[2018/06/03] 1

Jan Hoffman (lecture 1)

What is a prog. lang?
Today: progs. are mathematical objects.

How to define a PL?
1. Static semantics: what are (valid) progs?
2. Dynamic semantics: How to run progs?

Lang E

expr e ::= x
num[n]
bool[true]
bool[false]
if(e_1, e_2, e_3)
let(e_1, x, e_2)
plus(e_1; e_2) leq(e_1; e_2)

Static Semantics

- Option: progs = all expressions
  Not ideal. Progs that don't make sense should be excluded. (like '5 + true')

Observation: expressions come in 2 types: numbers & booleans

⇝ Type System

Ex: (1+2)+8 is valid (why?)
1+2 is a valid exp of type num.
8 is " "

⇝ need induction.

Jan Hoffman (Lect 1) [2018/06/03]

Notation: write $\vdash (1+2)+8 : num$

In general: $\vdash e : \tau$

we call this $\longrightarrow$ a judgment.

We often call things judgments and then say what these judgments mean using inductive definitions.

EX: Trees

1. emp is a tree
2. if n is a num & t_1, t_2 are trees, then $node(n, t_1, t_2)$ is a tree.

Judgment: $t : tree$

Could define this by saying the set of trees is the smallest set closed under rules 1 & 2

In PL we (instead) use inference rules:

Inference Rules

for defining judgments inductively

$$\frac{J_1 \;\; \ldots \;\; J_n}{J} \quad \begin{array}{l} \leftarrow \text{premises} \\ \leftarrow \text{conclusion} \end{array}$$

EX:

$$\frac{}{emp : tree} \qquad \frac{n : num \quad t_1 : tree \quad t_2 : tree}{node(n, t_1, t_2) : tree}$$

Jan Hoffman

[2018/06/03]

Derivations

$$\dfrac{\dfrac{\dfrac{}{z : num}(z)}{s(z) : num}(s) \qquad \dfrac{}{emp : tree}(1) \qquad \dfrac{}{emp : tree}(1)}{node(s(z), emp, emp) : tree}(2)$$

Type Rules

$$\dfrac{}{\vdash num[n] : num} \qquad \dfrac{}{\vdash bool[b] : bool}$$

What about variables? $\qquad \dfrac{}{\vdash x : ?}$

We need a context to keep track of the types of all variables.

$$\dfrac{}{\Gamma \vdash num[n] : num} \qquad \dfrac{}{\Gamma \vdash bool[b] : bool}$$

$$\dfrac{}{\Gamma, x:\tau \vdash x : \tau}$$

$$\dfrac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma, x:\tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash let(e_1; x.e_2) : \tau_2}$$ (Assume x does not appear in Γ)

$$\dfrac{\Gamma \vdash e_1 : num \qquad \Gamma \vdash e_2 : num}{\Gamma \vdash Plus(e_1; e_2) : num} \qquad \dfrac{\Gamma \vdash e_1 : num \qquad \Gamma \vdash e_2 : num}{\Gamma \vdash e_1 \le e_2 : bool}$$

$$\dfrac{\Gamma \vdash e : bool \qquad \Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash if(e, e_1, e_2) : \tau}$$

Jan Hoffmann

[2018/06/03]

Type Derivation

Ex: Judgment $\bullet \vdash \text{let } x = 5 \text{ in } x \leq 6 : \text{bool}$

We can derive this judgment using the type rules, as follows:

$$\dfrac{\dfrac{}{\bullet \vdash 5 : \text{num}}\,(\text{num}) \qquad \dfrac{x:\text{num} \vdash x : \text{num} \qquad x:\text{num} \vdash 6 : \text{num}}{x:\text{num} \vdash x \leq 6 : \text{bool}}}{\bullet \vdash \text{let } x = 5 \text{ in } x \leq 6 : \text{bool}}\,(\text{let})$$

Lemma: For every expr e and every context Γ there is at most one type τ s.t. $\Gamma \vdash e : \tau$.

(Proof: later)

Rule Induction

To prove $P(n)$ holds for all $n : \text{num}$, we prove
1. $P(z)$
2. If $P(n)$ then $P(s(n))$.

Then, since $\mathbb{N}$ is the smallest set closed under the two rules:

$$\dfrac{}{z : \text{num}} \qquad \dfrac{n : \text{num}}{s(n) : \text{num}},$$

it follows that $\mathbb{N} \subseteq \{ n \mid P(n) \}$.

Jan Hoffmann

[2018/06/03]

Rule Induction

To show $P(a)$, we show for every rule

$$\frac{a_1 \text{ J} \;\cdots\; a_n \text{ J} \quad b \text{ J}'}{a \text{ J}}$$ that

$P(a_1) \wedge \cdots \wedge P(a_n)$ implies $P(a)$

Inversion Principle

Show: if $\Gamma \vdash e : \tau_1$ $\Gamma \vdash e : \tau_2$ then $\tau_1 = \tau_2$

By induction on $\Gamma \vdash e : \tau_1$

Case "Var rule"

then $e = x$ and $\Gamma = \Gamma', x : \tau_1$

Example: Inversion for $Plus(e_1, e_2)$

Lemma If $\Gamma \vdash e_1 + e_2 : \tau$ then $\tau = num$ and $\Gamma \vdash e_1 : num$ and $\Gamma \vdash e_2 : num$.