CSCI-344
Programming Language Concepts (Section 3)

Lecture 4

ImpCore: Operational Semantics (2)

Instructor: Hossein Hojjat

August 31, 2016

Where we are

- Done:
  - Benefits of formal semantics for programming languages
    - Semantics based on mathematical models
  - Inference Systems: Inference rules and Judgments
- Now: Operational semantics of ImpCore

Refresher: Environments

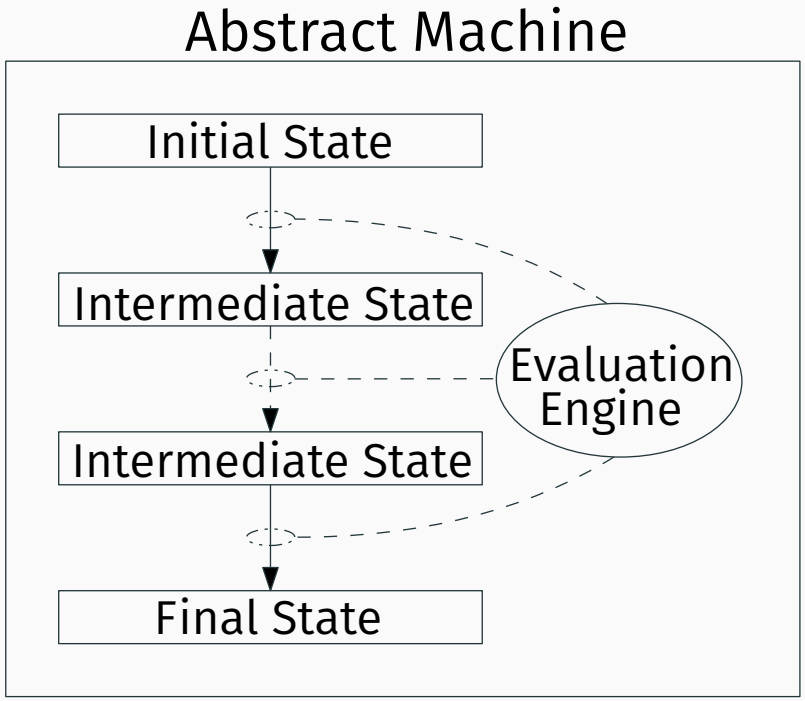
- An environment is a mapping from variables to values
  $\rho = \{x_1 \mapsto n_1, \cdots, x_k \mapsto n_k\}$
- Operations on environments:
  - $x \in dom(\rho)$: variable x is defined in environment ρ
  - $\rho(x)$: meaning of variable x in environment ρ
  - $\rho\{x \mapsto n\}$: extends/modifies environment ρ to map x to n

Refresher: Impcore Abstract Syntax

Def	=	VAL	(Name, Exp)
	|	EXP	(Exp)
	|	DEFINE	(Name, Namelist, Exp)
Exp	=	LITERAL	(Value)
	|	VAR	(Name)
	|	SET	(Name, Exp)
	|	IF	(Exp, Exp, Exp)
	|	WHILE	(Exp, Exp)
	|	BEGIN	(Explist)
	|	APPLY	(Name, Explist)

Abstract Machine

- Operational Semantics:
  how to compute a program on some abstract machine



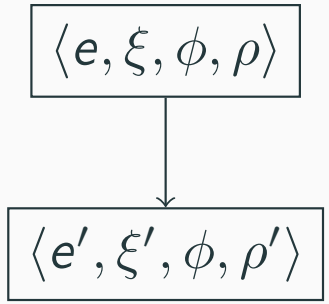
Abstract Machine

- State of the Impcore abstract machine has four parts:
  1. current definition d or expression e,
  2. environment ξ : global variables,
  3. environment ρ : formal parameters,
  4. environment ϕ : function definitions.

Abstract Machine Transitions (Expressions)

- We define judgments for transitions of the abstract machine

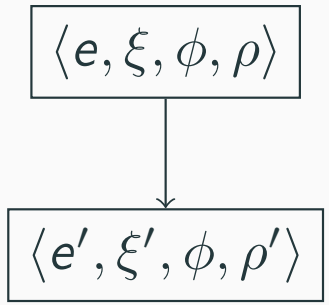
- $\langle e, \xi, \phi, \rho \rangle \Downarrow \langle e', \xi', \phi, \rho' \rangle$
- In the environments ξ, ϕ, and ρ, evaluating the expression e produces e', and it also produces new environments ξ' and ρ', while leaving ϕ unchanged



Abstract Machine Transitions (Expressions)

- We define judgments for transitions of the abstract machine

- $\langle e, \xi, \phi, \rho \rangle \Downarrow \langle e', \xi', \phi, \rho' \rangle$
- In the environments ξ, ϕ, and ρ, evaluating the expression e produces e', and it also produces new environments ξ' and ρ', while leaving ϕ unchanged



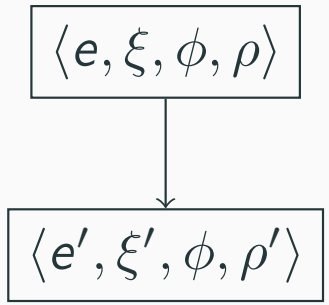
Evaluating an expression:

- always produces a value unless the machine gets stuck
  - For example evaluation of `(/ 1 0)` gets stuck
- may change the value of a global variable (from ξ) or a formal parameter (from ρ)
- never adds or changes a function definition (ϕ is unchanged)

Abstract Machine Transitions (Expressions)

- We define judgments for transitions of the abstract machine

- $\langle e, \xi, \phi, \rho \rangle \Downarrow \langle e', \xi', \phi, \rho' \rangle$
- In the environments ξ, ϕ, and ρ, evaluating the expression e produces e', and it also produces new environments ξ' and ρ', while leaving ϕ unchanged



- Valid judgment: $\langle \texttt{(+ 1 1)}, \xi, \phi, \rho \rangle \Downarrow \langle 2, \xi, \phi, \rho \rangle$
- Invalid judgment: $\langle \texttt{(+ 1 1)}, \xi, \phi, \rho \rangle \Downarrow \langle 5, \xi, \phi, \rho \rangle$
- We write inference rules to determine which judgments are valid

Impcore semantics: Literals

- Literal values evaluate to themselves without changing any environments

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

Impcore semantics: Variables

- Variables are either parameters

$$\frac{x \in \mathit{dom}(\rho)}{\langle \mathrm{VAR}(x), \xi, \phi, \rho \rangle \Downarrow \langle \rho(x), \xi, \phi, \rho \rangle} \ (\textsc{FormalVar})$$

- Or they are global variables (variable shadowing can happen)

$$\frac{x \notin \mathit{dom}(\rho) \qquad x \in \mathit{dom}(\xi)}{\langle \mathrm{VAR}(x), \xi, \phi, \rho \rangle \Downarrow \langle \xi(x), \xi, \phi, \rho \rangle} \ (\textsc{GlobalVar})$$

Impcore semantics: Assignment

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

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

Impcore semantics: Conditional

$$\frac{\langle e_1, \xi, \phi, \rho\rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1\rangle \qquad v_1 \neq 0 \qquad \langle e_2, \xi_1, \phi, \rho_1\rangle \Downarrow \langle v_2, \xi_2, \phi, \rho_2\rangle}{\langle \mathrm{IF}(e_1, e_2, e_3), \xi, \phi, \rho\rangle \Downarrow \langle v_2, \xi_2, \phi, \rho_2\rangle} \text{ (IfTrue)}$$

$$\frac{\langle e_1, \xi, \phi, \rho\rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1\rangle \qquad v_1 = 0 \qquad \langle e_3, \xi_1, \phi, \rho_1\rangle \Downarrow \langle v_3, \xi_3, \phi, \rho_3\rangle}{\langle \mathrm{IF}(e_1, e_2, e_3), \xi, \phi, \rho\rangle \Downarrow \langle v_3, \xi_3, \phi, \rho_3\rangle} \text{ (IfFalse)}$$

Impcore semantics: Iteration

- Operational semantics rules for while-loop have recursive call

$$\frac{\langle e_1, \xi, \phi, \rho\rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1\rangle \qquad v_1 \neq 0 \qquad \langle e_2, \xi_1, \phi, \rho_1\rangle \Downarrow \langle v_2, \xi_2, \phi, \rho_2\rangle \qquad \langle \text{WHILE}(e_1, e_2), \xi_2, \phi, \rho_2\rangle \Downarrow \langle v_3, \xi_3, \phi, \rho_3\rangle}{\langle \text{WHILE}(e_1, e_2), \xi, \phi, \rho\rangle \Downarrow \langle v_3, \xi_3, \phi, \rho_3\rangle} \text{ (WhileIterate)}$$

(e_2 is evaluated only for its side effects to the environments ξ and ρ)

$$\frac{\langle e_1, \xi, \phi, \rho\rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1\rangle \qquad v_1 = 0}{\langle \text{WHILE}(e_1, e_2), \xi, \phi, \rho\rangle \Downarrow \langle 0, \xi_1, \phi, \rho_1\rangle} \text{ (WhileEnd)}$$

Impcore semantics: Sequential Execution

$$\frac{}{\langle \text{BEGIN}(), \xi, \phi, \rho \rangle \Downarrow \langle 0, \xi, \phi, \rho \rangle} \ (\textsc{EmptyBegin})$$

$$\frac{\begin{array}{c} \langle e_1, \xi_0, \phi, \rho_0 \rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1 \rangle \\ \langle e_2, \xi_1, \phi, \rho_1 \rangle \Downarrow \langle v_2, \xi_2, \phi, \rho_2 \rangle \\ \vdots \\ \langle e_n, \xi_{n-1}, \phi, \rho_{n-1} \rangle \Downarrow \langle v_n, \xi_n, \phi, \rho_n \rangle \end{array}}{\langle \text{BEGIN}(e_1, e_2, \ldots, e_n), \xi_0, \phi, \rho_0 \rangle \Downarrow \langle v_n, \xi_n, \phi, \rho_n \rangle} \ (\textsc{Begin})$$

Impcore semantics: (User-defined) Function Application

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

Impcore semantics: (Primitive) Function Application

$$\frac{\begin{array}{c}\phi(f) = \text{PRIMITIVE}(+) \\ \langle e_1, \xi_0, \phi, \rho_0 \rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1 \rangle \\ \langle e_2, \xi_1, \phi, \rho_1 \rangle \Downarrow \langle v_2, \xi_2, \phi, \rho_2 \rangle\end{array}}{\langle \text{APPLY}(f, e_1, e_2), \xi_0, \phi, \rho_0 \rangle \Downarrow \langle v_1 + v_2, \xi_2, \phi, \rho_2 \rangle} \text{ (ApplyAdd)}$$

Derivation Tree Construction

- We construct a derivation tree from conclusion up, left to right
- (+ 2 3) evaluates to 5 in an environment where $\phi(+) = \text{PRIMITIVE}(+)$

$$\dfrac{\dfrac{}{\langle \text{LITERAL}(2), \xi, \phi, \rho\rangle \Downarrow \langle 2, \xi, \phi, \rho\rangle}\,(\textsc{Literal}) \quad \dfrac{}{\langle \text{LITERAL}(3), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi, \phi, \rho\rangle}\,(\textsc{Literal})}{\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle \Downarrow \langle 5, \xi, \phi, \rho\rangle}\,(\textsc{ApplyAdd})$$

To construct the derivation:

1. Start with $\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle$ on left side of bottom
2. Find applicable rule ApplyAdd and work up
3. Construct derivations for LITERAL(2) and LITERAL(3) recursively (Notice that ξ and ρ dont change.)
4. Finish with $\langle 5, \xi, \phi, \rho\rangle$ on right side of bottom

Derivation Tree Construction

- We construct a derivation tree from conclusion up, left to right
- (+ 2 3) evaluates to 5 in an environment where $\phi(+) = \text{PRIMITIVE}(+)$

$$\dfrac{\dfrac{}{\langle \text{LITERAL}(2), \xi, \phi, \rho\rangle \Downarrow \langle 2, \xi, \phi, \rho\rangle}\,(\textsc{Literal}) \quad \dfrac{}{\langle \text{LITERAL}(3), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi, \phi, \rho\rangle}\,(\textsc{Literal})}{\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle \Downarrow \langle 5, \xi, \phi, \rho\rangle}\,(\textsc{ApplyAdd})$$

To construct the derivation:

1. Start with $\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle$ on left side of bottom
2. Find applicable rule $\textsc{ApplyAdd}$ and work up
3. Construct derivations for $\text{LITERAL}(2)$ and $\text{LITERAL}(3)$ recursively (Notice that ξ and ρ dont change.)
4. Finish with $\langle 5, \xi, \phi, \rho\rangle$ on right side of bottom

Derivation Tree Construction

- We construct a derivation tree from conclusion up, left to right
- (+ 2 3) evaluates to 5 in an environment where $\phi(+) = \text{PRIMITIVE}(+)$

$$\dfrac{\dfrac{}{\langle \text{LITERAL}(2), \xi, \phi, \rho\rangle \Downarrow \langle 2, \xi, \phi, \rho\rangle}\,(\text{LITERAL}) \quad \dfrac{}{\langle \text{LITERAL}(3), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi, \phi, \rho\rangle}\,(\text{LITERAL})}{\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle \Downarrow \langle 5, \xi, \phi, \rho\rangle}\,(\text{APPLYADD})$$

To construct the derivation:

1. Start with $\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle$ on left side of bottom
2. Find applicable rule APPLYADD and work up
3. Construct derivations for LITERAL(2) and LITERAL(3) recursively (Notice that ξ and ρ dont change.)
4. Finish with $\langle 5, \xi, \phi, \rho\rangle$ on right side of bottom

Derivation Tree Construction

- We construct a derivation tree from conclusion up, left to right
- (+ 2 3) evaluates to 5 in an environment where $\phi(+) = \text{PRIMITIVE}(+)$

$$\dfrac{\dfrac{}{\langle \text{LITERAL}(2), \xi, \phi, \rho\rangle \Downarrow \langle 2, \xi, \phi, \rho\rangle}\,(\textsc{Literal}) \quad \dfrac{}{\langle \text{LITERAL}(3), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi, \phi, \rho\rangle}\,(\textsc{Literal})}{\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle \Downarrow \langle 5, \xi, \phi, \rho\rangle}\,(\textsc{ApplyAdd})$$

To construct the derivation:

1. Start with $\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle$ on left side of bottom
2. Find applicable rule $\textsc{ApplyAdd}$ and work up
3. Construct derivations for $\text{LITERAL}(2)$ and $\text{LITERAL}(3)$ recursively (Notice that ξ and ρ dont change.)
4. Finish with $\langle 5, \xi, \phi, \rho\rangle$ on right side of bottom

Derivation Tree Construction

- We construct a derivation tree from conclusion up, left to right
- (+ 2 3) evaluates to 5 in an environment where $\phi(+) = \text{PRIMITIVE}(+)$

$$\dfrac{\dfrac{}{\langle \text{LITERAL}(2), \xi, \phi, \rho\rangle \Downarrow \langle 2, \xi, \phi, \rho\rangle}\,(\textsc{Literal}) \quad \dfrac{}{\langle \text{LITERAL}(3), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi, \phi, \rho\rangle}\,(\textsc{Literal})}{\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle \Downarrow \langle 5, \xi, \phi, \rho\rangle}\,(\textsc{ApplyAdd})$$

To construct the derivation:

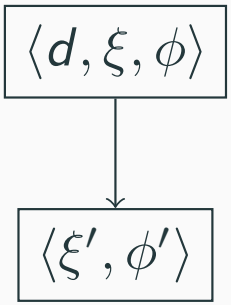
1. Start with $\langle \text{APPLY}(+, \text{LITERAL}(2), \text{LITERAL}(3)), \xi, \phi, \rho\rangle$ on left side of bottom
2. Find applicable rule $\textsc{ApplyAdd}$ and work up
3. Construct derivations for $\text{LITERAL}(2)$ and $\text{LITERAL}(3)$ recursively (Notice that ξ and ρ dont change.)
4. Finish with $\langle 5, \xi, \phi, \rho\rangle$ on right side of bottom

Exercise

- If $\rho = \{x \mapsto 2, y \mapsto 1\}$ then prove `(* x (+ y 1))` evaluates to 4

Abstract Machine Transitions (Definitions)

- $\langle d, \xi, \phi \rangle \rightarrow \langle \xi', \phi' \rangle$
- In the environments ξ and ϕ evaluating the definition d yields new environments ξ' and ϕ'



Impcore semantics: Variable Definition

$$\frac{\langle e, \xi, \phi, \{\}\rangle \Downarrow \langle v, \xi', \phi, \rho'\rangle}{\langle \mathrm{VAL}(x, e), \xi, \phi\rangle \to \langle \xi'\{x \mapsto v\}, \phi\rangle} \text{ (DefineGlobal)}$$

Impcore semantics: Function Definition

$$\frac{x_1, \ldots, x_n \text{ all distinct}}{\langle \textsc{define}(f, \langle x_1, \ldots, x_n \rangle, e), \xi, \phi \rangle \to \langle \xi, \phi\{f \mapsto \textsc{user}(\langle x_1, \ldots, x_n \rangle, e)\} \rangle} \quad (\textsc{DefineFunction})$$

Impcore semantics: Top-level Expression

$$\frac{\langle e, \xi, \phi, \{\} \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle}{\langle \mathrm{EXP}(e), \xi, \phi \rangle \to \langle \xi'\{\mathsf{it} \mapsto v\}, \phi \rangle} \ (\textsc{EvalExp})$$

Summary

This Lecture

- We discussed the operational semantics of ImpCore
- How to compute an Impcore program as sequences of transitions of an abstract machine

Next Lecture

- Theory and Metatheory