Lecture 2

EECS 483: COMPILER CONSTRUCTION

HW1: Hellocaml

- Homework 1 is available on the course web site.
  - Individual project – no groups
  - Due: *Tuesday, 23 January at 11:59pm*
  - Topic: OCaml programming, an introduction

- OCaml head start:
  - use `make test` to build the compiler

- We recommend using:
  - VSCode + OCaml Platform

- See the course web pages about the tool chain to get started

Midterm

- Date change: Now March 12, in DOW 1013 and 1014 7-9pm
  - If you cannot make this time please contact me ASAP.
- GSI/IA Office Hours posted online

How to represent programs as data structures.
How to write programs that process programs.

INTERPRETERS & A SIMPLE COMPILER

Factorial: Everyone's Favorite Function

- Consider this implementation of factorial in a hypothetical programming language:

```
X = 6;
ANS = 1;
whileNZ (x) {
    ANS = ANS * X;
    X = X + -1;
}
```

- We need to describe the constructs of this hypothetical language
  - ***Syntax***: which sequences of characters count as a legal "program"?
  - ***Semantics***: what is the meaning (behavior) of a legal "program"?

Grammar for a Simple Language

```
<exp> ::=
    |  <X>
    |  <exp> + <exp>
    |  <exp> * <exp>
    |  <exp> < <exp>
    |  <integer constant>
    |  (<exp>)

<cmd> ::=
    |  skip
    |  <X> = <exp>
    |  ifNZ <exp> { <cmd> } else { <cmd> }
    |  whileNZ <exp> { <cmd> }
    |  <cmd>; <cmd>
```

BNF grammars are themselves domain-specific metalanguages for describing the syntax of other languages…

- Concrete syntax (grammar) for a simple imperative language
  - Written in "Backus-Naur form"
  - <*exp*> and <*cmd*> are ***nonterminals***
  - '::=' , '|' , and <…> symbols are part of the *metalanguage*
  - keywords, like 'skip' and 'ifNZ' and symbols, like '{' and '+' are part of the *object language*

- Need to represent the ***abstract syntax*** (i.e. hide the irrelevant of the concrete syntax)
- Implement the ***operational semantics*** (i.e. define the behavior, or meaning, of the program)

OCaml Demo

simple.ml
translate.ml

Concepts from the Demo

- ″Object″ vs. “Meta″ language:
  - Object language: the language being represented, manipulated, analyzed and transformed
  - Metalanguage: the language in which the object language representation and transformations are implemented
  - SIMPLE vs. OCaml

- “Interpretation” vs. “Compilation”
  - Interpreter: uses the features of the metalanguage to evaluate an object-language program, producing a result
  - Compiler: translates the object language to another (often lower level) object language

- “Static” vs. “Dynamic”:
  - Static = determined before the program is executed
  - Dynamic = determined while the program is running

Correctness?

- What does it mean for a compiler to be correct?
- What constitutes the “observable behavior” of a program?
- How do these notions affect what program transformations are allowed?