

# CS 456

## Programming Languages Fall 2024

### Week 10

#### Introduction to Semantics, IMP

# Propositions

2

A **proposition** is a factual claim.

Have seen a couple of propositions:

equalities:  $0 + n = n$

implications:  $P \rightarrow Q$

universally quantified propositions: for all  $x$ ,  $P$

A **proof** is some evidence for the truth of a proposition

A **proof system** is a formalization of particular kinds of evidence.

# Propositions

3

Example:

## Proposition

- forall n, m: int, n = 0 /\ m = 0 -> n + m = 0

## Evidence

- Assume  $n = 0$  and  $m = 0$ . Substitute 0 for  $n$  and  $m$  in  $n + m = 0$ . By reflexivity of equality, the proposition is proven.

# Propositions

4

Propositions can be polymorphic and make claims about objects of arbitrary type, including functions:

Example:

```
forall A,B: Type, f: A->B,  
  forall x, y: A, f x = f y -> x = y
```

# Proofs and Judgements

5

A **judgement** is a claim of a proof system

The judgement  $\Gamma \vdash A$  is read as:  
“assuming the propositions in  $\Gamma$  are true,  $A$  is true”.

# Inference Rules

6

Proof systems construct evidence of judgements via inference rules:

Axioms

$$\overline{\Gamma \vdash \top}$$

$$\frac{A \in \Gamma}{\Gamma \vdash A}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \text{I} \rightarrow$$

$$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \text{E} \rightarrow$$

Inference Rules

# Example Proof

7

Want a proof of:

$$\vdash (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

$$\begin{array}{c} \frac{A \rightarrow (B \rightarrow C) \in \Gamma}{\Gamma \vdash A \rightarrow (B \rightarrow C)} \quad \frac{A \in \Gamma}{\Gamma \vdash A} \\ \hline \Gamma \vdash B \rightarrow C \\ \hline \Gamma = A \rightarrow (B \rightarrow C), A \rightarrow B, A \vdash C \\ \hline A \rightarrow (B \rightarrow C), A \rightarrow B \vdash A \rightarrow C \\ \hline A \rightarrow (B \rightarrow C) \vdash (A \rightarrow B) \rightarrow (A \rightarrow C) \\ \hline \vdash (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \end{array}$$

# Symbol Pushing

8

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \quad \mathbf{I} \wedge$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \quad \mathbf{E}_L \wedge$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \quad \mathbf{E}_R \wedge$$

Inference Rules for  $\wedge$



# Example

9

$$\frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C} E \vee$$

Introduction  
Rules for Or?

Inference Rules for  $\vee$

# Example

10

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \mathbf{I}_L \vee$$

$$\frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C} \quad \mathbf{E} \vee$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \quad \mathbf{I}_R \vee$$

Inference Rules for  $\vee$

# Example

11

Can you derive:  
 $\vdash A \rightarrow B \rightarrow B \wedge A$

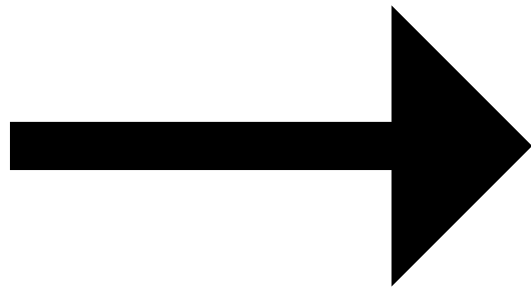
$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \quad \mathbf{I} \wedge$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \quad \mathbf{I} \rightarrow$$

$$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \quad \mathbf{E} \rightarrow$$

# Implication

12



Symbol (Math)

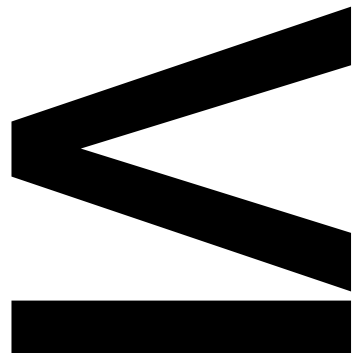
$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \mathbf{I} \rightarrow$$

$$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \mathbf{E} \rightarrow$$

Inference Rules for  $\rightarrow$

# Less Than

13



Symbol (Math)

$$n \leq m \equiv \exists k. n+k = m$$

Definition of  $\leq$

$$\frac{}{\Gamma \vdash n \leq n} \text{le}_n$$

$$\frac{\Gamma \vdash n \leq m}{\Gamma \vdash n \leq l + m} \text{les}$$

Inference Rules for  $\leq$

# Evenness

14

# EvenR

Symbol (Math)

$$\text{EvenR } n \equiv \exists k. n = k + k$$

Definition of EvenR

$$\frac{}{\Gamma \vdash \text{EvenR } 0} \quad \mathbf{ev}_0$$

$$\frac{\Gamma \vdash \text{EvenR } n}{\Gamma \vdash \text{EvenR } (2+n)} \quad \mathbf{ev}_2$$

Inference Rules for EvenR

## (OF ARITHMETIC + BOOLEAN EXPRESSIONS)

Backus-Naur Form (BNF) Definitions:

$A ::= N$   
|  $A + A$   
|  $A - A$   
|  $A * A$   
|  $X$

$X \in \text{Id}$

$N \in \mathbb{N}$

$B ::= \text{true}$   
|  $\text{false}$   
|  $A = A$   
|  $A \leq A$   
|  $\text{not } B$   
|  $B \text{ and } B$

## (OF IMP COMMANDS)

```
C ::= skip  
| X ::= A  
| C ; C  
| if B then C  
  else C end  
| while B do C end
```



# Imp Program

17

## ★ Key Feature: State\*

```
C := skip  
| x := A  
| C ; C  
| if B then C  
   else C end  
| while B do C end
```

```
X := 5;  
Z := X;  
Y := 1
```

# Imp Program

18

## ★ Key Feature: Control Flow

```
C := skip  
| x := A  
| C ; C  
| if B then C  
   else C end  
| while B do C end
```

```
X := 2;  
if (X ≤ 1)  
  then Y := 3;  
       X := 5 - Y  
  else Z := 4  
end;  
Y := 4
```

# Imp Program

19

## ★ Key Feature: Control Flow

```
C := skip  
| x := A  
| C ; C  
| if B then C  
  else C end  
| while B do C end
```

```
X := 2;  
Z := Y;  
while (0 ≤ X) do  
  X := X - 1;  
  Y := Y + Z  
end
```

# Imp Program

20

## ★ Key Feature: Control Flow

```
C := skip  
| x := A  
| C ; C  
| if B then C  
  else C end  
| while B do C end
```

```
X := 2;  
Z := Y;  
while (0 ≤ Y) do  
  X := X - 1;  
  Y := Y + Z  
end
```

# Semantics

21

```
let rec aeval (a : aexp) (st : var -> int): int =  
  match a with  
  | ANum n => n  
  | APlus a1 a2 => (aeval a1) + (aeval a2)  
  | AMinus a1 a2 => (aeval a1) - (aeval a2)  
  | AMult a1 a2 => (aeval a1) * (aeval a2)  
  | AId x => st x
```

Could equivalently have written this definition  
as a set of inference rules

# Semantics

22

```
let rec ceval (c : com) (st : var -> int) =  
  match c with  
  | Skip => st  
  | Assn x a => update st x (aeval st a)  
  | Seq c1 c2 => let st' = ceval st c1 in ceval st' c2  
  | If b c1 c2 => if (beval st b) then ceval st c1  
                  else ceval st c2  
  | While b c => st (* bogus *)
```

Not so clear what to do here: suppose the while loop does not terminate. Then, our formulation of the semantics as an interpreter won't be well-defined either

## AS A RELATION

**Key Idea:** Define evaluation as a Inductive Relation

$\text{aevalR}: \text{total\_map} \rightarrow A \rightarrow \mathbb{N} \rightarrow \text{Proposition}$

- ★ Ternary relation on states, expressions and values
- ★ Read ' $\sigma, a \Downarrow n$ ' as 'a evaluates to n in state  $\sigma$ '
- ★ Relation precisely spells out what values program can evaluate to
- ★ Put another way, rules define an 'abstract machine' for executing expression

## AS A RELATION

**Key Idea:** Define evaluation as a Inductive Relation ( $\Downarrow$ )

Inference Rules for  $\Downarrow$

$$\frac{}{\sigma, n \Downarrow n} \text{ENUM}$$

$$\frac{}{\sigma, x \Downarrow \sigma(x)} \text{EVAR}$$

$$\frac{\sigma, e_n \Downarrow v_n \quad \sigma, e_m \Downarrow v_m}{\sigma, e_n + e_m \Downarrow v_n +_{\mathbb{N}} v_m} \text{EADD}$$

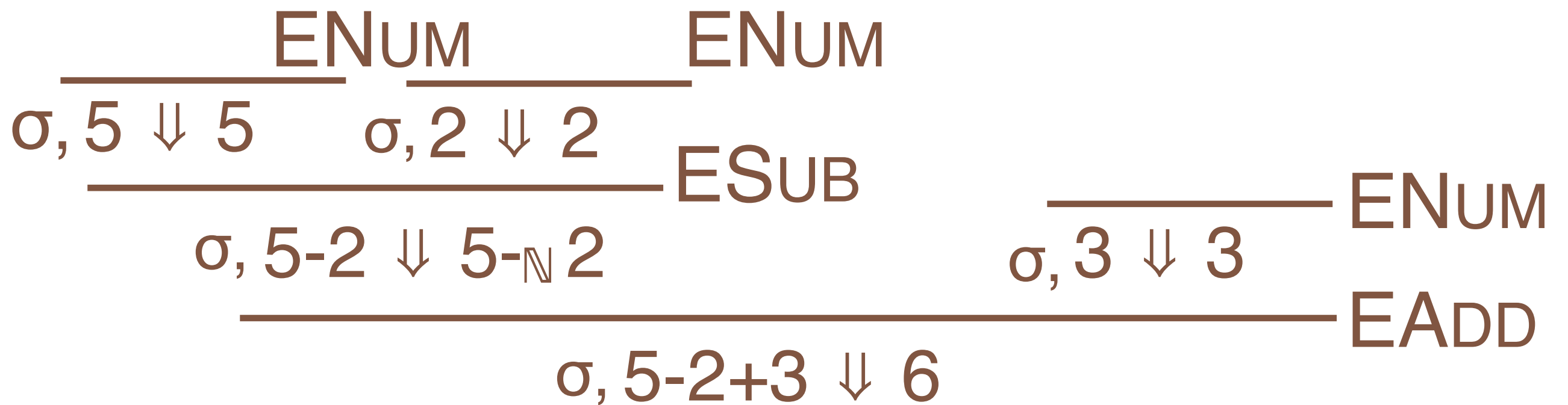
$$\frac{\sigma, e_n \Downarrow v_n \quad \sigma, e_m \Downarrow v_m}{\sigma, e_n - e_m \Downarrow v_n -_{\mathbb{N}} v_m} \text{ESUB}$$

$$\frac{\sigma, e_n \Downarrow v_n \quad \sigma, e_m \Downarrow v_m}{\sigma, e_n * e_m \Downarrow v_n *_{\mathbb{N}} v_m} \text{EMULT}$$



# Reduction

25



# Semantics

26

$\text{cevalR}: (\text{Id} \rightarrow \mathbb{N}) \rightarrow \text{C} \rightarrow (\text{Id} \rightarrow \mathbb{N}) \rightarrow \text{Proposition}$

- ★ Ternary relation on initial states, commands and final state
- ★ Read ' $\sigma, c \Downarrow \sigma'$ ' as 'when run in initial state  $\sigma$ ,  $c$  produces (i.e. evaluates to) final state  $\sigma'$ '

# Operational Semantics

27

Inference Rules for  $\Downarrow$  (commands)

$$\frac{}{\sigma, \text{skip} \Downarrow \sigma} \text{ESKIP}$$

$$\frac{\sigma, C_1 \Downarrow \sigma_1 \quad \sigma_1, C_2 \Downarrow \sigma_2}{\sigma, C_1; C_2 \Downarrow \sigma_2} \text{ESEQ}$$

$$\frac{\sigma, a \Downarrow v}{\sigma, x := a \Downarrow [x \mapsto v] \sigma} \text{EASSN}$$

# Operational Semantics

28

Inference Rules for  $\Downarrow$  (commands)

$$\frac{\sigma, b \Downarrow \mathbf{true} \qquad \sigma, c_1 \Downarrow \sigma_1}{\sigma, \mathbf{if } b \mathbf{ then } c_1 \mathbf{ else } c_2 \Downarrow \sigma_1} \text{EIFT}$$

$$\frac{\sigma, b \Downarrow \mathbf{false} \qquad \sigma, c_2 \Downarrow \sigma_1}{\sigma, \mathbf{if } b \mathbf{ then } c_1 \mathbf{ else } c_2 \Downarrow \sigma_1} \text{EIFF}$$

# Semantics

29

## Inference Rules for $\Downarrow$ (commands)

EWHILET

$$\frac{\sigma_1, b \Downarrow \text{true} \quad \sigma_1, c \Downarrow \sigma_2 \quad \sigma_2, \text{while } b \text{ do } c \text{ end} \Downarrow \sigma_3}{\sigma_1, \text{while } b \text{ do } c \text{ end} \Downarrow \sigma_3}$$

EWHILEF

$$\frac{\sigma, b \Downarrow \text{false}}{\sigma, \text{while } b \text{ do } c \text{ end} \Downarrow \sigma}$$

Why is this a better formulation than the definition of `ceval`?

# Imp Program

30

$\sigma,$

$X := 5;$
$Z := X;$
$Z := 3;$
$Y := 1$

$\Downarrow [Y \mapsto 1][Z \mapsto 3][Z \mapsto 5][X \mapsto 5]\sigma$

# Imp Program

31

$\sigma,$   $\left[ \begin{array}{l} X := 2; \\ \text{if } (X \leq 1) \\ \text{then } Y := 3 \\ \text{else } Z := 4 \\ \text{end} \end{array} \right] \Downarrow [X \mapsto 2]\sigma$

# Imp Program

32

```
 $\sigma,$   $X := 2;$   
 $Z := Y;$   
while  $(0 \leq X)$  do  
   $X := X - 1;$   
   $Y := Y + Z$   
end
```

$\Downarrow$   $[Y \mapsto \sigma(Y) + \sigma(Y) + \sigma(Y)][X \mapsto 0]$   
 $[Y \mapsto \sigma(Y) + \sigma(Y)][X \mapsto 1]$   
 $[Z \mapsto \sigma(Y)][X \mapsto 2]\sigma$




# Imp Program

33

$\sigma$ ,

```
X := 2;  
Z := Y;  
while (0 ≤ Y) do  
  X := X - 1;  
  Y := Y + Z  
end
```

⇓



# Defining IMP+FLIP

34

## 1. Syntax

```
C ::= skip
   | X := A
   | C ; C
   | if B then C
     else C end
   | while B do C end
   | if flip C
```

## 2. Semantics

$$\frac{\sigma, a \Downarrow v}{\sigma, x := a \Downarrow [x \mapsto v] \sigma} \text{EASSN}$$
$$\frac{\sigma_1, C \Downarrow \sigma_2}{\sigma_1, \text{if flip } c \Downarrow \sigma_2} \text{EFLIPT}$$
$$\frac{}{\sigma, \text{if flip } c \Downarrow \sigma} \text{EFLIPF}$$

# Concept Check

35

## Theorem [IMP+FLIP IS NOT DETERMINISTIC]:

For some commands  $c$ , from any starting state  $\sigma$ ,  $c$  can evaluate to multiple final states:

$\exists \sigma c \sigma_1 \sigma_2$ . If  $\sigma, c \Downarrow \sigma_1$  and  $\sigma, c \Downarrow \sigma_2$  and  $\sigma_1 \neq \sigma_2$ .

Can you write an IMP+Flip program that evaluates to different final states?

Can you write an IMP+Flip program that evaluates to an infinite number of final states?

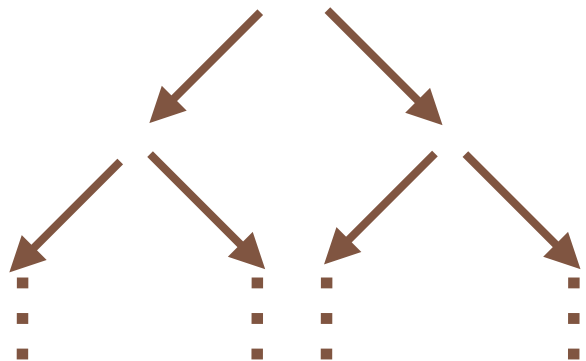
# Defining IMP + RAND

36

## 1. Syntax

```
C ::= skip
    | X := A
    | C ; C
    | if B then C
      else C end
    | while B do C end
    | X := Any
```

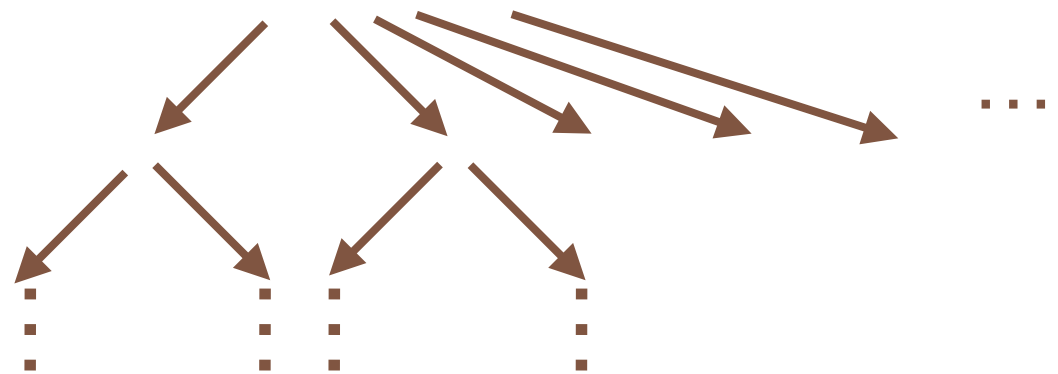
IMP+Flip: finite number of branches  
infinite final states



## 2. Semantics

$$\frac{v \in \mathbb{N}}{\sigma, x := \text{Any} \Downarrow [x \mapsto v] \sigma} \text{ERAND}$$

IMP+Rand: infinite number of branches  
infinite final states



## 1. Syntax

```
C := skip  
| X := A  
| C ; C  
| if B then C  
  else C end  
| while B do C end  
| X := F(A) —
```

$F \in \mathbb{F}$

$FD := F(X) \{C; \text{return } a\}$

```
Double(Y) {  
  skip;  
  return Y + Y}
```

```
Double(Y) {  
  Z := Y + Y;  
  return Z}
```

```
X := Double(5)
```

```
Y := 5;  
X := Double(Y)
```

# Fun IMP

38

```
C := skip
| X := A
| C ; C
| if B then C
  else C end
| while B do C end
| X := F(A)
```

$F \in \mathbb{F}$

$FD := F(X) \{C; \text{return } a\}$

- How to model set of function calls?
- Update the judgement!

$$\Delta \vdash \sigma_1, c \Downarrow \sigma_2$$
$$\Delta : \mathbb{F} \rightarrow FD$$

Read as ‘When run in initial state  $\sigma_1$  and using the function definitions in  $\Delta$ ,  $c$  produces (i.e. evaluates to) final state  $\sigma_2$ ’

# Fun IMP

39

$$\frac{\Delta \vdash \sigma, a \Downarrow v}{\Delta \vdash \sigma, x := a \Downarrow [x \mapsto v] \sigma} \text{EASSN}$$

$$\frac{\Delta(F) = F(y) \{c; \text{return } a_2\} \quad \Delta \vdash \sigma_1, \bar{a} \Downarrow \bar{v} \quad \Delta \vdash [\bar{y} \mapsto \bar{v}], c \Downarrow \sigma_2 \quad \Delta \vdash \sigma_2, a_2 \Downarrow v_2}{\Delta \vdash \sigma_1, x := F(\bar{a}) \Downarrow [x \mapsto v_2] \sigma} \text{ECALL}$$