CS292C-2: Computer-Aided Reasoning for Software

Lecture 2: IMP Syntax & Semantics

Spring 2025 | Prof. Yu Feng

Yu **by Yu Feng**

Why Define a Language?

Formal Verification Needs Precision

Verification requires precise semantics to reason about programs.

Real Languages Are Problematic

Too complex, underspecified, or large to prove properties about.

Clean Alternative Needed

IMP provides a simple, well-defined foundation for verification.

The Role of IMP in This Course

Ideal Sandbox

Just expressive enough while remaining simple to analyze.

Turing Complete

Can express any computation despite its minimal design.

Practical Applications

You'll build verifiers, symbolic executors, and synthesizers with it.

Syntax (Recap)

Expressions

```
e ::= n | x | e1 + e2
    | e1 - e2 | e1 * e2
```

Booleans

```
b ::= true | false | e1 = e2
    | e1 ≤ e2 | ¬b | b1 ∧ b2
```

Commands

```
c ::= skip | x := e | c1 ; c2
    | if b then c1 else c2
    | while b do c
```

Exercise: Identify Constructs

1

Given Program

```
while x ≤ 5 do
  x := x + 1;
  y := y * 2
```

2

Syntax Rules Used

While loop, comparison, assignment, sequence, addition, multiplication

3

Validity Check

Yes, this is valid IMP syntax following the grammar rules.

What is Semantics?

Syntax

Defines how code looks - the structure and form.

Semantics

Defines what code means - the behavior and execution.

Two Approaches

Big-step (results) and small-step (execution traces).

States

Definition

A map from variables to integers.

Example

$\sigma = \{x \mapsto 2, y \mapsto 7\}$

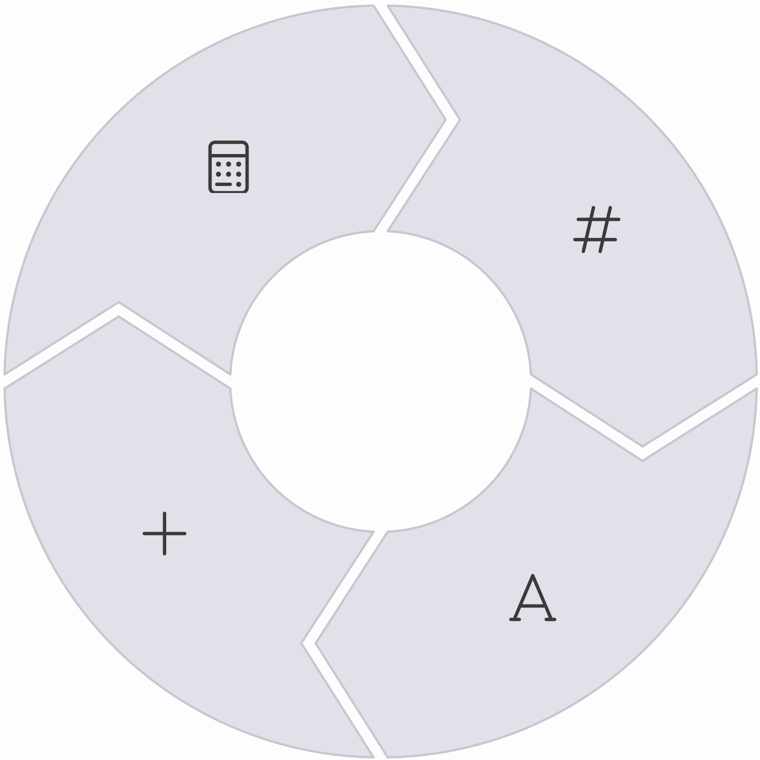
Notation

$\sigma(x) = 2$ and $\sigma[x \mapsto 5] = \sigma'$ (updated state)

Big-Step Semantics for Expressions

Meaning

⟨e, σ⟩ ⇓ n means: expression e evaluates to n under state σ



Constants

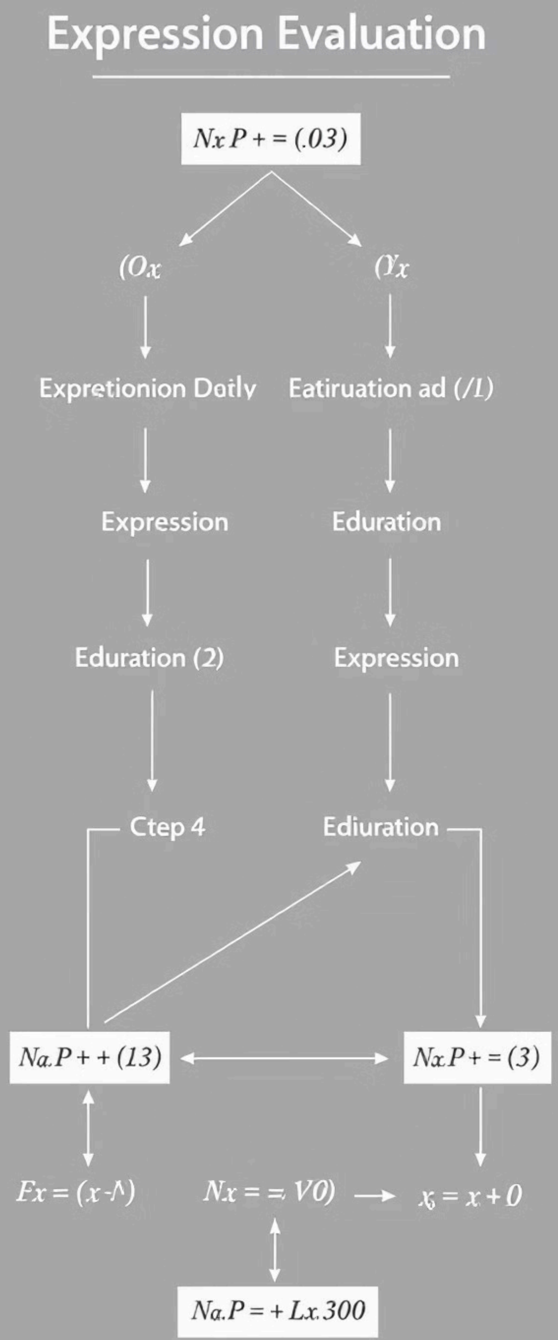
[Const] ⟨n, σ⟩ ⇓ n

Variables

[Var] ⟨x, σ⟩ ⇓ σ(x)

Addition

[Plus] ⟨e1, σ⟩ ⇓ n1, ⟨e2, σ⟩ ⇓ n2 ⇒ ⟨e1 + e2, σ⟩ ⇓ n1 + n2



Example Derivation (Expression)

1

Evaluate

□x + 1, {x □ 2}□ □ ?

2

Step 1

□x, σ□ □ 2 (by Var rule)

3

Step 2

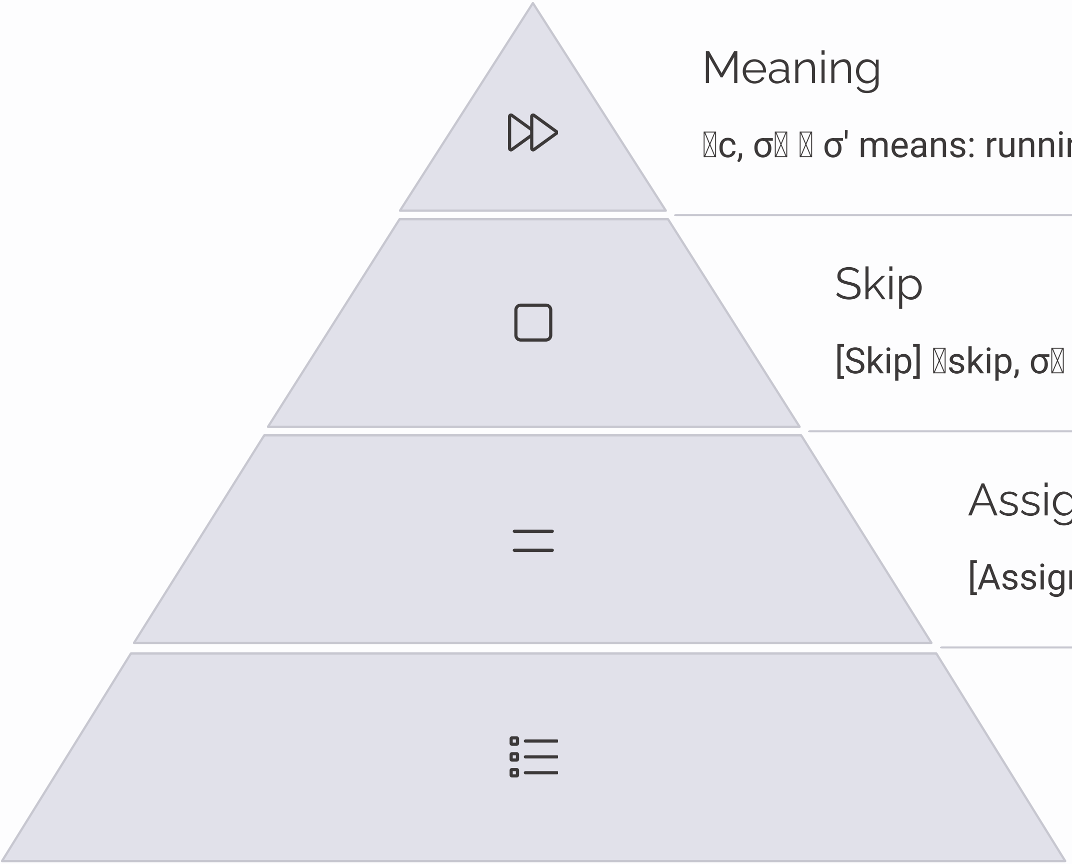
□1, σ□ □ 1 (by Const rule)

4

Result

⇒ □x + 1, σ□ □ 3 (by Plus rule)

Big-Step Semantics for Commands



Meaning

▯c, σ▯ ▯ σ' means: running command c in σ produces state σ'

Skip

[Skip] ▯skip, σ▯ ▯ σ

Assignment

[Assign] ▯e, σ▯ ▯ n ⇒ ▯x := e, σ▯ ▯ σ[x ▯ n]

Sequence

[Seq] ▯c1, σ▯ ▯ σ1, ▯c2, σ1▯ ▯ σ2 ⇒ ▯c1; c2, σ▯ ▯ σ2

Example Derivation (Command)

Evaluate

⟨x := 1; y := x + 2, {}⟩ ⇓ ?

First Command

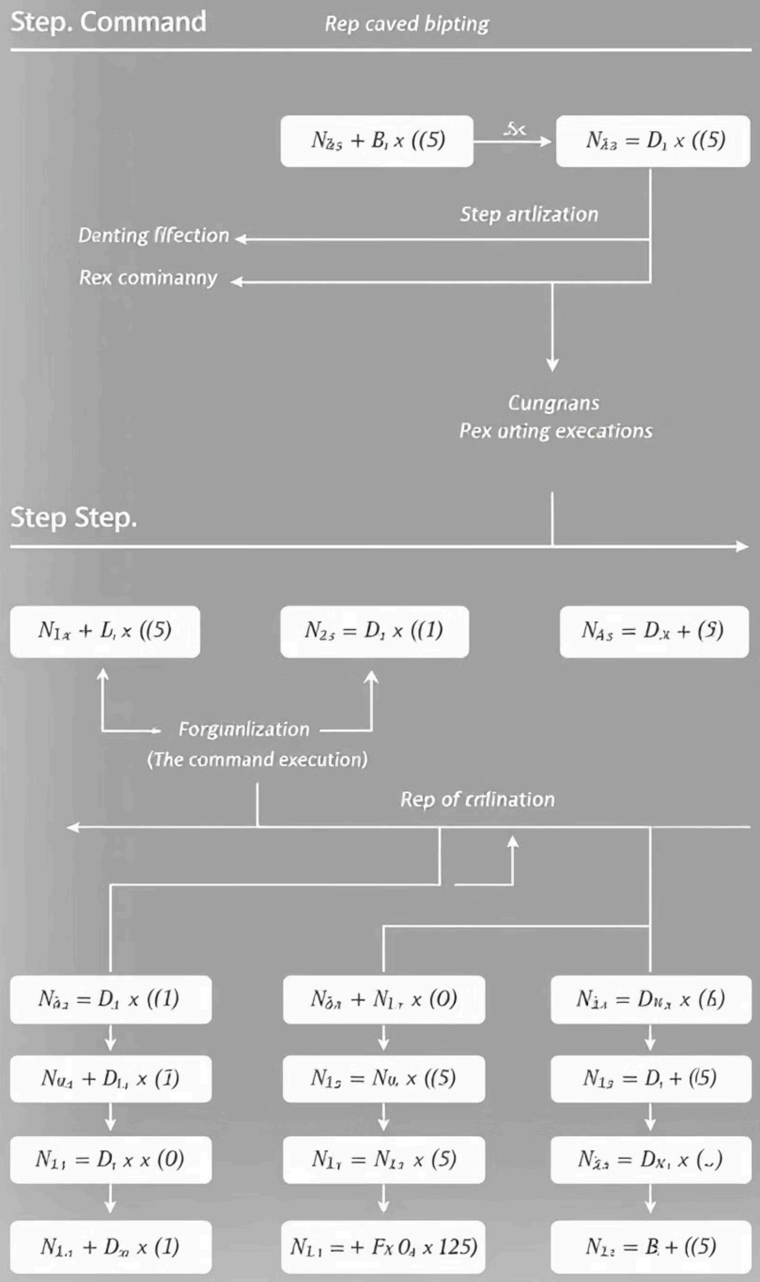
⟨x := 1, {}⟩ ⇓ {x ↦ 1} (by Assign rule)

Second Command

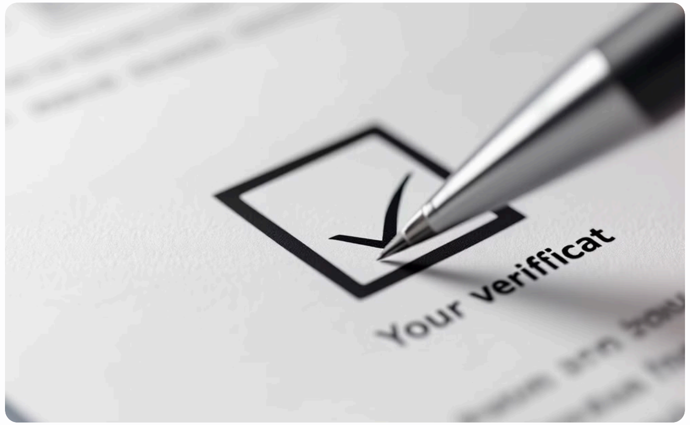
⟨y := x + 2, {x ↦ 1}⟩ ⇓ {x ↦ 1, y ↦ 3} (by Assign rule)

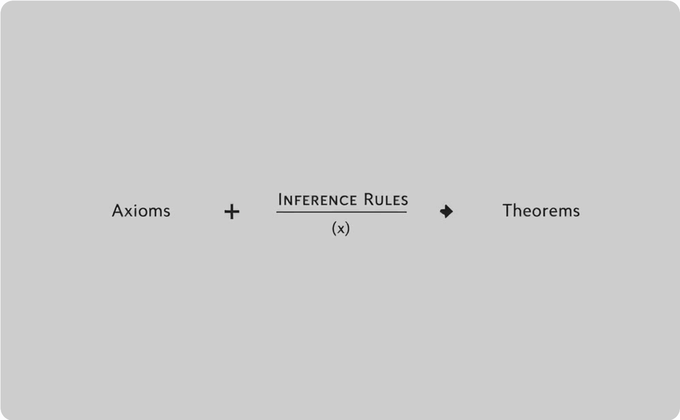
Final Result

⟨x := 1; y := x + 2, {}⟩ ⇓ {x ↦ 1, y ↦ 3} (by Seq rule)



Why Big-Step Semantics?





Direct Reasoning

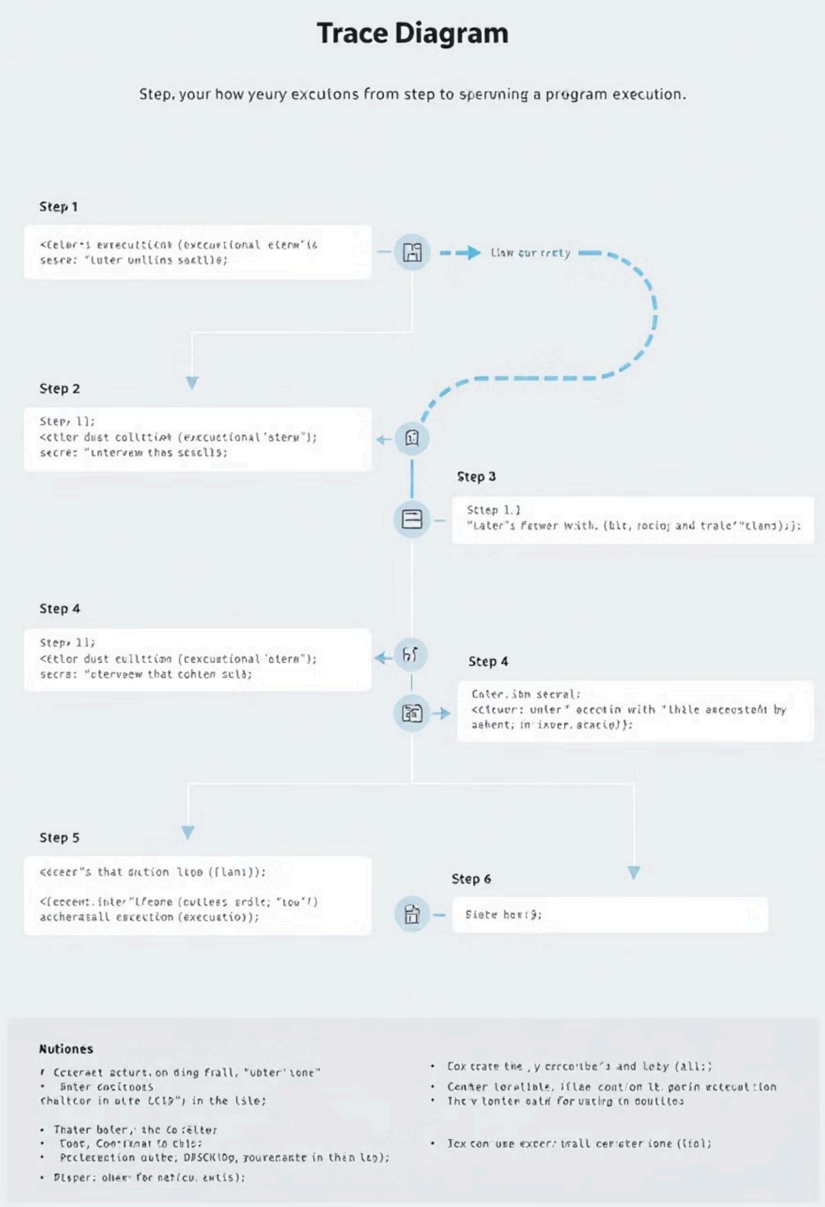
Enables straightforward reasoning about end results of programs.

Easier Proofs

Simplifies partial correctness proofs in verification.

Limitation

Can't distinguish non-termination from being "stuck" in execution.



Motivation for Small-Step

1

Results vs. Process

Big-step shows what happens, small-step shows how it happens.

3

Key Applications

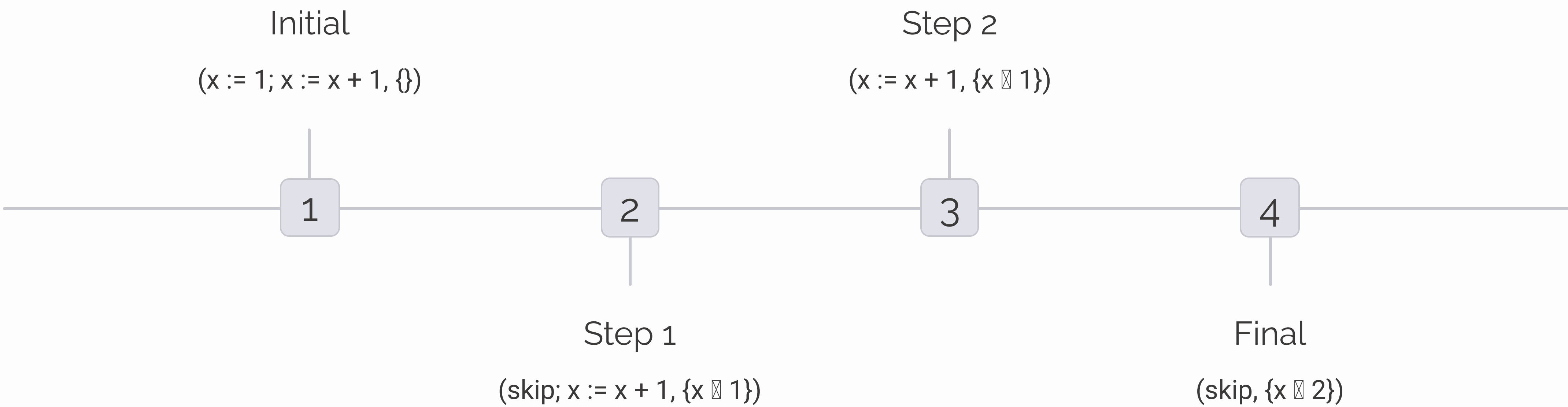
Symbolic execution, interactive proof assistants, debugging.

∞

Execution Traces

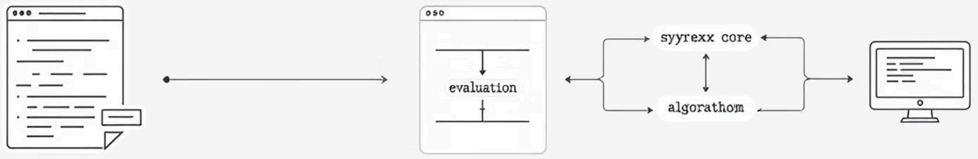
Captures step-by-step program behavior, even for non-terminating programs.

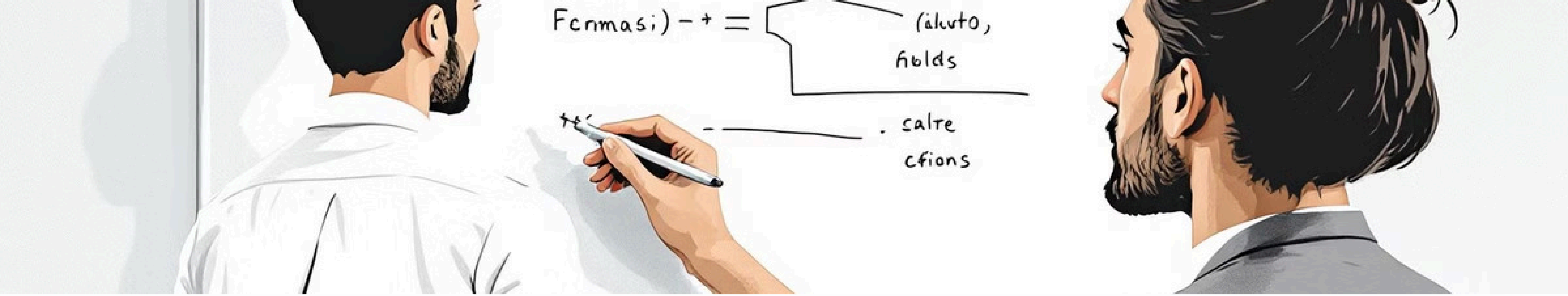
Small-Step Example



Syntax + Semantics Together

- IMP is executable, not just notation
- Semantics serves as an interpreter for logic
- You'll build symbolic and verification interpreters using this foundation

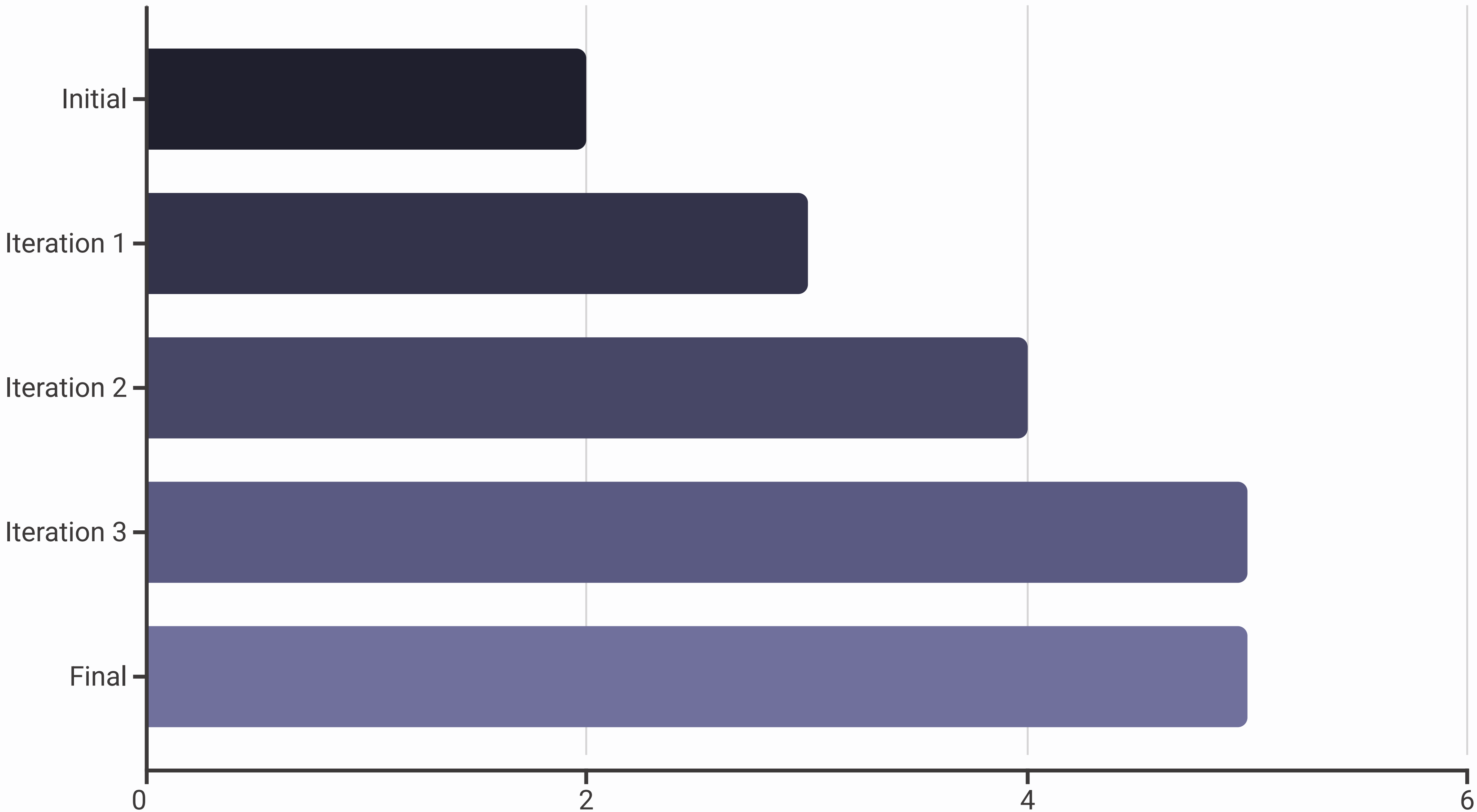




Exercise: Big-Step Evaluation

Given:	$\sigma = \{x \mapsto 2\}$
Command:	if $x \leq 3$ then $y := x + 1$ else $y := 0$
Question 1:	What is the resulting state?
Question 2:	What derivation tree justifies your answer?

Exercise: Loop Tracing



Given $\sigma = \{x \mapsto 2\}$ and command: while $x \leq 4$ do $x := x + 1$

Trace the execution steps manually on whiteboard.

What You've Learned

IMP Purpose

Understanding what IMP is and why it matters for verification.

Formal Definitions

How to formally define a programming language.

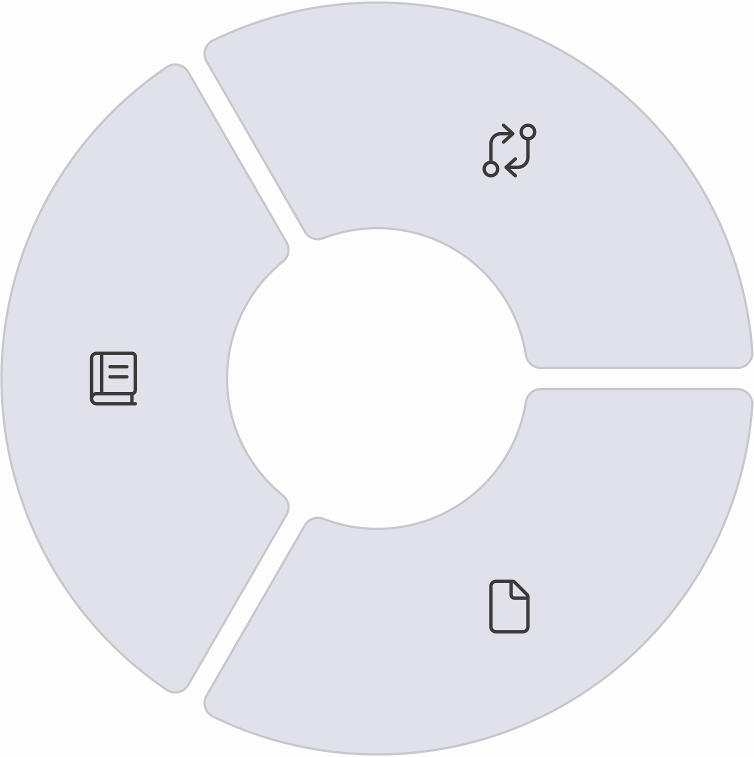
Semantic Approaches

Big-step vs. small-step semantics and their applications.

Foundation

Building blocks for everything else in this course.

What's Next



Assignment 1

Implementing a verifier for IMP

Hoare Logic

Reasoning with Pre/Post Conditions

Reading

Slides, reference implementation, Winskel Ch. 2 (optional)