Lecture 14

CIS 341: COMPILERS

Announcements

- HW4: OAT v. 1.0
  - Parsing & basic code generation
  - **Due: March 28th**
  - **START EARLY!**

- Midterm Exam
  - Grades Available on Gradescope
  - Solutions on the course web site

UNTYPED LAMBDA CALCULUS

(Untyped) Lambda Calculus

- The lambda calculus is a minimal programming language.
  - Note: we're writing `(fun x -> e)` lambda-calculus notation: λ x. e

Abstract syntax in OCaml:

```
type exp =
  | Var of var          (* variables                  *)
  | Fun of var * exp    (* functions: fun x -> e      *)
  | App of exp * exp    (* function application       *)
```

Concrete syntax:

exp ::=	
| x	*variables*
| `fun` x -> exp	*functions*
| exp_1 exp_2	*function application*
| `(` exp `)`	*parentheses*

Operational Semantics

- Specified using just two inference rules with judgments of the form exp ⇓ val
  - Read this notation a as "program exp evaluates to value val"
  - This is *call-by-value* semantics: function arguments are evaluated before substitution

$$\frac{}{v \Downarrow v}$$

"Values evaluate to themselves"

$$\frac{exp_1 \Downarrow (\text{fun } x \text{ -> } exp_3) \qquad exp_2 \Downarrow v \qquad exp_3\{v/x\} \Downarrow w}{exp_1\ exp_2 \Downarrow w}$$

"To evaluate function application: Evaluate the function to a value, evaluate the argument to a value, and then substitute the argument for the function. "

See fun.ml

IMPLEMENTING THE INTERPRETER