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
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:

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

Concrete syntax:

```latex
exp ::= 
| x variables
| fun x -> exp functions
| exp₁ exp₂ function application
| ( exp ) parentheses
```
Operational Semantics

- Specified using just two inference rules with judgments of the form \( \text{exp} \downarrow \text{val} \)
  - Read this notation as "program exp evaluates to value val"
  - This is call-by-value semantics: function arguments are evaluated before substitution

\[
\begin{align*}
\text{v} \downarrow \text{v} \\
\text{"Values evaluate to themselves"}
\end{align*}
\]

\[
\begin{align*}
\text{exp}_1 \downarrow (\text{fun } x \rightarrow \text{exp}_3) & \quad \text{exp}_2 \downarrow \text{v} & \quad \text{exp}_3\{v/x\} \downarrow \text{w} \\
\text{exp}_1 \text{ exp}_2 \downarrow \text{w}
\end{align*}
\]

“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