Programming Languages and Translators

COMS W4115

Pieter Bruegel, The Tower of Babel, 1563

Prof. Stephen A. Edwards
Fall 2003
Columbia University
Department of Computer Science

Instructor

Prof. Stephen A. Edwards
sedwards@cs.columbia.edu
http://www.cs.columbia.edu/~sedwards/
462 Computer Science Building
Office Hours: 4–5 PM Tuesday, Thursday

Schedule

Tuesdays and Thursdays, 5:40 PM to 6:55 PM
Room 717, Hamilton Hall
September 2 to December 4
Midterm: October 14
Holidays: November 4 (Election day), November 27 (Thanksgiving)

Objectives

Theory of language design
- Finer points of languages
- Different languages and paradigms

Practice of Compiler Construction
- Overall structure of a compiler
- Automated tools and their use
- Lexical analysis to assembly generation

Required Text

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman.
Addison-Wesley, 1985.
Available from Papyrus, 114th and Broadway.

Assignments and Grading

40% Programming Project
20% Midterm (near middle of term)
30% Final (at end of term)
10% Individual homework

Bottom line: do well on the project, you'll get a good grade.

Prerequisite: COMS W3156 Software Engineering

Teams will build a large software system
Makefiles, version control, test suites
Testing will be as important as development

Prerequisite: COMS W3261 Computability

You need to understand grammars.
We will be working with regular and context-free languages.

Class Website

Off my home page,
http://www.cs.columbia.edu/~sedwards/
Contains syllabus, lecture notes, and assignments.
Schedule will be continually updated during the semester.
Collaboration
Collaborate with your team on the project.
Homework is to be done by yourself.

The Project
Design and implement your own little language.

Five deliverables:
1. A white paper describing and motivating your language
2. A language reference manual defining it formally
3. A compiler or interpreter for your language running on some sample programs
4. A final project report
5. A final project presentation

The Project

Teams
Immediately start forming four-person teams to work on this project.
Each team will develop its own language.
Suggested division of labor: Front-end, back-end, testing, documentation.
All members of the team should be familiar with the whole project.

First Three Tasks
1. Decide who you will work with
   You’ll be stuck with them for the term; choose wisely.
2. Elect a team leader
   Languages come out better from dictatorships, not democracies. Besides, you’ll have someone to blame.
3. Select a weekly meeting time
   Harder than you might think. Might want to discuss with a TA you’d like to have so it is convenient for him/her as well.

White Paper
Follow the style of the Java white paper (see the class website for a link).
4–8 pages.
Answer the question, “why another language?” with a description of what your language is intended for.
Small snippets of code to show syntax is enough.

Language Reference Manual
A careful definition of the syntax and semantics of your language.
Follow the style of the C language reference manual (Appendix A of Kernighan and Ritchie, The C Programming Language; see the class website).

Final Report Sections
1. Introduction: the white paper
2. Language Tutorial
4. Project Plan
5. Architectural Design
6. Test Plan
7. Lessons Learned
8. Complete listing

Due Dates
White Paper
September 23 soon
Reference Manual
October 23
Final Report
December 12?
Design a language?
A small, domain-specific language.
Think of awk or php, not Java or C++.
Examples from last term:
Quantum computing language
Geometric figure drawing language
Projectile motion simulation language
Matlab-like array manipulation language
Screenplay animation language

Other language ideas
Simple animation language
Model train simulation language
Escher-like pattern generator
Music manipulation language (harmony)
Web surfing language
Mathematical function manipulator
Simple scripting language (à la Tcl)
Petri net simulation language

What's in a Language?

Components of a language: Syntax
How characters combine to form words, sentences, paragraphs.

The quick brown fox jumps over the lazy dog.
is syntactically correct English, but isn’t a Java program.

class Foo {
    public int j;
    public int foo(int k) { return j + k; }
}
is syntactically correct Java, but isn’t C.

Specifying Syntax
Usually done with a context-free grammar.

Typical syntax for algebraic expressions:

expr → expr + expr
    | expr − expr
    | expr × expr
    | expr/expr
    | digit
    | (expr)

Components of a language: Semantics
What a well-formed program “means.”

Semantics
Something may be syntactically correct but semantically nonsensical.

The rock jumped through the hairy planet.

Or ambiguous

The chickens are ready for eating.

Specifying Semantics
Doing it formally beyond the scope of this class, but basically two ways:

- Operational semantics
  Define a virtual machine and how executing the program evolves the state of the virtual machine

- Denotational semantics
  Shows how to build the function representing the behavior of the program (i.e., a transformation of inputs to outputs) from statements in the language.

Most language definitions use an informal operational semantics written in English.
Great Moments in Programming Language Evolution

Assembly

Before: numbers

Great moments in evolution

After: Symbols

55
89E5
8B4508
8B550C
39D0
74D0
39D0
78C8
29D0
39D0
75F6
C9
29C2
EBF6
gcd: pushl %ebp
movl %esp, %ebp
movl 8(%ebp), %eax
jle .L5
jne .L7
movl 12(%ebp), %edx
jne .L7
subl %edx, %eax
jmp .L2

After: Symbols

10 if (a .EQ. b) goto 20
if (a .LT. b) then
a = a - b
endif goto 10

FORTRAN

Before

10 if (a .EQ. b) goto 20
if (a .LT. b) then
a = a - b
endif goto 10

After: Expressions, control-flow

FORTRAN

10 if (a .EQ. b) goto 20
if (a .LT. b) then
a = a - b
endif goto 10

ALGOL, PASCAL, CLU, MODULA, ADA

Imperative, block-structured language, formal syntax definition, structured programming

COBOL

Added type declarations, record types, file manipulation

data division.

file section.

* describe the input file

fd employee-file-in

label records standard

block contains 5 records

record contains 31 characters

data record is employee-record-in.

01 employee-record-in.

02 employee-name-in pic x(20).

02 employee-rate-in pic 9(3)v99.

02 employee-hours-in pic 9(3)v99.

02 line-feed-in pic x(1).

LISP, Scheme, Common LISP

Functional, high-level languages

(defun gnome-doc-insert ()
"Add a documentation header to the current function.
Only C/C++ function types are properly supported currently."
(interactive)
(let (c-insert-here (point))
(save-excursion
(beginning-of-defun)
(let (c-arglist c-funcname c-comment-point c-isvoid c-doinsert)
(search-backward "\(")
(forward-line -2)
(while (or (looking-at "\$")
(looking-at "\*"))
(forward-line -1))
(while (or (looking-at "]")
(looking-at "\*"))
(forward-line 0))
)
)
)

SNOBOL, Icon

String-processing languages

10 PRINT "GUESS A NUMBER BETWEEN ONE AND TEN"
20 INPUT A$ 
30 IF A$ = "5" THEN PRINT "GOOD JOB, YOU GUESSED IT"
40 IF A$ = "5" GOTO 100
50 PRINT "YOU ARE WRONG. TRY AGAIN"
60 GOTO 10
100 END

BASIC

Programming for the masses

10 PRINT "GUESS A NUMBER BETWEEN ONE AND TEN"
20 INPUT A$ 
30 IF A$ = "5" THEN PRINT "GOOD JOB, YOU GUESSED IT"
40 IF A$ = "5" GOTO 100
50 PRINT "YOU ARE WRONG. TRY AGAIN"
60 GOTO 10
100 END
Simula, Smalltalk, C++, Java, C#

The object-oriented philosophy

class Shape(x, y); integer x; integer y;
virtual: procedure draw;
begin
  comment -- get the x & y components for the object --;
  integer procedure getX;
  integer procedure getY;
  comment -- set the x & y coordinates for the object --;
  integer procedure setX(newx); integer newx;
  integer procedure setY(newy); integer newy;
end Shape;

C

Efficiency for systems programming

int gcd(int a, int b)
{
    while (a != b) {
        if (a > b) a -= b;
        else b -= a;
    }
    return a;
}

ML, Miranda, Haskell

Purer functional language

structure RevStack = struct
    type 'a stack = 'a list
    exception Empty
    val empty = []
    fun isEmpty (s:'a stack):bool =
        (case s
         of [] => true
          | _ => false)
    fun top (s:'a stack): =
        (case s
         of [] => raise Empty
          | x::xs => x)
    fun pop (s:'a stack):'a stack =
        (case s
         of [] => raise Empty
          | x::xs => xs)
    fun push (s:'a stack,x: 'a):'a stack = x::s
    fun rev (s:'a stack):'a stack = rev (s)
end

sh, awk, perl, tcl, python

Scripting languages: glue for binding the universe together

class()
{
    classname=`echo "$1" | sed -n '1 s/ *:.*$//p'
    parent=`echo "$1" | sed -n '1 s/Ã.*: *//p'
    hppbody=`echo "$1" | sed -n '2,$p'
    forwarddefs="$forwarddefs
     class $classname;"
    if (echo $hppbody | grep -q "$classname());"); then
        defaultconstructor=""$classname() {}"
    else
        defaultconstructor="$classname() {}"
    fi
}

VisiCalc, Lotus 1-2-3, Excel

The spreadsheet style of programming

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hours</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>Wage per hour</td>
<td>$5.36</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Total Pay</td>
<td>= B1 * B2</td>
</tr>
</tbody>
</table>

SQL

Database queries

CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    color ENUM('red', 'blue', 'white', 'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL,
    references person(id),
    PRIMARY KEY (id)
);

INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', LAST_INSERT_ID()),
(NULL, 'dress', 'white', LAST_INSERT_ID()),
(NULL, 't-shirt', 'blue', LAST_INSERT_ID());

Prolog

Logic Language

edge(a, b). edge(b, c).
edge(c, d). edge(d, e).
edge(b, e). edge(d, f).
path(X, X).
path(X, Y) :-
    edge(X, Z), path(Z, Y).