CMPSCI 250: Introduction to Computation

Lecture #1: Things, Sets and Strings
David Mix Barrington
4 September 2013

Things, Sets, and Strings

- The Mathematical Method
- Administrative Stuff
- The Objects of Mathematics
- Pseudo-Java
- Set and String Definitions
- Languages and Decision Problems

The Mathematical Method

- The basic idea of mathematics is to take some aspect of the real world, create a mathematical system that shares some of its properties, and work with that mathematical system using logic and proof.
- Since computers follow logical rules, we can better understand what they do (and what we can do with them) by applying the mathematical method.

Overview of the Course

- We'll begin this course with the basic tools of logic -- **propositions**, **predicates**, and **induction** -- with **number theory** as our object of study.
- Later we'll use induction to reason about **trees** and **search algorithms**.
- Finally we'll study **finite-state machines** and have a very brief introduction to the theory of **computability**.

Administrative Stuff

- Most of you have taken CMPSCI 187 and MATH 132 -- there are a few exceptions.
- We will have two midterms (15% each), six homeworks (25% total), ten discussion assignments (10%), Moodle quizzes (5%), clicker questions (5%), and a final (25%).
- The final counts 50% if this helps you.
- The public course web site is http://www.cs.umass.edu/~barring/cs250

More Administration

- We are using Moodle, and quizzes and homeworks must be turned in through Moodle. You will be able to see your grades.
- Clicker questions will count starting a week from today (11 September).
- If you cheat on a test, I will fail you. You may work together on homework, but what you hand in must be *yours in presentation*. If it is identical to another's work, there's a problem.

The Textbook

- The text for the course is eight chapters of a book I am writing, called A Mathematical Foundation for Computer Science. It is sold at Collective Copies in downtown Amherst, in two volumes, for about $50.
- The texts for CMPSCI 250 the last three times I taught it (Fall 2010, Spring 2011, and Spring 2012) are very similar and should be usable for this course if you check the errata pages for those offerings.

More on the Textbook

- Each chapter has eight ordinary sections and three "excursion" sections -- the latter will often be the basis of discussion exercises.
- My lectures will follow the book closely, but they will assume that you are also reading the book. Homework exercises will also be from the "Problems" in book. The "Exercises" in the book have answers in the back.

The Objects of Mathematics

- Each area of mathematics has its own set of objects. In calculus you used real numbers, and functions from reals to reals. In geometry you used points, lines, triangles, and so forth.
- In this course we will use natural numbers or **naturals**, which are the integers 0, 1, 2, 3,... going on "forever". Less often we will use negative integers, and even less often rational numbers (fractions) or other real numbers.

More Objects

- We will use the boolean values **true** and **false** (sometimes written "1" and "0").
- Other objects come from data types. Any collection of objects can be a type, and we have a supertype **thing** that includes any object we might want to consider. The book uses the example type **novelist**, consisting of any person who has ever written a novel.

Pseudo-Java

- I will assume in this course that you are familiar with programming in Java. This is because we are learning about the mathematics of computation, and many abstract concepts will be clearer when you consider examples in Java.
- In our code examples, in the book and lecture, we will use a language called pseudo-Java rather than regular Java. The main difference is in the primitive types.

Types in Pseudo-Java

- In pseudo-Java, the natural numbers are a type called **natural**, and rather than stopping at a maximum value like real Java ints or longs, they go on forever.
- Strings in pseudo-Java come from a primitive type **string** rather than being objects. The letters used in strings will depend on context.
- Object-oriented concepts will be less important than in 187 (most of our methods will be static) but recursion will be used a lot.

Set Definitions

- A **set** is any collection of things, usually all of the same type. We will allow sets of sets only if they are from a common type (e.g., sets of sets of naturals).
- We can define the **elements** of a set by listing them all (e.g., {2, 3, 4, 7}) or by set builder notation (e.g, {n: n is an even number less than 24}).
- Two sets are **equal** if every element of one is an element of the other.

Practice Clicker Question

- Which statement is false?
- (a) {3, 7, 9} = {9, 3, 7}
- (b) {author of Harry Potter} = {J. K. Rowling}
- (c) {S: S has no elements} = {0}
- (d) {3, 4, 7, 4, 2} = {2, 7, 7, 7, 3, 4, 3, 2, 3}

Answer

- Which statement is false?
- (a) {3, 7, 9} = {9, 3, 7}
- (b) {author of Harry Potter} = {J. K. Rowling}
- (c) {S: S has no elements} = {0}
- (d) {3, 4, 7, 4, 2} = {2, 7, 7, 7, 3, 4, 3, 2, 3}

More Set Definitions

- A set A is a **subset** of a set B ($A \subseteq B$) if every element of A is also in B. We have the rule that "A = B" means the same thing as "$A \subseteq B$ and $B \subseteq A$".
- A set is **empty** if it contains no elements. Any two empty sets are equal.
- The **size** of a set is the number of elements in it, if the set is finite. The set of a finite set is always a natural.

String Definitions

- A **string** is a sequence of elements from a finite set, called the **alphabet**. A binary string is a string where the alphabet is {0, 1}. The **length** of a string is the number of elements (letters) in it, and this must be a natural.
- Two strings are **equal** if they have the same length and each letter in one is equal to the corresponding letter in the other. In pseudo-Java, we use **==** for equality of strings.

More String Definitions

- The **empty string** λ is the string with no letters -- any two empty strings are equal. The symbol "λ" is not a letter, it denotes the string with no letters. (The empty string in real Java is denoted "".)
- If Σ is an alphabet, the set of all strings over Σ is called Σ*. Every string in Σ* is finite, though Σ* itself is an infinite set. For Σ = {0, 1}, we can list Σ* as the set {λ, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111,...}.

Formal Languages

- Any set of strings over Σ is called a **language** over Σ. We can define languages using any of our ways of defining sets.
- Let Σ = {0, 1}. Define the language X to be all strings that have a 1 at the beginning and the end, and 0's in the middle. We can write X as {11, 101, 1001, 10001,...} or as {w: w starts and ends with 1 and has no other 1's}. Later we'll call this language "10*1".

Decision Problems

- The **decision problem** for a language X is to input a string w (over Σ, the correct alphabet) and return a boolean that is true if $w \in X$ and false if not.
- Given a language, how difficult is it for a computer to solve its decision problem? This is the central question of **formal language theory**. We'll touch on this at the end of the course.