

# Class 06: Introduction to Proof Rules and Proof Outlines

## A. Why?

To reason about programs, we use proof rules (rules of logic) especially designed for the purposes

## B. Outcomes

By the end of the class you should

- Know the proof rules for the textbook's system PD (Partial correctness of Deterministic programs).

## C. Quiz 2 (30 min)

## D. Formal Proofs and Proof Rules

- Now that we have a good idea of the semantics of correctness triples, we can develop proof rules for them.
- A formal proof is a syntactic object, with syntax rules.
- Various ways to write out proofs.
  - Hilbert-style proof: List of judgments and justifications. (High-school geometry)
    1. Length of AB = length of XY Assumption
    2. Angle ABC = Angle XYZ Assumption
    3. Length of BC = length of YZ Assumption
    4. Triangles ABC, XYZ are congruent Side-angle-side, lines 1, 2, 3
- Each line's assertion is an assumption, an axiom, or follows by some rule that appeals to earlier lines in the proof.
- Notation:  $\vdash \{p\} S \{q\}$  = "We can prove (that)  $\{p\} S \{q\}$  (is valid)"
  - The  $\vdash$  symbol is a single turnstile.
- Formally, a proof system over a set of formulas is determined by a set of axioms and proof rules.

## E. Proof System PD [Partial correctness of Deterministic programs]

### Axiom 1 (Skip)

- $\{p\} \text{skip} \{p\}$  (for every assertion  $p$ )

**Axiom 2 (Assignment)**

- The assignment rule is complicated/subtle because it needs to connect the states before and after the assignment.
- Textbook presents the “backward” assignment rule.
  - $\{p[u := t]\} u := t \{p\}$
  - Looks weird but does work.
- Example 1:  
 $\{b+1 \text{ odd}\} b := b+1 \{b \text{ odd}\}$
- Example 2:  
 $\{0 \leq i+1 \leq n \wedge s = \text{sum}(0, i+1)\}$   
 $i := i+1$   
 $\{0 \leq i \leq n \wedge s = \text{sum}(0, i)\}$

**Rule 6 (Consequence) [skipping ahead a bit]**

- How to read the rule: If the three conditions above the line (the antecedents) are provable, then the condition below the line (the consequent) is provable, using the rule.
- The correctness triple rules have one or more antecedents that are correctness triples. The consequence rule also has two predicate calculus antecedents

$$\frac{p_1 \rightarrow p_2 \quad \{p_2\} S \{q_1\} \quad q_1 \rightarrow q_2}{\{p_1\} S \{q_2\}}$$

- The consequence rule is a helper rule (it's not connected with a particular kind of statement). It allows you to strengthen preconditions and weaken postconditions.
- **Definition:**  $p_1$  is **stronger than**  $p_2$  iff  $p_2$  is **weaker than**  $p_1$  iff  $p_1 \rightarrow p_2$ .
- The consequence rule lets you simultaneously strengthen the precondition and weaken the postcondition. Often, we want to do one of these but not both. We can derive two simpler rules to do this.

**Weaken Postcondition**

$$\frac{\{p\} S \{q_1\} \quad q_1 \rightarrow q_2}{\{p\} S \{q_2\}}$$

### Strengthen Precondition

$$\frac{p_1 \rightarrow p_2 \quad \{p_2\} S \{q\}}{\{p_1\} S \{q\}}$$

- These implications get listed in the proof; each one introduces an obligation for a proof using everyday predicate logic (which we'll almost always omit).
- Example:
  - From  $\mathbf{true} \rightarrow 0 = 0$  and  $\{0 = 0\} i := 0 \{i = 0\}$  we can get  $\{\mathbf{true}\} i := 0 \{i = 0\}$  by strengthening the precondition:

Line	Claim	Rule
1	$\mathbf{true} \rightarrow 0 = 0$	Predicate calculus
2	$\{0 = 0\} i := 0 \{i = 0\}$	Assignment
3	$\{\mathbf{true}\} i := 0 \{i = 0\}$	Strengthen <b>precondition</b> 1, 2

### Rule 3 (Composition):

$$\frac{\{p\} S_1 \{r\} \quad \{r\} S_2 \{q\}}{\{p\} S_1 ; S_2 \{q\}}$$

- In English: “If you can prove  $\{p\} S_1 \{r\}$  and you can prove  $\{r\} S_2 \{q\}$ , then you can prove  $\{p\} S_1 ; S_2 \{q\}$  using the composition rule.”
- Expanded: If we can prove that starting in a state satisfying  $p$ , if  $S_1$  terminates, then  $r$  is true, and we can prove that starting in  $r$ , if  $S_2$  terminates then  $q$  is true, then by the composition rule, we can prove that starting in  $p$ , if  $S_1 ; S_2$  terminates, then  $q$  is true.
- Example

$$\frac{\{0 = 0 \wedge 0 = 0\} i := 0 \{i = 0 \wedge i = 0\} \quad \{i = 0 \wedge i = 0\} s := i \{i = 0 \wedge s = 0\}}{\{0 = 0 \wedge 0 = 0\} i := 0 ; s := i \{i = 0 \wedge s = 0\}}$$

- In a formal proof, the format would be

Line	Claim	Rule
1	$\{0 = 0 \wedge 0 = 0\} i := 0 \{i = 0 \wedge i = 0\}$	Assignment
2	$\{i = 0 \wedge i = 0\} s := i \{i = 0 \wedge s = 0\}$	Assignment

Line	Claim	Rule
3	$\{0 = 0 \wedge 0 = 0\} i := 0; s := 0 \{i = 0 \wedge s = 0\}$	Composition, lines 1, 2
4	$\mathbf{true} \rightarrow 0 = 0 \wedge 0 = 0$	Predicate calculus
5	$\{\mathbf{true}\} i := 0; s := 0 \{i = 0 \wedge s = 0\}$	Precondition Strengthening 4, 5

**Rule 4 (Conditional):**

$$\frac{\{p \wedge B\} S_1 \{q\} \quad \{p \wedge \neg B\} S_2 \{q\}}{\{p\} \mathbf{if} B \mathbf{then} S_1 \mathbf{else} S_2 \mathbf{fi} \{q\}}$$

- In English: If you know that:
  - Running the **true** branch  $S_1$  in a state satisfying  $p$  and the **if** test establishes  $q$
  - And running the **false** branch  $S_2$  in a state satisfying  $p$  and the negation of the **if** test establishes  $q$
- Then you know that running the **if-else** in a state satisfying  $p$  establishes  $q$ .
- Example:

$$\frac{\{\mathbf{true} \wedge x \geq 0\} y := x \{y \geq 0\} \quad \{\mathbf{true} \wedge x < 0\} y := -x \{y \geq 0\}}{\{\mathbf{true}\} \mathbf{if} x \geq 0 \mathbf{then} y := x \mathbf{else} y := -x \mathbf{fi} \{y \geq 0\}}$$

- There's no separate rule for an **if-then**: Treat it as an **if-else skip**.

**Rule 5 (Loop)**

- To understand the rule, we'll look at **loop invariants** after presenting the rule.

$$\frac{\{p \wedge B\} S_1 \{p\}}{\{p\} \mathbf{while} B \mathbf{do} S_1 \mathbf{od} \{p \wedge \neg B\}}$$

- Example: (where  $p \equiv 0 \leq i \leq n \wedge s = \mathit{sum}(0, i)$ )

$$\frac{\{p \wedge i < n\} i := i+1; s := s+i \{p\}}{\{p\} \mathbf{while} i < n \mathbf{do} i := i+1; s := s+i \mathbf{od} \{p \wedge i \geq n\}}$$

**F. Why Loop Invariants?**

- Let  $W$  be the loop **while**  $B$  **do**  $S_1$  **od**. For which  $\sigma$  do we get  $\sigma \models \{p\} W \{q\}$ ? I.e., if  $\sigma \models p$ , when does  $\mathcal{M}[\![W]\!](\sigma) \models q$ ?

- We defined the meaning of a loop using the states  $\tau_0, \tau_1, \dots, \tau_k$  that hold at each while loop test, with  $\tau_k$  being the first one that satisfies  $\neg B$ .
- How can we characterize/describe the states  $\tau_0, \tau_1, \dots, \tau_k$ ? We can't use  $k$  different predicates because in general, we don't know what  $k$  is.
- Three kinds of iterations: First, middle, and last.
  - Try using 3 predicates, one for each kind of iteration?
  - Gets complicated; easier to use one predicate  $r$  for all iterations.
- “ $r$  is a loop invariant” = “ $r$  is true at each loop test” — it generalizes the loop precondition, postcondition, and general condition.
  - We need (1)  $\tau_0, \tau_1, \dots, \tau_k \models r$ , (2) for all  $0 \leq j < k$ , we have  $\tau_j \models r \wedge B$ , and (3)  $\tau_k \models r \wedge \neg B$ .
  - Note  $\tau_j \models \{r \wedge B\} S_1 \{r \wedge B\}$  for  $j < k-1$  and  $\tau_{k-1} \models \{r \wedge B\} S_1 \{r \wedge \neg B\}$ .
  - Combine to get  $\tau_j \models \{r \wedge B\} S_1 \{r\}$  for every  $j$  (including  $j = k$ ).
- In that case,  $\tau_0 \models \{r\} W \{r \wedge \neg B\}$ .
  - Instead of  $\sigma \models \{p\} W \{q\}$ , let's use  $\sigma \models \{r\} W \{r \wedge \neg B\}$ . We can get  $p$  and  $q$  using consequence or composition if we really need them.
  - Use  $\{p\}$  initialization code  $\{r\}$  and  $\{r \wedge \neg B\}$  clean-up code  $\{q\}$ .

## G. Example of Loop Invariant

- Typical loop: Sum natural numbers  $0, 1, \dots, n$  where  $n \geq 0$ .
- Define  $sum(x, y) = x + (x+1) + \dots + y$  (or 0 if  $x > y$ )
- Take the program:
 

```

      {n ≥ 0}
      i := 0; s := 0;
      while i < n do
          i := i+1; s := s+i
      od
      {s = sum(0, n)}
      
```
- For loop invariant, use  $p \equiv 0 \leq i \leq n \wedge s = sum(0, i)$ .
- Initialization: Does  $\{n \geq 0\} i := 0; s := 0 \{p\}$  hold?
  - We know  $\{p[s := 0]\} s := 0 \{p\}$  and  $\{p[s := 0][i := 0]\} i := 0 \{p[s := 0]\}$  by the assignment axiom. (You can do calculations of substitutions.)
  - Turns out  $n \geq 0 \rightarrow p[s := 0][i := 0]$ .
  - Combine using composition and precondition strengthening.
- Loop body: Does  $\{p \wedge i < n\} i := i+1; s := s+i \{p\}$  hold? Use same pattern.
  - $\{p[s := s+i]\} s := s+i \{p\}$  holds by assignment axiom
  - $\{p[s := s+i][i := i+1]\} i := i+1 \{p[s := s+i]\}$  holds by assignment axiom.

- If you calculate the substitutions, we can verify  $p \wedge i < n \rightarrow p[s := s+i][i := i+1]$ .
- So we know  $\{p\} W \{p \wedge i \geq n\}$  where  $W$  is the loop. We can verify  $p \wedge i \geq n \rightarrow s = \text{sum}(0, n)$
- Combining everything gives us the program  $\{n \geq 0\} i := 0; s := 0; W \{s = \text{sum}(0, n)\}$ .

Here's a formal proof:

Line	Claim	Rule
1	$\{p[s := 0]\} s := 0 \{p\}$	Assignment
2	$\{p[s := 0][i := 0]\} i := 0 \{p[s := 0]\}$	Assignment
3	$\{p[s := 0][i := 0]\} i := 0; s := 0 \{p\}$	Composition, lines 2, 1
4	$n \geq 0 \rightarrow p[s := 0][i := 0]$	Predicate Calculus
5	$\{n \geq 0\} i := 0; s := 0 \{p\}$	Precondition Strengthening, 4, 3
6	$\{p[s := s+i]\} s := s+i \{p\}$	Assignment
7	$\{p[s := s+i][i := i+1]\} i := i+1 \{p[s := s+i]\}$	Assignment
8	$\{p[s := s+i][i := i+1]\} i := i+1; s := s+i \{p\}$	Composition 7, 6
9	$p \wedge i < n \rightarrow p[s := s+i][i := i+1]$	Predicate Calculus
10	$\{p \wedge i < n\} i := i+1; s := s+i \{p\}$	Precondition Strengthening, 9, 8
11	$\{p\} \text{ while } i < n \text{ do } i := i+1; s := s+i \text{ od } \{p \wedge i \geq n\}$	While loop, 10
12	$\{n \geq 0\} i := 0; s := 0; W \{p \wedge i \geq n\}$	Composition 5, 11
13	$p \wedge i \geq n \rightarrow s = \text{sum}(0, n)$	Predicate Calculus
14	$\{n \geq 0\} i := 0; s := 0; W \{s = \text{sum}(0, n)\}$	Postcondition Weakening, 13, 12

- The proof gives us four substitutions to do:
  - $p[s := 0] \equiv ???$
  - $p[s := 0][i := 0] \equiv ???$
  - $p[s := s+i] \equiv ???$
  - $p[s := s+i][i := i+1] \equiv ???$
- The proof gives us three “predicate calculus obligations” [implications we need to be true, otherwise the overall proof is incorrect].
  - $n \geq 0 \rightarrow p[s := 0][i := 0]$
  - $p \wedge i < n \rightarrow p[s := s+i][i := i+1]$

- $p \wedge i \geq n \rightarrow s = \text{sum}(0, n)$

## H. Proof Outlines

- Formal proofs are really long! :- ( Lots of repeated text.
- A **proof outline** contains all the information needed to generate a formal proof but without repeating so much text.
- The structure of a formal proof of correctness mirrors the structure of the program.
- We'll **annotate** (= **decorate**) the program with assertions/conditions.
- If a statement appears between two assertions, that stands for a correctness triple.
- If two assertions lie next to each other, that stands for an implication (the first one implies the second) and a use of precondition strengthening or postcondition weakening as appropriate.
- To specify a loop invariant, we write **{inv p} while B do S<sub>1</sub> od**.

### Example

- The outline  $\{n \geq 0\} \{p[s:=0][i:=0]\} i:=0; \{p[s:=0]\} s:=0 \{p\}$  stands for lines 1–5 of the formal proof earlier.
- The outline  $\{p \wedge i < n\} \{p[s:=s+i][i:=i+1]\} i:=i+1; \{p[s:=s+i]\} s:=s+i \{p\}$  stands for lines 6–10 of the formal proof.
- Combining these two outlines and adding the final postcondition stands for lines 11–14 of the formal proof. The whole outline is

$$\begin{aligned} & \{n \geq 0\} \{p[s:=0][i:=0]\} i:=0; \{p[s:=0]\} s:=0; \\ & \{\text{inv } p\} \text{ while } i < n \text{ do} \\ & \quad \{p \wedge i < n\} \{p[s:=s+i][i:=i+1]\} i:=i+1; \{p[s:=s+i]\} s:=s+i \{p\} \\ & \text{od} \\ & \{p \wedge i \geq n\} \\ & \{s = \text{sum}(0, n)\} \end{aligned}$$

- Outlines for loops
  - **{inv p} while B do {p ∧ B} S<sub>1</sub> {p} od {p ∧ ¬B}**
- Outlines for conditionals
  - There are a couple of different styles.
  - One is: **{p} if B then {p ∧ B} {q<sub>1</sub>} S<sub>1</sub> {q} else {p ∧ ¬B} {q<sub>2</sub>} S<sub>2</sub> {q} fi {q}**
  - The other is:
 
$$\{(p \wedge B \rightarrow q_1) \wedge (p \wedge \neg B \rightarrow q_2)\} \text{ if } B \text{ then } \{q_1\} S_1 \{q\} \text{ else } \{q_2\} S_2 \{q\} \text{ fi } \{q\}$$