CSCI-GA-2110

Lecture 04

A Bit of Notation

- For the most part we will focus on implementation of interpreters
- But it is helpful to introduce some notation for describing programming languages with "pencil-and-paper"

Describing Syntax of Language: Backus-Naur Form

```
(define-type Expr
  [numC (n : number)]
  [plusC (e1 : Expr) (e2 : Expr)]
  [timesC (e1 : Expr) (e2 : Expr)]
  [letC (x : symbol) (e1 : Expr) (e2 : Expr)]
  [lambdaC (x : symbol) (e : Expr)]
  [appC (e1 : Expr) (e2 : Expr)]
  [idC (x : symbol)]
  )
```

Becomes:

$$
\begin{aligned}
e ::= n \mid (+\ e_1\ e_2) \mid (*\ e_1\ e_2) \mid (\mathit{let}\ x\ e_1\ e_2) \mid (\mathit{lambda}\ x\ e) \\
\mid (e_1\ e_2) \mid x
\end{aligned}
$$

Other Syntax Styles

Can also use this format to describe non-sexpression format languages

$$
\begin{aligned}
e ::= {} & n \mid (+\ e_1\ e_2) \mid (*\ e_1\ e_2) \mid (\mathit{let}\ x\ e_1\ e_2) \mid (\mathit{lambda}\ x\ e) \\
& \mid (e_1\ e_2) \mid x
\end{aligned}
$$

$$
\begin{aligned}
e ::= {} & n \mid e_1 + e_2 \mid e_1 * e_2 \mid \text{let } x = e_1 \text{ in } e_2 \mid \lambda x.\, e \\
& \mid e_1\ e_2 \mid x
\end{aligned}
$$

Substitution

Common to write $[e_1/x]e_2$ for capture-avoiding substitution of the expression e_1 in for the free variable x in e_2.

i.e. what we called `(subst e1 x e2)` in Racket.

α-equivalence

Typically, we want to think of programs as "the same" if they only differ in the names of variables.

Formally we say e_1 and e_2 are α-equivalent, written $e_1 \equiv_\alpha e_2$, if e_1 and e_2 only differ in names of bound variables.

α-equivalence

Typically, we want to think of programs as "the same" if they only differ in the names of variables.

Formally we say e_1 and e_2 are α-equivalent, written $e_1 \equiv_\alpha e_2$, if e_1 and e_2 only differ in names of bound variables.

$$\texttt{(let ([x 3]) (+ x x))} \equiv_\alpha \texttt{(let ([y 3]) (+ y y))}$$

$$\texttt{(lambda x (* 3 x))} \equiv_\alpha \texttt{(lambda y (* 3 y))}$$

Describing Evaluation

Write $e \Downarrow v$ to mean "e evaluates to the value v"

This is a **mathematical relation**. We can describe when this relation should hold using **inference rules**

$$\frac{P_1 \qquad P_2 \qquad \dots \qquad P_n}{Q}$$

" if $P_1, P_2, \dots P_n$ are true, then Q is true"

Describing Evaluation

Write $e \Downarrow v$ to mean "e evaluates to the value v"

This is a **mathematical relation**. We can describe when this relation should hold using **inference rules**

$$\frac{P_1 \qquad P_2 \qquad \dots \qquad P_n}{Q}$$

" if $P_1, P_2, \dots P_n$ are true, then Q is true"

$$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2 \qquad n_1 + n_2 = n}{(+\ e_1\ e_2) \Downarrow n}$$

Describing Eager vs Lazy Evaluation

$$\frac{e_1 \Downarrow v_1 \qquad [v_1/x]e_2 \Downarrow v_2}{(\mathit{let}\ x\ e_1\ e_2) \Downarrow v_2}$$

$$\frac{[e_1/x]e_2 \Downarrow v}{(\mathit{let}\ x\ e_1\ e_2) \Downarrow v}$$