Lecture 1

Course Roadmap and Historical Perspective

The goal of this course is to understand the foundations of computation. We will ask some very basic questions, such as

- What does it mean for a function to be computable?
- Are there any noncomputable functions?
- How does computational power depend on programming constructs?

These questions may appear simple, but they are not. They have intrigued scientists for decades, and the subject is still far from closed.

In the quest for answers to these questions, we will encounter some fundamental and pervasive concepts along the way: *state, transition, nondeterminism, reduction*, and *undecidability*, to name a few. Some of the most important achievements in theoretical computer science have been the crystallization of these concepts. They have shown a remarkable persistence, even as technology changes from day to day. They are crucial for every good computer scientist to know, so that they can be recognized when they are encountered, as they surely will be.

Various models of computation have been proposed over the years, all of which capture some fundamental aspect of computation. We will concentrate on the following three classes of models, in order of increasing power:

D. C. Kozen, *Automata and Computability*

(i) finite memory: finite automata, regular expressions;

(ii) finite memory with stack: pushdown automata;

(iii) unrestricted:

- Turing machines (Alan Turing [120]),
- Post systems (Emil Post [99, 100]),
- μ-recursive functions (Kurt Gödel [51], Jacques Herbrand),
- λ-calculus (Alonzo Church [23], Stephen C. Kleene [66]),
- combinatory logic (Moses Schönfinkel [111], Haskell B. Curry [29]).

These systems were developed long before computers existed. Nowadays one could add PASCAL, FORTRAN, BASIC, LISP, SCHEME, C++, JAVA, or any sufficiently powerful programming language to this list.

In parallel with and independent of the development of these models of computation, the linguist Noam Chomsky attempted to formalize the notion of *grammar* and *language*. This effort resulted in the definition of the *Chomsky hierarchy*, a hierarchy of language classes defined by grammars of increasing complexity:

(i) right-linear grammars;

(ii) context-free grammars;

(iii) unrestricted grammars.

Although grammars and machine models appear quite different on a superficial level, the process of parsing a sentence in a language bears a strong resemblance to computation. Upon closer inspection, it turns out that each of the grammar types (i), (ii), and (iii) are equivalent in computational power to the machine models (i), (ii), and (iii) above, respectively. There is even a fourth natural class called the *context-sensitive* grammars and languages, which fits in between (ii) and (iii) and which corresponds to a certain natural class of machine models called *linear bounded automata*.

It is quite surprising that a naturally defined hierarchy in one field should correspond so closely to a naturally defined hierarchy in a completely different field. Could this be mere coincidence?

Abstraction

The machine models mentioned above were first identified in the same way that theories in physics or any other scientific discipline arise. When studying real-world phenomena, one becomes aware of recurring patterns and themes that appear in various guises. These guises may differ substantially on a superficial level but may bear enough resemblance to one another to suggest that there are common underlying principles at work. When this happens, it makes sense to try to construct an abstract model that captures these underlying principles in the simplest possible way, devoid of the unimportant details of each particular manifestation. This is the process of *abstraction*. Abstraction is the essence of scientific progress, because it focuses attention on the important principles, unencumbered by irrelevant details.

Perhaps the most striking example of this phenomenon we will see is the formalization of the concept of *effective computability*. This quest started around the beginning of the twentieth century with the development of the *formalist* school of mathematics, championed by the philosopher Bertrand Russell and the mathematician David Hilbert. They wanted to reduce all of mathematics to the formal manipulation of symbols.

Of course, the formal manipulation of symbols is a form of computation, although there were no computers around at the time. However, there certainly existed an awareness of computation and algorithms. Mathematicians, logicians, and philosophers knew a constructive method when they saw it. There followed several attempts to come to grips with the general notion of *effective computability*. Several definitions emerged (Turing machines, Post systems, etc.), each with its own peculiarities and differing radically in appearance. However, it turned out that as different as all these formalisms appeared to be, they could all simulate one another, thus they were all computationally equivalent.

The formalist program was eventually shattered by Kurt Gödel's incompleteness theorem, which states that no matter how strong a deductive system for number theory you take, it will always be possible to construct simple statements that are true but unprovable. This theorem is widely regarded as one of the crowning intellectual achievements of twentieth century mathematics. It is essentially a statement about computability, and we will be in a position to give a full account of it by the end of the course.

The process of abstraction is inherently mathematical. It involves building models that capture observed behavior in the simplest possible way. Although we will consider plenty of concrete examples and applications of these models, we will work primarily in terms of their mathematical properties. We will always be as explicit as possible about these properties.

We will usually start with definitions, then subsequently reason purely in terms of those definitions. For some, this will undoubtedly be a new way of thinking, but it is a skill that is worth cultivating.

Keep in mind that a large intellectual effort often goes into coming up with just the right definition or model that captures the essence of the principle at hand with the least amount of extraneous baggage. After the fact, the reader often sees only the finished product and is not exposed to all the misguided false attempts and pitfalls that were encountered along the way. Remember that it took many years of intellectual struggle to arrive at the theory as it exists today. This is not to say that the book is closed—far from it!