CSCI-344
Programming Language Concepts (Section 3)

Lecture 5

Theory and Meta-theory

Instructor: Hossein Hojjat

September 2, 2016

Where we are

- Done:
  - ImpCore concrete syntax
  - ImpCore abstract syntax
  - ImpCore operational semantics
- Now: Theory and Metatheory

ImpCore Inference Rules

Operational Semantics Game:

Apply inference rules to derive transitions that a program can perform

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

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

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

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

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

ImpCore Inference Rules

Operational Semantics Game:

Apply inference rules to derive transitions that a program can perform

If $\rho = \{x \mapsto 3, y \mapsto 2\}$ then `(+ (* x x) y)` evaluates to 11

$$\langle \text{APPLY}(+, \text{APPLY}(*, \text{VAR}(x), \text{VAR}(x)), \text{VAR}(y)), \xi, \phi, \rho \rangle \Downarrow \langle 11, \xi, \phi, \rho \rangle$$

$$\textsc{ApplyAdd}\ \dfrac{\textsc{ApplyMul}\ \dfrac{\textsc{FormalVar}\ \dfrac{x \in dom(\rho) \qquad \rho(x) = 3}{\langle \text{VAR}(x), \xi, \phi, \rho \rangle \Downarrow \langle 3, \xi, \phi, \rho \rangle} \quad \textsc{FormalVar}\ \dfrac{x \in dom(\rho) \qquad \rho(x) = 3}{\langle \text{VAR}(x), \xi, \phi, \rho \rangle \Downarrow \langle 3, \xi, \phi, \rho \rangle}}{\langle \text{APPLY}(*, \text{VAR}(x), \text{VAR}(x)), \xi, \phi, \rho \rangle \Downarrow \langle 9, \xi, \phi, \rho \rangle} \quad \dfrac{y \in dom(\rho) \qquad \rho(y) = 2}{\langle \text{VAR}(y), \xi, \phi, \rho \rangle \Downarrow \langle 2, \xi, \phi, \rho \rangle}\ \textsc{FormalVar}}{\langle \text{APPLY}(+, \text{APPLY}(*, \text{VAR}(x), \text{VAR}(x)), \text{VAR}(y)), \xi, \phi, \rho \rangle \Downarrow \langle 11, \xi, \phi, \rho \rangle}$$

ImpCore Inference Rules

Operational Semantics Game:

> Apply inference rules to derive transitions that a program can perform

Theorem:

If $\rho = \{x \mapsto 3, y \mapsto 2\}$ then `(+ (* x x) y)` evaluates to 11

$$\langle \text{APPLY}(+, \text{APPLY}(*, \text{VAR}(x), \text{VAR}(x)), \text{VAR}(y)), \xi, \phi, \rho\rangle \Downarrow \langle 11, \xi, \phi, \rho\rangle$$

Proof:

$$\textsc{ApplyAdd}\ \dfrac{\textsc{ApplyMul}\ \dfrac{\textsc{FormalVar}\ \dfrac{x \in dom(\rho) \qquad \rho(x) = 3}{\langle \text{VAR}(x), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi, \phi, \rho\rangle} \qquad \textsc{FormalVar}\ \dfrac{x \in dom(\rho) \qquad \rho(x) = 3}{\langle \text{VAR}(x), \xi, \phi, \rho\rangle \Downarrow \langle 3, \xi, \phi, \rho\rangle}}{\langle \text{APPLY}(*, \text{VAR}(x), \text{VAR}(x)), \xi, \phi, \rho\rangle \Downarrow \langle 9, \xi, \phi, \rho\rangle} \qquad \dfrac{y \in dom(\rho) \qquad \rho(y) = 2}{\langle \text{VAR}(y), \xi, \phi, \rho\rangle \Downarrow \langle 2, \xi, \phi, \rho\rangle}\ \textsc{FormalVar}}{\langle \text{APPLY}(+, \text{APPLY}(*, \text{VAR}(x), \text{VAR}(x)), \text{VAR}(y)), \xi, \phi, \rho\rangle \Downarrow \langle 11, \xi, \phi, \rho\rangle}$$

ImpCore Inference Rules

Operational Semantics Game:

> Apply inference rules to derive transitions that a program can perform

Meta-Theory:

- Looks into the operational semantics from an **outside** view of the game
- Asks questions about derivations themselves
- Example:
  - **Claim:** It is impossible to add/remove global variables when evaluating an expression in ImpCore
  - **Proof:** Show in all derivation trees for expression evaluation it is impossible to add/remove global variables
- Meta-Theory: a theory the subject matter of which is another theory (Encyclopædia Britannica)

Why Meta-Theory?

- Meta-theory: Reasoning about general **program-independent** properties
- Helps language designers to show non-trivial properties of their languages
  - Does evaluating an expression always terminate with a value?
- Helps compiler (interpreter) implementation
  - Is it OK to put environments on a stack?
    - (Exercise 22 - Chapter 2)
  - Optimization: Is it OK to replace `(if x x 0)` with `x`?
    - (Exercise 13,14 - Chapter 2)
- Guides programmers

Examples

If $\langle e, \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle$ and $\langle e, \xi, \phi, \rho \rangle \Downarrow \langle v', \xi'', \phi, \rho'' \rangle$, then $v = v'$ and $\xi' = \xi''$ and $\rho' = \rho''$.

Examples

If $\langle e, \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle$ and $\langle e, \xi, \phi, \rho \rangle \Downarrow \langle v', \xi'', \phi, \rho'' \rangle$, then $v = v'$ and $\xi' = \xi''$ and $\rho' = \rho''$.

- ImpCore expression evaluation is **deterministic**
- If we run an ImpCore program multiple times, we will always get the same answer
- If we run an ImpCore program under multiple interpreters, we should always get the same answer
- Non-deterministic behavior: concurrent programs where global variables are not shared between multiple processes

Examples

If $\langle e, \xi, \phi, \rho \rangle \Downarrow \langle v', \xi', \phi, \rho' \rangle$,
then $dom(\xi) = dom(\xi')$ and $dom(\rho) = dom(\rho')$.

Examples

If $\langle e, \xi, \phi, \rho \rangle \Downarrow \langle v', \xi', \phi, \rho' \rangle$,
then $dom(\xi) = dom(\xi')$ and $dom(\rho) = dom(\rho')$.

- ImpCore expression evaluation doesn't change the domain of the global and formal environments

Examples

If $\langle d, \xi, \phi \rangle \rightarrow \langle \xi', \phi' \rangle$,
then $dom(\xi) \subseteq dom(\xi')$ and $dom(\phi) \subseteq dom(\phi')$.

Examples

If $\langle d, \xi, \phi \rangle \rightarrow \langle \xi', \phi' \rangle$,
then $dom(\xi) \subseteq dom(\xi')$ and $dom(\phi) \subseteq dom(\phi')$.

- ImpCore definition evaluation doesn't remove the previous global variable and function definitions

Meta-theoretical Proofs

- How to prove a claim about any derivation?

Meta-theoretical Proofs

- How to prove a claim about any derivation?
- Derivation tree is a **recursive** data structure:

$$\mathcal{D}: \quad \dfrac{\mathcal{D}_1 : \dfrac{\vdots}{h_1}\,(R') \qquad \mathcal{D}_2 : \dfrac{\vdots}{h_2}\,(R'')}{c}\,(R)$$

- c is the conclusion of the last rule (R) and the whole derivation $(\mathcal{D})$
- Premises h_1 and h_2 are also derived in subderivations D_1 and D_2 with conclusions h_1 and h_2
- Axioms (inference rules with no premises) do not decompose into subderivations

Meta-theoretical Proofs

- How to prove a claim about any derivation?
- Derivation tree is a **recursive** data structure:

$$\mathcal{D}: \quad \dfrac{\mathcal{D}_1: \dfrac{\vdots}{h_1}\,(R') \qquad \mathcal{D}_2: \dfrac{\vdots}{h_2}\,(R'')}{c}\,(R)$$

- c is the conclusion of the last rule (R) and the whole derivation $(\mathcal{D})$
- Premises h_1 and h_2 are also derived in subderivations D_1 and D_2 with conclusions h_1 and h_2
- Axioms (inference rules with no premises) do not decompose into subderivations
- A meta-theoretical proof uses **structural induction** on the derivations

Structural Induction

To prove that a property $\mathcal{P}$ holds for all derivation $\mathcal{D}$ we need to confirm two conditions:

1. **Basis:** For every axiom A the property must hold: $\mathcal{P}(A)$
2. **Induction step:** For each rule of the form:

$$\frac{h_1 \cdots h_n}{c}$$

- Prove that any derivation $\mathcal{D}$ ending with an instance of this rule satisfies $\mathcal{P}$
- $\mathcal{D}$ has sub-derivations $\mathcal{D}_1, \cdots, \mathcal{D}_n$ with conclusions h_1,...,h_n
- Assume $\mathcal{P}$ holds for each of these sub-derivations as inductive hypothesis

$$\mathcal{P}(\mathcal{D}_1) ,..., \mathcal{P}(\mathcal{D}_n)$$

This concludes $\forall \mathcal{D}.\ (\mathcal{D} \text{ valid}) \rightarrow \mathcal{P}(D)$

Example: Natural Numbers

- Proof a property $\mathcal{P}$ holds for all natural numbers

$$\textsc{Zero}\quad \frac{}{0 \in \mathbb{N}} \qquad\qquad \textsc{Succ}\quad \frac{n \in \mathbb{N}}{n + 1 \in \mathbb{N}}$$

- For $\textsc{Zero}$, prove $\mathcal{P}(0)$,
- For $\textsc{Succ}$, assume $\mathcal{P}(n)$, prove $\mathcal{P}(n+1)$

Conclude $\forall n \in \mathbb{N} : \mathcal{P}(n)$.

Exercise

- Additive expression are defined with the following abstract syntax:

$$E = n \mid \mathrm{SUM}(E, E) \qquad (n \in \mathbb{N})$$

- Judgment $E \Downarrow n$ means E evaluates to n
- Inference rules:

$$\frac{}{n \Downarrow n}\ (\textsc{Num}) \qquad \frac{E_1 \Downarrow n_1 \qquad E_2 \Downarrow n_2}{\mathrm{SUM}(E_1, E_2) \Downarrow n_1 + n_2}\ (\textsc{Add})$$

- Prove that evaluation is deterministic in additive expressions: for any expression E, if $E \Downarrow n$ and $E \Downarrow n'$ then $n = n'$.

Summary

- Theory involves proofs about individual derivations
    - Evaluation of a single program
- Metatheory involves proofs about collections of derivations
    - Evaluation of entire classes of programs (or all programs)