cs2220:
Engineering
Software

Class 28:
Past and
Future

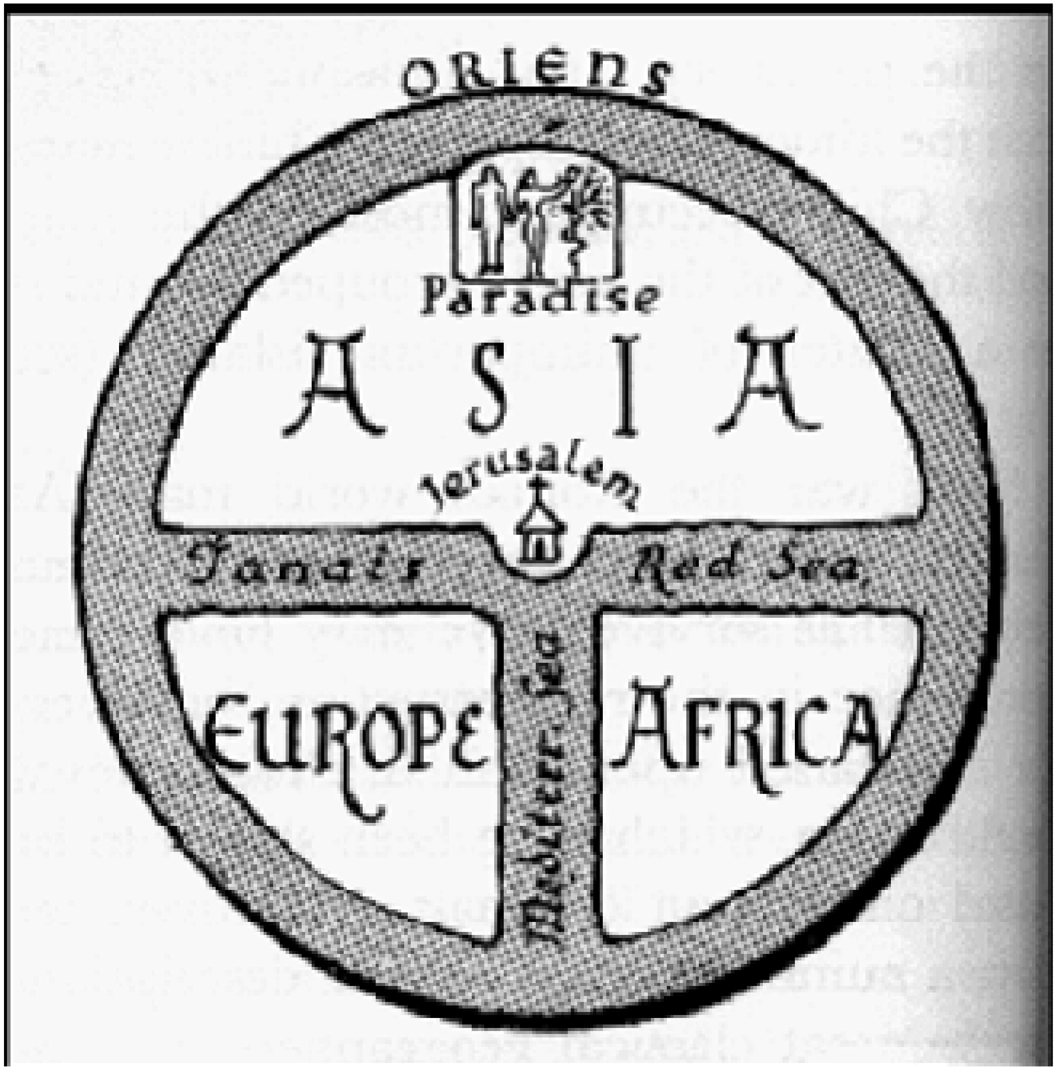
Fall 2010
UVa
David Evans

Menu

- One Word Course Summary
- Is this a Computer Science course?
- Presentations Tuesday
- Parenthesizing

One Word Course Summary?

Abstraction



Adapted from Gerard Holzmann’s FSE Slides

Abstraction in cs2220

Abstraction by Specification

Abstract away *how* by saying *what* a procedure does

Procedural Abstraction

Abstract away specific *inputs* from *what* is done

Data Abstraction

Abstract away *representation* details by specifying what you can do with something

Subtyping

Abstract away actual type details by allowing many types to be used in the same way

Concurrency Abstraction

Abstract away (some) *when* details

~~CS~~

~~cs2220 is Science~~

CS is Science

Therefore, cs2200 is CS.

Is cs2220 a *Computer Science* course?

Is Computer Science a "science"?

Geometry vs. Computer Science

- Geometry (mathematics) is about *declarative* knowledge: “what is”

  If now *CD* measures *AB*, since it also measures itself, then *CD* ***is*** a common measure of *CD* and *AB*

- Computer Science is about *imperative* knowledge: “how to”

Computer Science

"How to" knowledge:

- Ways of describing information processes (computations)

Language

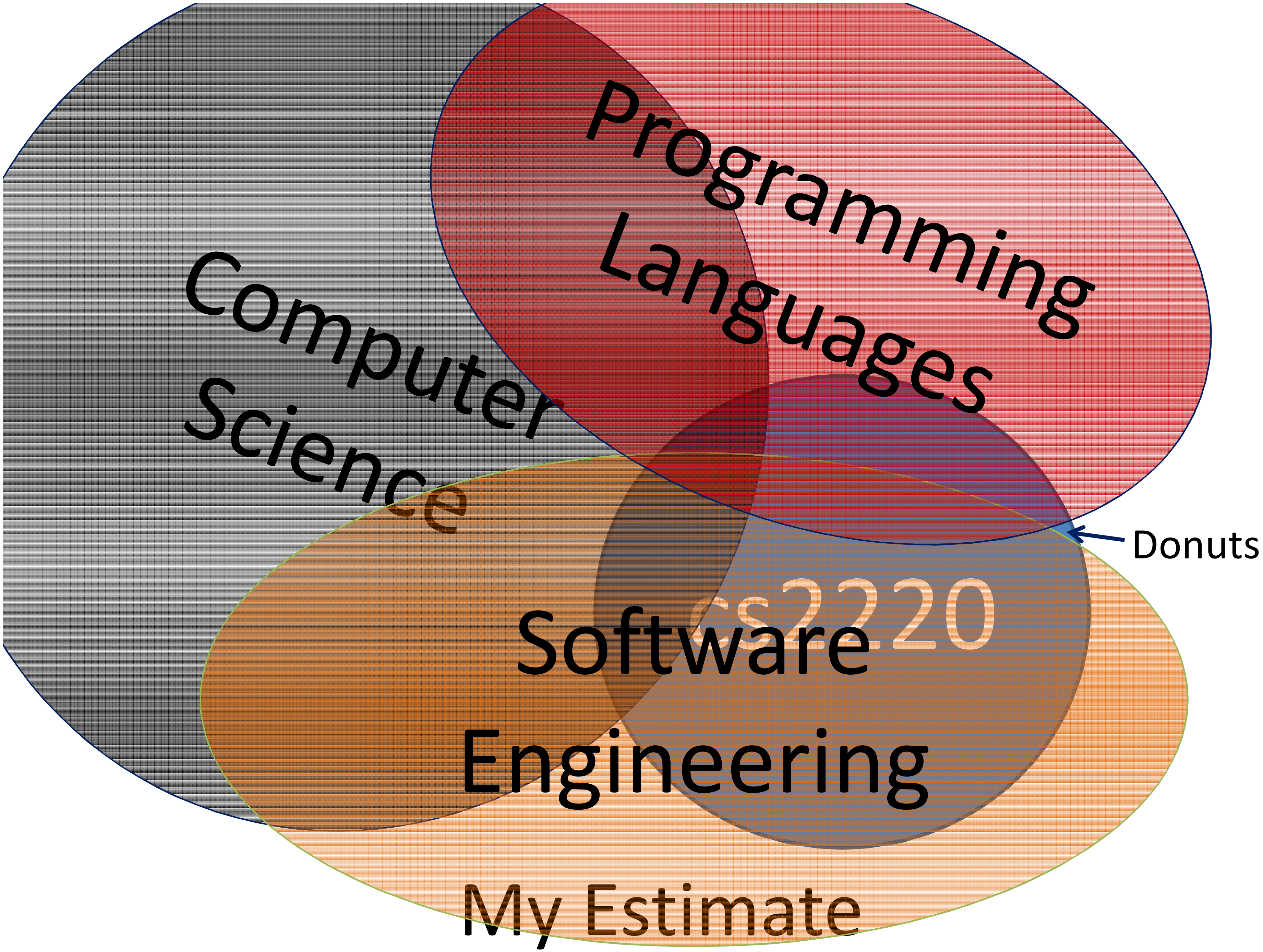
- Ways of predicting properties of information processes

Logic

What kinds of things do we want to predict?

- Ways of executing information processes

Machines


Programming Languages
Computer Science
Donuts
cs2220
Software Engineering
My Estimate

Project Deliverables

- **Project Demos/Presentations**: Tuesday, 7 December (last class)
- **Project Final Reports and Teammate Assessments**: 11:59pm, Friday, 10 December

Project Presentations

- Like all good presentations, your presentation should *tell a story* not *convey a list*.
  - Find a way to present a coherent and compelling story, not just list what you have done!
  - Provide a clear motivation for the software you have built, explain what problem it solves, and show how someone would use it to solve that problem.
- Your presentation should be ***prepared***. There should be a plan for how you will use your time effectively to get the main points across well and how to fit in what you say with your demo.

Up to 10 minutes per team.

Project Reports

- Due by 11:59pm Friday, 10 December
- **Zip file** containing all of your code
- A single **PDF file** or **paper document**:
  - An updated description of your project idea.
  - An updated design document.
  - A description of your testing strategy,
  - An explanation of what is working and what problems remain. For the problems, explain as much as you understand what the problem is.

If you do an **excellent project presentation** Tuesday (including a working demo of your project) your team **does not need to submit a project report!** (You will be notified shortly after class Tuesday.)

Parenthesizing Question

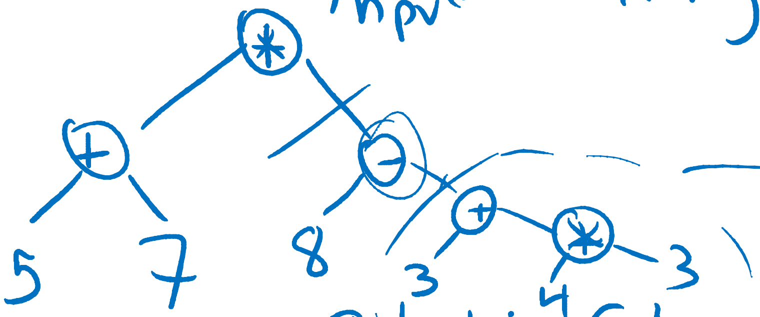
Given an arithmetic expression involving addition, subtraction, and multiplication of natural numbers, add parentheses to maximize the **value** of the expression.

Input: String e

"5+7*8-3+4*3"

["5", "+", "7", "*", "8" ... "3"]

*8 - 3+ (4*3)



→ "(5+7)*(8-3+4*3)"

Output: String

(with parens added)

```
public class ExpressionTree {                    ~~Tree[Operator, Number]~~
  //OVERVIEW: An immutable expression tree. A typical
  //    ExpressionTree is:
  // [ T => < T, op, T >          private ExprTree left, right;
  //   T => Number,               [ private int val;
  //   op => "+", "*", "-"          private Operator op;

  public ExpressionTree (~~String~~ s) {
      // REQUIRES: s is a valid arith
      // ~~EFFECTS: Initializes~~ this to the
             value-maximizing ~~tree~~ from s.
  public ExpressionTree (int val){

  }
  public ExpressionTree (ExprTree t1, Op pop, ExprTree t2)
       left = t1;  right = t2;  op = pop;
```

```
public int value() {
    if (isLeaf()) {
        return val;
    } else {

        return
            op.value(left.value(),
                     right.value());

    }
}
```

```
public class Operator {
    :

    public int value(int a, int b) {
        : a op b
    }
```