In Impcore, a name stands for a value

Environment associates each variable with one value

Written $\rho = \{x_1 \mapsto n_1, \ldots x_k \mapsto n_k\}$,
associates variable x_i with value n_i.

Environment is finite map, aka partial function

$x \in \operatorname{dom} \rho$	**x is defined in environment ρ**
$\rho(x)$	**the value of x in environment ρ**
$\rho\{x \mapsto v\}$	**extends/modifies environment ρ to map x to v**

Find behavior using environment

Recall

```
(* y 3)    ;; what does it mean?
```

Impcore uses three environments

Global variables ξ ("ksee")

Functions ϕ ("fee")

Formal parameters ρ ("roe")

There are no local variables

- Just like `awk`; if you need temps, use extra formal parameters
- For homework, you'll add local variables

Function environment ϕ not shared with variables—just like Perl

Syntax & environments determine meaning

Initial state of abstract machine:

$$\langle e, \xi, \phi, \rho \rangle$$

State $\langle e, \xi, \phi, \rho \rangle$ is

e	**Expression being evaluated**
ξ	**Values of global variables**
ϕ	**Definitions of functions**
ρ	**Values of formal parameters**

Three environments make a basis

Meaning expressed as "Evaluation judgment"

We say

$$\langle e, \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle$$

(Big-step judgment form.)

Notes:

- ξ and ξ' may differ
- ρ and ρ' may differ
- ϕ must equal ϕ

Exercise: what do we know about globals, functions?

Primes and not primes

A prime says "something might change"
(Impcore: value of a variable)

Primes are:

- **Always in the *form of judgment*:**

  $$\langle e, \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle$$

- **Often in *judgments***
- **Usually in *rules***
- **Never in a "derivation" (computation)**

Prime implies "we don't know"

Impcore atomic form: literal

"Literal" generalizes "numeral"

$$\frac{\textsc{Literal}}{\langle \mathrm{LITERAL}(v), \xi, \phi, \rho \rangle \Downarrow \langle v, \xi, \phi, \rho \rangle}$$

We know nothing changes!

(Numeral converted to $\mathrm{LITERAL}(v)$ in parser)

Impcore atomic form: variable name

Parameters hide global variables.

FORMALVAR

$$\frac{x \in \operatorname{dom} \rho}{\langle \textsf{VAR}(x), \xi, \phi, \rho \rangle \Downarrow \langle \rho(x), \xi, \phi, \rho \rangle}$$

GLOBALVAR

$$\frac{x \notin \operatorname{dom} \rho \qquad x \in \operatorname{dom} \xi}{\langle \textsf{VAR}(x), \xi, \phi, \rho \rangle \Downarrow \langle \xi(x), \xi, \phi, \rho \rangle}$$

Impcore compound form: assignment

In SET(x, e), e is any expression

FORMALASSIGN

$$\frac{x \in \mathrm{dom}\,\rho \qquad \langle e, \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle}{\langle \mathsf{SET}(x, e), \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho'\{x \mapsto v\} \rangle}$$

GLOBALASSIGN

$$\frac{x \notin \mathrm{dom}\,\rho \qquad x \in \mathrm{dom}\,\xi \qquad \langle e, \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle}{\langle \mathsf{SET}(x, e), \xi, \phi, \rho \rangle \Downarrow \langle v, \xi'\{x \mapsto v\}, \phi, \rho' \rangle}$$

Impcore can assign only to existing variables

Semantics corresponds to code

We compose rules to make proofs

Math	*Code*
Semantics	Interpreter
Evaluation judgment	Result of evaluation
Proof of judgment	Computation of result
Rule of semantics	Case in the interpreter

Interpreter succeeds if and only if a proof exists

(Homework: result is unique!)

Implementing variables: two rules

FormalVar

$$\frac{x \in \operatorname{dom} \rho}{\langle \textsf{VAR}(x), \xi, \phi, \rho \rangle \Downarrow \langle \rho(x), \xi, \phi, \rho \rangle}$$

GlobalVar

$$\frac{x \notin \operatorname{dom} \rho \qquad x \in \operatorname{dom} \xi}{\langle \textsf{VAR}(x), \xi, \phi, \rho \rangle \Downarrow \langle \xi(x), \xi, \phi, \rho \rangle}$$

How do we tell them apart?

Implementing evaluation (`VAR`): three cases

Consult `formals` ρ or `globals` ξ:

```
switch (e->alt) {
  case VAR:
    if (isvalbound(e->u.var, formals))
      return fetchval(e->u.var, formals);
    else if (isvalbound(e->u.var, globals))
      return fetchval(e->u.var, globals);
    else
      runerror("unbound variable %n", e->u.var);
  ...
```

Why a third case?

- **When no proof, run-time error**

Application math: user-defined function

ApplyUser

$$\frac{\begin{array}{c}\phi(f) = \textsf{USER}(\langle x_1, \ldots, x_n \rangle, e) \\ x_1, \ldots, x_n \textsf{ all distinct} \\ \langle e_1, \xi_0, \phi, \rho_0 \rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1 \rangle \\ \langle e_2, \xi_1, \phi, \rho_1 \rangle \Downarrow \langle v_2, \xi_2, \phi, \rho_2 \rangle \\ \vdots \\ \langle e_n, \xi_{n-1}, \phi, \rho_{n-1} \rangle \Downarrow \langle v_n, \xi_n, \phi, \rho_n \rangle \\ \langle e, \xi_n, \phi, \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\} \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle\end{array}}{\langle \textsf{APPLY}(f, e_1, \ldots, e_n), \xi_0, \phi, \rho_0 \rangle \Downarrow \langle v, \xi', \phi, \rho_n \rangle}$$

Simpler math: function of two parameters

ApplyUser

$$\frac{\begin{array}{c}\phi(f) = \textsf{USER}(\langle x_1, x_2\rangle, e) \\ x_1, x_2 \text{ distinct} \\ \langle e_1, \xi_0, \phi, \rho_0\rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1\rangle \\ \langle e_2, \xi_1, \phi, \rho_1\rangle \Downarrow \langle v_2, \xi_2, \phi, \rho_2\rangle \\ \langle e, \xi_2, \phi, \{x_1 \mapsto v_1, x_2 \mapsto v_2\}\rangle \Downarrow \langle v, \xi', \phi, \rho'\rangle\end{array}}{\langle \textsf{APPLY}(f, e_1, e_2), \xi_0, \phi, \rho_0\rangle \Downarrow \langle v, \xi', \phi, \rho_2\rangle}$$

Evaluating function application

The math demands these steps:

- Find function in ϕ environment

```
f = fetchfun(e->u.apply.name, functions);
```

- Using caller's ρ, evaluate actuals

```
vs = evallist(e->u.apply.actuals,globals,functions,
                                 formals);
```

N.B. `actuals` evaluated in current environment

- Make a new environment: bind formals to actuals

```
new_formals = mkValenv(f.u.userdef.formals, vs);
```

- Evaluate body in new environment

```
return eval(f.u.userdef.body, globals, functions,
                              new_formals);
```

Next up: Good and bad judgments

Which judgments describe real computations?

Your turn: good and bad judgments

Which *correctly* describes what happens at run time?

1. $\langle$`(+ 2 2)`$, \xi, \phi, \rho\rangle \Downarrow \langle 4, \xi, \phi, \rho\rangle$
2. $\langle$`(+ 2 2)`$, \xi, \phi, \rho\rangle \Downarrow \langle 99, \xi, \phi, \rho\rangle$
3. $\langle$`(+ 2 2)`$, \xi, \phi, \rho\rangle \Downarrow \langle 0, \xi, \phi, \rho\rangle$
4. $\langle$`(while 1 0)`$, \xi, \phi, \rho\rangle \Downarrow \langle 77, \xi, \phi, \rho\rangle$
5. $\langle$`(begin (set n (+ n 1)) 17)`$, \xi, \phi, \rho\rangle$
   $\Downarrow \langle 17, \xi, \phi, \rho\rangle$

To know for sure, we need a proof

Judgment speaks truth when “derivable”

Special kind of proof: derivation

- It’s a data structure (derivation tree)
- Made inductively, by composing rules
- Valid derivation matches rules (by substitution)
- Spacelike representation of timelike behavior (think flip-book animation)

A form of “syntactic proof”

Recursive evaluator travels inductive proof

Root of derivation at the bottom (surprise!)

Build

- Start on the left, go up
- Cross the $\Downarrow$
- Finish on the right, go down

The "Tony Hawk" algorithm

First let's see a movie

Evaluating $(10+1) \times (10-1)$

$\langle$ `(* (+ 10 1) (- 10 1))`$, \xi, \phi, \rho\rangle \Downarrow$

Evaluating $(10 + 1) \times (10 - 1)$

$$\dfrac{\dfrac{}{\langle \texttt{(+ 10 1)}, \xi, \phi, \rho \rangle \Downarrow}}{\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho \rangle \Downarrow}$$

Evaluating $(10 + 1) \times (10 - 1)$

$$\dfrac{\dfrac{\dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow}}{\langle \texttt{(+ 10 1)}, \xi, \phi, \rho \rangle \Downarrow}}{\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho \rangle \Downarrow}$$

Evaluating $(10 + 1) \times (10 - 1)$

$$\dfrac{\dfrac{\dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow \langle \mathbf{10}, \ldots \rangle}}{\langle \texttt{(+ 10 1)}, \xi, \phi, \rho \rangle \Downarrow}}{\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho \rangle \Downarrow}$$

Evaluating $(10 + 1) \times (10 - 1)$

$$
\dfrac{
  \dfrac{
    \dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow \langle \mathbf{10}, \ldots \rangle}
    \qquad
    \dfrac{}{\langle \texttt{1}, \ldots \rangle \Downarrow}
  }{
    \langle \texttt{(+ 10 1)}, \xi, \phi, \rho \rangle \Downarrow
  }
}{
  \langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho \rangle \Downarrow
}
$$

Evaluating $(10 + 1) \times (10 - 1)$

$$
\cfrac{\cfrac{\cfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow \langle \mathbf{10}, \ldots \rangle} \qquad \cfrac{}{\langle \texttt{1}, \ldots \rangle \Downarrow \langle \mathbf{1}, \ldots \rangle}}{\langle \texttt{(+ 10 1)}, \xi, \phi, \rho \rangle \Downarrow}}{\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho \rangle \Downarrow}
$$

Evaluating $(10+1) \times (10-1)$

$$\frac{\dfrac{\dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow \langle \mathbf{10}, \ldots \rangle} \qquad \dfrac{}{\langle \texttt{1}, \ldots \rangle \Downarrow \langle \mathbf{1}, \ldots \rangle}}{\langle \texttt{(+ 10 1)}, \xi, \phi, \rho \rangle \Downarrow \langle \mathbf{11}, \xi, \phi, \rho \rangle}}{\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho \rangle \Downarrow}$$

Evaluating $(10+1)\times(10-1)$

$$\dfrac{\dfrac{\overline{\langle \texttt{10},\ldots\rangle \Downarrow \langle 10,\ldots\rangle} \qquad \overline{\langle \texttt{1},\ldots\rangle \Downarrow \langle 1,\ldots\rangle}}{\langle \texttt{(+ 10 1)}, \xi,\phi,\rho\rangle \Downarrow \langle 11,\xi,\phi,\rho\rangle} \qquad \dfrac{}{\langle \texttt{(- 10 1)},\xi,\phi,\rho\rangle \Downarrow}}{\langle \texttt{(* (+ 10 1) (- 10 1))},\xi,\phi,\rho\rangle \Downarrow}$$

Evaluating $(10 + 1) \times (10 - 1)$

$$\dfrac{\dfrac{\dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow \langle \mathbf{10}, \ldots \rangle} \qquad \dfrac{}{\langle \texttt{1}, \ldots \rangle \Downarrow \langle \mathbf{1}, \ldots \rangle}}{\langle \texttt{(+ 10 1)}, \xi, \phi, \rho \rangle \Downarrow \langle \mathbf{11}, \xi, \phi, \rho \rangle} \qquad \dfrac{\dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow}}{\langle \texttt{(- 10 1)}, \xi, \phi, \rho \rangle \Downarrow}}{\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho \rangle \Downarrow}$$

Evaluating $(10+1)\times(10-1)$

$$\dfrac{\dfrac{\dfrac{}{\langle \texttt{10}, \ldots\rangle \Downarrow \langle 10, \ldots\rangle} \qquad \dfrac{}{\langle \texttt{1}, \ldots\rangle \Downarrow \langle 1,\ldots\rangle}}{\langle \texttt{(+ 10 1)}, \xi, \phi, \rho\rangle \Downarrow \langle 11, \xi,\phi,\rho\rangle} \qquad \dfrac{\dfrac{}{\langle \texttt{10},\ldots\rangle \Downarrow \langle 10,\ldots\rangle}}{\langle \texttt{(- 10 1)}, \xi,\phi,\rho\rangle \Downarrow}}{\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi,\phi,\rho\rangle \Downarrow}$$

Evaluating $(10 + 1) \times (10 - 1)$

$$\dfrac{\dfrac{\dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow \langle 10, \ldots \rangle} \qquad \dfrac{}{\langle \texttt{1}, \ldots \rangle \Downarrow \langle 1, \ldots \rangle}}{\langle \texttt{(+ 10 1)}, \xi, \phi, \rho \rangle \Downarrow \langle 11, \xi, \phi, \rho \rangle} \qquad \dfrac{\dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow \langle 10, \ldots \rangle} \qquad \dfrac{}{\langle \texttt{1}, \ldots \rangle \Downarrow}}{\langle \texttt{(- 10 1)}, \xi, \phi, \rho \rangle \Downarrow}}{\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho \rangle \Downarrow}$$

Evaluating $(10 + 1) \times (10 - 1)$

$$
\dfrac{
\dfrac{
\dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow \langle 10, \ldots \rangle}
\qquad
\dfrac{}{\langle \texttt{1}, \ldots \rangle \Downarrow \langle 1, \ldots \rangle}
}{
\langle \texttt{(+ 10 1)}, \xi, \phi, \rho \rangle \Downarrow \langle 11, \xi, \phi, \rho \rangle
}
\qquad
\dfrac{
\dfrac{}{\langle \texttt{10}, \ldots \rangle \Downarrow \langle 10, \ldots \rangle}
\qquad
\dfrac{}{\langle \texttt{1}, \ldots \rangle \Downarrow \langle 1, \ldots \rangle}
}{
\langle \texttt{(- 10 1)}, \xi, \phi, \rho \rangle \Downarrow
}
}{
\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho \rangle \Downarrow
}
$$

Evaluating $(10 + 1) \times (10 - 1)$

$$\dfrac{\dfrac{\dfrac{}{\langle \texttt{10}, \ldots\rangle \Downarrow \langle \mathbf{10}, \ldots\rangle} \quad \dfrac{}{\langle \texttt{1}, \ldots\rangle \Downarrow \langle \mathbf{1}, \ldots\rangle}}{\langle \texttt{(+ 10 1)}, \xi, \phi, \rho\rangle \Downarrow \langle \mathbf{11}, \xi, \phi, \rho\rangle} \quad \dfrac{\dfrac{}{\langle \texttt{10}, \ldots\rangle \Downarrow \langle \mathbf{10}, \ldots\rangle} \quad \dfrac{}{\langle \texttt{1}, \ldots\rangle \Downarrow \langle \mathbf{1}, \ldots\rangle}}{\langle \texttt{(- 10 1)}, \xi, \phi, \rho\rangle \Downarrow \langle \mathbf{9}, \xi, \phi, \rho\rangle}}{\langle \texttt{(* (+ 10 1) (- 10 1))}, \xi, \phi, \rho\rangle \Downarrow}$$

Evaluating $(10+1)\times(10-1)$

$$\dfrac{\dfrac{\dfrac{}{\langle \texttt{10},\ldots\rangle \Downarrow \langle 10,\ldots\rangle} \qquad \dfrac{}{\langle \texttt{1},\ldots\rangle \Downarrow \langle 1,\ldots\rangle}}{\langle \texttt{(+ 10 1)},\xi,\phi,\rho\rangle \Downarrow \langle 11,\xi,\phi,\rho\rangle} \qquad \dfrac{\dfrac{}{\langle \texttt{10},\ldots\rangle \Downarrow \langle 10,\ldots\rangle} \qquad \dfrac{}{\langle \texttt{1},\ldots\rangle \Downarrow \langle 1,\ldots\rangle}}{\langle \texttt{(- 10 1)},\xi,\phi,\rho\rangle \Downarrow \langle 9,\xi,\phi,\rho\rangle}}{\langle \texttt{(* (+ 10 1) (- 10 1))},\xi,\phi,\rho\rangle \Downarrow \langle 99,\xi,\phi,\rho\rangle}$$

Algorithm for building derivations

Want to solve

$$\langle e, \xi, \phi, \rho \rangle \Downarrow ?$$

What rule can I use to prove it?

1. Syntactic form of e narrows to a few choices (usually 1 or 2)
2. Look for form in conclusion
3. Now check premises
4. When premise is evaluation judgment, build sub-derivation recursively

Derivation is written $\mathcal{D}$