More Symple Types
Progress And Preservation

Armando Solar-Lezama
Computer Science and Artificial Intelligence Laboratory
M.I.T.

September 28, 2015

Formalizing a Type System
Recap

Static Semantics

- Typing rules
  - Typing rules tell us how to derive typing judgments
  - Very similar to derivation rules in Big Step OS

$$\frac{premises}{Judgment}$$

- Ex. Language of Expressions

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \qquad \frac{}{\Gamma \vdash N : int} \qquad \frac{\Gamma \vdash e1 : int \qquad \Gamma \vdash e2 : int}{\Gamma \vdash e1 + e2 : int}$$

Ex. Language of Expressions

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \qquad \frac{}{\Gamma \vdash N : int} \qquad \frac{\Gamma \vdash e1 : int \qquad \Gamma \vdash e2 : int}{\Gamma \vdash e1 + e2 : int}$$

- Show that the following Judgment is valid

$$x : int, y : int \vdash x + (y + 5) : int$$

$$\frac{x : int, y : int \vdash x : int \qquad x : int, y : int \vdash (y + 5) : int}{x : int, y : int \vdash x + (y + 5) : int}$$

$$\frac{\dfrac{x : int \in x : int, y : int}{x : int, y : int \vdash x : int} \qquad \dfrac{x : int, y : int \vdash y : int \qquad x : int, y : int \vdash 5 : int}{x : int, y : int \vdash (y + 5) : int}}{x : int, y : int \vdash x + (y + 5) : int}$$

Simply Typed λ Calculus (F_1)

- Basic Typing Rules

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash (\lambda x : \tau_1\ e) : \tau_1 \to \tau_2} \qquad \frac{\Gamma \vdash e_1 : \tau' \to \tau \quad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1 e_2 : \tau}$$

- Extensions

$$\frac{}{\Gamma \vdash N : int} \qquad \frac{\Gamma \vdash e1 : int \qquad \Gamma \vdash e2 : int}{\Gamma \vdash e1 + e2 : int} \qquad \frac{\Gamma \vdash e1 : int \qquad \Gamma \vdash e2 : int}{\Gamma \vdash e1 = e2 : bool}$$

$$\frac{\Gamma \vdash e : bool \qquad \Gamma \vdash e_t : \tau \qquad \Gamma \vdash e_f : \tau}{\Gamma \vdash if\ e\ then\ e_t\ else\ e_f : \tau}$$

Example

- Is this a valid typing judgment?

$$\vdash (\lambda x{:}\,bool\ \lambda y{:}\,int\ \ if\ x\ then\ y\ else\ y+1){:}\,bool \rightarrow int \rightarrow int$$

- How about this one?

$$\vdash (\lambda x{:}\,int\ \ \lambda y{:}\,bool\ x+y){:}\,int \rightarrow bool \rightarrow int$$

Example

- What's the type of this function?

  (λ f. λ x. *if* x = 1 *then* x *else* (f f (x-1)) * x)

$$\frac{x:\tau \in \Gamma}{\Gamma \vdash x:\tau}$$

$$\frac{\Gamma, x:\tau_1 \vdash e:\tau_2}{\Gamma \vdash (\lambda x:\tau_1\ e):\tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1:\tau' \to \tau \quad \Gamma \vdash e_2:\tau'}{\Gamma \vdash e_1 e_2:\tau}$$

$$\frac{}{\Gamma \vdash N : int}$$

$$\frac{\Gamma \vdash e1 : int \qquad \Gamma \vdash e2 : int}{\Gamma \vdash e1 + e2 : int}$$

$$\frac{\Gamma \vdash e1 : int \qquad \Gamma \vdash e2 : int}{\Gamma \vdash e1 = e2 : bool}$$

$$\frac{\Gamma \vdash e : bool \qquad \Gamma \vdash e_t : \tau \qquad \Gamma \vdash e_f : \tau}{\Gamma \vdash if\ e\ then\ e_t\ else\ e_f : \tau}$$

  - Hint: This IS a trick question

Simply Typed λ Calculus (F_1)

- We have defined a really strong type system on λ-calculus
  - It's so strong, it won't even let us write non-terminating computation
  - We can actually prove this!

Progress and Preservation

What makes a type system "correct"

- "Well typed programs never go wrong"

- Inductive argument
  - **Preservation:** If a program is well typed it will stay well typed in the next step of evaluation
  - **Progress:** If a program is well typed now, it won't go wrong in the next step of evaluation

- What do we mean by "step of evaluation"?

Preservation

- Using Big-Step semantics we can argue global preservation

$$\Gamma \vdash e_1 : \tau \land \; e_1 \to e_2 \Rightarrow \quad \Gamma \vdash e_2 : \tau$$

- Prove by induction on the structure of derivation of $e_1 \to e_2$

Proof by induction on Structure of Evaluation

- Base cases: trivial

$$\frac{}{x \rightarrow x}$$

$$\frac{}{\lambda x. e \rightarrow \lambda x. e}$$

- Inductive case is a little trickier

$$\frac{e_1 \rightarrow \lambda x. e_1' \qquad e_1'[e_2/x] \rightarrow e_3}{e_1\ e_2 \ \rightarrow e_3}$$

Induction on the Structure of the Derivation

- Inductive case $$\frac{e_1 \to \lambda x.\, e_1' \qquad e_1'[e_2/x] \to e_3}{e_1\ e_2 \ \to e_3}$$
  - Given $\Gamma \vdash e_1 e_2 : \tau_{e12}$ we want to show that $\Gamma \vdash e_3 : \tau_{e12}$
  - By our typing rule, we have

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau_{e12} \quad \Gamma \vdash e_2 : \tau'}{\Gamma \ \vdash e_1 e_2 : \tau_{e12}}$$

  - And by the IH, we have that $\lambda x.\, e_1' : \tau' \to \tau_{e12}$
  - Which again by the typing rule $$\frac{\Gamma, x\!:\tau' \ \vdash e_1' : \tau_{e12}}{\Gamma \ \vdash (\lambda x\!:\tau'\ e_1') : \tau \to \tau_{e12}}$$
  - Now, we need to show that

$$\Gamma, x\!: \tau' \vdash e_1' : \tau_{e12} \wedge \Gamma \vdash e_2 : \tau' \Rightarrow \ \Gamma \vdash e_1'[e_2/x] : \tau_{e12}$$

  - And from our IH

$$\Gamma \vdash e_1'[e_2/x] : \tau_{e12} \Rightarrow \Gamma \vdash e_3 : \tau_{e12}$$

Small Step Semantics

- Big step goes directly from initial program to result

- Small Step evaluates one step at a time

Small Step Example

- Contexts

  H ::= o | H e1 | H + e | n + H |
  if H then e1 else e2 |
  H == e1 | n == H

- Local Reduction Rules
  - n1 + n2 → n (where n = plus n1 n2)
  - n1 == n2 → b (where b =(equals n1 n2))
  - if true then e1 else e2 → e1
  - if false then e1 else e2 → e2
  - (λx:τ.e1) v2 → [v2/x] e1

- Global Reduction Rules
  - H[*r*] → H[*e*] iff *r* → *e*

The proof strategy

- Progress Theorem

  If $\vdash e{:}\tau$ and e is not a value, then there is an e' s.t. $e \rightarrow e'$

- We can prove this through a decomposition lemma
  - If $\vdash e{:}\tau$ and e is not a value, then there are H and r s.t. e = H[r]
  - This guarantees one step of progress

Proving the Progress Theorem

If $\vdash e{:}\tau$ and e is not a value, then there is an e' s.t. $e \rightarrow e'$

or equivalently, e = H[r]

- Proved by induction on the derivation of $\vdash e{:}\tau$

- Base case:
  - Irreducible values

$$\frac{}{\Gamma \vdash false : bool} \quad \frac{}{\Gamma \vdash N : int} \quad \frac{x{:}\tau \in \Gamma}{\Gamma \vdash x : \tau} \quad \frac{}{\Gamma \vdash true : bool} \quad \frac{\Gamma, x{:}\tau_1 \vdash e{:}\tau_2}{\Gamma \vdash (\lambda x{:}\tau_1\ e){:}\ \tau_1 \rightarrow \tau_2}$$

Proving the Progress Theorem

- Inductive case

$$\frac{\Gamma \vdash e : bool \quad \Gamma \vdash e_t : \tau \quad \Gamma \vdash e_f : \tau}{\Gamma \vdash if\ e\ then\ e_t\ else\ e_f : \tau}$$

  - by the IH, e can be irreducible,
    - in which case it must be true or false and the whole thing is a redex
  - Or, it can be decomposed into H[r]
    - in which case if H then e1 else e2 is a valid context.

6.820 Fundamentals of Program Analysis
Fall 2015