CS4430/7430
Introduction to Compiler Construction

Dr William Harrison

Spring 2017

HarrisonWL@missouri.edu

Administrivia

- Office hours:
  - By appointment only: feel free to email me to set up an appointment otherwise
  - 318 EBN
  - Course website:
    - https://harrisonwl.github.io/doc/cs4430.html

Today's Lecture

- **What is this course about?**
  - Programming Language Implementation
    - Overview of implementation styles
      - Interpreters, compilers, formal semantics…
    - High-level view of compiler structure
      - Lexing, parsing, code generation & optimization
- **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) { … }
}
```

- 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?

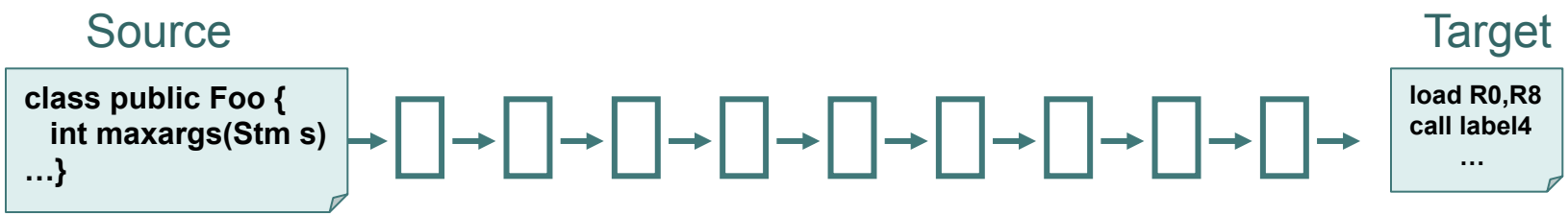
c	l	a	s	s		p	u	b	l	i	c		F	o	o		{	…

- 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 language definition
  - 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
    - Doesn’t run as fast as possible
- Compilers
  - Translate programs into “stand-alone executables”
  - 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

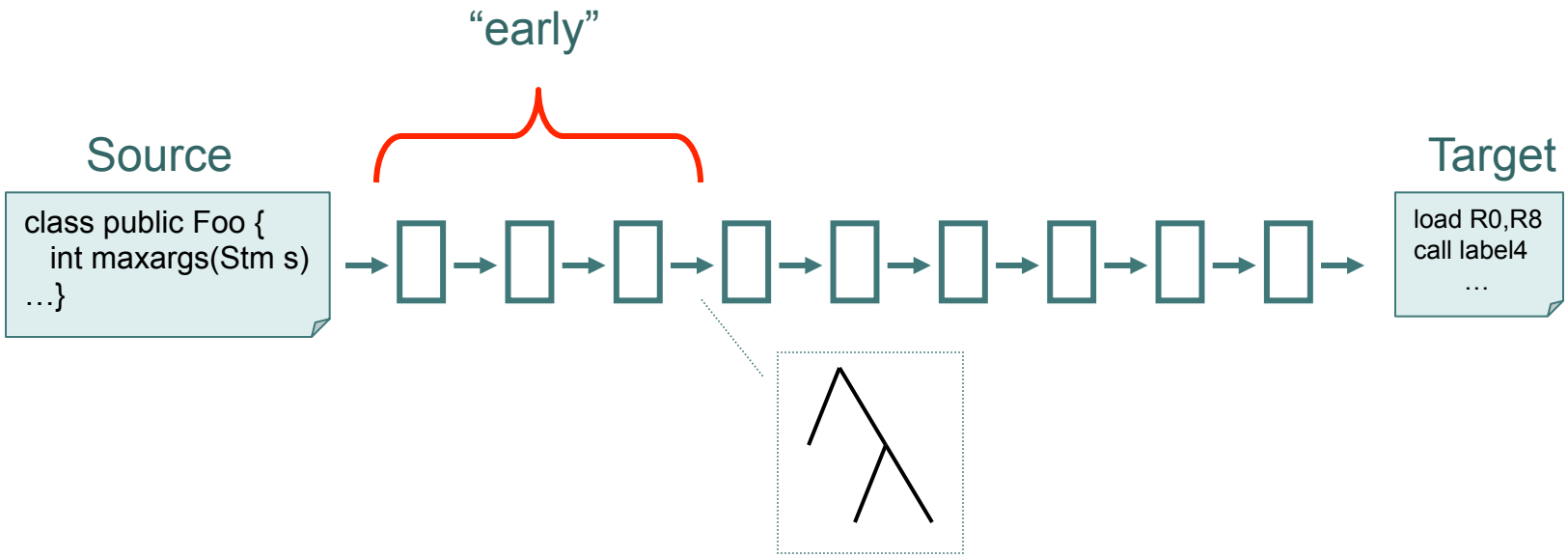
Traditional Compiler Structure



Compilers have “phases”:

- each phase has an input and an output
- each phase transforms its input code into output code
- they are typically classified into “early,” “middle,” and “late” phases which accomplish different kinds of transformations

Compiler phases



- early phases transform input sequence into tree representation (AST)
- ensure that input stream is, indeed, a program in the source language
- lexing, parsing, type-checking

What a lexer does

ascii form

c	l	a	s	s		p	u	b	l	i	c		F	o	o		{		i	n	t	...

lexer

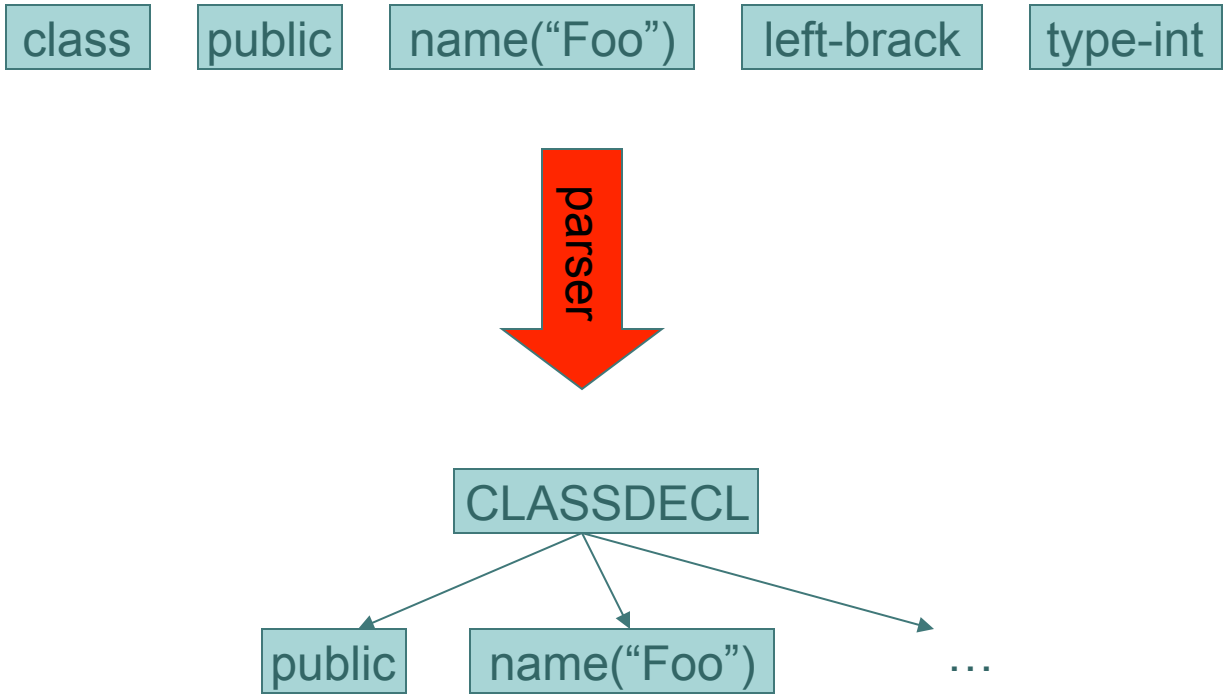
symbolic form

class	public	name(“Foo”)	left-brack	type-int	…

Key Concept: regular expressions

What parsing does

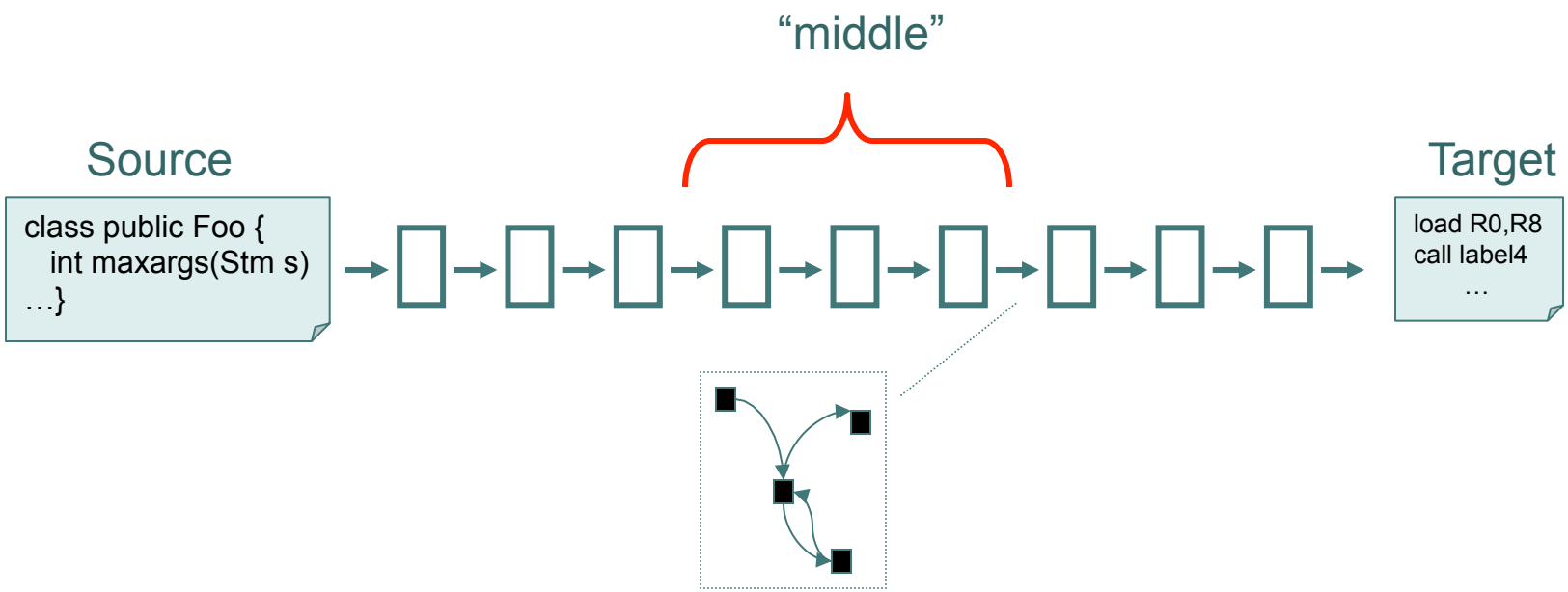
symbolic form



abstract syntax tree

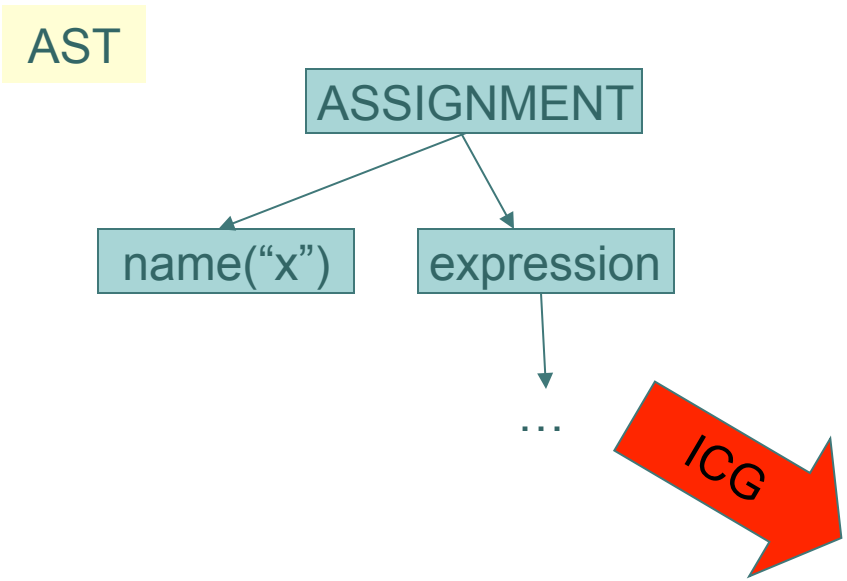
Key Concept: “Backus-Naur form” grammars (BNF)

Compiler phases



- middle phases transform AST into directed graph(s) of code chunks called basic blocks; graphs describe control & data-flow information of program
- "intermediate code generation" – code chunks look like high-level assembly language called "intermediate representation" (IR)
- IR: `r0:=r8+5; jump label3;`

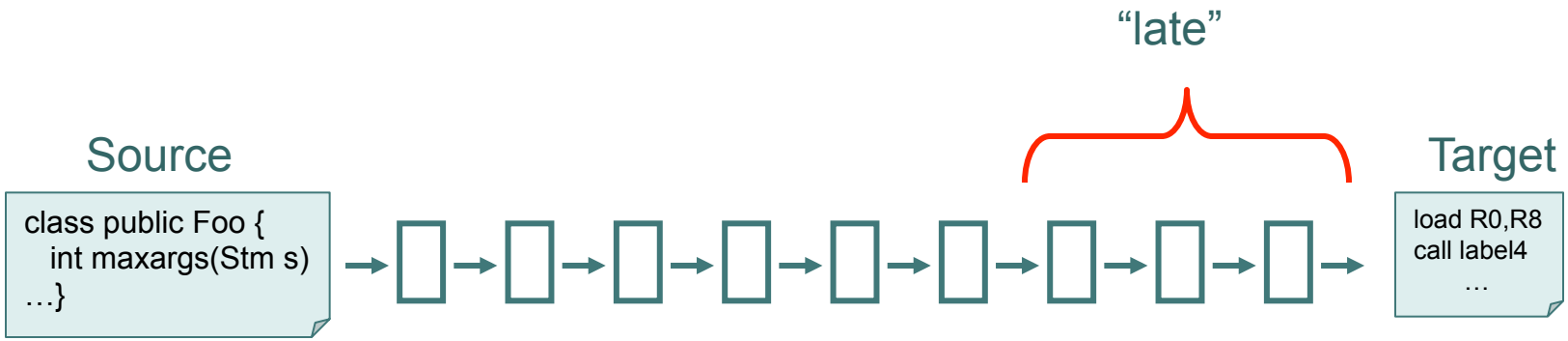
What intermediate code generation does




```
<code for expression> ;
Rx ← result
```

Key Concept: “three address code”

Compiler phases



- late phases transform IR into the target language
- "code optimization" – change IR in a meaning preserving manner which improves its performance (hopefully!)
- Code optimization has been an area of intense research for some time, but still remains something of a "black art"

What code optimization does

Intermediate Code

```
Rx ← Ry - Ry
```


```
Rx ← 0
```

Key Concept: change code beneficially; i.e., it runs faster, is smaller, etc.

Administrivia

- Textbook: “Compiler Design: Syntactic and Semantic Analysis”
  - Book is available at amazon (and in kindle format) and at the bookstore
- Programming assignments **MUST** be written in Haskell
  - 4450 is a prerequisite for this class, so you should have learned Haskell there
- **First assignment**: get the text!

Administrivia

- Grading
  - Final (30%)
  - Midterm 1 & 2 (15% + 15%)
  - Homework/programming assignments/project (40%)
- Academic Honesty: students are encouraged to <u>discuss</u> 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.

About those programming assignments…

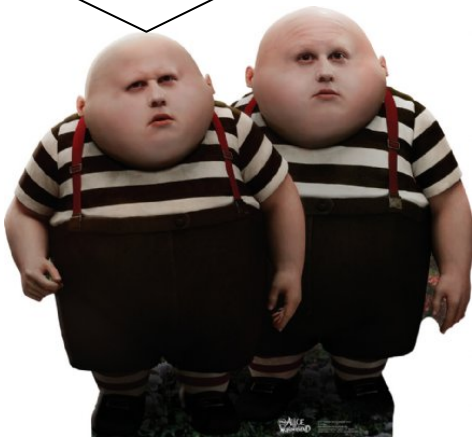
- Course has a **significant** amount of programming
- Much of this is **independent** in nature
  - Will require learning “nitty-gritty” details of certain tools on your own
  - Requires self-discipline on your part

Academic Dishonesty is Not Tolerated in my Class

A Cautionary Tale of Woe

Tweedledee explains…

Okay, I know you may think I cheated on this assignment, but I can assure you that I did not. I work very hard on these assignments and spend a lot of time trying to understand and get them right. For this assignment alone, I spent about 5 hours straight trying to get it working. **I may slightly work on them with another person, but there is no code sharing going on.** No offense to my friend, but a lot of the time I am the one that figures out the answer to these problems and **I do not give him the exact code/answer right after I figure it out. Instead I let the him try to figure it out and if he needs help, then I will assist.** If you do not decide to change the grade, that is fine. I just wanted to let you know that I have never cheated on anything in college, especially when it comes to programming since that is the one thing I rally understand how to do.



Tweedledum complains…

I do not agree with this change to my grade for the last homework. Tweedledee*, 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.*

Contrast & Compare for yourself

A sample of Tweedledee's code:

```
eval Plus rho = 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)
```

```
eval Plus rho = FunVal consVal
    where consVal :: [Value] -> Value
          consVal [v1,v2]    = plus' v1 v2
          consVal            = error "cons: 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 Tweedledum'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)
```

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
    - BTW, I have heard of google, too.
- Consequences of academic dishonesty:
  - 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
- Continued enrollment in this class implies your consent to these rules.