

CS292C-2: Computer-Aided Reasoning for Software

Lecture 2: IMP Syntax & Semantics

Spring 2025 | Prof. Yu Feng

 **by Yu Feng**



Why Define a Language?

Formal Verification Needs Precision

Verification requires precise semantics to reason about programs.

Clean Alternative Needed

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

Real Languages Are Problematic

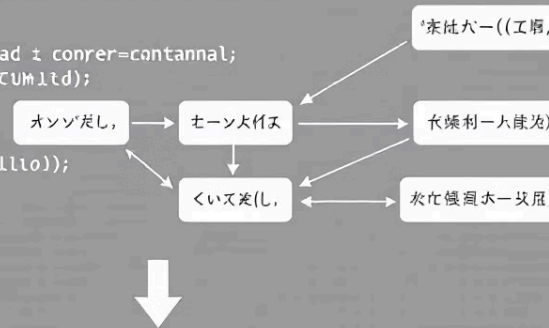
Too complex, underspecified,
or large to prove properties
about.

大鰐付糸一ツミンサ

```

1 mination syntactfletuens: Income townal);
  frinntling..f calls commpnational(t2), An compre flonts:
    fi atting ==の(左めが夫勝), (又lm)にててて(6);
    ftaapt uutte Apppl(るlen));
    0mutiations, notating an decridetling (tar同(king(115麗),
    comple notwall(て人がて(6));
    {
      com pater sortunes . fil ((0020+, 'て根銀(8);
    }
  );
al) cancur(etutive-(utad i conrer=contannal;
41; notuをててに足る節集てum1ld);
}
u);
    compent: "b1A(S11110));
} );
2);

```

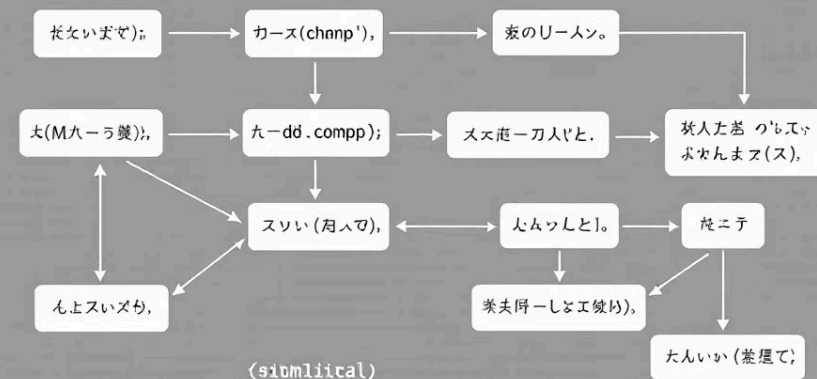


大人レて寝ん庚(らけ)は

```

simplificant simplifying notation);
Anate Butckl 5entaion(), 習得, 2rko(12歳の幼女に宛て);
collection, jany; "Fgone(12)にさ(ざんが)の(理)制、て(先)生(素)問、て(d)).

```





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

$e ::= n \mid x \mid e1 + e2$
 $\mid e1 - e2 \mid e1 * e2$

Booleans

$b ::= \text{true} \mid \text{false} \mid e1 = e2$
 $\mid e1 \leq e2 \mid \neg b \mid b1 \wedge b2$

Commands

$c ::= \text{skip} \mid x := e \mid c1 ; c2$
 $\mid \text{if } b \text{ then } c1 \text{ else } c2$
 $\mid \text{while } b \text{ do } c$

Exercise: Identify Constructs

1

Given Program

```
while  $x \leq 5$  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).

Code sinex
this a swith
programmall
prograting

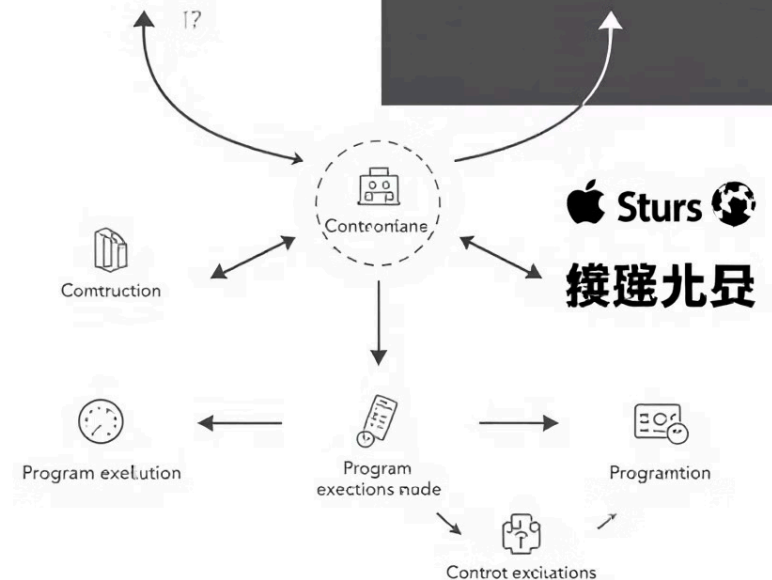
A Program, certecitions:

```
(eal stuf - thies (uvition);  
{  
  Coularizist avatercall,  
  vrutial aesting;  
  { trie, llise - patiction;  
    eroplication - and/ineaction;  
  }  
}
```

Comfunt,
Dulicatio
and thave
interuptions
protlution

Couptant in Programming

```
{  
  () vuall eartyst();  
  couallertecutions (Skingla)  
};  
complotacttion  
{  
  esyllt exeeing certian l;  
  ewulle exacting();  
  porbliens conigiel)  
};  
// rowll - rasyeraly  
// ewulls ehattractionf;  
// rowllows fagr avellections);  
// coula (ssvalvariscation  
  no kalll e raryating inlol;  
  coull lers (miration)  
};  
Code in exection {;  
  lntal - feauestalings  
  rowll exetecions letetion);  
};  
Ovrisite exection;  
  null: longrowwamng {};  
  conta billes;  
};  
vwuall exection();  
Syuallke entarolection;  
};  
culta incitles  
};  
comll cangrantistecion l;  
};  
emiple exetecion();  
Inseetecion)  
};  
(vwllc covwregrecations)  
  lmpolseual interyartacttim);  
};  
twaly - program;  
};  
ewall exetecion;
```



States



Definition

A map from variables to integers.



Example

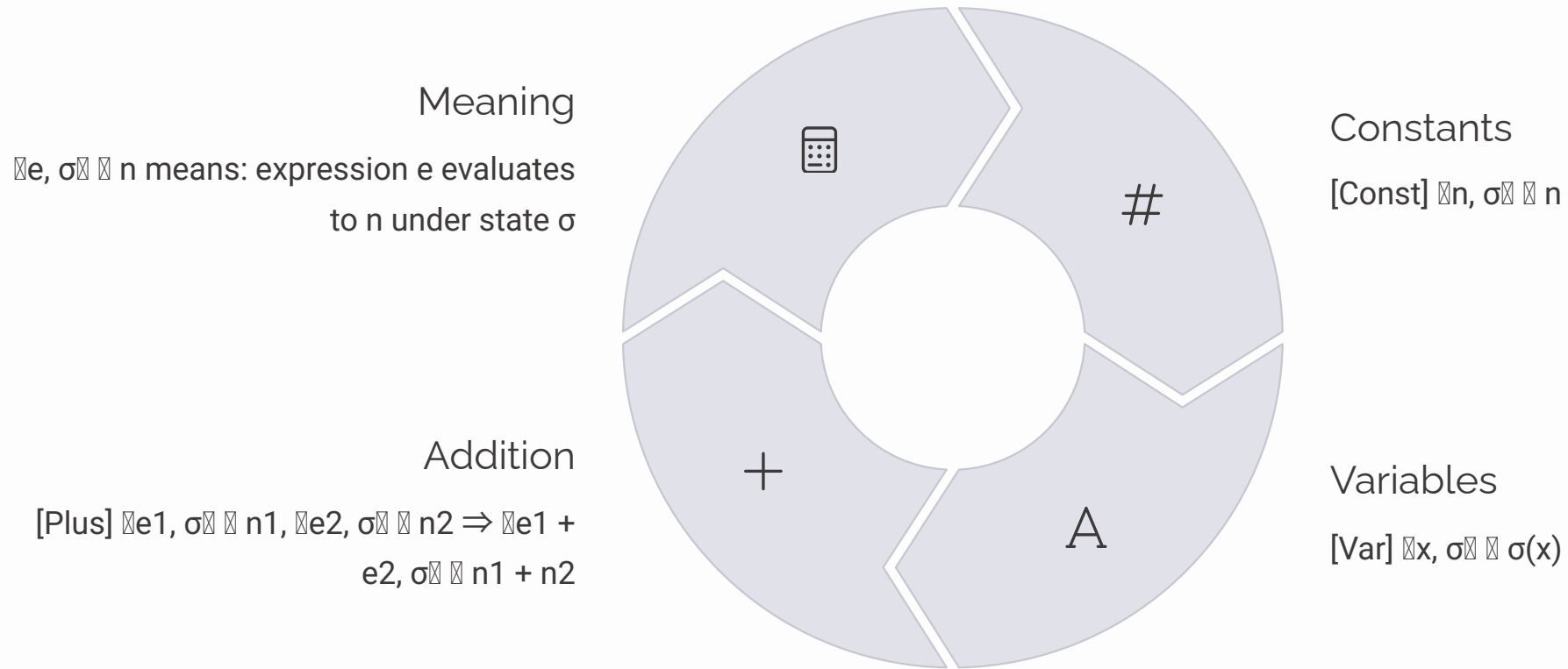
$\sigma = \{x \mapsto 2, y \mapsto 7\}$



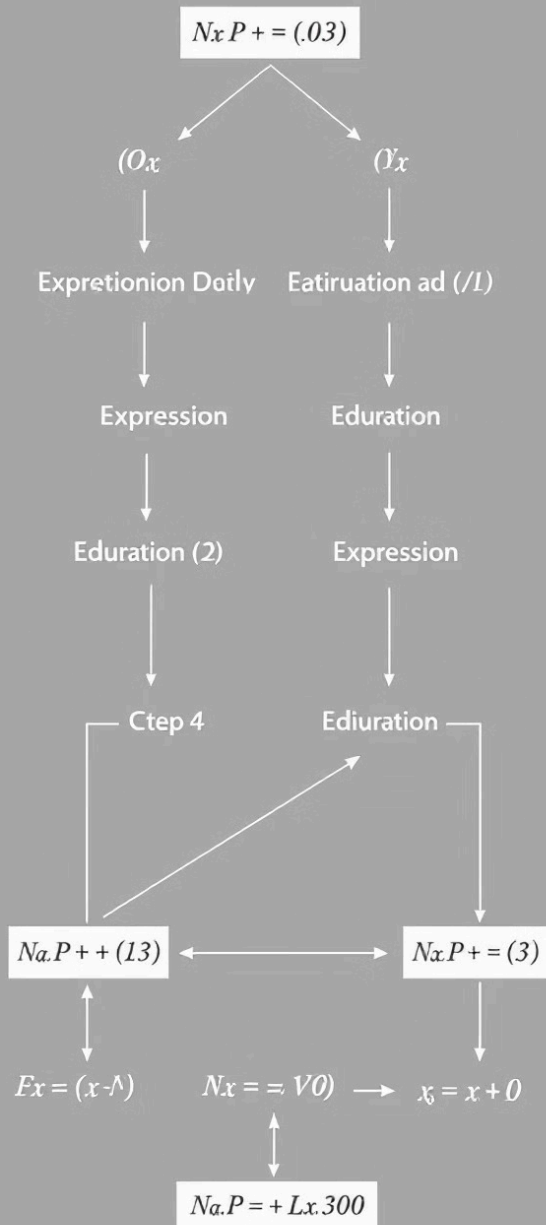
Notation

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

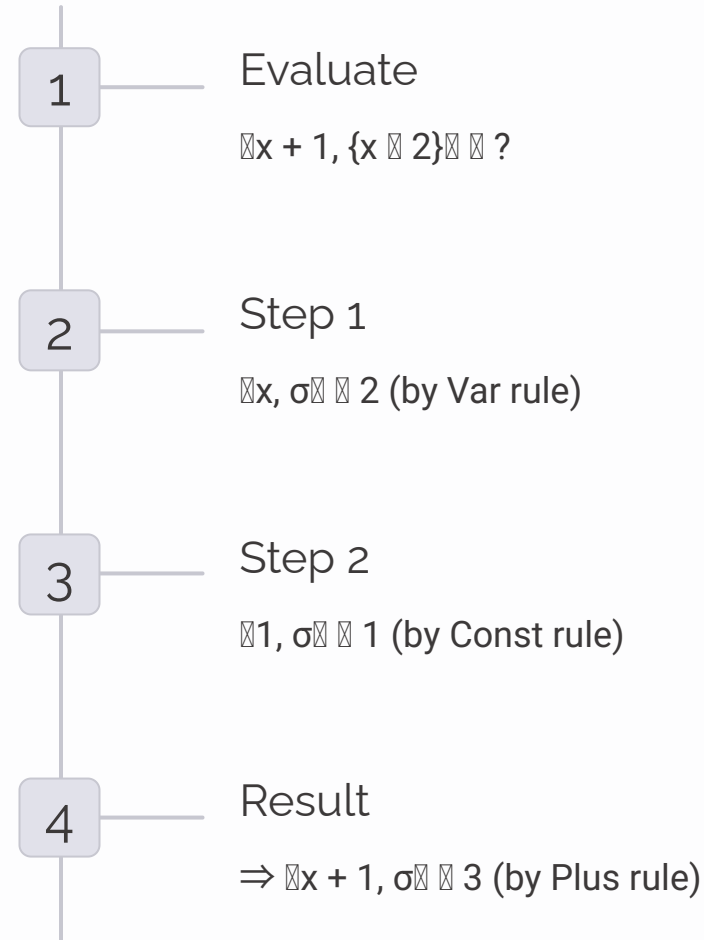
Big-Step Semantics for Expressions



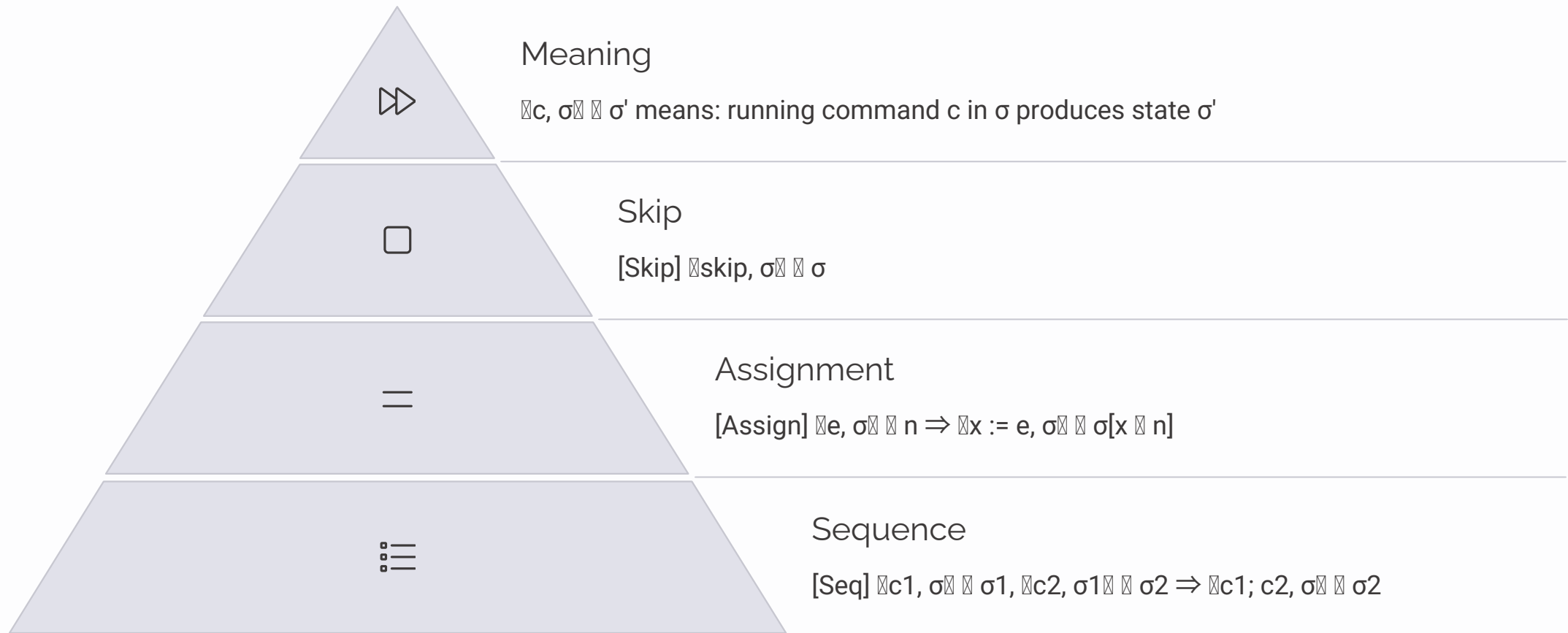
Expression Evaluation



Example Derivation (Expression)



Big-Step Semantics for Commands



Example Derivation (Command)

Evaluate

$\{x := 1; y := x + 2, \}\ ?$

First Command

$\{x := 1, \}\ \{x \neq 1\}$ (by Assign rule)

Second Command

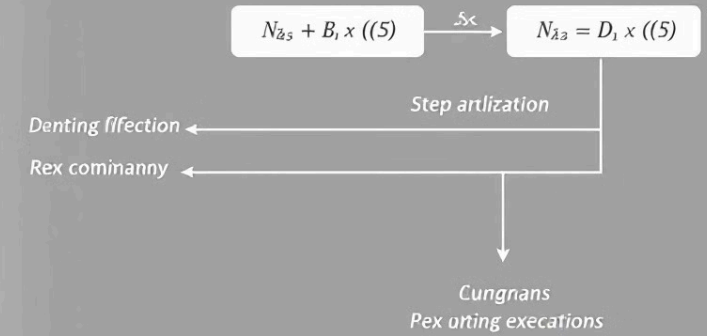
$\{y := x + 2, \{x \neq 1\}\ \{x \neq 1, y \neq 3\}$ (by Assign rule)

Final Result

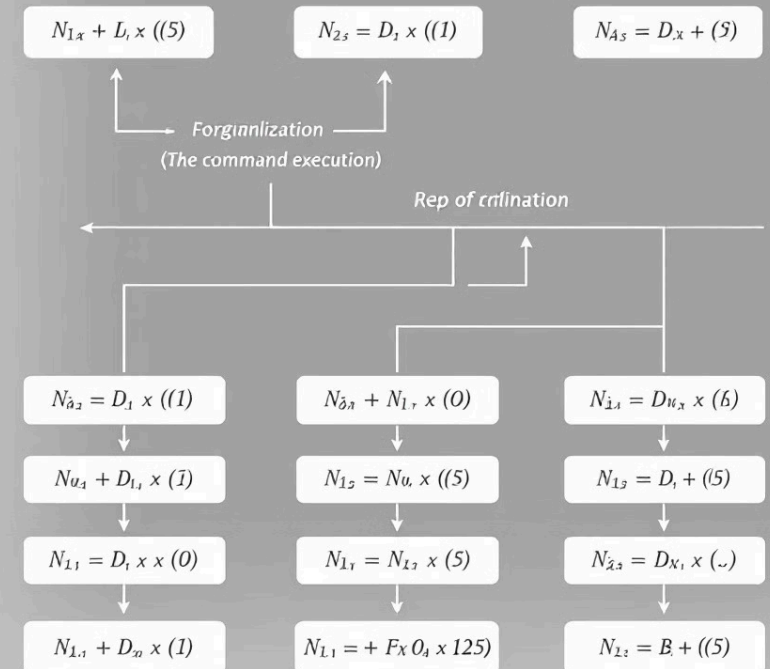
$\{x := 1; y := x + 2, \}\ \{x \neq 1, y \neq 3\}$ (by Seq rule)

Step. Command

Rep. caved bipting



Step Step.

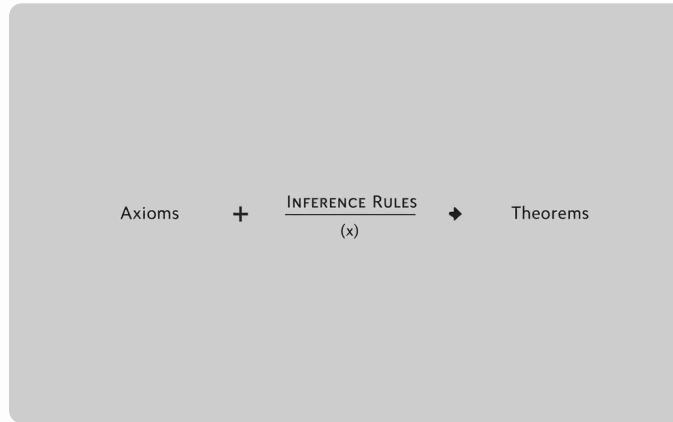


Why Big-Step Semantics?



Direct Reasoning

Enables straightforward reasoning about end results of programs.



Easier Proofs

Simplifies partial correctness proofs in verification.

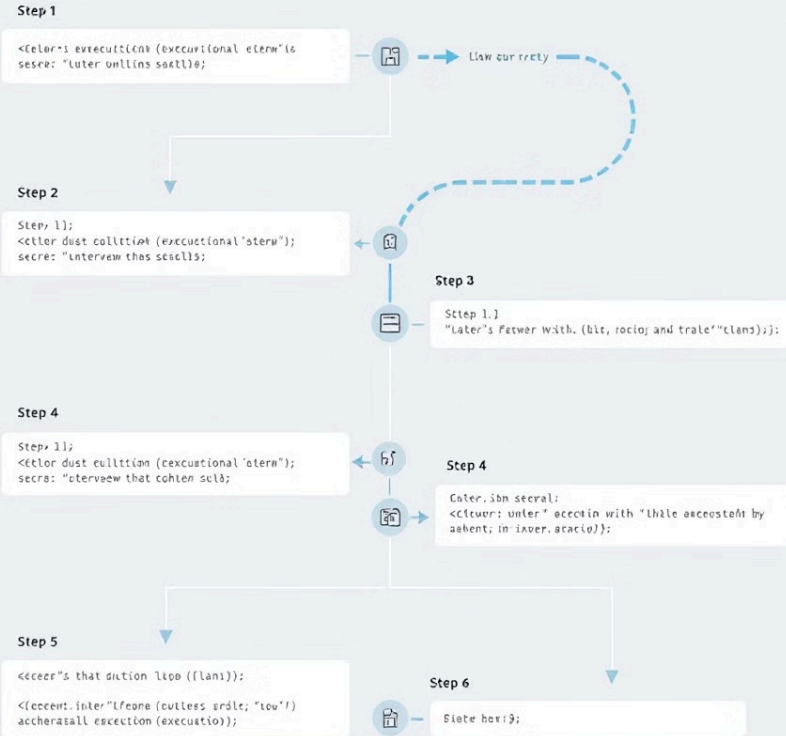


Limitation

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

Trace Diagram

Step, your how yeury excutons from step to spervning a program execution.



Notiones

r C@eract actus on ding frall, "ubter" tone
• Enter co@t@o@s
r@l@or in ult@ (C1@) in the l@le;

- Th@ter b@ter, the co s@lter
- T@st, Co@f@n@l to c@l@;
- P@ct@ec@on out@; D@SCKID@, r@ur@n@te in th@n l@p;
- D@p@er, o@l@er for n@t@c@u, ext@is;

- Co@trate the ,y @r@p@r@b@'s and l@b@ (All);
- Co@ter l@r@l@le, if l@e co@t@on l@ p@r@n @ct@u@on
- Th@v l@nt@r o@l@ for v@st@n in o@l@l@es

- T@x co@ use ext@er; tr@ll co@ter@one (l@l);

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.



Step 1: Initialization

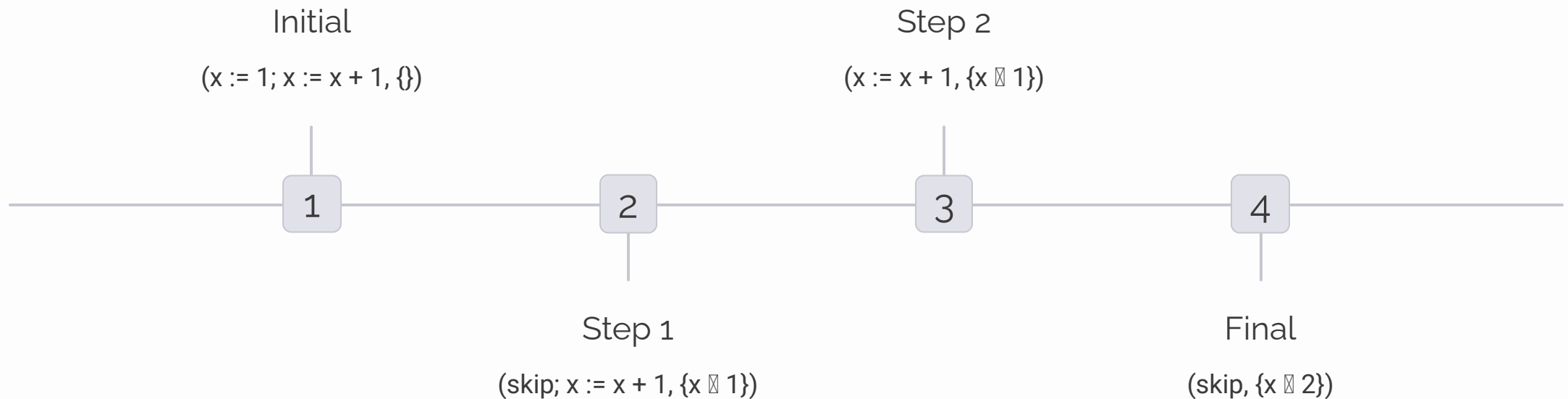


Step 2: Data Input



Step 4: Output

Small-Step Example



Syntax + Semantics Together



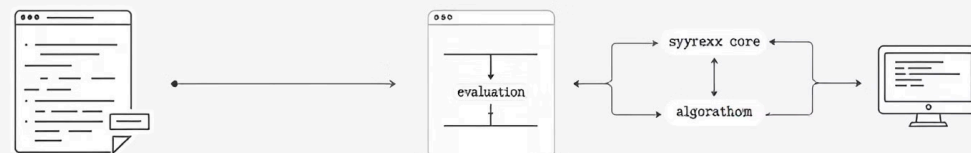
IMP is executable, not just notation

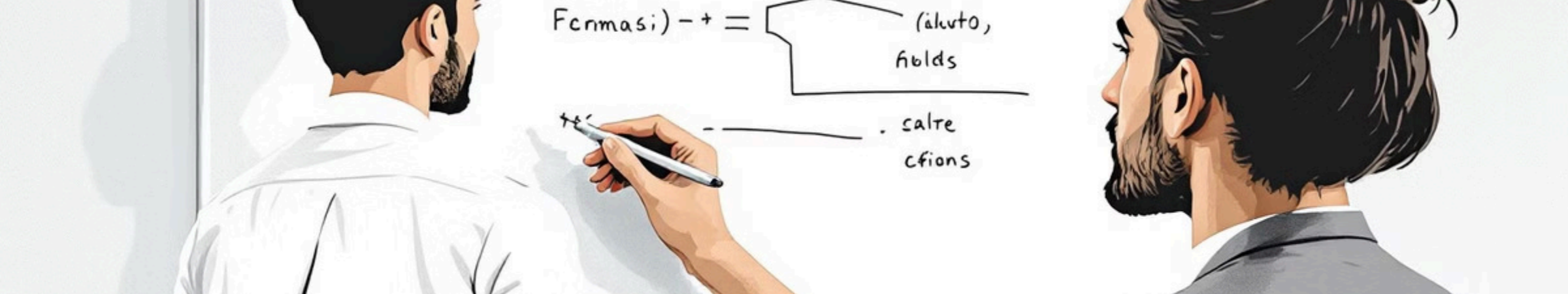


Semantics serves as an interpreter for logic



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 $y := 0$

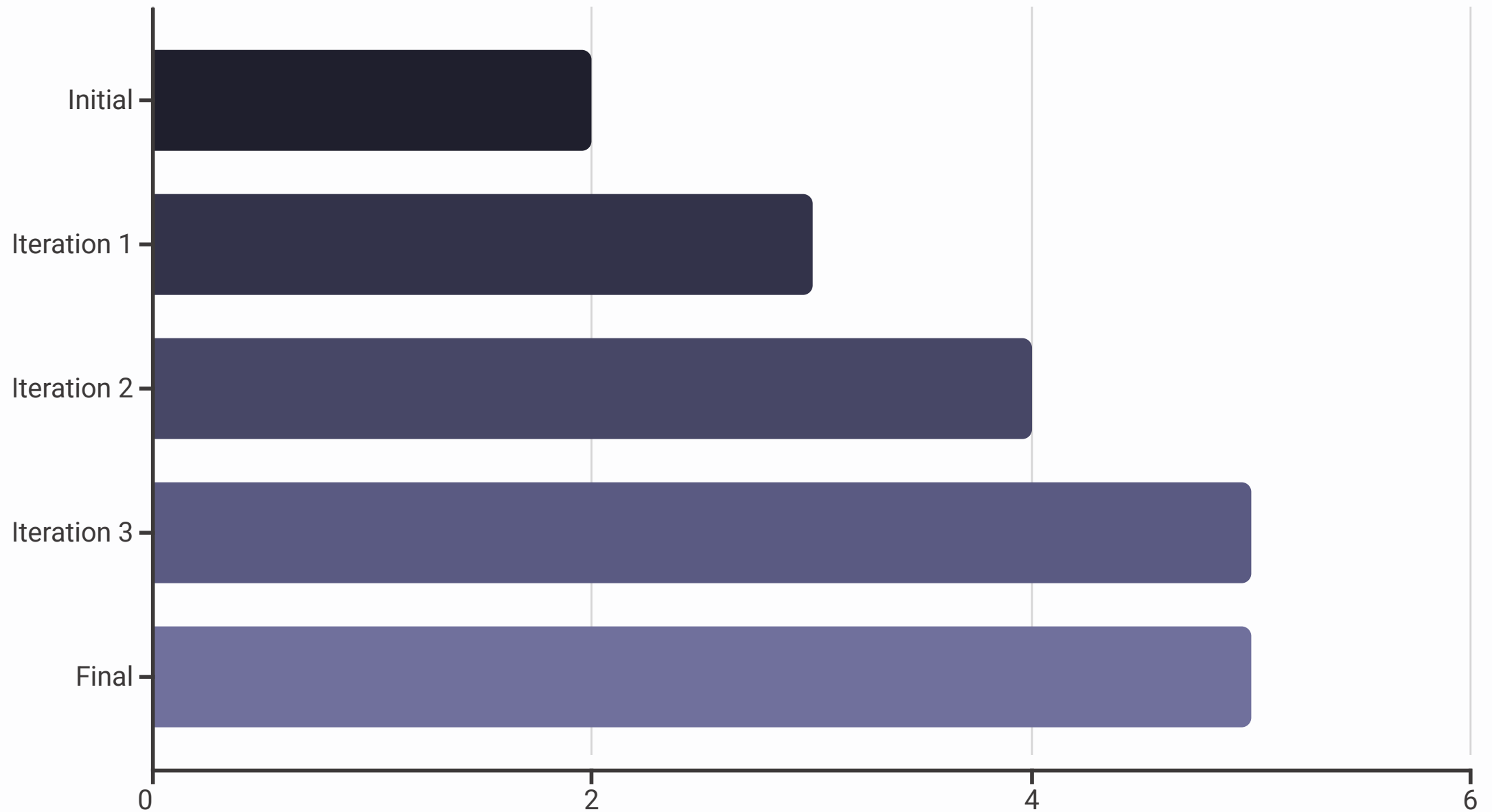
Question 1:

What is the resulting state?

Question 2:

What derivation tree justifies your answer?

Exercise: Loop Tracing



Given $\sigma = \{x \geq 2\}$ 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.



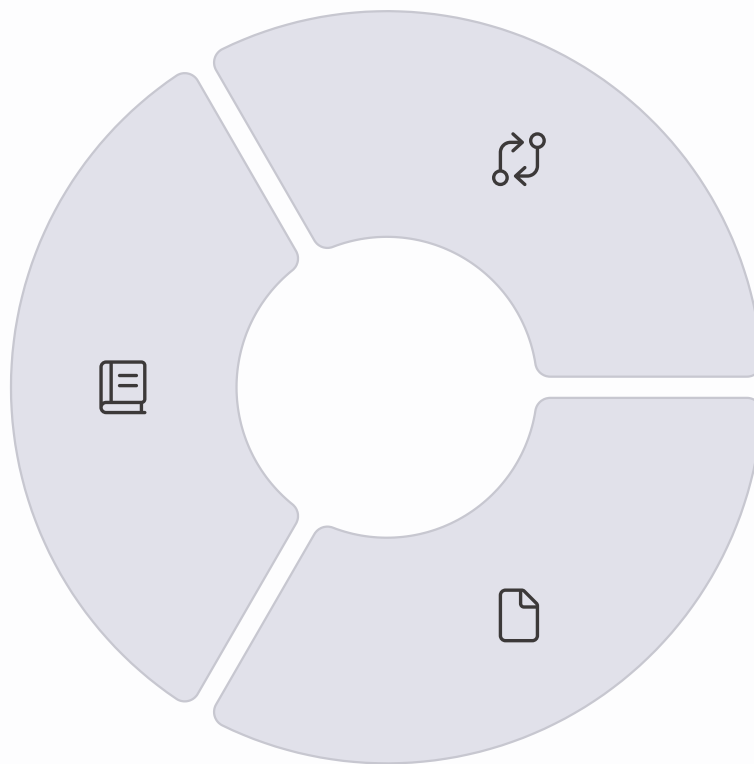
Foundation

Building blocks for everything else in this course.



What's Next

Hoare Logic
Reasoning with Pre/Post Conditions



Assignment 1
Implementing a verifier for IMP

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