before and after
3. Work backwards using rule corresponding to prog. statement to get a mid condition

Using the proof rules in practice!

Writing trees is unwieldy (and difficult in latex!). So we use a different convention...

Example 1: Show that $\vdash_{par} \{y < 5\}\ x := y + 1\ \{x < 6\}$

$\{y < 5\}$

$\{y + 1 < 6\}$ Implied

$x := y + 1$

$\{x < 6\}$ Assignment

1. Start by writing down the program
2. Write down the pre- and post-conditions before and after
3. Work backwards using rule corresponding to prog. statement to get a mid condition
4. At any midcondition, you can go further up by using other rules

Another example

Consider program P, which is $z := x; z := z + y; u := z$.

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4**. Write a Hoare triple!

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4.** Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4**. Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

$z := x;$

$z := z + y;$

$u := z;$

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4**. Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

$\{\top\}$

$z := x;$

$z := z + y;$

$u := z;$

$\{u = x + y\}$

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4.** Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

$\{\top\}$

$z := x;$

$z := z + y;$
$\{x + y = z\}$
$u := z;$
$\{u = x + y\}$ Assignment

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4.** Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

$\{\top\}$

$z := x;$
$\{x + y = z + y\}$
$z := z + y;$
$\{x + y = z\}$ Assignment
$u := z;$
$\{u = x + y\}$ Assignment

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4.** Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

$\{\top\}$

$\{x + y = x + y\}$

$z := x;$

$\{x + y = z + y\}$ Assignment

$z := z + y;$

$\{x + y = z\}$ Assignment

$u := z;$

$\{u = x + y\}$ Assignment

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4.** Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

$\{\top\}$	
$\{x + y = x + y\}$	Implied
$z := x;$	
$\{x + y = z + y\}$	Assignment
$z := z + y;$	
$\{x + y = z\}$	Assignment
$u := z;$	
$\{u = x + y\}$	Assignment

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4**. Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

$$
\begin{array}{ll}
\{\top\} & \\
\{x + y = x + y\} & \text{Implied} \\
z := x; & \\
\{x + y = z + y\} & \text{Assignment} \\
z := z + y; & \\
\{x + y = z\} & \text{Assignment} \\
u := z; & \\
\{u = x + y\} & \text{Assignment}
\end{array}
$$

- Though you constructed the proof from bottom to up, you can read it top to bottom!

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4**. Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

$\{\top\}$	
$\{x + y = x + y\}$	Implied
$z := x;$	
$\{x + y = z + y\}$	Assignment
$z := z + y;$	
$\{x + y = z\}$	Assignment
$u := z;$	
$\{u = x + y\}$	Assignment

- Though you constructed the proof from bottom to up, you can read it top to bottom!
- Implied rule always refers to immediate preceding line.

Another example

Consider program P, which is $z := x; z := z + y; u := z$. **Exercise 20.4**. Write a Hoare triple!

Example 2: Show that $\vdash_{par} \{\top\}\ P\ \{u = x + y\}$

$$
\begin{array}{ll}
\{\top\} & \\
\{x + y = x + y\} & \text{Implied} \\
z := x; & \\
\{x + y = z + y\} & \text{Assignment} \\
z := z + y; & \\
\{x + y = z\} & \text{Assignment} \\
u := z; & \\
\{u = x + y\} & \text{Assignment}
\end{array}
$$

- Though you constructed the proof from bottom to up, you can read it top to bottom!
- Implied rule always refers to immediate preceding line.
- Weakest pre-condition

Using the If-then-Else rule

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

We will use a slight variant of the rule in practice...

Using the If-then-Else rule

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

$$\frac{\{\varphi_1\}\ C_1\ \{\psi\} \qquad \{\varphi_2\}\ C_2\ \{\psi\}}{\{(B \to \varphi_1) \wedge (\neg B \to \varphi_2)\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

We will use a slight variant of the rule in practice...

Using the If-then-Else rule

$$\frac{\{\varphi \wedge B\}\ C_1\ \{\psi\} \qquad \{\varphi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

$$\frac{\{\varphi_1\}\ C_1\ \{\psi\} \qquad \{\varphi_2\}\ C_2\ \{\psi\}}{\{(B \to \varphi_1) \wedge (\neg B \to \varphi_2)\}\ \textbf{if}\ B\ \{C_1\}\ \textbf{else}\ \{C_2\}\ \{\psi\}}\ \text{If-then-Else}$$

We will use a slight variant of the rule in practice...

Exercise 20.5

- Show that any instance of the modified rule in a proof can be replaced by an instance of the original rule and instances of rule Implied.
- Is the converse true?