Jan Hoffmann (lecture 1) [2018/07/03] 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 and booleans
⇝ type system

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

Jan Hoffmann (Lect 1) [2018/07/03] ②

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

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

EX:

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

Derivations

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

Type Rules

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

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

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

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

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

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

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

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

Jan Hoffmann [2018/07/03] 4

Type Derivation

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

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

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

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

Rule Induction

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

$$\frac{a_1 \text{ J} \ \ldots \ a_n \text{ J} \quad b \text{ J}'}{a \text{ J}}$$

that

$P(a_1) \wedge \ldots \wedge P(a_n)$ 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 $\Gamma \vdash e_2 : num$.