CS101

Special Topics in Computer Science

Language-Based Security

Lecture 2: Introduction to OCaml and λ-Calculus

Aleksey Nogin

October 7, 2005

Reminder: λ-Calculus Syntax

Expressions:

$e := n$	(Numerical constants)
$\mid e_1\ op\ e_2$	(Binary operations; $op = +, -, *$)
$\mid \lambda x : t.e$	(Functions — "`fun x:t -> e`")
$\mid e_1\ e_2$	(Function applications)

Types:

$t := \texttt{int}$	(Integers)
$\mid t_1 \rightarrow t_2$	(Functions)

Reminder: λ-Calculus Evaluation

Notation:

$e \longrightarrow v$ Expression e *evaluates to* (computes to) *value* v

$\dfrac{A_1 \quad \cdots \quad A_n}{C}$ Whenever *assumptions* A_i are true, the *conclusion* C must be true as well. If n is 0, C must be true unconditionally.

Numbers and functions are values:

$$\frac{}{n \longrightarrow n} \qquad \frac{}{\lambda x : t.e \longrightarrow \lambda x : t.e}$$

Binary operations: $\dfrac{e_1 \longrightarrow n_1 \quad e_2 \longrightarrow n_2 \quad n = n_1\ op\ n_2}{e_1\ op\ e_2 \longrightarrow n}$

Reminder: λ-Calculus Evaluation

Numbers and functions are values:

$$\frac{}{n \longrightarrow n} \qquad \frac{}{\lambda x : t.e \longrightarrow \lambda x : t.e}$$

Binary operations: $\dfrac{e_1 \longrightarrow n_1 \quad e_2 \longrightarrow n_2 \quad n = n_1\ op\ n_2}{e_1\ op\ e_2 \longrightarrow n}$

Function applications:

$$\frac{e_1 \longrightarrow \lambda x : t.e_1' \quad e_2 \longrightarrow v \quad e_1'[v/x] \longrightarrow v'}{e_1\ e_2 \longrightarrow v'}$$

Here $e_1'[v/x]$ stands for the result of *substitution* of the value v for the variable x in expression e_1'.

Motivation

What does all this have to do with security?

Two answers:

- Making sure that an "untrusted" function does not get "secret" data is similar to (but more complicated than) making sure that a function that expects an int will not get a string as its input.
- Before making sure that program does not violate some security policy, we need to be sure that program does not act "randomly" (for example, uses an integer value as a pointer and writes something to that location).

Reminder: λ-Calculus Example

In OCaml we had:

```
# let twice f x = f (f x);;
val twice : ('a -> 'a) -> 'a -> 'a = <fun>
# let add1 x = x + 1;;
val add1 : int -> int = <fun>
# let add2 = twice add1;;
val add2 : int -> int = <fun>
# add2 3;;
- : int = 5
```

In λ-Calculus, `add2 3` is $\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3$

Reminder: λ-Calculus Evaluation Example

$$\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3 \longrightarrow$$

Reminder: λ-Calculus Evaluation Example

$$(\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1) \longrightarrow$$

$$\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3 \longrightarrow$$

Reminder: λ-Calculus Evaluation Example

$$\lambda f.\lambda x.f\ (f\ x) \longrightarrow$$

$$(\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1) \longrightarrow$$

$$\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3 \longrightarrow$$

Reminder: λ-Calculus Evaluation Example

$$\lambda f.\lambda x.f\ (f\ x) \longrightarrow \lambda f.\lambda x.f\ (f\ x)$$

$$(\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1) \longrightarrow$$

$$\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3 \longrightarrow$$

Reminder: λ-Calculus Evaluation Example

$$\lambda f.\lambda x.f\ (f\ x) \longrightarrow \lambda f.\lambda x.f\ (f\ x) \qquad \lambda x.x+1 \longrightarrow$$

$$(\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1) \longrightarrow$$

$$\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3 \longrightarrow$$

Reminder: λ-Calculus Evaluation Example

$$\lambda f.\lambda x.f\ (f\ x) \longrightarrow \lambda f.\lambda x.f\ (f\ x) \qquad \lambda x.x+1 \longrightarrow \lambda x.x+1$$

$$(\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1) \longrightarrow$$

$$\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3 \longrightarrow$$

Reminder: λ-Calculus Evaluation Example

$$\dfrac{\dfrac{\dfrac{}{\lambda f.\lambda x.f\ (f\ x) \longrightarrow \lambda f.\lambda x.f\ (f\ x)} \quad \dfrac{}{\lambda x.x+1 \longrightarrow \lambda x.x+1} \quad \dfrac{}{\lambda x.(\lambda y.y+1)\ ((\lambda z.z+1)\ x) \longrightarrow \lambda x.\Big((\lambda y.y+1)\ ((\lambda z.z+1)\ x)\Big)}}{(\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1) \longrightarrow \lambda x.\Big((\lambda y.y+1)\ ((\lambda z.z+1)\ x)\Big)}}{\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3 \longrightarrow}$$

Reminder: λ-Calculus Evaluation Example

$$\dfrac{\dfrac{\dfrac{}{\lambda f.\lambda x.f\ (f\ x) \longrightarrow \lambda f.\lambda x.f\ (f\ x)} \quad \dfrac{}{\lambda x.x+1 \longrightarrow \lambda x.x+1} \quad \dfrac{}{\lambda x.(\lambda y.y+1)\ ((\lambda z.z+1)\ x) \longrightarrow \lambda x.\Big((\lambda y.y+1)\ ((\lambda z.z+1)\ x)\Big)}}{(\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1) \longrightarrow \lambda x.\Big((\lambda y.y+1)\ ((\lambda z.z+1)\ x)\Big)}}{\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3 \longrightarrow}$$

Reminder: λ-Calculus Evaluation Example

$$\dfrac{\dfrac{\dfrac{}{\lambda f.\lambda x.f\ (f\ x) \longrightarrow \lambda f.\lambda x.f\ (f\ x)} \quad \dfrac{}{\lambda x.x+1 \longrightarrow \lambda x.x+1} \quad \dfrac{}{\lambda x.(\lambda y.y+1)\ ((\lambda z.z+1)\ x) \longrightarrow \lambda x.\Big((\lambda y.y+1)\ ((\lambda z.z+1)\ x)\Big)}}{(\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1) \longrightarrow \lambda x.\Big((\lambda y.y+1)\ ((\lambda z.z+1)\ x)\Big)}}{\Big((\lambda f.\lambda x.f\ (f\ x))\ (\lambda x.x+1)\Big)\ 3 \longrightarrow}$$

Homework I

Homework I will be posted tonight, due next Friday in class.

- A few λ-Calculus problems.
- OCaml programming.

λ-Calculus Typing Judgements

Notation:

$e \in t$	Expression e has type t
$x_1 : t_1; \cdots ; x_n : t_n \vdash e \in t$	Provided each variable x_i has type t_i, the expression e (which might have x_i in it) has type t

Example:
The following *sequents* are true judgments:

$$x : \texttt{int}; y : \texttt{int} \vdash x + y \in \texttt{int}$$
$$z : \texttt{int} \vdash 3 \in \texttt{int}$$
$$a : \texttt{int} \vdash (\lambda x.x + a) \in (\texttt{int} \rightarrow \texttt{int})$$

λ-Calculus Typing Rules

Here Γ and Δ stand for arbitrary number of *hypotheses* in a sequent.

$$\frac{}{\Gamma \vdash n \in \texttt{int}}\ (\textit{Number}) \qquad \frac{}{\Gamma; x : t; \Delta \vdash x \in t}\ (\textit{Var})$$

$$\frac{\Gamma \vdash e_1 \in \texttt{int} \quad \Gamma \vdash e_2 \in \texttt{int}}{\Gamma \vdash e_1\ op\ e_2 \in \texttt{int}}\ (\textit{Binop})$$

$$\frac{\Gamma \vdash e_1 \in t_1 \rightarrow t_2 \quad \Gamma \vdash e_2 \in t_1}{\Gamma \vdash (e_1\ e_2) \in t_2}\ (\textit{Apply})$$

$$\frac{\Gamma; x : t_1 \vdash e \in t_2}{\Gamma \vdash \lambda x : t_1.e \in (t_1 \rightarrow t_2)}\ (\textit{Fun})$$

Reminder: Basic OCaml Syntax

Operations on integers:	`1 + 2`
Operations on floats:	`1.0 +. 2.0`
Variable definitions:	`let x = 2 + 4`
Function definitions:	`let f x = x + 1`
Anonymous functions:	`fun x -> x + 2`