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 classify computational problems in terms of which models can solve them and which 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.
- Mathematical proofs.
- Certain biological processes.

The precise definition of what constitutes a computation can be debated (which is something we will not 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 low-level similarity: 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 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, such as 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 will be assumed throughout these notes that you are familiar with naive set theory and basic propositional logic.

Basic set theory

Naive set theory treats the concept of a set to be self-evident. 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 by the condition that a set must satisfy to be included in S, 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, something along the lines of axiomatic set theory would be needed.

On the other hand, axiomatic set theory is more complicated than naive set theory, and it is also outside of the scope of this course. Fortunately, there will be no specific situations that arise in this course for which the advantages of axiomatic set theory over naive set theory explicitly appear, and for this reason 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.

Countability

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*

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

and *rational numbers*

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

The sets of *real* and *complex numbers* are also infinite, but we will not define these sets here because they will not play a major role in this course 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 in these notes 0 is included among the natural numbers. It is not right or wrong to make such a choice, it is only a definition, and what is most important is that we make clear the precise meaning of the terms we use.

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 a 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. Notice that substituting f for the function g in statement 2 above makes that statement true, and likewise for statement 3 when f is substituted for the function h.

The function $f(n) = n$ is not the only function that works to establish that $\mathbb{N}$ is countable. For example, the function

$$f(n) = \begin{cases} n+1 & \text{if } n \text{ is even} \\ n-1 & \text{if } n \text{ is odd} \end{cases} \tag{1.5}$$

also works. The first few values of this function are

$$f(0) = 1, \quad f(1) = 0, \quad f(2) = 3, \quad f(3) = 2, \tag{1.6}$$

and it is not too hard to see that this function is both one-to-one and onto. There are (infinitely) many other choices of functions that work equally well to establish that $\mathbb{N}$ is countable.

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

$$f : \mathbb{N} \to \mathbb{Z}. \tag{1.7}$$

As in the previous example, there are many possible choices of f that work, one of which is this function:

$$f(n) = \begin{cases} 0 & \text{if } n = 0 \\ \frac{n+1}{2} & \text{if } n \text{ is odd} \\ -\frac{n}{2} & \text{if } n \text{ is even.} \end{cases} \tag{1.8}$$

Thus, we have

$$f(0) = 0, \quad f(1) = 1, \quad f(2) = -1, \quad f(3) = 2, \quad f(4) = -2, \tag{1.9}$$

and so on. This is a well-defined[2] function of the correct form $f : \mathbb{N} \rightarrow \mathbb{Z}$, and it is onto: for every integer m, there is a natural number $n \in \mathbb{N}$ so that $f(n) = m$, as is evident from the pattern exhibited by (1.9).

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

First, imagine that we create a sequence of finite sequences (or lists)

$$(L_0, L_1, L_2, \ldots) \tag{1.10}$$

that starts like this:

$$L_0 = (0), \tag{1.11}$$

$$L_1 = (-1, 1), \tag{1.12}$$

$$L_2 = \left(-2, -\frac{1}{2}, \frac{1}{2}, 2\right), \tag{1.13}$$

$$L_3 = \left(-3, -\frac{3}{2}, -\frac{2}{3}, -\frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{3}{2}, 3\right), \tag{1.14}$$

$$L_4 = \left(-4, -\frac{4}{3}, -\frac{3}{4}, -\frac{1}{4}, \frac{1}{4}, \frac{3}{4}, \frac{4}{3}, 4\right), \tag{1.15}$$

$$L_5 = \left(-5, -\frac{5}{2}, -\frac{5}{3}, -\frac{5}{4}, -\frac{4}{5}, -\frac{3}{5}, -\frac{2}{5}, -\frac{1}{5}, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, \frac{5}{4}, \frac{5}{3}, \frac{5}{2}, 5\right). \tag{1.16}$$

In general, for any $n \geq 1$ we let L_n be the sorted list of all numbers that can be written as k/m for $k, m \in \{-n, \ldots, n\}$ satisfying $m \neq 0$, as well as not being included in any of the previous lists L_j, for $j < n$. The sequences get longer and longer, but for every natural number n it is surely the case that L_n is a *finite* sequence.

Now consider the single sequence S obtained by concatenating together the sequences L_0, L_1, L_2, etc. The beginning of S looks like this:

$$S = \left(0, -1, 1, -2, -\frac{1}{2}, \frac{1}{2}, 2, -3, -\frac{3}{2}, -\frac{2}{3}, -\frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{3}{2}, 3, -4, -\frac{4}{3}, -\frac{3}{4}, \ldots\right). \tag{1.17}$$

It is a well-defined sequence because each of the lists L_j is finite. (In contrast, it would *not* be clear what was meant by the concatenation of two or more *infinite* sequences.)

[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 or where they consider the definition.

Finally, let

$$f : \mathbb{N} \to \mathbb{Q} \tag{1.18}$$

be the function we obtain by setting $f(n)$ to be the number in position n of the sequence S, assuming that S begins with position 0. For example,

$$f(0) = 0, \quad f(1) = -1, \quad \text{and} \quad f(8) = -3/2. \tag{1.19}$$

Even though we did not write down an *explicit formula* for the function f, it is a well-defined function of the proper form (1.18).

Most importantly, f is an onto function—for any rational number you choose, you will eventually find that rational number in the list constructed above. It follows from the fact that f is onto that $\mathbb{Q}$ is countable.

The function f also happens to be one-to-one, but we do not need to know this to conclude that $\mathbb{Q}$ is countable.

An uncountable set

It is natural at this point to ask a question: Is every set countable? The answer is no, and we will soon see an example of an uncountable set. First, however, 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:

$$\mathcal{P}(A) = \{B : B \subseteq A\}. \tag{1.20}$$

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

$$\mathcal{P}(\{1,2,3\}) = \{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}. \tag{1.21}$$

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 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, the power set of the natural numbers $\mathcal{P}(\mathbb{N})$ is the set containing all subsets of $\mathbb{N}$.

The set $\mathcal{P}(\mathbb{N})$ is, 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, so that there exists an onto function of the form $f : \mathbb{N} \to \mathcal{P}(\mathbb{N})$. From this function we may define a subset of natural numbers as follows:

$$S = \{n \in \mathbb{N} : n \notin f(n)\}. \tag{1.22}$$

This definition makes sense because, for each $n \in \mathbb{N}$, $f(n)$ is an element of $\mathcal{P}(\mathbb{N})$, which is equivalent to $f(n)$ being a subset of $\mathbb{N}$.

Now, the set S is a subset of $\mathbb{N}$, or equivalently, $S \in \mathcal{P}(\mathbb{N})$. We have assumed that f is onto, and therefore there must 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.

We will now consider whether or not the number m is contained in the set S. The statement that $m \in S$ is equivalent to the statement that $m \in f(m)$ by the requirement that $S = f(m)$. The statement that $m \in f(m)$ is, however, equivalent to the statement that $m \notin S$ by the definition of the set S. That is to say, $m \in S$ if and only if $m \notin S$, which is a contradiction.

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

There is a technique at work in this proof, known as *diagonalization*. It is a fundamentally important technique in the theory of computation, and we will see instances of it later.

Using this technique, one can also prove that the sets $\mathbb{R}$ and $\mathbb{C}$ of real and complex numbers are uncountable. The central idea is the same as the proof above, but there is a small inconvenience caused by the fact that the natural approach of associating real numbers with infinite sequences of digits, as in the decimal or binary representation we are familiar with, does not define a one-to-one and onto function—there will always be real numbers having multiple representations. If you are interested, try to make it work!

1.3 Alphabets, strings, and languages

The last thing we will do for this lecture is to introduce some basic terminology that will be used throughout the course.

Alphabets

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, encoding information, 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*, *encoding*, or *calculating* 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 used for alphabets in this course are capital Greek letters such as Σ, Γ, and Δ. We refer to elements of alphabets as *symbols*, and we will often use lower-case letters appearing at the beginning of the Roman alphabet, such as a, b, c, and d, as variable names when referring to symbols.

Our favorite alphabet in these notes will be the *binary alphabet* $\Sigma = \{0,1\}$.

Sometimes we will refer to the *unary alphabet* $\Sigma = \{0\}$ that has just one symbol. Although it is not a very efficient choice for encoding information, the unary alphabet is a valid alphabet, and we will find good uses for it.

We can also think about alphabets more abstractly. For instance, we may consider the alphabet

$$\Sigma = \{0,1,\dots,n-1\}, \tag{1.23}$$

where n is a large positive integer, like $n = 1{,}000{,}000$, or it may even be the case that n is a hypothetical positive integer that has not been explicitly chosen. Of course we do not need to dream up different symbols in order to contemplate such an alphabet in a mathematical sense.

Alphabets can also contain other symbols, such as

$$\Sigma = \{A,B,C,\dots,Z\}, \quad \Sigma = \{\heartsuit,\diamondsuit,\spadesuit,\clubsuit\}, \quad \text{or} \quad \Sigma = \{\text{ZoSo}, \text{☘}, \text{⚭}, \text{⦶}\}, \tag{1.24}$$

but from the viewpoint of this course the actual symbols that appear in alphabets will not really matter all that much. From a mathematical point of view, there is nothing special about the alphabets $\{\heartsuit,\diamondsuit,\spadesuit,\clubsuit\}$ and $\{\text{ZoSo}, \text{☘}, \text{⚭}, \text{⦶}\}$ that distinguishes them from the alphabet $\{0,1,2,3\}$. For this reason, when it is convenient to do so, we may assume without loss of generality that a given alphabet we are working with takes the form (1.23) for some positive integer n.

On the other hand, when it is convenient for us to choose symbols other than those suggested above, meaning 0, 1, 2, etc., we will not hesitate to do that. Sometimes it is very convenient to pick different symbols for alphabets, such as in a construction (or formal description) of a machine or algorithm that performs a complicated task, as we will see. A simple example is that we often choose the symbol # to suggest a separator between strings not containing this symbol.

Strings

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

Definition 1.8. A *string* over an alphabet Σ is a finite sequence of symbols drawn from Σ. The *length* of a string is the total number of symbols it contains, counting repetitions.

For example, 11010 is a string of length 5 over the binary alphabet $\Sigma = \{0, 1\}$. It is also a string over the alphabet $\Gamma = \{0, 1, 2\}$; it just does not happen to include the symbol 2.

On the other hand,

$$0101010101 \cdots \quad \text{(repeating forever)} \tag{1.25}$$

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, but in this course we will not refer to such things as *strings*.

There is a special string, called the *empty string*, that has length zero, meaning that it is an empty sequence with no symbols in it. We will denote this string by ε. (You may find that other writers choose a different symbol, such as λ, to represent the empty string.)

We will typically use lower-case letters appearing near the end of the Roman 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 are not thinking about u, v, w, x, y, and z as being single symbols from the Roman alphabet in this context. Because we are 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 $|w|$.

Languages

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

Definition 1.9. A *language* over an alphabet Σ is any set of strings, with each one 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 includes 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 letters near the beginning of the Roman alphabet, such as A, B, C, and D, to refer to languages. Sometimes we will also give special languages special names, such as PAL and DIAG, as you will see later.

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

$$A = \big\{0010, 110110, 011000010110, 111110000110100010010\big\}, \tag{1.26}$$
$$B = \big\{x \in \Sigma^* \,:\, x \text{ starts with 0 and ends with 1}\big\}, \tag{1.27}$$
$$C = \big\{x \in \Sigma^* \,:\, x \text{ is a binary representation of a prime number}\big\}, \tag{1.28}$$
$$D = \big\{x \in \Sigma^* \,:\, \lvert x\rvert \text{ and } \lvert x\rvert + 2 \text{ are prime numbers}\big\}. \tag{1.29}$$

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).