CS 421: Programming Languages and Compilers

Summer 2009

Kirill Mechitov

Based in part on slides developed by Mattox Beckman, Vikram Adve, Gul Agha, Elsa Gunter, and Sam Kamin

Announcements

- MP1 has been posted
  - Due 9:00AM Wednesday
  - Automatic 48 hour extension (at -20%)

Lecture Outline

- Intro to CS 421
- Administrative information
- Intro to OCaml
- MP1 information

Why CS421?

Why learn about compilers?

- Complete picture of how programs go from keyboard to execution
- Understand translation from high-level language to machine language
- Learn to build compilers and other programs that process structured input
- Learn interesting algorithms

Why CS421?

Why learn about programming languages?

- Increase ability to learn new languages
- Learn correct terminology for describing languages
- Make informed decisions about choosing the right language for the task
- Become better programmers by seeing different perspectives on programming

Course Information

- CS 421: Programming Languages and Compilers

- Lectures
  - Mon, Tue, Thu 11:30am – 12:45pm
  - 1304 Siebel Center
  - Lecture slides and videos will be posted same day

Contact Information

- Instructor – Kirill Mechitov
  - Office: 2107 SC
  - Office hours
    - Tuesday 1:00 – 2:00pm
    - And by appointment
  - E-mail: mechitov@illinois.edu
- TA – Amir Nayyeri
  - Office: TBA
  - Office hours: TBA
  - E-mail: nayyeri2@uiuc.edu

Resources

- Web page:
  - http://www.cs.uiuc.edu/class/cs421/
- Lecture videos (I2CS site):
  - https://agora.cs.uiuc.edu/display/I2CS/CS421
- Compass (grades):
  - https://compass.illinois.edu/
- Newsgroups:
  - Server: news.cs.uiuc.edu
  - class.cs421.announce – announcements
  - class.cs421 – general discussion

Course Website

- Main page – summary of news items
- Policy – rules governing course
- Lectures – slides and videos
- MPs – information about assignments
- Exams
- Unit Projects – for 4 credit students
- Resources – tools and helpful info
- FAQ

MPs, Exams, Grading

- Machine problems (MPs) 35%
  - About 8 MPs (in OCaml)
  - MPs submitted by via Compass
  - Automatic 48 hour extension with late submission penalty
    - 20% of assignments total value
- Midterm 25%
  - On-campus students: in class – **July 2**
  - I2CS students: **July 2**
- Final 40%
  - On-campus students: **August 8**, 1:00pm - 3:00pm
  - I2CS students: **August 7**

- **DO NOT MISS EXAM DATES!**

Assignments

- Collaboration is *not* allowed, unless explicitly stated otherwise in the assignment
  - If collaboration is allowed, each student must still turn in their assignment separately
- You may look at examples from class and other similar examples from any source
  - Note: University policy on plagiarism still holds
- Ungraded (practice) homework assignments may be given prior to the exams
- Problems from MPs and homeworks may appear verbatim, or with some modification, on exams

Course Outline

- First half: compilers
  - First week: Ocaml
- Second half: languages

Full details on web page (subject to change)

OCaml

- Functional programming language*
  - One of the two leading language paradigms (the other is object-oriented)
  - Defined mainly by *no assignment statements*
    - Heavy use of *dynamically-allocated data structures and recursion*
- Everything we will do in the first half of the class could also be done in Java, but:
  - OCaml notationally much more concise
  - Using OCaml now will prepare you for more advanced uses of Ocaml in second half

OCaml History

- Caml is European descendant of original ML
  - American/British version is SML
  - O is for object-oriented extension
- ML stands for Meta-Language
- ML family designed for implementing theorem provers
  - It was the meta-language for programming the "object" language of the theorem prover
  - Despite obscure original application area, OCaml is a full general-purpose programming language

OCaml Features

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

- It's fast - winners of the 1999 and 2000 ICFP Programming Contests used OCaml

OCaml Resources

- Compiler is on the EWS-Linux systems at /usr/local/bin/ocaml
- For the OCaml code for today's lecture see
  - http://www.cs.uiuc.edu/class/cs421/lectures/01-Intro.ml
- Main CAML homepage
  - http://caml.inria.fr/index.en.html
- To install OCaml on your computer see
  - http://caml.inria.fr/ocaml/release.en.html
- See also: Resources on the course web page

OCaml

- Interactive system:

```
> ocaml
# 2+2;;
- 4 : int
```

- Loading files:

```
#use "filename";;
```

OCaml

- Define variables and functions:

```
let x = 3;;
let f x = x+1;;
let f y = y+x;;
```

- Functions of multiple arguments:

```
let f x y = x+y;;
```

OCaml

- Arithmetic comparison operators: usual
- Boolean operators: =, && , ||, not
- Conditional expressions:

```
if x > y then x else y;;
```

OCaml

- Use of parentheses: precedence only

```
2 * (1 + 1);;
(f x) + 3;;
f (x + 3);;
```

- Strings: "…", ^ for concatenation
  - String module contains length, get, etc.

```
open String;;
length "Hello world";;
```

OCaml

- Characters: ‘c’, int_of_char, char_of_int

```
let x = (int_of_char ‘a’) + 3;;
```

- Printing to console: print_int, print_string

```
print_string “Hello“ ^ “ world!\n”;;
```

OCaml

- Sequencing: use ;

```
print_string "CS "; print_int 421;;
```

- Comments (* ... *)

```
(* this is a comment (* and so is this *) and this *)
```

OCaml

- Recursive function definitions

```
let rec fact n = if n < 2 then 1 else n * fact (n-1);;
```

MP1 Information

- Define a few simple functions in OCaml
  - See course web page
  - Follow instructions carefully
  - Skeleton ML file and grading script are provided

- Submitting the MP
  - zip netid.zip mp1.ml [your helper files, if any]
  - Upload via Illinois Compass
  - Compass assignment submissions not open yet, watch the course web page and/or newsgroup for announcements