Vorwort

Die Vorlesung behandelt logische Strukturen und Methoden, die in vielen Bereichen der Informatik angewendet werden. Zu den Anwendungsgebieten zählen Datenbanken, Künstliche Intelligenz, Programmiersprachen sowie die Verifikation von Hardware und Software. Zu den von uns behandelten logischen Strukturen gehören Aussagenlogik, Lambda-Kalkül, Prädikatenlogik und Hoare-Logik. Dabei lernen wir grundlegende Konzepte wie Rekursion, Induktion, Syntax, Semantik, Deduktion, Typen und Berechenbarkeit kennen. Die Vorlesung gehört zur Theoretischen Informatik. Das bedeutet, dass wir mit mathematischen Methoden idealisierte Modelle von Phänomenen untersuchen, die in der Informatik von Bedeutung sind.

Die Themen der Vorlesung sind Gegenstand einer wissenschaftlichen Disziplin, die man Logik nennt. Man unterscheidet zwischen philosophischer, mathematischer und computationaler Logik, drei Teildisziplinen, die zeitlich nacheinander entstanden sind. Philosophische Logik wurde bereits im Altertum betrieben. Mathematische Logik ist als Teilgebiet der Mathematik ab der zweiten Hälfte des neunzehnten Jahrhunderts entstanden (Boole, Frege, Hilbert, Russel, Herbrand, Gödel, Tarski, Church, Kleene, Turing) und sah ihre Hauptaufgabe zunächst in der Fundierung der Mathematik. Computationale Logik ist eine Fortführung der Mathematischen Logik, bei der jedoch Anwendungen in der Informatik und operationale Methoden im Vordergrund stehen.

Wir wollen kurz versuchen, eine zentrale Fragestellung der Computationalen Logik zu erklären. Dabei geht es darum, wichtige Eigenschaften von Computersystemen automatisch oder halbautomatisch zu verifizieren. Dazu beschreibt man zunächst die relevanten Aspekte des Sytems mit einem mathematischen *Modell*. Darauf aufbauend kann man *Eigenschaften* des Modells formulieren. Da Modell und Eigenschaften mathematisch präzise formuliert sind, steht für jede Eigenschaft fest, ob sie erfüllt oder verletzt ist. Der nächste Schritt ist die Konstruktion eines Software-Werkzeugs, das uns bei der *Verifikation* der Eigenschaften unterstützt. Unter Verifikation verstehen wir die Klärung der Frage, ob ein Modell eine Eigenschaft erfüllt oder verletzt. Um Verifikationswerkzeuge entwickeln zu können, benötigen wir einerseits implementierbare Verifikationsmethoden, und andererseits eine Beschreibungssprache für Modelle und Eigenschaften, damit wir diese dem Werkzeug mitteilen können.

Beschreibungssprachen und Verifikationsmethoden sind ein zentraler Gegenstand der Logik. Statt von Beschreibungssprachen spricht man von *logischen Sprachen*. Logische Sprachen unterscheiden sich in ihrer *Expressivität* (Ausdrucksstärke) und in der *Komplexität* ihrer Verfikationsprobleme. Für Anwendungen versucht man logische Sprachen so zu entwickeln, dass die relevanten Verifikationsprobleme möglichst effizient zu lösen sind.

Logische Sprachen haben viele Gemeinsamkeiten mit Programmiersprachen. Schon bevor es Computer gab, haben Logiker idealisierte Programmiersprachen betrachtet, deren Programme Funktionen im mathematischen Sinne berechnen.

Die Programmiersprache Standard ML bietet uns eine praktische Umsetzung einiger grundlegender logischer Konzepte (insbesondere Funktionen, Typen, Rekursion und abstrakte Syntax). Oft erweist es sich als hilfreich, neu eingeführte Konzepte in Standard ML zu realisieren, um mit ihnen in vertrauter Umgebung experimentieren zu können.

