# Chapter 11: Languages with Higher Types

Hongfei Fu

John Hopcroft Center for Computer Science
Department of Computer Science
Shanghai Jiao Tong University

# Previous Lecture

## Domain Theory

- construction of complete partial orders (cpo's)
- construction of continuous functions
- construction of a metalanguage

# Today's Topic

## Typed Languages

- functional programming languages
- operational semantics
- denotational semantics

# Today's Topic

## Typed Languages

- functional programming languages
- operational semantics
- denotational semantics

## Main Theorem

- agreement of operational and denotational semantics

# A Basic Functional Language

# A Basic Functional Language

## Types

The types $\tau$ are generated from the grammar:

$$\tau ::= \textbf{int} \mid \tau_1 * \tau_2 \mid \tau_1 \to \tau_2$$

- **int**: the basic type for integers
- $\tau_1 * \tau_2$: product type for ordered pairs
  (e.g., type **int** $*$ **int** for integer pairs $(0, 1), (-1, 2), \ldots$)
- $\tau_1 \to \tau_2$: function type from $\tau_1$ to $\tau_2$
  (e.g., type **int** $\to$ **int** for functions from integers to integers)

# A Basic Functional Language

## Variables

- **Var** $= \{x, y, \dots\}$: a set of variables
- **type**$(x)$: the uniquely-fixed type for the variable $x$

We write $x : \tau$ to stress that **type**$(x) = \tau$.

# A Basic Functional Language

## Terms

The terms $t$ are generated from the grammar:

$$
\begin{array}{rcl}
t & ::= & x \text{ (variables)} \\
& | & n \text{ (integer constants)} \\
& | & t_1 \bowtie t_2 \ (\bowtie \in \{=, -, \times\}) \text{ (arithmetic operations)} \\
& | & \textbf{if } t_0 \textbf{ then } t_1 \textbf{ else } t_2 \\
& | & (t_1, t_2) \text{ (ordered pairs)} \\
& | & \textbf{fst}(t) \text{ (first entry of ordered pairs)} \\
& | & \textbf{snd}(t) \text{ (second entry of ordered pairs)} \\
& | & \lambda x.t \ (\lambda\text{-abstraction}) \\
& | & (t_1 \ t_2) \text{ (function application } t_1(t_2)) \\
& | & \textbf{let } x \Leftarrow t_1 \textbf{ in } t_2 \text{ (let-notation } t_2\,[t_1/x]) \\
& | & \textbf{rec}y.(\lambda x.t) \text{ (recursion)}
\end{array}
$$

# Terms

## "Legal" Terms

- $\lambda x.(x + 1)$
- $(\lambda x.(x + 1), 2)$
- $\mathbf{rec} y.(\lambda x.\mathbf{if}\ x\ \mathbf{then}\ 1\ \mathbf{else}\ x \times (y\ (x - 1)))$

Types:

- $x : \mathbf{int}$
- $y : \mathbf{int} \to \mathbf{int}$

# Terms

## "Illegal" Terms

- $(\lambda x.(x)) + 1$
- $(\lambda x.(x)) + (\lambda x.(x + 1))$

Types:

- $x$ : **int**

# Typing Rules

## Variables

$$\frac{\textbf{type}(x) = \tau}{x : \tau}$$

# Typing Rules

## Arithmetic Operations

$$\frac{n \in \mathbb{Z}}{n : \textbf{int}} \qquad \frac{t_1 : \textbf{int},\ t_2 : \textbf{int}}{t_1\ \mathrm{op}\ t_2 : \textbf{int}\ (\mathrm{op} \in \{+, -, \times\})}$$

# Typing Rules

## Conditional Branch

$$\frac{t_0 : \textbf{int},\ \ t_1 : \tau,\ \ t_2 : \tau}{\textbf{if}\ t_0\ \textbf{then}\ t_1\ \textbf{else}\ t_2 : \tau}$$

# Typing Rules

## Products

$$\frac{t_1 : \tau,\ \ t_2 : \tau}{(t_1, t_2) : \tau_1 * \tau_2} \qquad \frac{t : \tau_1 * \tau_2}{\mathbf{fst}(t) : \tau_1} \qquad \frac{t : \tau_1 * \tau_2}{\mathbf{snd}(t) : \tau_2}$$

# Typing Rules

## Functions

$$\frac{x : \tau',\ t : \tau}{\lambda x.t : \tau' \to \tau} \qquad \frac{t_1 : \tau' \to \tau,\ t_2 : \tau'}{(t_1\ t_2) : \tau}$$

# Typing Rules

## "Let" Notation

$$\frac{x : \tau_1,\ \ t_1 : \tau_1,\ \ t_2 : \tau_2}{\textbf{let}\ x \Leftarrow t_1\ \textbf{in}\ t_2 : \tau_2}$$

# Typing Rules

## Recursion

$$\frac{y : \tau,\ \lambda x.t : \tau}{\mathbf{rec}y.(\lambda x.t) : \tau}$$

# Typing Rules

## Typable Terms

- A term $t$ is *typable* if $t : \tau$ for some type $\tau$.
- A term $t$ is *uniquely typable* if $t : \tau$ for some unique type $\tau$. (i.e., $t : \tau_1$ and $t : \tau_2$ implies $\tau_1 = \tau_2$)

# Typing Rules

## Typable Terms

- A term $t$ is *typable* if $t : \tau$ for some type $\tau$.
- A term $t$ is *uniquely typable* if $t : \tau$ for some unique type $\tau$. (i.e., $t : \tau_1$ and $t : \tau_2$ implies $\tau_1 = \tau_2$)

## A Simple Exercise

Every typable term is uniquely typable.

# Terms

## Free Variables

- $FV(n) := \emptyset$;
- $FV(x) := \{x\}$;
- $FV(t_1 \text{ op } t_2) := FV(t_1) \cup FV(t_2)$;
- $FV(\textbf{if } t_0 \textbf{ then } t_1 \textbf{ else } t_2) := FV(t_0) \cup FV(t_1) \cup FV(t_2)$;
- $FV((t_1,\ t_2)) = FV((t_1\ t_2)) := FV(t_1) \cup FV(t_2)$;
- $FV(\textbf{fst}(t)) = FV(\textbf{snd}(t)) := FV(t)$;
- $FV(\lambda x.t) := FV(t) \setminus \{x\}$;
- $FV(\textbf{let } x \Leftarrow t_1 \textbf{ in } t_2) := FV(t_1) \cup (FV(t_2)\, x)$;
- $FV(\textbf{rec}y.(\lambda x.t)) := FV(\lambda x.t) \setminus \{y\}$.

# Terms

## Closed Terms

- A term $t$ is *closed* if $FV(t) = \emptyset$.

# Terms

## Substitution

- $t$: a term
- $t'$: a closed term
- $x$: a free variable in $t$

Then we have

- $t\,[t'/x]$: the term obtained from substituting all free occurrences of $x$ by $t'$ in $t$
- $t[t_1/x_1, \ldots, t_k/x_k]$: the term obtained from substituting all free occurrences of $x_i$ by $t_i$ $(1 \leq i \leq k)$ in $t$

# Terms

## Example

- $x : \mathbf{int}$
- $t = \mathbf{let}\ x \Leftarrow x\ \mathbf{in}\ x + 1$
- $t\,[4/x] = \mathit{let}\ x \Leftarrow 4.x + 1$

# Eager Operational Semantics

# Eager Operational Semantics

## Eager Semantics

- $t = (3 + 1, (\lambda x.(x + 1)\ 4))$

How can we evaluate **fst**$(t)$:

- first we evaluate both $3 + 1$ and $(\lambda x.(x + 1)\ 4)$;
- then the final result is the evaluation from $3 + 1$.

# Eager Operational Semantics

## Eager Semantics

- $t_1 = \lambda x.1$;
- $t_2 = (\mathbf{rec} y.(\lambda x.(y\ x))\ 4)$;
- $t = (t_1\ t_2)$

How can we evaluate $t$:

- first we evaluate both $t_1$ and $t_2$;
- then the final result is the function application.

# Eager Operational Semantics

## Canonical Forms

- $\tau$: a type

The set $C^{\mathfrak{e}}_{\tau}$ is recursively defined as follows:

- $C^{\mathfrak{e}}_{\mathbf{int}} := \mathbb{Z}$;
- $C^{\mathfrak{e}}_{\tau_1 * \tau_2} := C^{\mathfrak{e}}_{\tau_1} \times C^{\mathfrak{e}}_{\tau_2}$;
- $C^{\mathfrak{e}}_{\tau_1 \to \tau_2} := \{\lambda x.t \mid \lambda x.t : \tau_1 \to \tau_2 \text{ is closed.}\}$

# Eager Operational Semantics

## Canonical Forms

- $\tau$: a type

The set $C^{\mathfrak{e}}_{\tau}$ is recursively defined as follows:

- $C^{\mathfrak{e}}_{\mathbf{int}} := \mathbb{Z}$;
- $C^{\mathfrak{e}}_{\tau_1 * \tau_2} := C^{\mathfrak{e}}_{\tau_1} \times C^{\mathfrak{e}}_{\tau_2}$;
- $C^{\mathfrak{e}}_{\tau_1 \to \tau_2} := \{\lambda x.t \mid \lambda x.t : \tau_1 \to \tau_2 \text{ is closed.}\}$

## Exercise

Prove that for any type $\tau$ and term $t \in C^{\mathfrak{e}}_{\tau}$, $t$ is closed.

# Eager Operational Semantics

## The Evaluation Relation

- $t$: a closed term with type $\tau$
- $c$: a canonical term in $C_\tau^{\mathfrak{e}}$

Then

- $t \to^{\mathfrak{e}} c$: $t$ evaluates to $c$ in eager operational semantics

# Eager Operational Semantics

## Evaluation Rules

- canonical terms:

$$\frac{c \in C_\tau^{\mathfrak{e}}}{c \rightarrow^{\mathfrak{e}} c}$$

# Eager Operational Semantics

## Evaluation Rules

- arithmetic operations:

$$\frac{t_1 \to^{\mathfrak{e}} n_1, \;\; t_2 \to^{\mathfrak{e}} n_2}{t_1 \;\mathrm{op}\; t_2 \to^{\mathfrak{e}} n_1 \;\mathrm{op}\; n_2} \quad \mathrm{op} \in \{+, -, \times\}$$

# Eager Operational Semantics

## Evaluation Rules

- conditional branch:

$$\frac{t_0 \to^{\mathfrak{e}} 0 \;,\; t_1 \to^{\mathfrak{e}} c_1}{\textbf{if } t_0 \textbf{ then } t_1 \textbf{ else } t_2 \to^{\mathfrak{e}} c_1}$$

$$\frac{t_0 \to^{\mathfrak{e}} n \;,\; t_2 \to^{\mathfrak{e}} c_2}{\textbf{if } t_0 \textbf{ then } t_1 \textbf{ else } t_2 \to^{\mathfrak{e}} c_2} \; (n \neq 0)$$

# Eager Operational Semantics

## Evaluation Rules

- Product:

$$\frac{t_1 \to^{e} c_1 \ , t_2 \to^{e} c_2}{(t_1, t_2) \to^{e} (c_1, c_2)}$$

$$\frac{t \to^{e} (c_1, c_2)}{\mathbf{fst}(t) \to^{e} c_1} \qquad \frac{t \to^{e} (c_1, c_2)}{\mathbf{snd}(t) \to^{e} c_2}$$

# Eager Operational Semantics

## Evaluation Rules

- Function Application:

$$\frac{t_1 \rightarrow^{e} \lambda x.t_1' \ , t_2 \rightarrow^{e} c_2, \ \ t_1'\,[c_2/x] \rightarrow^{e} c}{(t_1 \ t_2) \rightarrow^{e} c}$$

# Eager Operational Semantics

## Evaluation Rules

- "Let" Notation:

$$\frac{t_1 \rightarrow^{\mathfrak{e}} c_1 \ , t_2\,[c_1/x] \rightarrow^{\mathfrak{e}} c_2}{\textbf{let}\ x \Leftarrow t_1\ \textbf{in}\ t_2 \rightarrow^{\mathfrak{e}} c_2}$$

# Eager Operational Semantics

## Evaluation Rules

- Recursion:

$$\frac{}{\mathbf{rec}y.(\lambda x.t) \rightarrow^{\mathfrak{e}} \lambda x.(t\,[\mathbf{rec}y.(\lambda x.t)/y])}$$

# Eager Operational Semantics

## Proposition

- $t$: a closed term with type $\tau$
- $c, c'$: canonical terms

Then we have that:

- if $t \to^{\mathfrak{e}} c$ and $t \to^{\mathfrak{e}} c'$ then $c = c'$;
- if $t \to^{\mathfrak{e}} c$ then $c : \tau$.

# Eager Operational Semantics

## Proposition

- $t$: a closed term with type $\tau$
- $c, c'$: canonical terms

Then we have that:

- if $t \to^{\mathfrak{e}} c$ and $t \to^{\mathfrak{e}} c'$ then $c = c'$;
- if $t \to^{\mathfrak{e}} c$ then $c : \tau$.

# Eager Operational Semantics

## Proposition

- $t$: a closed term with type $\tau$
- $c, c'$: canonical terms

Then we have that:

- if $t \to^{\mathfrak{e}} c$ and $t \to^{\mathfrak{e}} c'$ then $c = c'$;
- if $t \to^{\mathfrak{e}} c$ then $c : \tau$.

## Proof.

By a simple rule induction. □

# Eager Operational Semantics

## Exercise

- $\mathit{fact} := \mathbf{rec} y.(\lambda x.\,(\mathbf{if}\ x\ \mathbf{then}\ 1\ \mathbf{else}\ x \times (y\ (x-1))))$
- How can one evaluate $(\mathit{fact}\ 2)$?

# Eager Denotational Semantics

# Eager Denotational Semantics

## An Overview

- terms as values (e.g., integers, pairs, functions)
- complete partial orders (cpo's) as mathematical backbone

# Eager Denotational Semantics

## Values

- $\tau$: a type

The discrete cpo $V^{\mathbf{e}}_{\tau}$ of values associated with type $\tau$ is recursively defined as follows:

- $V^{\mathbf{e}}_{\mathbf{int}} := \mathbb{Z}$;
- $V^{\mathbf{e}}_{\tau_1 * \tau_2} := V^{\mathbf{e}}_{\tau_1} \times V^{\mathbf{e}}_{\tau_2}$;
- $V^{\mathbf{e}}_{\tau_1 \to \tau_2} := [V^{\mathbf{e}}_{\tau_1} \to (V^{\mathbf{e}}_{\tau_2})_{\perp}]$

# Eager Denotational Semantics

## Values

- $\tau$: a type

The discrete cpo $V^{\mathbf{e}}_{\tau}$ of values associated with type $\tau$ is recursively defined as follows:

- $V^{\mathbf{e}}_{\mathbf{int}} := \mathbb{Z}$;
- $V^{\mathbf{e}}_{\tau_1 * \tau_2} := V^{\mathbf{e}}_{\tau_1} \times V^{\mathbf{e}}_{\tau_2}$;
- $V^{\mathbf{e}}_{\tau_1 \to \tau_2} := [V^{\mathbf{e}}_{\tau_1} \to (V^{\mathbf{e}}_{\tau_2})_{\perp}]$

## Question

Why do we incorporate $\perp$ in the last definition?

# Eager Denotational Semantics

## Environments

- **Var**: the set of variables

An *environment* $\rho$ is a function

$$\rho : \mathbf{Var} \to \bigcup \{ V_\tau^{\mathfrak{e}} \mid \tau \text{ a type} \}$$

such that

$$\forall x \in \mathbf{Var}.(x : \tau \Rightarrow \rho(x) \in V_\tau^{\mathfrak{e}}) \ .$$

# Eager Denotational Semantics

## Environments

- **Var**: the set of variables

An *environment* $\rho$ is a function

$$\rho : \mathbf{Var} \to \bigcup \{ V_\tau^{\mathfrak{e}} \mid \tau \text{ a type} \}$$

such that

$$\forall x \in \mathbf{Var}.(x : \tau \Rightarrow \rho(x) \in V_\tau^{\mathfrak{e}}) \ .$$

We denote by $\mathbf{Env}^{\mathfrak{e}}$ the set of environments under eager semantics.

# Eager Denotational Semantics

## Environments

- **Var**: the set of variables

An *environment* $\rho$ is a function

$$\rho : \mathbf{Var} \to \bigcup \{ V_\tau^{\mathfrak{e}} \mid \tau \text{ a type} \}$$

such that

$$\forall x \in \mathbf{Var}.(x : \tau \Rightarrow \rho(x) \in V_\tau^{\mathfrak{e}}) \ .$$

# Eager Denotational Semantics

## Environments

- **Var**: the set of variables

An *environment* $\rho$ is a function

$$\rho : \mathbf{Var} \to \bigcup \{ V_\tau^{\mathfrak{e}} \mid \tau \text{ a type} \}$$

such that

$$\forall x \in \mathbf{Var}.(x : \tau \Rightarrow \rho(x) \in V_\tau^{\mathfrak{e}}) \ .$$

We denote by $\mathbf{Env}^{\mathfrak{e}}$ the set of environments under eager semantics.

# Eager Denotational Semantics

## Intuition

- $t$: a typable term with type $\tau$
- $[\![t]\!]^{\mathfrak{e}} : \mathbf{Env}^{\mathfrak{e}} \to (V_\tau^{\mathfrak{e}})_\perp$: the denotational semantics of $t$

# Eager Denotational Semantics

## Inductive Definition

- $[\![x]\!]^{\mathfrak{e}} := \lambda\rho.\lfloor\rho(x)\rfloor$;
- $[\![n]\!]^{\mathfrak{e}} := \lambda\rho.\lfloor n\rfloor$;

# Eager Denotational Semantics

## Inductive Definition

- $[\![t_1 \ \mathbf{op} \ t_2]\!]^{\mathfrak{e}} := \lambda\rho.([\![t_1]\!]^{\mathfrak{e}}(\rho) \ \mathbf{op}_{\perp} \ [\![t_2]\!]^{\mathfrak{e}}(\rho))$;
- $$[\![\mathbf{if} \ t_0 \ \mathbf{then} \ t_1 \ \mathbf{else} \ t_2]\!]^{\mathfrak{e}} := \lambda\rho.\mathit{Cond}([\![t_0]\!]^{\mathfrak{e}}(\rho), [\![t_1]\!]^{\mathfrak{e}}(\rho), [\![t_2]\!]^{\mathfrak{e}}(\rho));$$

# Eager Denotational Semantics

## Inductive Definition

- $[\![(t_1, t_2)]\!]^{\mathfrak{e}} := \lambda\rho.\mathit{let}\ v_1 \Leftarrow [\![t_1]\!]^{\mathfrak{e}}(\rho), v_2 \Leftarrow [\![t_2]\!]^{\mathfrak{e}}(\rho).\lfloor(v_1, v_2)\rfloor$;
- $[\![\mathbf{fst}(t)]\!]^{\mathfrak{e}} := \lambda\rho.\mathit{let}\ v \Leftarrow [\![t]\!]^{\mathfrak{e}}(\rho).\lfloor\pi_1(v)\rfloor$
- $[\![\mathbf{snd}(t)]\!]^{\mathfrak{e}} := \lambda\rho.\mathit{let}\ v \Leftarrow [\![t]\!]^{\mathfrak{e}}(\rho).\lfloor\pi_2(v)\rfloor$

# Eager Denotational Semantics

## Inductive Definition

- $[\![\lambda x.t]\!]^{\mathfrak{e}} := \lambda\rho.\lfloor\lambda v \in V^{\mathfrak{e}}_{\tau_1}.[\![t]\!]^{\mathfrak{e}}(\rho\,[v/x])\rfloor$ for $\lambda x.t : \tau_1 \to \tau_2$
- $[\![(t_1\ t_2)]\!]^{\mathfrak{e}} := \lambda\rho.\mathit{let}\ F \Leftarrow [\![t_1]\!]^{\mathfrak{e}}(\rho), v \Leftarrow [\![t_2]\!]^{\mathfrak{e}}(\rho).F(v)$

# Eager Denotational Semantics

## Inductive Definition

- $[\![\textbf{let } x \Leftarrow t_1 \textbf{ in } t_2]\!]^{\mathfrak{e}} := \lambda\rho.let\ v \Leftarrow [\![t_1]\!]^{\mathfrak{e}}(\rho).[\![t_2]\!]^{\mathfrak{e}}(\rho\,[v/x])$
- $[\![\textbf{rec}y.(\lambda x.t)]\!]^{\mathfrak{e}} := \lambda\rho.\lfloor\mu F.(\lambda v.[\![t]\!]^{\mathfrak{e}}(\rho[v/x, F/y]))\rfloor$

# Eager Denotational Semantics

## Lemma

- $t$: a typable term
- $\rho, \rho'$: two environments such that for all $x \in FV(t)$, we have $\rho(x) = \rho'(x)$

Then we have $[\![t]\!]^{\mathfrak{e}}(\rho) = [\![t]\!]^{\mathfrak{e}}(\rho')$.

## Proof.

A simple structural induction on $t$. $\square$

# Eager Denotational Semantics

## Substitution Lemma

- $s$: a typable closed term with type $\tau$
- $x$: a variable with type $\tau$
- $t$: a typable term with type $\tau'$

If $[\![s]\!]^{\mathfrak{e}}(\rho) = \lfloor v \rfloor$, then we have that:

- $t\,[s/x] : \tau'$;
- $[\![t\,[s/x]]\!]^{\mathfrak{e}}(\rho) = [\![t]\!]^{\mathfrak{e}}(\rho\,[v/x])$.

## Proof.

By structural induction on $t$. $\square$

# Eager Denotational Semantics

## Lemma

- If $t : \tau$, then for all $\rho$ we have $[\![t]\!]^{\mathfrak{e}}(\rho) \in (V_\tau^{\mathfrak{e}})_\perp$.
- If $c \in C_\tau^{\mathfrak{e}}$, then for all $\rho$ we have $[\![c]\!]^{\mathfrak{e}}(\rho) \neq \perp$ (the bottom element of $(V_\tau^{\mathfrak{e}})_\perp$).

# Agreement of Eager Semantics

# Agreement of Eager Semantics

## A First Statement

- $t$: a closed typable term
- $c$: a canonical term

Then we may expect that $t \to^{\mathfrak{e}} c$ iff $[\![t]\!]^{\mathfrak{e}}(\rho) = [\![c]\!]^{\mathfrak{e}}(\rho)$.

# Agreement of Eager Semantics

## A Problem

- $t$: a closed typable term
- $c$: a canonical term

$[\![t]\!]^{\mathfrak{e}}(\rho) = [\![c]\!]^{\mathfrak{e}}(\rho) \Rightarrow t \rightarrow^{\mathfrak{e}} c$ may not hold.

# Agreement of Eager Semantics

## The Correct Theorem

- $t$: a closed typable term
- $c$: a canonical term

Then we have:

- $t \to^{\mathfrak{e}} c$ implies $[\![t]\!]^{\mathfrak{e}}(\rho) = [\![c]\!]^{\mathfrak{e}}(\rho)$;
- The two eager semantics agree on the convergence of $t$.

# Agreement of Eager Semantics

## Operational Convergence

- $t$: a typable closed term

We say that $t$ is *operationally convergent*, denoted by $t{\downarrow^{\mathfrak{e}}}$, if it holds that $\exists c.t{\rightarrow^{\mathfrak{e}}}c$.

# Agreement of Eager Semantics

## Denotational Convergence

- $t$: a typable closed term with type $\tau$

We say that $t$ is *denotationally convergent*, denoted by $t{\Downarrow}^{\mathfrak{e}}$, if it holds that $\exists v \in V_{\tau}^{\mathfrak{e}}.[\![t]\!]^{\mathfrak{e}}(\rho) = \lfloor v \rfloor$.

# Agreement of Eager Semantics

## The Correct Theorem

- $t$: a closed typable term
- $c$: a canonical term

Then we have:

- $t \to^{\mathfrak{e}} c$ implies $[\![t]\!]^{\mathfrak{e}}(\rho) = [\![c]\!]^{\mathfrak{e}}(\rho)$;
- $t{\downarrow}^{\mathfrak{e}}$ iff $t{\Downarrow}^{\mathfrak{e}}$.

# Agreement of Eager Semantics

## A Corollary

- $t$: a closed typable term with type **int**

Then we have that $t \rightarrow^{\mathfrak{e}} n$ iff $[\![t]\!]^{\mathfrak{e}}(\rho) = \lfloor n \rfloor$.

# Agreement of Eager Semantics

## The Correct Theorem

- $t$: a closed typable term
- $c$: a canonical term

Then we have:

- $t \to^{\mathfrak{e}} c$ implies $[\![t]\!]^{\mathfrak{e}}(\rho) = [\![c]\!]^{\mathfrak{e}}(\rho)$;
- $t\downarrow^{\mathfrak{e}}$ iff $t\Downarrow^{\mathfrak{e}}$.

## Main Task

How can we prove this theorem?

# Agreement of Eager Semantics

## Lemma

- $t$: a closed typable term
- $c$: a canonical term

If $t \to^{\mathfrak{e}} c$ then $[\![t]\!]^{\mathfrak{e}}(\rho) = [\![c]\!]^{\mathfrak{e}}(\rho)$ (for any environment $\rho$).

## Proof.

By rule induction on evaluation of terms. $\square$

# Agreement of Eager Semantics

## The Rule for **fst**$(-)$

$$\frac{t \to^{\mathfrak{e}} (c_1,\ c_2)}{\mathbf{fst}(t) \to^{\mathfrak{e}} c_1}$$

- $[\![t]\!]^{\mathfrak{e}}(\rho) = [\![(c_1,\ c_2)]\!]^{\mathfrak{e}}(\rho)$;
- $[\![(c_1,\ c_2)]\!]^{\mathfrak{e}}(\rho) = \mathit{let}\ v_1 \Leftarrow [\![c_1]\!]^{\mathfrak{e}}(\rho), v_2 \Leftarrow [\![c_2]\!]^{\mathfrak{e}}(\rho).\lfloor (v_1, v_2) \rfloor$;
- $(c_1,\ c_2)\Downarrow^{\mathfrak{e}}$;
- $v_1 = [\![c_1]\!]^{\mathfrak{e}}(\rho)$ and $v_2 = [\![c_2]\!]^{\mathfrak{e}}(\rho)$;
- $[\![\mathbf{fst}(t)]\!]^{\mathfrak{e}}(\rho) = \mathit{let}\ v \Leftarrow [\![t]\!]^{\mathfrak{e}}(\rho).\lfloor \pi_1(v) \rfloor = \lfloor v_1 \rfloor = [\![c_1]\!]^{\mathfrak{e}}(\rho)$.

# Agreement of Eager Semantics

## The Rule for Function Application

$$\frac{t_1 \rightarrow^{\mathfrak{e}} \lambda x.t_1',\ t_2 \rightarrow^{\mathfrak{e}} c_2,\ t_1'\,[c_2/x] \rightarrow^{\mathfrak{e}} c}{(t_1\ t_2) \rightarrow^{\mathfrak{e}} c}$$

- $[\![t_1]\!]^{\mathfrak{e}}(\rho) = [\![\lambda x.t_1']\!]^{\mathfrak{e}}(\rho)$, $[\![t_2]\!]^{\mathfrak{e}}(\rho) = [\![c_2]\!]^{\mathfrak{e}}(\rho)$ and $[\![t_1'\,[c_2/x]]\!]^{\mathfrak{e}}(\rho) = [\![c]\!]^{\mathfrak{e}}(\rho)$;

$$\begin{aligned}
[\![(t_1\ t_2)]\!]^{\mathfrak{e}}(\rho) &= \mathit{let}\ F \Leftarrow [\![t_1]\!]^{\mathfrak{e}}(\rho), v \Leftarrow [\![t_2]\!]^{\mathfrak{e}}(\rho).F(v) \\
&= \mathit{let}\ F \Leftarrow [\![\lambda x.t_1']\!]^{\mathfrak{e}}(\rho), v \Leftarrow [\![t_2]\!]^{\mathfrak{e}}(\rho).F(v) \\
&= \mathit{let}\ F \Leftarrow \lfloor \lambda v.[\![t_1']\!]^{\mathfrak{e}}(\rho\,[v/x]) \rfloor, v \Leftarrow [\![c_2]\!]^{\mathfrak{e}}(\rho).F(v) \\
&= [\![t_1']\!]^{\mathfrak{e}}(\rho\,[v/x]) \text{ where } \lfloor v \rfloor = [\![c_2]\!]^{\mathfrak{e}}(\rho) \\
&= [\![t_1'\,[c_2/x]]\!]^{\mathfrak{e}}(\rho) \\
&= [\![c]\!]^{\mathfrak{e}}(\rho)
\end{aligned}$$

# Agreement of Eager Semantics

## The Rule for Recursion

$$\frac{}{\mathbf{rec}y.(\lambda x.t) \to^{\mathfrak{e}} \lambda x.(t\,[\mathbf{rec}y.(\lambda x.t)/y])}$$

- $[\![\mathbf{rec}y.(\lambda x.t)]\!]^{\mathfrak{e}}(\rho) = \lfloor F^* \rfloor$;
- $F^* = \mu F.(\lambda v.[\![t]\!]^{\mathfrak{e}}(\rho[v/x, F/y]))$;

$$\begin{aligned}
[\![\lambda x.t\,[\mathbf{rec}y.(\lambda x.t)/y]]\!]^{\mathfrak{e}}(\rho) &= [\![\lambda x.t]\!]^{\mathfrak{e}}(\rho\,[F^*/y]) \\
&= \lfloor \lambda v.[\![t]\!]^{\mathfrak{e}}(\rho[v/x, F^*/y]) \rfloor \\
&= \lfloor F^* \rfloor \\
&= [\![\mathbf{rec}y.(\lambda x.t)]\!]^{\mathfrak{e}}(\rho)
\end{aligned}$$

# Agreement of Eager Semantics

## Lemma

- $t$: a closed typable term
- $c$: a canonical term

If $t \to^{e} c$ then $[\![t]\!]^{e}(\rho) = [\![c]\!]^{e}(\rho)$ (for any environment $\rho$).

## Corollary

- $t$: a closed typable term

Then we have that $t\downarrow^{e} \Rightarrow t\Downarrow^{e}$.

# Agreement of Eager Semantics

## The Difficult Part

- $t$: a closed typable term

Then we have that $t{\Downarrow^{\mathfrak{e}}} \Rightarrow t{\downarrow^{\mathfrak{e}}}$.

## Proof.

The First Attempt: structural induction on $t$ □

# Agreement of Eager Semantics

## The Case $t = (t_1\ t_2)$

- induction hypothesis: $(t_1{\Downarrow^e} \Rightarrow t_1{\downarrow^e})\ \&\ (t_2{\Downarrow^e} \Rightarrow t_2{\downarrow^e})$

Then

- $t{\Downarrow^e}$
- $[\![t]\!]^e(\rho) = \mathit{let}\ F \Leftarrow [\![t_1]\!]^e(\rho), v \Leftarrow [\![t_2]\!]^e(\rho).F(v)$
- $t_1{\Downarrow^e}\ \&\ t_2{\Downarrow^e}$
- $t_1{\rightarrow^e}\lambda x.t_1'\ \&\ t_2{\rightarrow^e}c_2$
- $[\![t]\!]^e(\rho) = F(v)$ where $F = \lambda u.[\![t_1']\!]^e(\rho\,[u/x])$ and $\lfloor v \rfloor = [\![c_2]\!]^e(\rho)$
- $[\![t]\!]^e(\rho) = [\![t_1']\!]^e(\rho\,[v/x]) = [\![t_1'\,[c_2/x]]\!]^e(\rho)$
- $t_1'\,[c_2/x]{\downarrow^e}$ and $t_1'\,[c_2/x]{\rightarrow^e}c$
- $t{\rightarrow^e}c$

# Agreement of Eager Semantics

## Question

What's wrong with the proof?

# Agreement of Eager Semantics

## Question

What's wrong with the proof?

## Proof

- ...
- $t_1 \to^{\mathfrak{e}} \lambda x.t_1'$ & $t_2 \to^{\mathfrak{e}} c_2$
- ...
- $[\![t]\!]^{\mathfrak{e}}(\rho) = [\![t_1']\!]^{\mathfrak{e}}(\rho\,[v/x]) = [\![t_1'\,[c_2/x]]\!]^{\mathfrak{e}}(\rho)$
- $t_1'\,[c_2/x]{\downarrow^{\mathfrak{e}}}$ and $t_1'\,[c_2/x] \to^{\mathfrak{e}} c$ ??
- $t \to^{\mathfrak{e}} c$

# Agreement of Eager Semantics

## Question

What's wrong with the proof?

## The Problem

- It is not guaranteed that $t_1' [c_2/x]\Downarrow^{\mathfrak{e}} \Rightarrow t_1' [c_2/x]\downarrow^{\mathfrak{e}}$ .

# Agreement of Eager Semantics

## Solution

- a stronger induction hypothesis
- a *logical relation* between values and types

# Agreement of Eager Semantics

## Logical Relations

- $\tau$: a type

Then we will define:

- $\lesssim^{\circ}_{\tau} \subseteq V^{\mathfrak{e}}_{\tau} \times C^{\mathfrak{e}}_{\tau}$
- $\lesssim_{\tau} \subseteq (V^{\mathfrak{e}}_{\tau})_{\perp} \times \mathit{ClosedTerms}$

# Agreement of Eager Semantics

## The Relation $\lesssim_\tau$

- $\tau$: a type

We define the relation $\lesssim_\tau \subseteq (V_\tau^{\mathfrak{e}})_\perp \times \mathit{ClosedTerms}$ by:

$$d \lesssim_\tau t \text{ iff } \forall v \in V_\tau^{\mathfrak{e}}.\, [d = \lfloor v \rfloor \Rightarrow (\exists c.(t \rightarrow^{\mathfrak{e}} c \;\&\; v \lesssim_\tau^{\circ} c))]$$

# Agreement of Eager Semantics

## The Relations $\lesssim^{\circ}_{\tau} \subseteq V^{\mathfrak{e}}_{\tau} \times C^{\mathfrak{e}}_{\tau}$

- ground types:

$$n \lesssim^{\circ}_{\mathbf{int}} n \text{ for all integers } n$$

- product types:

$$(v_1, v_2) \lesssim^{\circ}_{\tau_1 * \tau_2} (c_1, c_2) \text{ iff } v_1 \lesssim^{\circ}_{\tau_1} c_1 \text{ and } v_2 \lesssim^{\circ}_{\tau_2} c_2$$

- function types:

$$F \lesssim^{\circ}_{\tau_1 \to \tau_2} \lambda x.t \text{ iff } \forall v \in V^{\mathfrak{e}}_{\tau_1}, c \in C^{\mathfrak{e}}_{\tau_2}. v \lesssim^{\circ}_{\tau_1} c \Rightarrow F(v) \lesssim_{\tau_2} t\,[c/x]$$

# Agreement of Eager Semantics

## Lemma

- $t$: a closed term with type $\tau$

We have that

- $\bot_{(V_\tau^{\mathfrak{e}})_\bot} \lesssim_\tau t$;
- for any $d, d' \in (V_\tau^{\mathfrak{e}})_\bot$, it holds that

$$\left(d \sqsubseteq d' \;\&\; d' \lesssim_\tau t \Rightarrow d \lesssim_\tau t\right);$$

- for any $d_0 \sqsubseteq d_1 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \ldots$ in $(V_\tau^{\mathfrak{e}})_\bot$, it holds that

$$(\forall n. d_n \lesssim_\tau t) \Rightarrow \bigsqcup_{n\in\omega} d_n \lesssim_\tau t.$$

# Agreement of Eager Semantics

## Lemma

- $\bot_{(V_\tau^{\mathfrak{e}})_\bot} \lesssim_\tau t$;

## Proof

By definition:

$$d \lesssim_\tau t \text{ iff } \forall v \in V_\tau^{\mathfrak{e}}.\, [d = \lfloor v \rfloor \Rightarrow (\exists c.(t \rightarrow^{\mathfrak{e}} c \;\&\; v \lesssim_\tau^{\circ} c))]$$

# Agreement of Eager Semantics

## Lemma

- for any $d_0 \sqsubseteq d_1 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \ldots$ in $(V_\tau^{\mathfrak{e}})_\perp$, it holds that

$$(\forall n. d_n \lesssim_\tau t) \Rightarrow \bigsqcup_{n \in \omega} d_n \lesssim_\tau t.$$

## Proof (Structural Induction on Types)

base type: $\tau = \textbf{int}$. Straightforward.

# Agreement of Eager Semantics

## Lemma

- for any $d_0 \sqsubseteq d_1 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \ldots$ in $(V^{\mathfrak{e}}_{\tau})_{\perp}$, it holds that

$$(\forall n. d_n \lesssim_{\tau} t) \Rightarrow \bigsqcup_{n \in \omega} d_n \lesssim_{\tau} t.$$

## Proof (Structural Induction on Types)

function types: $\tau = \tau_1 \to \tau_2$.

- Suppose $d_0 \sqsubseteq d_1 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \ldots$ in $(V^{\mathfrak{e}}_{\tau_1 \to \tau_2})_{\perp}$.
- Suppose that $\forall n. d_n \lesssim_{\tau_1 \to \tau_2} t$.
- easy case: $\forall n. d_n = \perp$

# Agreement of Eager Semantics

## Lemma

- for any $d_0 \sqsubseteq d_1 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \ldots$ in $(V^{\mathfrak{e}}_{\tau})_{\perp}$, it holds that

$$(\forall n.d_n \lesssim_{\tau} t) \Rightarrow \bigsqcup_{n\in\omega} d_n \lesssim_{\tau} t.$$

## Proof (Structural Induction on Types)

function types: $\tau = \tau_1 \to \tau_2$.

- Suppose that $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$ in $(V^{\mathfrak{e}}_{\tau_1\to\tau_2})_{\perp}$ and $\forall n.d_n \lesssim_{\tau_1\to\tau_2} t$.
- nontrivial case: $\exists n.d_n \neq \perp$ and $t \to^{\mathfrak{e}} \lambda x.t'$.
- from definition: $\forall n \geq N.(d_n = \lfloor F_n \rfloor \;\&\; F_n \lesssim^{\circ}_{\tau_1\to\tau_2} \lambda x.t')$
- $\forall n \geq N.\forall (v,c) \in V^{\mathfrak{e}}_{\tau_1} \times C^{\mathfrak{e}}_{\tau_1}.v \lesssim^{\circ}_{\tau_1} c \Rightarrow F_n(v) \lesssim_{\tau_2} t'\,[c/x]$
- induction hypothesis: $(\bigsqcup_n F_n)(v) = \bigsqcup_n (F_n(v)) \lesssim_{\tau_2} t'\,[c/x]$
- $\forall (v,c).v \lesssim^{\circ}_{\tau_1} c \Rightarrow (\bigsqcup_n F_n)(v) \lesssim_{\tau_2} t'\,[c/x]$
- $\bigsqcup_{n\in\omega} F_n \lesssim^{\circ}_{\tau_1\to\tau_2} \lambda x.t'$
- $\bigsqcup_{n\in\omega} d_n = \lfloor \bigsqcup_{n\in\omega} F_n \rfloor \lesssim_{\tau_1\to\tau_2} t$

# Agreement of Eager Semantics

## Lemma

- $t$: a typable close term

We have that $t{\Downarrow^{\mathfrak{e}}} \Rightarrow t{\downarrow^{\mathfrak{e}}}$.

## Proof (by Structural Induction)

We prove by structural induction on terms that:

- $t : \tau$: a term
- $x_1 : \tau_1, \ldots, x_k : \tau_k$: free variables in $t$
- $s_1 : \tau_1, \ldots, s_k : \tau_k$: closed terms
- $v_i \in V^{\mathfrak{e}}_{\tau_i}$ $(1 \leq i \leq k)$: elements such that $\lfloor v_i \rfloor \lesssim_{\tau_i} s_i$

Then $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_{\tau} t[s_1/x_1, \ldots, s_k/x_k]$.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

We prove by structural induction on terms that:

- $t : \tau$: a term
- $x_1 : \tau_1, \ldots, x_k : \tau_k$: free variables in $t$
- $s_1 : \tau_1, \ldots, s_k : \tau_k$: closed terms
- $v_i \in V^{\mathfrak{e}}_{\tau_i}$ $(1 \leq i \leq k)$: elements such that $\lfloor v_i \rfloor \lesssim_{\tau_i} s_i$

Then $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_{\tau} t[s_1/x_1, \ldots, s_k/x_k]$.

## Corollary

- $t : \tau$: a closed term
- $t{\Downarrow^{\mathfrak{e}}}$
- $[\![t]\!]^{\mathfrak{e}}(\rho) = \lfloor v \rfloor$
- $\exists c.(t{\rightarrow^{\mathfrak{e}}}c \;\&\; v \lesssim^{\circ}_{\tau} c)$
- $t{\downarrow^{\mathfrak{e}}}$

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $t : \tau$: a term
- $x_1 : \tau_1, \ldots, x_k : \tau_k$: free variables in $t$
- $s_1 : \tau_1, \ldots, s_k : \tau_k$: closed terms
- $v_i \in V^{\mathfrak{e}}_{\tau_i}$ $(1 \leq i \leq k)$: elements such that $\lfloor v_i \rfloor \lesssim_{\tau_i} s_i$

Then $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_{\tau} t[s_1/x_1, \ldots, s_k/x_k]$.

## Base Step: $t = x$ and $x : \tau$

- Suppose $\lfloor v \rfloor \lesssim_{\tau} s$.
- $[\![x]\!]^{\mathfrak{e}}(\rho\,[v/x]) = \lfloor v \rfloor \lesssim_{\tau} s = x\,[s/x]$.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $t : \tau$: a term
- $x_1 : \tau_1, \ldots, x_k : \tau_k$: free variables in $t$
- $s_1 : \tau_1, \ldots, s_k : \tau_k$: closed terms
- $v_i \in V^{\mathfrak{e}}_{\tau_i}$ $(1 \leq i \leq k)$: elements such that $\lfloor v_i \rfloor \lesssim_{\tau_i} s_i$

Then $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_{\tau} t[s_1/x_1, \ldots, s_k/x_k]$.

## Base Step: $t = n$

- $n \lesssim^{\circ}_{\mathbf{int}} n$.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = t_1$ **op** $t_2$

- Suppose that $[\![t_1 \textbf{ op } t_2]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor n \rfloor$.
- Then
  - $[\![t_1]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor n_1 \rfloor$.
  - $[\![t_2]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor n_2 \rfloor$.
  - $n = n_1$ **op** $n_2$.
- By induction hypothesis,
  - $\lfloor n_1 \rfloor \lesssim_{\textbf{int}} t_1[v_1/x_1, \ldots, v_k/x_k]$
  - $\lfloor n_2 \rfloor \lesssim_{\textbf{int}} t_2[v_1/x_1, \ldots, v_k/x_k]$

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = t_1$ **op** $t_2$

- By induction hypothesis,
  - $\lfloor n_1 \rfloor \lesssim_{\mathbf{int}} t_1[v_1/x_1, \ldots, v_k/x_k]$
  - $\lfloor n_2 \rfloor \lesssim_{\mathbf{int}} t_2[v_1/x_1, \ldots, v_k/x_k]$
- From the definition of $\lesssim_{\mathbf{int}}$,
  - $t_1[v_1/x_1, \ldots, v_k/x_k] \rightarrow^{\mathfrak{e}} n_1$
  - $t_2[v_1/x_1, \ldots, v_k/x_k] \rightarrow^{\mathfrak{e}} n_2$
- Hence, $t[v_1/x_1, \ldots, v_k/x_k] \rightarrow^{\mathfrak{e}} n$.
- Finally, $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t =$ **if** $t_0$ **then** $t_1$ **else** $t_2$

- Suppose that $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u \rfloor$.
- Then either
  - $[\![t_0]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor 0 \rfloor$.
  - $[\![t_1]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u_1 \rfloor$.

  or
  - $[\![t_0]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor n \rfloor$ $(n > 0)$.
  - $[\![t_2]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u_2 \rfloor$.
- ...

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t =$ **if** $t_0$ **then** $t_1$ **else** $t_2$

- Suppose that $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u \rfloor$.
- Then either
  - $[\![t_0]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor 0 \rfloor$.
  - $[\![t_1]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u_1 \rfloor$.

  or
  - $[\![t_0]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor n \rfloor$ $(n > 0)$.
  - $[\![t_2]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u_2 \rfloor$.
- ...

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = (t_1, t_2)$

- Suppose $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u \rfloor$ and $t_1 : \tau_1, t_2 : \tau_2$.
- Then
  - $[\![t_1]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u_1 \rfloor$.
  - $[\![t_2]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u_2 \rfloor$.
  - $u = (u_1, u_2)$.
- By induction hypothesis,
  - $\lfloor u_1 \rfloor \lesssim_{\tau_1} t_1[s_1/x_1, \ldots, s_k/x_k]$;
  - $\lfloor u_2 \rfloor \lesssim_{\tau_2} t_2[s_1/x_1, \ldots, s_k/x_k]$.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = (t_1, t_2)$

- By induction hypothesis,
  - $\lfloor u_1 \rfloor \lesssim_{\tau_1} t_1[s_1/x_1, \ldots, s_k/x_k]$;
  - $\lfloor u_2 \rfloor \lesssim_{\tau_2} t_2[s_1/x_1, \ldots, s_k/x_k]$.
- By definition,
  - $t_1[s_1/x_1, \ldots, s_k/x_k] \rightarrow^{\mathfrak{e}} c_1$ and $u_1 \lesssim^{\circ}_{\tau_1} c_1$;
  - $t_2[s_1/x_1, \ldots, s_k/x_k] \rightarrow^{\mathfrak{e}} c_2$ and $u_2 \lesssim^{\circ}_{\tau_2} c_2$.
- Then $(u_1, u_2) \lesssim^{\circ}_{\tau} (c_1, c_2)$ and $t[s_1/x_1, \ldots, s_k/x_k] \rightarrow^{\mathfrak{e}} (c_1, c_2)$.
- Finally, $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_{\tau} t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = \mathbf{fst}(t')$

- Suppose that $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u \rfloor$ and $t' : \tau_1 * \tau_2$;
- Then $[\![t']\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor (u_1, u_2) \rfloor$ and $u = u_1$;
- By induction hypothesis, $\lfloor (u_1, u_2) \rfloor \lesssim_{\tau_1 * \tau_2} t'[s_1/x_1, \ldots, s_k/x_k]$;
- Hence, $t'[s_1/x_1, \ldots, s_k/x_k] \rightarrow^{\mathfrak{e}} (c_1, c_2)$ and $(u_1, u_2) \lesssim^{\circ}_{\tau_1 * \tau_2} (c_1, c_2)$;
- Then, $t[s_1/x_1, \ldots, s_k/x_k] \rightarrow^{\mathfrak{e}} c_1$ and $u_1 \lesssim^{\circ}_{\tau_1} c_1$;
- Finally, $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_{\tau} t[s_1/x_1, \ldots, s_k/x_k]$.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = \mathbf{snd}(t')$

By similar proof.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = \lambda x.t'$ with $x : \tau$, $t' : \tau'$

- $[\![\lambda x.t']\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor F \rfloor$.
- Then $F = \lambda v \in V_\tau^{\mathfrak{e}}.[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k, v/x])$
- For any $v, c$ such that $v \lesssim^\circ_\tau c$, $\lfloor v \rfloor \lesssim_\tau c$ and $F(v) = [\![t']\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k, v/x])$
- By induction hypothesis,

$$[\![t']\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k, v/x]) \lesssim_{\tau'} t'[s_1/x_1, \ldots, s_k/x_k, c/x]$$

- $F \lesssim^\circ_{\tau\to\tau'} \lambda x.t'[s_1/x_1, \ldots, s_k/x_k]$
- $\lfloor F \rfloor \lesssim^\circ_{\tau\to\tau'} t[s_1/x_1, \ldots, s_k/x_k]$

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_{\tau} t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = (t_1\ t_2)$ with $t_1 : \tau' \to \tau,\ t_2 : \tau'$

- Suppose that $[\![(t_1\ t_2)]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor u \rfloor$.
- Then we have
  - $[\![t_1]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor F \rfloor$.
  - $[\![t_2]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor v \rfloor$.
  - $F(v) = \lfloor u \rfloor$.
- By induction hypothesis,
  - $\lfloor F \rfloor \lesssim_{\tau' \to \tau} t_1[s_1/x_1, \ldots, s_k/x_k]$.
  - $\lfloor v \rfloor \lesssim_{\tau'} t_2[s_1/x_1, \ldots, s_k/x_k]$.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = (t_1\ t_2)$ with $t_1 : \tau' \to \tau''$, $t_2 : \tau'$

- By induction hypothesis,
  - $\lfloor F \rfloor \lesssim_{\tau' \to \tau} t_1[s_1/x_1, \ldots, s_k/x_k]$.
  - $\lfloor v \rfloor \lesssim_{\tau'} t_2[s_1/x_1, \ldots, s_k/x_k]$.
- By definition,
  - $t_1[s_1/x_1, \ldots, s_k/x_k] \to^{\mathfrak{e}} \lambda x.t_1'$ and $F \lesssim^{\circ}_{\tau' \to \tau} \lambda x.t_1'$;
  - $t_2[s_1/x_1, \ldots, s_k/x_k] \to^{\mathfrak{e}} c_2$ and $v \lesssim^{\circ}_{\tau'} c_2$.
- From $F \lesssim^{\circ}_{\tau' \to \tau} \lambda x.t_1'$, we have $\lfloor u \rfloor = F(v) \lesssim_\tau t_1'\,[c_2/x]$.
- Then, there is $c \in C^{\mathfrak{e}}_\tau$ such that $t_1'\,[c_2/x] \to^{\mathfrak{e}} c$ and $u \lesssim^{\circ}_\tau c$;
- $(t_1\ t_2)[s_1/x_1, \ldots, s_k/x_k] \to^{\mathfrak{e}} c$;
- Finally, $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1,\ldots,v_k/x_k]) \lesssim_{\tau} t[s_1/x_1,\ldots,s_k/x_k]$.

## Inductive Step: $t = \mathit{let}\ x \Leftarrow t_1.t_2$ with $x:\tau_1,\ t_1:\tau_1,\ t_2:\tau$

- Suppose that $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1,\ldots,v_k/x_k]) = \lfloor u \rfloor$
- Then there is $u_1 \in V^{\mathfrak{e}}_{\tau_1}$ such that
  - $[\![t_1]\!]^{\mathfrak{e}}(\rho[v_1/x_1,\ldots,v_k/x_k]) = \lfloor u_1 \rfloor$;
  - $[\![t_2]\!]^{\mathfrak{e}}(\rho[v_1/x_1,\ldots,v_k/x_k][u_1/x]) = \lfloor u \rfloor$;
- From induction hypothesis, there are canonical forms $c_1, c$ such that
  - $u_1 \lesssim^{\circ}_{\tau_1} c_1$ and $t_1[s_1/x_1,\ldots,s_k/x_k] \rightarrow^{\mathfrak{e}} c_1$
  - $u \lesssim^{\circ}_{\tau_1} c$ and $t_2[s_1/x_1,\ldots,s_k/x_k][c_1/x] \rightarrow^{\mathfrak{e}} c$
- Thus, $t[s_1/x_1,\ldots,s_k/x_k] \rightarrow^{\mathfrak{e}} c$.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = \mathbf{rec}y.(\lambda x.t')$ with $x : \tau''$, $t' : \tau'$

- Suppose that $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) = \lfloor G \rfloor$ for $G \in V^{\mathfrak{e}}_{\tau'' \to \tau'}$
- $G = \mu F.(\lambda v.[\![t']\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k, v/x, F/y]))$
- $G = \bigsqcup_{n \in \omega} G_n = \bigcup_{n \in \omega} G_n$ where
  - $G_0 := \bot_{V^{\mathfrak{e}}_{\tau'' \to \tau'}}$
  - $G_{n+1} := \lambda v.[\![t']\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k, v/x, G_n/y])$
- By induction: for all $n$, $G_n \lesssim^{\circ}_{\tau'' \to \tau'} (\lambda x.t')[\mathbf{s}/\mathbf{x}, t[\mathbf{s}/\mathbf{x}]/y]$
- Then $G \lesssim^{\circ}_{\tau'' \to \tau'} (\lambda x.t')[\mathbf{s}/\mathbf{x}, t[\mathbf{s}/\mathbf{x}]/y]$.
- Note that $t[s_1/x_1, \ldots, s_k/x_k] \to^{\mathfrak{e}} (\lambda x.t')[\mathbf{s}/\mathbf{x}, t[\mathbf{s}/\mathbf{x}]/y]$.
- Thus, $\lfloor G \rfloor \lesssim^{\circ}_{\tau'' \to \tau'} t[s_1/x_1, \ldots, s_k/x_k]$.

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = \mathbf{rec}y.(\lambda x.t')$ with $x : \tau''$, $t' : \tau'$

- $G_0 := \bot_{V^{\mathfrak{e}}_{\tau'' \to \tau'}}$ and $G_{n+1} := \lambda v.[\![t']\!]^{\mathfrak{e}}(\rho[\vec{\mathbf{v}}/\vec{\mathbf{x}}, v/x, G_n/y])$
- By induction: for all $n$, $G_n \lesssim^{\circ}_{\tau'' \to \tau'} (\lambda x.t')[\mathbf{s}/\mathbf{x}, t[\mathbf{s}/\mathbf{x}]/y]$
- Base Step: $\bot_{V^{\mathfrak{e}}_{\tau'' \to \tau'}} \lesssim^{\circ}_{\tau'' \to \tau'} (\lambda x.t')[\mathbf{s}/\mathbf{x}, t[\mathbf{s}/\mathbf{x}]/y]$
- For all $v, c$ such that $v \lesssim^{\circ}_{\tau''} c$,

$$\bot_{V^{\mathfrak{e}}_{\tau'' \to \tau'}}(v) \lesssim_{\tau'} t'[\mathbf{s}/\mathbf{x}, c/x, t[\mathbf{s}/\mathbf{x}]/y].$$

# Agreement of Eager Semantics

## Proof (by Structural Induction)

- $[\![t]\!]^{\mathfrak{e}}(\rho[v_1/x_1, \ldots, v_k/x_k]) \lesssim_\tau t[s_1/x_1, \ldots, s_k/x_k]$.

## Inductive Step: $t = \mathbf{rec}y.(\lambda x.t')$ with $x : \tau''$, $t' : \tau'$

- $G_0 := \bot_{V^{\mathfrak{e}}_{\tau'' \to \tau'}}$ and $G_{n+1} := \lambda v.[\![t']\!]^{\mathfrak{e}}(\rho[\vec{\mathbf{v}}/\vec{\mathbf{x}}, v/x, G_n/y])$
- By induction: for all $n$, $G_n \lesssim^{\circ}_{\tau'' \to \tau'} (\lambda x.t')[\mathbf{s}/\mathbf{x}, t[\mathbf{s}/\mathbf{x}]/y]$
- Inductive Step: suppose $G_n \lesssim^{\circ}_{\tau'' \to \tau'} (\lambda x.t')[\mathbf{s}/\mathbf{x}, t[\mathbf{s}/\mathbf{x}]/y]$.
- Then $\lfloor G_n \rfloor \lesssim_{\tau'' \to \tau'} t[\mathbf{s}/\mathbf{x}]$.
- Consider any $v \lesssim^{\circ}_{\tau''} c$ so that $\lfloor v \rfloor \lesssim_{\tau''} c$.
- Note that $G_{n+1}(v) = [\![t']\!]^{\mathfrak{e}}(\rho[\vec{\mathbf{v}}/\vec{\mathbf{x}}, v/x, G_n/y])$.
- By the main induction hypothesis,

$$G_{n+1}(v) \lesssim_{\tau'} t'[\mathbf{s}/\mathbf{x}, c/x, t[\mathbf{s}/\mathbf{x}]/y].$$

- $G_{n+1} \lesssim^{\circ}_{\tau'' \to \tau'} (\lambda x.t')[\mathbf{s}/\mathbf{x}, t[\mathbf{s}/\mathbf{x}]/y]$.

# Agreement of Eager Semantics

## Corollary

- $t$ : **int**: a closed term

Then $t \to^{\mathfrak{e}} n$ iff $[\![t]\!]^{\mathfrak{e}}(\rho) = \lfloor n \rfloor$.

# Lazy Semantics

# Lazy Semantics

Before the final evaluation of a term, . . .

- early semantics: evaluate every sub-term
- late semantics: evaluate only necessary sub-terms

# Lazy Semantics

## Terms

The terms $t$ are generated from the grammar:

$$
\begin{array}{rcl}
t & ::= & x \text{ (variables)} \\
& | & n \text{ (integer constants)} \\
& | & t_1 \bowtie t_2 \ (\bowtie \in \{=, -, \times\}) \text{ (arithmetic operations)} \\
& | & \textbf{if } t_0 \textbf{ then } t_1 \textbf{ else } t_2 \\
& | & (t_1, t_2) \text{ (ordered pairs)} \\
& | & \textbf{fst}(t) \text{ (first entry of ordered pairs)} \\
& | & \textbf{snd}(t) \text{ (second entry of ordered pairs)} \\
& | & \lambda x.t \ (\lambda\text{-abstraction}) \\
& | & (t_1 \ t_2) \text{ (function application } t_1(t_2)) \\
& | & \textbf{let } x \Leftarrow t_1 \textbf{ in } t_2 \text{ (let-notation } t_2\,[t_1/x]) \\
& | & \textbf{rec}y.t \text{ (recursion)}
\end{array}
$$

# Typing Rules

## Recursion

$$\frac{y : \tau,\ \lambda t : \tau}{\mathbf{rec} y.(t) : \tau}$$

# Typing Rules

## Typable Terms

- A term $t$ is *typable* if $t : \tau$ for some type $\tau$.
- A term $t$ is *uniquely typable* if $t : \tau$ for some unique type $\tau$. (i.e., $t : \tau_1$ and $t : \tau_2$ implies $\tau_1 = \tau_2$)

# Typing Rules

## Typable Terms

- A term $t$ is *typable* if $t : \tau$ for some type $\tau$.
- A term $t$ is *uniquely typable* if $t : \tau$ for some unique type $\tau$. (i.e., $t : \tau_1$ and $t : \tau_2$ implies $\tau_1 = \tau_2$)

## A Simple Exercise

Every typable term is uniquely typable.

# Terms

## Free Variables

- $FV(\mathbf{rec}y.t) := FV(t) \setminus \{y\}$.
- A term $t$ is *closed* if $FV(t) = \emptyset$.

# Lazy Operational Semantics

## Canonical Forms

- $\tau$: a type

The set $C^{\mathfrak{l}}_{\tau}$ is recursively defined as follows:

- $C^{\mathfrak{l}}_{\mathbf{int}} := \mathbb{Z}$;
- $C^{\mathfrak{l}}_{\tau_1 * \tau_2} := \{(t_1, t_2) \mid t_1 : \tau_1, t_2 : \tau_2 \text{ and } t_1, t_2 \text{ are closed.}\}$;
- $C^{\mathfrak{l}}_{\tau_1 \to \tau_2} := \{\lambda x.t \mid \lambda x.t : \tau_1 \to \tau_2 \text{ is closed.}\}$

# Lazy Operational Semantics

## Canonical Forms

- $\tau$: a type

The set $C^{\mathfrak{l}}_{\tau}$ is recursively defined as follows:

- $C^{\mathfrak{l}}_{\mathbf{int}} := \mathbb{Z}$;
- $C^{\mathfrak{l}}_{\tau_1 * \tau_2} := \{(t_1, t_2) \mid t_1 : \tau_1, t_2 : \tau_2 \text{ and } t_1, t_2 \text{ are closed.}\}$;
- $C^{\mathfrak{l}}_{\tau_1 \to \tau_2} := \{\lambda x.t \mid \lambda x.t : \tau_1 \to \tau_2 \text{ is closed.}\}$

## Exercise

Prove that for any type $\tau$ and term $t \in C^{\mathfrak{e}}_{\tau}$, $t$ is closed.

# Lazy Operational Semantics

## The Evaluation Relation

- $t$: a closed term with type $\tau$
- $c$: a canonical term in $C_\tau^{\mathfrak{e}}$

Then

- $t \to^{\mathfrak{l}} c$: $t$ evaluates to $c$ in lazy operational semantics

# Lazy Operational Semantics

## Evaluation Rules

- canonical terms:

$$\frac{c \in C_\tau^\iota}{c \rightarrow^\iota c}$$

# Lazy Operational Semantics

## Evaluation Rules

- arithmetic operations:

$$\frac{t_1 \to^{\mathsf{l}} n_1, \;\; t_2 \to^{\mathsf{l}} n_2}{t_1 \;\mathrm{op}\; t_2 \to^{\mathsf{l}} n_1 \;\mathrm{op}\; n_2} \quad \mathrm{op} \in \{+, -, \times\}$$

# Lazy Operational Semantics

## Evaluation Rules

- conditional branch:

$$\frac{t_0 \to^{\mathfrak{l}} 0 \ , t_1 \to^{\mathfrak{l}} c_1}{\textbf{if } t_0 \textbf{ then } t_1 \textbf{ else } t_2 \to^{\mathfrak{l}} c_1}$$

$$\frac{t_0 \to^{\mathfrak{l}} n \ , t_2 \to^{\mathfrak{l}} c_2}{\textbf{if } t_0 \textbf{ then } t_1 \textbf{ else } t_2 \to^{\mathfrak{l}} c_2} \ (n \neq 0)$$

# Lazy Operational Semantics

## Evaluation Rules

- Product:

$$\frac{t \to^{\mathsf{l}} (t_1, t_2),\ \ t_1 \to^{\mathsf{l}} c_1}{\mathbf{fst}(t) \to^{\mathsf{l}} c_1} \qquad \frac{t \to^{\mathsf{l}} (t_1, t_2),\ \ t_2 \to^{\mathsf{l}} c_2}{\mathbf{snd}(t) \to^{\mathsf{l}} c_2}$$

# Lazy Operational Semantics

## Evaluation Rules

- Function Application:

$$\frac{t_1 \rightarrow^{\mathfrak{l}} \lambda x.t_1' \ , t_1'\,[t_2/x] \rightarrow^{\mathfrak{l}} c}{(t_1\ t_2) \rightarrow^{\mathfrak{l}} c}$$

# Lazy Operational Semantics

## Evaluation Rules

- "Let" Notation:

$$\frac{t_2\,[t_1/x] \;\rightarrow^{\ell}\; c}{\textbf{let}\; x \Leftarrow t_1\; \textbf{in}\; t_2 \;\rightarrow^{\ell}\; c}$$

# Lazy Operational Semantics

## Evaluation Rules

- Recursion:

$$\frac{t\,[\mathbf{rec}y.t/y] \;\rightarrow^{\ell}\; c}{\mathbf{rec}y.t \;\rightarrow^{\ell}\; c}$$

# Lazy Operational Semantics

## Proposition

- $t$: a closed term with type $\tau$
- $c, c'$: canonical terms

Then we have that:

- if $t \to^{\mathfrak{e}} c$ and $t \to^{\mathfrak{e}} c'$ then $c = c'$;
- if $t \to^{\mathfrak{e}} c$ then $c : \tau$.

# Lazy Operational Semantics

## Proposition

- $t$: a closed term with type $\tau$
- $c, c'$: canonical terms

Then we have that:

- if $t \to^{!} c$ and $t \to^{!} c'$ then $c = c'$;
- if $t \to^{!} c$ then $c : \tau$.

# Lazy Operational Semantics

## Proposition

- $t$: a closed term with type $\tau$
- $c, c'$: canonical terms

Then we have that:

- if $t \to^{\mathfrak{l}} c$ and $t \to^{\mathfrak{l}} c'$ then $c = c'$;
- if $t \to^{\mathfrak{l}} c$ then $c : \tau$.

## Proof.

By a simple rule induction. □

# Lazy Denotational Semantics

# Lazy Denotational Semantics

## Values

- $\tau$: a type

The discrete cpo $V^{\mathfrak{l}}_{\tau}$ of values associated with type $\tau$ is recursively defined as follows:

- $V^{\mathfrak{l}}_{\mathbf{int}} := \mathbb{Z}$;
- $V^{\mathfrak{l}}_{\tau_1 * \tau_2} := (V^{\mathfrak{l}}_{\tau_1})_{\perp} \times (V^{\mathfrak{l}}_{\tau_2})_{\perp}$;
- $V^{\mathfrak{l}}_{\tau_1 \to \tau_2} := [(V^{\mathfrak{l}}_{\tau_1})_{\perp} \to (V^{\mathfrak{l}}_{\tau_2})_{\perp}]$

# Lazy Denotational Semantics

## Values

- $\tau$: a type

The discrete cpo $V^{\mathfrak{l}}_{\tau}$ of values associated with type $\tau$ is recursively defined as follows:

- $V^{\mathfrak{l}}_{\mathbf{int}} := \mathbb{Z}$;
- $V^{\mathfrak{l}}_{\tau_1 * \tau_2} := (V^{\mathfrak{l}}_{\tau_1})_{\perp} \times (V^{\mathfrak{l}}_{\tau_2})_{\perp}$;
- $V^{\mathfrak{l}}_{\tau_1 \to \tau_2} := [(V^{\mathfrak{l}}_{\tau_1})_{\perp} \to (V^{\mathfrak{l}}_{\tau_2})_{\perp}]$

## Question

Why do we have extra $\perp$'s?

# Lazy Denotational Semantics

## Environments

- **Var**: the set of variables

An *environment* $\rho$ is a function

$$\rho : \mathbf{Var} \to \bigcup \{ (V_\tau^{\mathfrak{l}})_\perp \mid \tau \text{ a type} \}$$

such that

$$\forall x \in \mathbf{Var}.(x : \tau \Rightarrow \rho(x) \in (V_\tau^{\mathfrak{l}})_\perp) \ .$$

# Lazy Denotational Semantics

## Environments

- **Var**: the set of variables

An *environment* $\rho$ is a function

$$\rho : \mathbf{Var} \to \bigcup \{ (V_\tau^{\mathfrak{l}})_\perp \mid \tau \text{ a type} \}$$

such that

$$\forall x \in \mathbf{Var}. (x : \tau \Rightarrow \rho(x) \in (V_\tau^{\mathfrak{l}})_\perp) \ .$$

We denote by $\mathbf{Env}^{\mathfrak{l}}$ the set of environments under lazy semantics.

# Lazy Denotational Semantics

## Intuition

- $t$: a typable term with type $\tau$
- $[\![t]\!]^{\mathfrak{l}} : \mathbf{Env}^{\mathfrak{l}} \to (V_\tau^{\mathfrak{l}})_\perp$: the denotational semantics of $t$

# Lazy Denotational Semantics

## Inductive Definition

- $[\![x]\!]^{\mathfrak{l}} := \lambda\rho.\rho(x)$;
- $[\![n]\!]^{\mathfrak{l}} := \lambda\rho.\lfloor n \rfloor$;

# Lazy Denotational Semantics

## Inductive Definition

- $[\![t_1 \ \mathbf{op} \ t_2]\!]^{\mathfrak{l}} := \lambda\rho.([\![t_1]\!]^{\mathfrak{l}}(\rho) \ \mathbf{op}_{\perp} \ [\![t_2]\!]^{\mathfrak{l}}(\rho))$;
- $$[\![\mathbf{if} \ t_0 \ \mathbf{then} \ t_1 \ \mathbf{else} \ t_2]\!]^{\mathfrak{l}} := \lambda\rho.\mathit{Cond}([\![t_0]\!]^{\mathfrak{l}}(\rho), [\![t_1]\!]^{\mathfrak{l}}(\rho), [\![t_2]\!]^{\mathfrak{l}}(\rho));$$

# Lazy Denotational Semantics

## Inductive Definition

- $[\![(t_1, t_2)]\!]^{\mathfrak{l}} := \lambda\rho.\lfloor([\![t_1]\!]^{\mathfrak{l}}(\rho), [\![t_2]\!]^{\mathfrak{l}}(\rho))\rfloor$;
- $[\![\mathbf{fst}(t)]\!]^{\mathfrak{l}} := \lambda\rho.let\ v \Leftarrow [\![t]\!]^{\mathfrak{l}}(\rho).\pi_1(v)$
- $[\![\mathbf{snd}(t)]\!]^{\mathfrak{l}} := \lambda\rho.let\ v \Leftarrow [\![t]\!]^{\mathfrak{l}}(\rho).\pi_2(v)$

# Lazy Denotational Semantics

## Inductive Definition

- $[\![\lambda x.t]\!]^{\mathfrak{l}} := \lambda\rho.\lfloor\lambda v \in (V^{\mathfrak{l}}_{\tau_1})_{\perp}.[\![t]\!]^{\mathfrak{l}}(\rho\,[v/x])\rfloor$ for $\lambda x.t : \tau_1 \to \tau_2$
- $[\![(t_1\ t_2)]\!]^{\mathfrak{l}} := \lambda\rho.let\ F \Leftarrow [\![t_1]\!]^{\mathfrak{l}}(\rho).F([\![t_2]\!]^{\mathfrak{l}}(\rho))$

# Lazy Denotational Semantics

## Inductive Definition

- $[\![\textbf{let } x \Leftarrow t_1 \textbf{ in } t_2]\!]^{\mathfrak{l}} := \lambda\rho.[\![t_2]\!]^{\mathfrak{l}}\left(\rho\left[[\![t_1]\!]^{\mathfrak{l}}(\rho)/x\right]\right)$
- $[\![\textbf{rec}y.t]\!]^{\mathfrak{l}} := \lambda\rho.(\mu F.[\![t]\!]^{\mathfrak{l}}(\rho[F/y]))$

# Agreement of Lazy Semantics

## Operational Convergence

- $t$: a typable closed term

We say that $t$ is *operationally convergent*, denoted by $t{\downarrow^{\mathsf{r}}}$, if it holds that $\exists c.t{\rightarrow^{\mathsf{r}}}c$.

# Agreement of Lazy Semantics

## Denotational Convergence

- $t$: a typable closed term with type $\tau$

We say that $t$ is *denotationally convergent*, denoted by $t{\Downarrow^{\mathfrak{l}}}$, if it holds that $\exists v \in V_\tau^{\mathfrak{l}}.[\![t]\!]^{\mathfrak{l}}(\rho) = \lfloor v \rfloor$.

# Agreement of Lazy Semantics

## The Theorem

- $t$: a closed typable term
- $c$: a canonical term

Then we have:

- $t \to^{\mathfrak{e}} c$ implies $[\![t]\!]^{\mathfrak{e}}(\rho) = [\![c]\!]^{\mathfrak{e}}(\rho)$;
- $t{\downarrow}^{\mathfrak{e}}$ iff $t{\Downarrow}^{\mathfrak{e}}$.

# Agreement of Lazy Semantics

## A Corollary

- $t$: a closed typable term with type **int**

Then we have that $t \rightarrow^{\mathfrak{l}} n$ iff $[\![t]\!]^{\mathfrak{l}}(\rho) = \lfloor n \rfloor$.

# Summary

- functional language
- early and late semantics
- agreement of the semantics