Logic and Proof Hilary 2024

Logic in Computer Science

James Worrell

1 Introduction

When I was a student, even the topologists regarded mathematical logicians as living in outer space. Today the connections between logic and computers are a matter of engineering practice at every level of computer organization.

Martin Davis. *Influences of Mathematical Logic on Computer Science.*

1.1 This Course

Logic is fundamental to computer science. This is not surprising, given that computers are built from Boolean circuits. However, what has been called the *unusual effectiveness of logic in computer science* goes far beyond hardware design: it applies, among other things, to knowledge representation, programming-language theory, automated verification, complexity theory, databases, and constraint solving. The role of logic in computer science has been compared to that of calculus in physics and engineering.

This course focusses on the foundations of logic rather than its computer-science applications. We mostly leave applications to subsequent courses in the areas mentioned above. However our emphasis is on the parts of the logic that are most relevant to computer science. In particular, we study questions of decidability using notions from the *Models of Computation* course, including finite-state automata and Post's Correspondence Problem. We will also present the satisfiability problem in propositional logic as a prototypical search problem, making connections with the first-year *Algorithms* course.

1.2 A Very Brief History of Logic

The study of logic arose from a desire to understand reasoning and argumentation. Aristotle (384–322 BC) compiled a list of *syllogisms*, which can be seen as arguments in which the conclusion follows from the hypotheses *merely by virtue of the meaning of the words if, then, and, or, is, all, are, some and none.* For example,

$$\frac{\begin{array}{l}\text{All students are lazy people}\\ \text{All lazy people are happy}\end{array}}{\text{All students are happy}} \qquad \frac{\begin{array}{l}\text{All } S \text{ are } L\\ \text{All } L \text{ are } H\end{array}}{\text{All } S \text{ are } H}$$

While Aristotle wrote down a compendium of valid arguments, Leibniz (1646–1716) envisioned a system of rules (or *calculus*) by which arguments could be systematically constructed and tested for validity. An important step toward this goal was worked out by George Boole (1815–1864) who proposed a set of equational rules for *propositional logic*. A particularly influential contribution of Boole was to give an algebraic formulation of logic. Boole's work was picked up by William Stanley Jevons (1835–1882) who built a mechanical computer, the *logic piano*, to carry out logical deductions.

A more expressive and powerful system than propositional logic, called *predicate logic*, was invented independently by Gottlob Frege (1848–1925) and Charles Sanders Pierce (1839–1914), partly motivated by problems in the foundations of mathematics. Propositional logic and predicate logic are the two main logical systems that we study in this course.

In the first half of the twentieth century logic played a central role in the study of the foundations of mathematics. Russell (1872–1970) and Whitehead (1861–1947) attempted to show in their *Principia Mathematica* how theorems in set theory, arithmetic, real analysis and geometry could be derived from well-defined axioms and rules of inference within a formal system of predicate logic. One of the most celebrated outcomes of this research program is a negative result, by Kurt Gödel (1906–1978), who showed that no logical system of arithmetic could be both *consistent* (free of contradiction) and *complete* (capable of proving all true facts). Shortly thereafter Alonzo Church (1903–1995) and Alan Turing (1912–1954) independently showed that there is no algorithm for the *Entscheidungsproblem*, that is, the problem of deciding the validity of a given logic statement. The formulation and proof of this last result led directly to the notions of *Turing machine* and λ-*calculus*, thus laying the foundations of theoretical computer science. We will give a proof of this result in this course, building on things you have learned in Models of Computation.

More recent developments in logic have been heavily influenced by computer science. We highlight two among many important contributions. Claude Shannon (1916–2001) showed how to use electrical switches to compute Boolean functions, and is regarded as the founder of digital circuit design. Alan Robinson (1925–) discovered *resolution* and *unification*, thus contributing to the foundations of automated reasoning and logic programming. Resolution will be one of the main proof systems considered in this course.

A form of resolution underlies modern *SAT solvers*–computer programs for determining satisfiability of propositional formulas. The dramatic improvement in the performance of SAT solvers over the last 20 years has led to their successful application in areas such as automated verification, cryptography, and AI planning.