



CS4450/7450: Principles of Programming Languages

Fall 2009



Who am I?

- Dr William Harrison
Associate Professor of CS
17 EBN
HarrisonWL@missouri.edu
- Research Areas:
 - Language-based Security
 - Language Semantics
 - Software Verification



Today's Lecture

- **What is this course about?**

- Programming Language Design

- Overview of useful PL features

- data abstraction, imperative features, procedures, objects, types,...

- ... and their implementation using “interpreters”

- **Administrivia:** grading, textbook, syllabus,

- ...



What is a Programming Language?

```
class Foo {  
    String name; int age; Widget w;  
    int alg1 (int a, Widget w) { ... }  
    ...  
    void algk (String n, Widget w) { ... }  
}
```

- Is it just a syntax for describing data and associated algorithms?
- And, there are many such syntax:
 - Java, C++, ML, Scheme, Haskell, Prolog, Perl, ...

● ● ● | Language implementation

- After you have typed in a program, what have you got?

```
class public Foo { ...
```

- This sequence of characters must be given some “meaning” or definition to be useful
 - Translation into machine code, JVM code,...
 - Evaluation by a program in another high-level programming language
 - Mathematical specifications of some kind



Varieties of Language Definition

- Mathematical (aka “denotational”) semantics
 - Precise: programs **are** particular constructions in Mathematics
 - Suitable for proving properties of programs
- Interpreter
 - An “evaluator” program for the new language
 - Usually written in another, existing high-level PL
 - Relatively easy to write, but
 - may not execute as fast as possible
- Compiler
 - Translates program into “stand-alone executable”
 - Efficiency of executable is usually the biggest concern
 - take a long time to write,
 - are notoriously tricky to get correct,
 - are large and difficult to maintain
 - Gnu GCC-1750 (version 1.0) C++ compiler has **278,949** lines of code in **168** separate files



Bad Programming Language Design: COBOL

- Most of the languages you know have benefited from years of PL research:
 - Java, C#, Smalltalk,...
- ...& some languages have benefited more than others:
 - Haskell, ML, Scheme,...
- ...and some haven't benefited at all:
 - COBOL, Perl, ...

- ● ● | Bad Programming Language Design: COBOL

- COBOL has a built-in “date” type
- Ex: represents “Nov. 6th, 1999” as “991106”
- There are 500,000,000,000 lines of COBOL with statements like:
 - `if curYear+1 > 90 then...`



Bad Programming Language Design: COBOL

- Source of the infamous “Millennium bug”
 - when 1999→2000,
 - the year part becomes “00”
 - COBOL programs in banks crash
 - life as we know it ends
- Estimated cost of fixing/worrying about Millennium bug: \$100 Billion→\$1 Trillion

● ● ● | What did COBOL's design have to do with this?

- Basic sin: the implementation of the “date” type...
 - “Nov. 6th, 1999” as “991106”
- ...is as an integer in actual programs
 - “if curYear+1 > 90 then...”
- When is an integer an integer and when is it a date?
- Q: What's the moral of this story?



What is the purpose of the following program?

```
n := i;  
acc := 1;  
while (n > 0) {  
    acc := acc * n;  
    n := n - 1;  
}
```



Functions in Mathematics

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n - 1)! & \text{otherwise} \end{cases}$$



Functions in Mathematics

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n - 1)! & \text{otherwise} \end{cases}$$

It's not so easy to
spot the relationship!

```
n := i;  
acc := 1;  
while (n > 0) {  
    acc := acc * n;  
    n := n - 1;  
}
```



Functions in Programming

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{otherwise} \end{cases}$$

Haskell

```
fac 0 = 1
fac n = n * (fac (n-1))
```

ML

```
fun fac n = if n=0 then 1 else n * (fac (n-1));
```

Scheme

```
(define fac (lambda (n)
              (if (zero? n)
                  1
                  (* n (fac (- n 1))))))
```

● ● ● | Why functional languages?

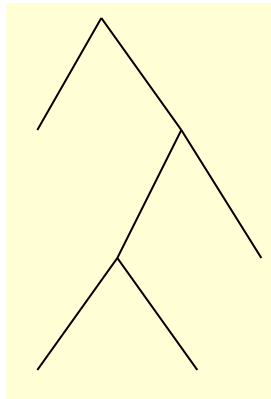
- Quick to learn “the basics” of functional programming
- Functional languages are very expressive
 - programs tend to be quite concise
 - this course will involve much programming, but few lines of code will be produced
- Their fundamental abstraction (i.e., *functions*) is “close to mathematics”
 - especially when compared with other programming paradigms like OO, etc...



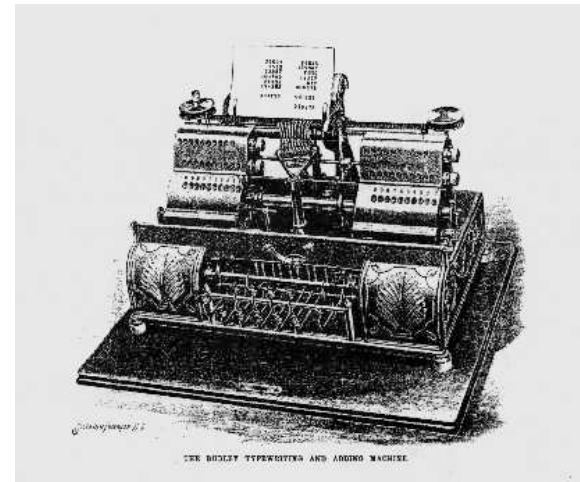
What is an interpreter?

An interpreter is a **function** s.t.

Representation
of Source Program



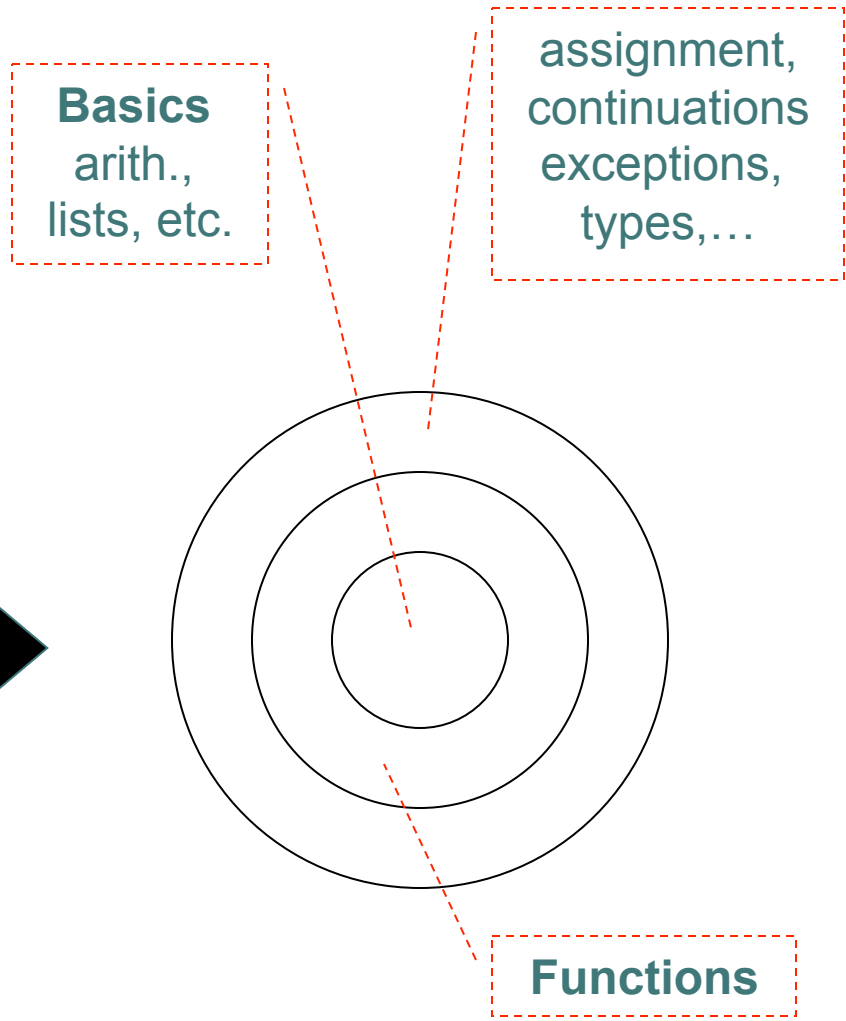
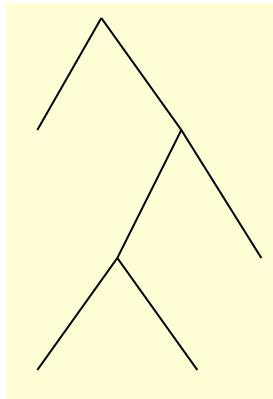
Executes/evaluates
with “abstract machine”





CS4450 at a high level

Representation
of Source Program





Administrivia

- Office hour:
 - By appointment only: feel free to email me to set up an appointment any time
 - that does not mean “show up anytime”!!!
 - I’m free most days, so it’s only a matter of making the appointment
- Course website is:
 - <http://www.cs.missouri.edu/~harrison/cs4450/>
 - all slides, homework assignments & solutions will end up there
 - they’re not there now



Administrivia

- **Required:** “Programming in Haskell” by Graham Hutton
 - will use this to teach the basics of functional programming in Haskell
 - Ramsey & Kamin’s: “Programming Languages: An Interpreter-based Approach”
 - Available from Mizzou Media
 - Draft of new edition of Classic Text
 - R&K is a classic text
 - describes building a Scheme interpreter for Scheme (aka, a “metacircular interpreter”)
 - we will build interpreters in Haskell



What do I expect of you?

- Eager to learn
- Responsible, independent and honest
- You know C and you are not afraid to learn new languages
- Some mathematical maturity: you're not afraid of a Greek letter or two
- This course involves writing a lot of “little” programs in several languages



Administrivia

- **Grading**
 - One midterm (25% each)
 - Final (35%)
 - Homework/programming assignments (40%)
- **Academic Honesty:** students are encouraged to discuss coursework.
- **Re-grades:** requests for re-grades must be made **in writing within 7 days** of receiving graded HW/test. There will be absolutely no exceptions.



Grading Scale: no curve

Undergraduate

A: 90-100%

B: 80-89%

C: 70-79%

D: 60-69%

F: < 60%

+ - grades possible

Graduate

A: 90-100%

B: 80-89%

C: 70-79%

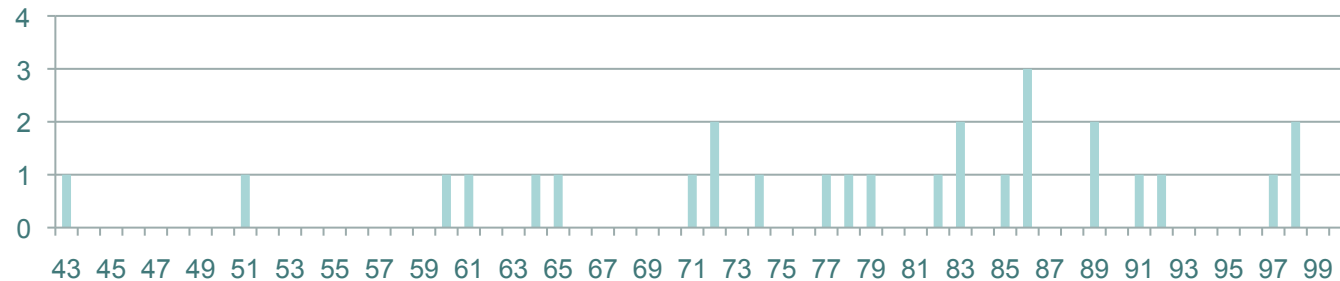
F: < 70%

+ - grades not possible



Previous Grading Experience

Midterm Scores: Frequency



If you're not confident about working independently and thinking for yourself, please consider taking another course



Some advice

- BSCS Requirement: “One of CS 4410 or 4430 or 4450”
 - I teach 4430 (compilers) and 4450
 - 4410 (Theory of Computation, taught in Spring) is much easier than either of these



“Assignment” 0

- Download and install the Haskell 98
 - we’ll use “Hugs interpreter”
 - Get it for free at: <http://haskell.org/hugs/>
 - read Chapter 1 & 2 of Hutton
 - Brief introduction to Haskell’s features and the use of Hugs

Academic
Dishonesty is Not
Tolerated in my
Class



Academic Honesty

- Your work **must** be your own
 - discussion with classmates is fine (and encouraged!)
 - that involves speaking with your mouth or writing on a chalkboard
- Consequences of academic dishonesty (undergraduate) :
 - 1st offense: Receive a zero on that assignment or test
 - 2nd offense: Automatic “F” grade in the class and I forward the evidence to the Provost
 - No exceptions
- Consequences of academic dishonesty (graduate)
 - The penalty for a graduate student caught cheating in this course is an automatic F grade for the course.
- Continued enrollment in this class implies your consent to these rules.



Academic Integrity

- Be truthful
- Always hand in your own work
- Never present the work of others as your own
- Give proper credit to sources
- Present your data accurately
- Violations of academic integrity will be taken very seriously.



Plagiarism

- Copying text or presenting ideas without attribution is plagiarism
- Plagiarism is a violation of academic integrity
- Undergraduates: If you commit plagiarism you will get a grade of 0 and be reported to the university
- I know how to use google
- I will accept no excuses
- There will be no second chances



Spring 2009

- The last time I taught this course, I caught two groups of cheaters
 - One group of three undergrads (one of whom I call “Beavis”)
 - One group of three graduate students
 - Upshot: The penalty for a graduate student caught cheating in this course is an automatic F grade for the course.



Judge for yourself

A sample of Beavis's code:

```
eval Plus rho = FunVal consVal
eval Plus env = FunVal consVal
  where consVal :: [Value] -> Value
        consVal [v1,v2]      = plus' v1 v2
        consVal _           = error "error: too few or too many args!"
        plus' :: Value -> Value -> Value
        plus' (I x) NilVal  = I (x)
        plus' (I x) (I y)   = I (x + y)
```

A sample of Butthead's code:

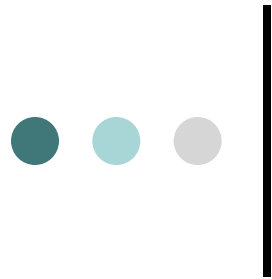
```
eval Plus env = FunVal consVal
  where consVal :: [Value] -> Value
        consVal [v1,v2]      = splus v1 v2
        consVal _           = error "error"
        splus :: Value -> Value -> Value
        splus (I x) NilVal  = I (x)
        splus (I x) (I y)   = I (x + y)
```



Beavis's Complaint

I do not agree with this change to my grade for the last homework. Beavis*, Butthead*, and I all met in the computer lab at EBW and worked on this assignment together. **All three of us discussed the problems and contributed to the solutions.** We did not simply copy and paste code from each other. I have collaborated with fellow students on numerous assignments in this class and other classes, and have never been penalized. I do not view this as a form of cheating or academic dishonesty, especially because my teachers often encourage me to work with other students. **You yourself encouraged collaboration in your initial set of slides for this class.** I deserve the initial grade that you sent to me for this assignment.

** Names changed to protect the stupid.*



Judge for Yourself: Beavis's Complaint

I do not agree with this change to my grade for the last homework. Beavis*, Butthead*, and I all met in the computer lab at EBW and worked on this assignment together. **All three of us discussed the problems and contributed to the solutions.** We did not simply copy and paste code from each other. I have collaborated with fellow students on numerous assignments in this class and other classes, and have never been penalized. I do not view this as a form of cheating or academic dishonesty, especially because my teachers often encourage me to work with other students. **You yourself encouraged collaboration in your initial set of slides for this class.** I deserve the initial grade that you sent to me for this assignment.

Here's what I said:

- Your work **must** be your own
 - discussion with classmates is fine (and encouraged!)
 - that involves speaking with your mouth or writing on a chalkboard