Lecture 1

Course overview and mathematical foundations

1.1 Course overview

This course is about the *theory of computation*, which deals with mathematical properties of abstract models of computation and the problems they solve. An important idea to keep in mind as we begin the course is this:

> *Computational problems, devices, and processes can themselves be viewed as mathematical objects.*

We can, for example, think about each program written in a particular programming language as a single element in the set of all programs written in that language, and we can investigate not only those programs that might be interesting to us, but also properties that must hold for all programs. We can also consider problems that some computational models can solve and that others cannot.

The notion of a *computation* is very general. Examples of things that can be viewed or described as computations include the following.

- Computers running programs (of course).
- Networks of computers running protocols.
- People performing calculations with a pencil and paper.
- Proofs of theorems (in a sense to be discussed from time to time throughout this course).
- Certain biological processes.

One could debate the definition of a computation (which is not something we will do), but a reasonable starting point for a definition is that a computation is a manipulation of symbols according to a fixed set of rules.

One interesting connection between computation and mathematics, which is particularly important from the viewpoint of this course, is that *mathematical proofs* and *computations* performed by the models we will discuss throughout this course have a similarity at the lowest level: they both involve symbolic manipulations according to fixed sets of rules. Indeed, fundamental questions about proofs and mathematical logic have played a critical role in the development of theoretical computer science.

We will begin the course working with very simple models of computation (finite automata, regular expressions, context-free grammars, and related models), and later on we will discuss more powerful computational models (especially the Turing machine model). Before we get to any of these models, however, it is appropriate that we discuss some of the mathematical foundations and definitions upon which our discussions will be based.

1.2 Sets and countability

It is assumed throughout these notes that you are familiar with naive set theory and basic propositional logic.

Naive set theory treats the concept of a set to be self-evident, and this will not be problematic for the purposes of this course—but it does lead to problems and paradoxes, such as Russell's paradox, when it is pushed to its limits. Here is one formulation of Russell's paradox, in case you are interested:

> **Russell's paradox.** Let S be the set of all sets that are not elements of themselves:
> $$S = \{T : T \notin T\}.$$
> Is it the case that S is an element of itself?
>
> If $S \in S$, then we see by the condition that a set must satisfy to be included in S that it must be that $S \notin S$. On the other hand, if $S \notin S$, then the definition of S says that S is to be included in S. It therefore holds that $S \in S$ if and only if $S \notin S$, which is a contradiction.

If you want to avoid this sort of paradox, you need to replace naive set theory with *axiomatic set theory*, which is quite a bit more formal and disallows objects such as *the set of all sets* (which is what opens the door to let in Russell's paradox).

Set theory is the foundation on which mathematics is built, so axiomatic set theory is the better choice for making this foundation sturdy. Moreover, if you really wanted to reduce mathematical proofs to a symbolic form that a computer can handle, along the lines of what was mentioned above, something along the lines of axiomatic set theory is needed.

On the other hand, axiomatic set theory is quite a bit more complicated than naive set theory and well outside the scope of this course, and there is no point in this course where the advantages of axiomatic set theory over naive set theory will appear explicitly. So, we are safe in thinking about set theory from the naive point of view—and meanwhile we can trust that everything would work out the same way if axiomatic set theory had been used instead.

The *size* of a *finite* set is the number of elements if contains. If A is a finite set, then we write $|A|$ to denote this number. For example, the empty set is denoted $\varnothing$ and has no elements, so $|\varnothing| = 0$. A couple of simple examples are

$$|\{a,b,c\}| = 3 \quad \text{and} \quad |\{1,\ldots,n\}| = n. \tag{1.1}$$

In the second example, we are assuming n is a positive integer, and $\{1,\ldots,n\}$ is the set containing the positive integers from 1 to n.

Sets can also be *infinite*. For example, the set of *natural numbers*[1]

$$\mathbb{N} = \{0,1,2,\ldots\} \tag{1.2}$$

is infinite, as are the sets of *integers*

$$\mathbb{Z} = \{\ldots,-2,-1,0,1,2,\ldots\}, \tag{1.3}$$

rational numbers

$$\mathbb{Q} = \left\{\frac{n}{m} \,:\, n,m \in \mathbb{Z},\; m \neq 0\right\}, \tag{1.4}$$

as well as the *real* and *complex numbers* (which we won't define here because these sets don't really concern us and the definitions are a bit more complicated than one might initially expect).

While it is sometimes sufficient to say that a set is infinite, we will require a more refined notion, which is that of a set being *countable* or *uncountable*.

Definition 1.1. A set A is *countable* if either (i) A is empty, or (ii) there exists an onto (or surjective) function of the form $f : \mathbb{N} \to A$. If a set is not countable, then we say that it is *uncountable*.

[1] Some people choose not to include 0 in the set of natural numbers, but for this course we will include 0 as a natural number. It is not right or wrong to make such a choice, it is only a definition. What is important is that the meaning is clear, and now that we're clear on the meaning we can move on.

These three statements are equivalent for any choice of a set A:

1. A is countable.
2. There exists a one-to-one (or injective) function of the form $g : A \to \mathbb{N}$.
3. Either A is finite or there exists a one-to-one and onto (or bijective) function of the form $h : \mathbb{N} \to A$.

It is not obvious that these three statements are actually equivalent, but it can be proved. We will, however, not discuss the proof.

Example 1.2. The set of natural numbers $\mathbb{N}$ is countable. Of course this is not surprising, but it is sometimes nice to start out with an extremely simple example. The fact that $\mathbb{N}$ is countable follows from the fact that we may take $f : \mathbb{N} \to \mathbb{N}$ to be the identity function, meaning $f(n) = n$ for all $n \in \mathbb{N}$, in Definition 1.1. We may also notice that by substituting f for the function g in statement 2 makes that statement true, and likewise for statement 3 when f is substituted for the function h.

Example 1.3. The set $\mathbb{Z}$ of integers is countable. To prove that this is so, it suffices to show that there exists an onto function of the form

$$f : \mathbb{N} \to \mathbb{Z}. \tag{1.5}$$

There are many possible choices of f that work—one good choice of a function is this one:

$$f(n) = \begin{cases} 0 & \text{if } n = 0 \\ \frac{n+1}{2} & \text{if } n \text{ is odd} \\ -\frac{n}{2} & \text{if } n \text{ is even.} \end{cases} \tag{1.6}$$

Thus, we have

$$f(0) = 0, \quad f(1) = 1, \quad f(2) = -1, \quad f(3) = 2, \quad f(4) = -2, \tag{1.7}$$

and so on. This is a well-defined function[2] of the correct form $f : \mathbb{N} \to \mathbb{Z}$, and it is onto; for every integer m, there is a natural number $n \in \mathbb{N}$ so that $f(n) = m$, as is quite evident from the pattern in (1.7).

Example 1.4. The set $\mathbb{Q}$ of rational numbers is countable, which we can prove by defining an onto function taking the form $f : \mathbb{N} \to \mathbb{Q}$. Again, there are many choices of functions that would work, and we'll pick just one.

[2] We can think of *well-defined* as meaning that there are no "undefined" values, and moreover that every reasonable person that understands the definition would agree on the values the function takes, irrespective of when and where they lived.

First, imagine that we create a sequence of ordered lists of numbers, starting like this:

List 0: 0

List 1: $-1, 1$

List 2: $-2, -\frac{1}{2}, \frac{1}{2}, 2$

List 3: $-3, -\frac{3}{2}, -\frac{2}{3}, -\frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{3}{2}, 3$

List 4: $-4, -\frac{4}{3}, -\frac{3}{4}, -\frac{1}{4}, \frac{1}{4}, \frac{3}{4}, \frac{4}{3}, 4$

List 5: $-5, -\frac{5}{2}, -\frac{5}{3}, -\frac{5}{4}, -\frac{4}{5}, -\frac{3}{5}, -\frac{2}{5}, -\frac{1}{5}, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, \frac{5}{4}, \frac{5}{3}, \frac{5}{2}, 5$

and so on. In general, for $n \geq 1$ we let the n-th list be the sorted list of all numbers that can be written as

$$\frac{k}{m}, \tag{1.8}$$

where $k, m \in \{-n, \ldots, n\}$, $m \neq 0$, and the value of the number k/m does not already appear in one of the previous lists. The lists get longer and longer, but for every natural number n it is surely the case that the corresponding list is finite.

Now consider the single list obtained by concatenating all of the lists together, starting with List 0, then List 1, and so on. Because the lists are finite, we have no problem defining the concatenation of all of them, and every number that appears in any one of the lists above will also appear in the single concatenated list. For instance, this single list begins as follows:

$$0, -1, 1, -2, -\frac{1}{2}, \frac{1}{2}, 2, -3, -\frac{3}{2}, -\frac{2}{3}, -\frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{3}{2}, 3, -4, -\frac{4}{3}, -\frac{3}{4}, \ldots \tag{1.9}$$

and naturally the complete list is infinitely long. Finally, let $f : \mathbb{N} \rightarrow \mathbb{Q}$ be the function we obtain by setting $f(n)$ to be the number in position n in the infinitely long list we just defined, starting with position 0. For example, we have $f(0) = 0$, $f(1) = -1$, and $f(8) = -3/2$.

Even though we didn't write down an explicit formula for the function f, it is a well-defined function of the proper form $f : \mathbb{N} \rightarrow \mathbb{Q}$. Moreover, it is an onto function: for any rational number you choose, you will eventually find that rational number in the list constructed above. It therefore holds that $\mathbb{Q}$ is countable. The function f also happens to be one-to-one, although we don't need to know this to conclude that $\mathbb{Q}$ is countable.

It is natural at this point to ask a question: Is every set countable? The answer is "no," and we will now see an example of an uncountable set. We will need the following definition.

Definition 1.5. For any set A, the *power set* of A is the set $\mathcal{P}(A)$ containing all subsets of A:

$$\mathcal{P}(A) = \{B : B \subseteq A\}. \tag{1.10}$$

For example, the power set of $\{1,2,3\}$ is

$$\mathcal{P}(\{1,2,3\}) = \big\{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\big\}. \tag{1.11}$$

Notice in particular that the empty set $\varnothing$ and the set $\{1,2,3\}$ itself are contained in the power set $\mathcal{P}(\{1,2,3\})$. For any finite set A, the power set $\mathcal{P}(A)$ always contains $2^{|A|}$ elements, which is one of the reasons why it is called the power set.

Also notice that there is nothing that prevents us from taking the power set of an infinite set. For instance, $\mathcal{P}(\mathbb{N})$, the power set of the natural numbers, is the set containing all subsets of $\mathbb{N}$. This set, in fact, is our first example of an uncountable set.

Theorem 1.6 (Cantor). *The power set of the natural numbers, $\mathcal{P}(\mathbb{N})$, is uncountable.*

Proof. Assume toward contradiction that $\mathcal{P}(\mathbb{N})$ is countable, which implies that there exists an onto function of the form $f : \mathbb{N} \rightarrow \mathcal{P}(\mathbb{N})$. From this function we may define[3] a subset of natural numbers as follows:

$$S = \{n \in \mathbb{N} : n \notin f(n)\}. \tag{1.12}$$

Because S is a subset of $\mathbb{N}$, it is the case that $S \in \mathcal{P}(\mathbb{N})$. We have assumed that f is onto, so there must therefore exist a natural number $m \in \mathbb{N}$ such that $f(m) = S$. Fix such a choice of m for the remainder of the proof.

Now we may ask ourselves a question: Is m contained in S? We have

$$[m \in S] \Leftrightarrow [m \in f(m)] \tag{1.13}$$

because $S = f(m)$. On the other hand, by the definition of the set S we have

$$[m \in S] \Leftrightarrow [m \notin f(m)]. \tag{1.14}$$

It therefore holds that

$$[m \in f(m)] \Leftrightarrow [m \notin f(m)], \tag{1.15}$$

or, equivalently,

$$[m \in S] \Leftrightarrow [m \notin S], \tag{1.16}$$

which is a contradiction.

Having obtained a contradiction, we conclude that our assumption that $\mathcal{P}(\mathbb{N})$ is countable was wrong, so the theorem is proved. $\square$

[3] This definition makes sense because, for each $n \in \mathbb{N}$, $f(n)$ is an element of $\mathcal{P}(\mathbb{N})$, which means it is a subset of $\mathbb{N}$. It therefore holds that either $n \in f(n)$ or $n \notin f(n)$, so if you knew what the function f was, you could determine whether or not a given number n is contained in S or not.

The method used in the proof above is called *diagonalization*, for reasons we will discuss later in the course. This is a fundamentally important proof technique in the theory of computation. Using this technique, one can prove that the sets $\mathbb{R}$ and $\mathbb{C}$ of real and complex numbers are uncountable—the central idea of the proof is the same, but the fact that some real numbers have multiple decimal representations (or, for any other choice of a base b, that some real numbers have multiple base b representations) gives a slight complication to the proof.

1.3 Alphabets, strings, and languages

The last thing we will do for this lecture is to introduce some terminology that you may already be familiar with from other courses.

First let us define what we mean by an *alphabet*. Intuitively speaking, when we refer to an alphabet, we mean a collection of symbols that could be used for writing or performing calculations. Mathematically speaking, there is not much to say—there is nothing to be gained by defining what is meant by the words *symbol*, *writing*, or *calculation* in this context, so instead we keep things as simple as possible and stick to the mathematical essence of the concept.

Definition 1.7. An *alphabet* is a finite and nonempty set.

Typical names for alphabets in this course are capital Greek letters such as Σ, Γ, and Δ. We will often use lower-case Greek letters, such as σ and τ, as variable names when we refer elements of alphabets (which we naturally will refer to as *symbols* when it is convenient to do this). Our favorite alphabet throughout this course will be the *binary alphabet* $\Sigma = \{0,1\}$.

Next we have *strings*, which are defined with respect to a particular alphabet as follows.

Definition 1.8. Let Σ be an alphabet. A *string* over the alphabet Σ is a finite, ordered sequence of symbols from Σ. The *length* of a string is the total number of symbols in the sequence.

For example, 11010 is a string of length 5 over the binary alphabet $\Sigma = \{0,1\}$. It is also a string over the alphabet $\Gamma = \{0,1,2\}$ that doesn't happen to include the symbol 2. On the other hand,

$$0101010101\cdots \quad \text{(repeating forever)} \tag{1.17}$$

is not a string because it is not finite. There are situations where it is interesting or useful to consider infinitely long sequences of symbols like this, and on a couple of occasions in this course we will encounter such sequences—but we won't call them strings.

There is a special string, called the *empty string* and denoted ε, that has no symbols in it (and therefore it has length 0). It is a string over every alphabet.

We will typically use lower-case Roman letters at the end of the alphabet, such as u, v, w, x, y, and z, as names that refer to strings. Saying that these are *names that refer to strings* is just meant to clarify that we're not thinking about u, v, w, x, y, and z as being single symbols from the Roman alphabet in this context. Because we're essentially using symbols and strings to communicate ideas about symbols and strings, there is hypothetically a chance for confusion, but once we establish some simple conventions this will not be an issue. If w is a string, we denote the length of w as $|w|$.

Finally, the term *language* refers to any collection of strings over some alphabet.

Definition 1.9. Let Σ be an alphabet. A *language* over Σ is a set of strings, with each string being a string over the alphabet Σ.

Notice that there has to be an alphabet associated with a language—we would not, for instance, consider a set of strings that included infinitely many different symbols appearing among all of the strings to be a language.

A simple but nevertheless important example of a language over a given alphabet Σ is the set of *all* strings over Σ. We denote this language as Σ^*. Another simple and important example of a language is the *empty language*, which is the set containing no strings at all. The empty language is denoted $\varnothing$ because it is the same thing as the empty set—there is no point in introducing any new notation here because we already have a notation for the empty set. The empty language is a language over an arbitrary choice of an alphabet.

In this course we will typically use capital Roman letters near the beginning of the alphabet, such as A, B, C, and D, to refer to languages.

We will see many other examples of laguages throughout the course. Here are a few examples involving the binary alphabet $\Sigma = \{0, 1\}$:

$$A = \big\{0010, 110110, 011000010110, 111110000110100010010\big\}. \tag{1.18}$$

$$B = \big\{x \in \Sigma^* \,:\, x \text{ starts with 0 and ends with 1}\big\}. \tag{1.19}$$

$$C = \big\{x \in \Sigma^* \,:\, x \text{ is a binary representation of a prime number}\big\}. \tag{1.20}$$

$$D = \big\{x \in \Sigma^* \,:\, |x| \text{ and } |x| + 2 \text{ are prime numbers}\big\}. \tag{1.21}$$

The language A is finite, B and C are not finite (they both have infinitely many strings), and at this point in time nobody knows if D is finite or infinite (because the so-called *twin primes conjecture* remains unproved).