Three Main Topics of the Course

I. New Programming Paradigm
   - Functional Programming
   - Environments and Closures
   - Patterns of Recursion
   - Continuation Passing Style

II. Language Translation
   - Lexing and Parsing
   - Type Systems
   - Interpretation

III. Language Semantics
   - Order of Evaluation
   - Specification to Implementation
Lexing and Parsing
Type Systems
Interpretation
Order of Evaluation
Specification to Implementation

Operational Semantics
Lambda Calculus
Axiomatic Semantics

Office: 2112 SC
Office hours:
TBD
Today 11:00am – 11:50 pm
Also by appointment
Email: egunter@illinois.edu
Do not use DM in Campuswire if you want a timely response. It does not email me notifications of that and it may take days for a response.

https://courses.engr.illinois.edu/cs421/sp2023
Main page - summary of news items
Policy - rules governing course
Lectures - syllabus and slides
MPs - information about assignments
Exams – Syllabi and review material for Midterms and finals
Unit Projects - for 4 credit students
Resources - tools and helpful info
FAQ
Some Course References

- No required textbook
- Some suggested references

![Books](image1.png)

1/18/23 15

Course Grading

- Assignments 10%
  - Web Assignments (WA) (~3-6%)
  - MPs (in Ocaml) (~4-7%)
  - All WAs and MPs Submitted in PrairieLearn
- May include necessary reading material
- Late submission:
  - 48 hours, unless otherwise specified
  - capped at 80% of total

1/18/23 17

Course Assignments – WA & MP

- You may discuss assignments and their solutions with others
- You may work in groups, but you must list members with whom you worked if you share solutions or detailed solution outlines
- Each student must write up and turn in their own solution separately
  - No direct copy-paste – type it yourself from your understanding
- You may look at examples from class and other similar examples from any source – cite appropriately
  - Note: University policy on plagiarism still holds - cite your sources if not the sole author of your solution
  - Do not have to cite course notes or me

1/18/23 19

OCAML

- Locally:
  - Will use ocaml inside VSCode inside PrairieLearn problems this semester
- Globally:
  - Main OCAML home: [http://ocaml.org](http://ocaml.org)
  - To install OCAML on your computer see: [http://ocaml.org/docs/install.html](http://ocaml.org/docs/install.html)
  - To try on the web: [https://try.ocamlpro.com](https://try.ocamlpro.com)
  - More notes on this later

1/18/23 20
References for OCaml

- Supplemental texts (not required):
  - The Objective Caml system release 4.05, by Xavier Leroy, online manual
  - Introduction to the Objective Caml Programming Language, by Jason Hickey
  - Developing Applications With Objective Caml, by Emmanuel Chailloux, Pascal Manoury, and Bruno Pagano, on O’Reilly
  - Available online from course resources

Features of OCAML

- Higher order applicative language
- Call-by-value parameter passing
- Modern syntax
- Parametric polymorphism
  - Aka structural polymorphism
- Automatic garbage collection
- User-defined algebraic data types

Why learn OCAML?

- Many features not clearly in languages you have already learned
- Assumed basis for much research in programming language research
- OCAML is particularly efficient for programming tasks involving languages (eg parsing, compilers, user interfaces)
- Industrially Relevant:
  - Jane Street trades billions of dollars per day using OCaml programs
  - Major language supported at Bloomberg
- Similar languages: Microsoft F#, SML, Haskell, Scala

Session in OCAML

% ocaml

Objective Caml version 4.07.1
#

(* Read-eval-print loop; expressions and declarations *)
2 + 3;;

(* Expression *)
- : int = 5
# 3 < 2;;
- : bool = false

Declarations; Sequencing of Declarations

# let x = 2 + 3;; /* declaration */
val x : int = 5
# let test = 3 < 2;;
val test : bool = false
# let a = 1 let b = a + 4;; /* Sequence of decl */
val a : int = 1
val b : int = 5

Functions

# let plus_two n = n + 2;;
val plus_two : int -> int = <fun>
# plus_two 17;;
- : int = 19
### Functions

```ocaml
define plus_two n = n + 2

plus_two 17
- : int = 19
```

### Extra Material

### No Overloading for Basic Arithmetic Operations

```ocaml
# 15 * 2;;
- : int = 30
# 1.35 + 0.23;; (* Wrong type of addition *)
Characters 0-4:
  1.35 + 0.23;; (* Wrong type of addition *)
  ^^^
Error: This expression has type float but an expression was expected of type int

# 1.35 +. 0.23;;
- : float = 1.58
```

### Booleans (aka Truth Values)

```ocaml
# true;;
- : bool = true
# false;;
- : bool = false
// p7 = {c → 4, test → 3.7, a → 1, b → 5}
# if b > a then 25 else 0;;
- : int = 25
```

### Booleans and Short-Circuit Evaluation

```ocaml
# 3 > 1 && 4 > 6;;
- : bool = false
# 3 > 1 || 4 > 6;;
- : bool = true
# (print_string "Hi\n"; 3 > 1) || 4 > 6;;
Hi
- : bool = true
# 3 > 1 || (print_string "Bye\n"; 4 > 6);;
- : bool = true
# not (4 > 6);;
- : bool = true
```
Sequencing Expressions

```ocaml
# "Hi there";; (* has type string *)
- : string = "Hi there"
# print_string "Hello world\n";; (* has type unit *)
Hello world
- : unit = ()
# (print_string "Bye\n"; 25);; (* Sequence of exp *)
Bye
- : int = 25
```

Recursive Functions

```ocaml
# let rec factorial n = 
  if n = 0 then 1 else n * factorial (n - 1);;
val factorial : int -> int = <fun>
# factorial 5;;
- : int = 120
# (* rec  is needed for recursive function declarations *)
```

Recursion Example

```ocaml
Compute \( n^2 \) recursively using:
\[
 n^2 = (2 \cdot n - 1) + (n - 1)^2 
\]

# let rec nthsq n =         (* rec for recursion *)
  match n              (* pattern matching for cases *)
  with 0              (* base case *)
    | n -> (2 * n -1) + nthsq (n -1) ;;   (* recursive call *)
val nthsq : int -> int = <fun>
# nthsq 3;;
- : int = 9
```

Recursion and Induction

```ocaml
# let rec nthsq n = match n with 0 -> 0 
  | n -> (2 * n -1) + nthsq (n -1) ;;
- : int -> int = <fun>
```

Environments

```ocaml
- **Environments** record what value is associated with a given identifier
- Central to the semantics and implementation of a language
- Notation
  \[ \rho = \{ \text{name}_1 \rightarrow \text{value}_1, \text{name}_2 \rightarrow \text{value}_2, \ldots \} \]
  Using set notation, but describes a partial function
- Often stored as list, or stack
  - To find value start from left and take first match
```

End of Extra Material
Environments

\[
X \rightarrow 3 \quad \text{name} \rightarrow \text{"Steve"} \\
y \rightarrow 17 \quad \text{region} \rightarrow (5.4, 3.7) \\
b \rightarrow \text{true} \quad \text{id} \rightarrow \{\text{Name} = \text{"Paul"}, \text{Age} = 23, \text{SSN} = 999888777\}
\]

Global Variable Creation

\[
\# \ 2 + 3;; \quad (* \ \text{Expression} *) \\
// \text{doesn’t affect the environment} \\
\# \ \text{let test} = 3 < 2;; \quad (* \ \text{Declaration} *) \\
\text{val test : bool} = \text{false} \\
// \ \rho_1 = \{\text{test} \rightarrow \text{false}\} \\
\# \ \text{let a} = 1 \ \text{let b} = a + 4;; \quad (* \ \text{Seq of dec} *) \\
// \ \rho_2 = \{b \rightarrow 5, a \rightarrow 1, \text{test} \rightarrow \text{false}\}
\]

New Bindings Hide Old

\[
// \ \rho_2 = \{b \rightarrow 5, a \rightarrow 1, \text{test} \rightarrow \text{false}\} \\
\text{let test} = 3.7;; \\
\]

What is the environment after this declaration?

\[
// \ \rho_3 = \{\text{test} \rightarrow 3.7, a \rightarrow 1, b \rightarrow 5\}
\]
Now it’s your turn

You should be able to do WA1-IC Problem 1, parts (* 1 *) - (* 3 *)