CMPSCI 250: Introduction to Computation

Lecture #1: Introduction, Things, Sets, and Strings
22 January 2012

Introduction: Things, Sets, and Strings

- What We're Doing Here -- The Mathematical Method
- Administrative Stuff
- The Objects of Mathematics
- Pseudo-Java
- Set Definitions: Sets, Subsets, Empty Sets, Size
- String Definitions: Alphabets, Length, λ, Σ^*
- Languages and Decision Problems

What We're Doing Here: 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.

- 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 (half of you with me!) and MATH 132. I've let in some students who have one of those but not the other.

- We will have two midterms (20% each), nine homeworks (25% total), weekly discussion assignments (10% total), and a final exam (25%). If it helps you, I will count the final exam as 50% and scale everything else down.

- The course web site (http://www.cs.umass.edu/~barring/cs250) contains a blog with important course announcements. I will email you all in an emergency, using your *@student.umass.edu* email address.

- If you cheat on a test, I will fail you. On homeworks, you may work together but what you hand in must be *yours in presentation*. If it is copied from the web, or identical to what another student turns in, we have 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 two times I taught it, Fall 2010 and Spring 2011, are very similar and should be usable for this course if you check the errata pages for those offerings.

- 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 book.

The Objects of Mathematics

- Each area of mathematics has its set of objects. In calculus you used real numbers, and functions from real numbers to real numbers. 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.

- 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

- We will assume in this course that you are familiar with programming in Java. This is because we are learning about mathematics that talks about 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. 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 range of characters in a `string` will depend on the context.

- Object-oriented concepts will be less important than in 187 (most of our methods will be static) but **recursion** will be used constantly.

Set Definitions: Sets, Subsets, Empty Set, Size

- 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. 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: Alphabets, Strings, λ, Σ*

- 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 the == operator for equality of strings.

- 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, 0000,...}.

Languages and Decision Problems

- 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 learn a notation for languages where this X will be 10*1.
- 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?