# Einleitung

## Was ist theoretische Informatik?

»It is unnecessary to design various new machines to do various computing processes. They can all be done with one digital computer, suitably programmed for each case.«

(Es ist nicht nötig, immer neue, verschiedene Maschinen für verschiedene Rechenprozesse zu entwerfen. Sie können alle von einem einzigen digitalen Computer bearbeitet werden, der für jeden Fall geeignet programmiert wird.)

Alan Turing, in »Computing Machinery and Intelligence«, Mind, 1950

Für uns heute, die wir gewohnt sind, Computer als allgegenwärtige und universelle Werkzeuge anzusehen, ohne die unsere Alltag fast undenkbar wäre, bedeuten diese Sätze nicht viel mehr als eine Selbstverständlichkeit. Aber als Alan Turing sie 1950 in der philosophischen Zeitschrift *Mind* veröffentlichte, dürften die meisten seiner Leser unter dem Wort »Computer« noch eine menschliche Person, die Rechenaufgaben mit Papier und Bleistift löst, verstanden haben. Natürlich gab es bereits mechanische und elektromechanische Rechenmaschinen auf dem Markt, aber dies waren Maschinen, die ausschließlich addieren, subtrahieren, multiplizieren oder dividieren konnten. Die Idee, dass es eine einzige Maschine für eine Vielzahl von Aufgaben geben könnte, war für die breite Öffentlichkeit neu. Und was sollte es bedeuten, diese Maschine geeignet zu »programmieren«?

Die theoretischen Grundlagen für seine Aussagen von 1950 hatte Turing bereits 1936 in einem bahnbrechenden Aufsatz gelegt, in dem er ein einfaches mathematisches Modell einer »Logic Computing Machine« entwarf und dann zeigte, dass so eine Maschine tatsächlich universell ist, also alle möglichen Aufgaben übernehmen kann, wenn man sie nur mit den richtigen Eingabedaten füttert. Gleichzeitig bewies er aber auch, dass auch der universelle Computer nicht alles und jedes berechnen kann, indem er Probleme angab, die mit einer solchen Maschine prinzipiell nicht gelöst werden können. Die Vision einer universellen Maschine hatten vor Turing schon andere gehabt, aber erst Turing konnte beweisen, dass es sie tatsächlich gibt, und zeigen, welche Konponenten sie benötigt. In den fünfziger Jahren gingen dann Turing und seine Kollegen in England und den USA daran, die Vision Wirklichkeit werden zu lassen.

Für mich verkörpert Alan Turings Aufsatz von 1936 »On Computable Numbers, with an Application to the Entscheidungsproblem« die Quintessenz der theoretischen Informatik: Mit Hilfe eines mathematischen Modells wird eine abstrakte Maschine beschrieben, die mächtig genug ist, um im Prinzip alle Aufgaben lösen zu können, die auch heutige Computer lösen können. Andererseits ist die Maschine so einfach gebaut, dass es möglich ist, sowohl ihre Universalität als auch ihre Beschränkungen mit mathematischen Mitteln zu beweisen. Die theoretische Informatik nutzt also mathematische Konzepte und Beweisverfahren, um allgemeingültige Aussagen über das Wesen und die Fähigkeiten von Computern

und den von ihnen bearbeiteten Problemen treffen zu können. Deswegen besitzt die theoretische Informatik einen hohen Mathematisierungs- und Abstraktionsgrad, der auf viele abschreckend wirkt und sie nicht gerade zum Lieblingsfach angehender InformatikerInnen im Grundstudium werden lässt. Aber die Abstraktion ist kein Selbstzweck, sondern hat eine wichtige Konsequenz: Die Aussagen der theoretischen Informatik sind unabhängig von der verwendeten Hard- und Software, von CPU-Taktfrequenzen, Betriebssystemen und Programmiersprachen. Was man hier einmal bewiesen hat, das gilt (und hat durchaus auch Konsequenzen für den Alltag eines Informatikers).

## Über dieses Buch

Machen wir uns nichts vor – die Wahrscheinlichkeit, dass Sie dieses Buch gekauft haben, um den Stoff einer Vorlesung in theoretischer Informatik an einer Hochschule oder Universität besser zu verstehen, ist relativ hoch. Genau dabei möchte Ihnen dieses Buch helfen, und zwar durch möglichst ausführliche Erklärungen, viele komplett ausgeführte Rechnungen und, wenn möglich, bildliche Darstellungen. Das bedeutet aber auch, dass dieses Buch nicht ohne einen gewissen mathematischen Formalismus auskommen kann. Die theoretische Informatik liegt nun einmal an der Schnittstelle zwischen Informatik und Mathematik und macht ausgiebig von den Methoden und Begriffen der Mathematik Gebrauch. Ich habe aber - vor allem zu Beginn - versucht, Ihnen den Einstieg in den Formalismus der theoretischen Informatik zu erleichtern, indem ich viele Formeln auch noch einmal in Umgangssprache erklärt habe. Natürlich wäre es auch möglich, die wesentlichen Konzepte der theoretischen Informatik »formelfrei« zu besprechen, aber ein solches Buch wird Ihnen nicht dabei helfen, Ihre Hausaufgaben und Klausuren in theoretischer Informatik zu bestehen. Ebensowenig ist es möglich, bei Stoffauswahl und -darstellung komplett neue, nie dagewesene Wege zu beschreiten. Ein solches Buch wäre vielleicht für Spezialisten interessant, die den Standard-Stoff ohnehin beherrschen, wäre aber für den durchschnittlichen Studenten so gut wie nutzlos. Machen Sie sich also - trotz des unkonventionellen Titels - auf ein relativ konventionelles Lehrbuch gefasst.

Ich habe meine Vorlesungen immer so gehalten, wie ich als Student mir eine Vorlesung gewünscht hätte, und ich habe versucht, beim Schreiben dieses Buches der gleichen Devise zu folgen: Ich wollte ein Buch schreiben, wie ich es mir für mich selbst beim Erlernen der Inhalte in diesem Buch gewünscht hätte. Dies mag eine Selbstverständlichkeit sein, aber ich möchte es hier trotzdem erwähnen, um zu zeigen, dass Lehren und Lernen sehr subjektive, vom Individuum abhängige Dinge sind und es den einen Königsweg nicht gibt.

## Wie dieses Buch aufgebaut ist

Das Buch beginnt in Teil I mit der **Automatentheorie.** Ausgehend von sehr einfachen Automaten und ihren mathematischen Modellen fügen wir den Automaten nach und nach weitere Fähigkeiten hinzu, bis wir beim bis heute gültigen mathematischen Modell für unsere

Computer angekommen sind, der **Turing-Maschine**. Die Beweise dafür, dass die Turing-Maschine einerseits wirklich eine universelle Maschine darstellt und andererseits trotzdem gewisse Einschränkungen bei dem besitzt, was sie berechnen kann, bilden einen ersten inhaltlichen Höhepunkt.

Nachdem es in Teil I unter anderem darum ging, formale Sprachen mithilfe von Automaten zu »erkennen« und sie durch die Art der erkennenden Automaten zu charakterisieren, geht es in Teil II um das »Erzeugen« formaler Sprachen mit Hilfe von Regelsätzen, so genannten **Grammatiken**. Der Nachweis, dass bestimmte Grammatiktypen bei der Erzeugung bzw. Akzeptanz von Sprachen äquivalent zu den in Teil I betrachteten Automatentypen sind, bildet das Leitmotiv in diesem Teil.

Teil I und Teil II handelten davon, was mit Computern prinzipiell möglich und was prinzipiell unmöglich ist - in Teil III fragen wir schließlich danach, wie schnell ein Problem auf dem Computer lösbar ist. In der **Komplexitätstheorie** wird die Schwierigkeit von Problemen grob in so genannte Komplexitätsklassen eingeteilt, und wir werden sehen, dass es bestimmte, besonders schwierige Probleme gibt, die mit dem Computer nicht effizient gelöst werden können. Der Nachweis, dass es einen harten Kern von besonders schwierigen Problemen gibt, auf die alle anderen schwierigen Probleme zurückgeführt werden können, bildet den Höhepunkt und Abschluss von Teil III und damit auch den Abschluss des klassischen Stoffs der Theoretischen Informatik, den man im Rahmen einer vierstündigen Vorlesung abdecken kann.

Ich gehe davon aus, dass die meisten meiner Leser bereits eine oder mehrere Anfängervorlesungen in Mathematik gehört haben, da die Theoretische Informatik zumeist im zweiten oder dritten Semester auf dem Lehrplan steht. Trotzdem finden es manche vielleicht nützlich, wenn sie für die Theoretische Informatik wichtige mathematische Grundbegriffe direkt hier im Buch nachschlagen können. In Teil IV habe ich diese Grundbegriffe kurz und knapp und zumeist ohne Beweis zusammen gestellt. Eine Ausnahme bildet die Graphentheorie, die ich etwas ausführlicher als unbedingt nötig dargestellt habe, weil sie zum einen nicht unbedingt zum Standard-Kanon der ersten Semester gehört, zum anderen relativ leicht zugänglich ist und nicht nur ein Hilfskonzept für die Theoretische Informatik darstellt, sondern auch eine Fülle von Anwendungsbeispielen und Problemen mit praktischer Relevanz bietet.

Kein Buch der »für Dummies« - Reihe ohne abschließenden Top-Ten-Teil. Anstatt hier noch einmal die wichtigsten Aussagen zu wiederholen (in gewisser Weise passiert dies auf den Schummelseiten zu Beginn), habe ich mich dazu entschlossen, hier noch kurz auf die Menschen hinter den wichtigsten Resultaten der Theoretischen Informatik einzugehen. Wie oben beschrieben, war das Jahr 1936 wie der Startschuss der Theoretischen Informatik, und bekanntermaßen waren die dreißiger Jahre des letzten Jahrhunderts auch sonst eine sehr bewegte Zeit. Dies spiegelt sich auch in den Lebensläufen der beteiligten Wissenschaftler wider, die keineswegs immer geradlinig verliefen. Gerade Alan Turing, von seiner Persönlichkeit her eigentlich der Prototyp des »Nerds«, war alles andere als ein akademischer Stubenhocker und schaffte es mit seiner intellektuellen Brillianz, den Lauf der Weltgeschichte zu beeinflussen.

# Symbole in diesem Buch

In diesem Buch werden Sie immer wieder bestimmte Symbole zu Ihrer Orientierung finden:

Mit diesem Symbol werden **Definitionen** markiert. In Definitionen werden neue Begriffe mit Hilfe bereits eingeführter Begriffe erklärt. Der neue Begriff ist dabei in **Fettdruck** hervorgehoben.

### Ein Beispielsatz

Mit diesem Symbol werden wichtige neue Aussagen bzw. **Sätze**, die wir aus dem, was wir schon wissen, hergeleitet haben (oder noch herleiten müssen), hervorgehoben. Zusätzlich sind diese Aussagen durch einen grauen Kasten hervorgehoben.

Dieses Symbol ist für das Verständnis der Inhalte des Buches besonders wichtig, denn es markiert Beispiele, in denen Definitionen oder neue Aussagen illustriert werden. Die Rechnungen in Beispielen sind im Normalfall sehr detailliert gehalten, damit Sie sie gut nachvollziehen können.

# Wie Sie dieses Buch lesen sollten

Idealerweise lesen Sie in diesem Buch die Teile I, II und III hintereinander und schlagen gegenenfalls einen mathematischen Begriff in Teil IV nach. Falls Ihnen die Zeit fehlt und Sie schon gewisse Vorkenntnisse besitzen, können Sie auch versuchen, die Teile I bis III unabhängig voneinander zu lesen. Dabei sollten Sie immer einen Bleistift und Papier zur Hand haben. Vollziehen Sie die Beispiele selbst nach und denken Sie sich eigene Beispiele aus. Am Ende jedes Kapitels finden Sie einige Aufgaben. Diese sind normalerweise nicht besonders schwer und dienen vor allem dazu, dass Sie für sich selbst testen können, ob Sie die Inhalte des Kapitels verstanden haben. Als kleine Hilfe, so zur Sicherheit für Sie, stelle ich Ihnen einige der Lösungen unter `http://wiley-vch.de/ISBN978-3-527-71431-5` zur Verfügung, aber nicht alle. Einige Lösungen müssen Sie selbst ausknobeln, so bleiben sie auch besser im Gedächtnis haften.

Und nun wünsche ich Ihnen viel Spaß und Erfolg bei Ihrer Reise durch die theoretische Informatik!