Einführung in die Theoretische Informatik

Formale Sprachen und Automatentheorie

von
Prof. Dr. Ulrich Hedtstück

5., überarbeitete Auflage

Oldenbourg Verlag München

Prof. Dr. Ulrich Hedtstück ist Dozent an der Fachhochschule Konstanz mit den Lehrgebieten Theoretische Informatik, Algorithmen und Datenstrukturen sowie Simulation. Seine Forschungsschwerpunkte sind Künstliche Intelligenz, Simulation und Virtual Reality. Er ist Mitglied des CIM-Arbeitskreises der FH Konstanz mit dem Teilprojekt Simulation in CIM-Konzepten.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
www.oldenbourg-verlag.de

Lektorat: Johannes Breimeier
Herstellung: Constanze Müller
Titelbild: Ulrich Hedtstück, Grafik: Irina Apetrei
Einbandgestaltung: hauser lacour
Gesamtherstellung: Grafik & Druck GmbH, München

Dieses Papier ist alterungsbeständig nach DIN/ISO 9706.

ISBN 978-3-486-71404-3
eISBN 978-3-486-71896-6

Für

Geli, Anemone und Johannes

Vorwort

Die junge Wissenschaft der Informatik ist geprägt durch das praktische Arbeiten am Computer. Der rasante Fortschritt auf dem Hardwaresektor sowie die in gleichem Maße wachsende Erwartungshaltung gegenüber den Anwendungsmöglichkeiten von Computern verleiten viele Informatiker dazu, neue Software durch schnelles Programmieren und Ausprobieren zu erstellen. Unsystematisches Vorgehen bei der Softwareentwicklung hat zur Folge, dass man häufig in Sackgassen gerät oder Umwege einschlagen muss, um ans Ziel zu gelangen.

Dies mag ein Grund dafür sein, dass sich die Software, obwohl hervorragende Fortschritte erzielt werden, hauptsächlich in die Breite entwickelt. Es werden immer neue Spezialprobleme gelöst, die einzeln kaum größere Schwierigkeiten aufweisen als frühere Probleme und die selten grundlegend neue Techniken erfordern.

In Richtung vorwärts bewegt sich die kommerzielle Softwareentwicklung nur sehr langsam, was z. B. daran erkannt werden kann, wie lange es dauert, bis Konzepte, die von Informatik-Forschern entwickelt worden sind, in die Praxis umgesetzt werden. Das deutlichste Beispiel ist das objektorientierte Programmieren, das schon in den siebziger Jahren bekannt war und erst jetzt den Sprung in die kommerzielle Programmierung geschafft hat.

Ein Hauptgrund für die langsame geradeausgerichtete Softwareentwicklung scheint mir die mangelnde Kenntnis der theoretischen Zusammenhänge zu sein, die entweder auf diesem neuen Gebiet noch nicht bekannt sind, oder die sich die Programmierer aus Eile nicht aneignen können bzw. wollen.

Es gibt ein hervorragendes Beispiel, wie ad hoc-programmierte Anwendungen einen fundamentalen Schub nach vorne erfuhren, nachdem eine theoretische Durchleuchtung der Problematik erfolgt war: der Compilerbau. Als die Com-

puterwissenschaftler zusammen mit den Linguisten die Theorie der formalen Sprachen entwickelt hatten, konnten plötzlich völlig neuartige Techniken bei der Entwicklung von Compilern realisiert werden, die sogar dazu führten, dass heute wesentliche Teile eines Compilers automatisch erstellt werden.

Hier wird die Aufgabe der Theoretischen Informatik, die sich inzwischen zu einer fruchtbaren Forschungsdisziplin entwickelt hat, besonders anschaulich. Mit Ergebnissen aus der Theoretischen Informatik sollen neue Wege erschlossen werden und qualitative Fortschritte in der praktischen Informatik erzielt werden, die einen grundlegenden Schritt nach vorne bedeuten.

In diesem Buch wird die Theorie der formalen Sprachen und Automaten als ein Teil der Theoretischen Informatik, der besonders ausgereift ist und heute zum Standardwissen eines jeden Informatikers gehören sollte, in Form einer Einführung dargestellt. Der behandelte Stoff ist die theoretische Grundlage nicht nur des Compilerbaus, sondern auch der Verarbeitung natürlicher Sprache und aller Softwaretechniken, die das automatische Verarbeiten von formalen Objekten zum Inhalt haben.

Durch meine Tätigkeit an einer Fachhochschule war ich vor die Situation gestellt, einen schon für Universitätsstudenten oft zu abstrakten, schwierigen und häufig ungeliebten Stoff für eine Zuhörerschaft aufzubereiten, die noch mehr durch Praxisorientiertheit gekennzeichnet ist. Ich habe mir deshalb zum Ziel gesetzt, möglichst solche Inhalte zu vermitteln, die konstruktiv mit anschaulichen Algorithmen erklärt werden können. Es war mir auch wichtig, die Stofffülle überschaubar zu halten, damit sie in einer vierstündigen Vorlesung in einem Semester bewältigt werden kann, deshalb habe ich auf Optimierungsaspekte hier weitgehend verzichtet.

An der formalen Strenge der Darstellung möchte ich keine Abstriche machen, denn das sichere Umgehen mit Formalismen im Stile der Theoretischen Informatik ist durchaus eines der Lernziele, die ich verfolge.

Ich hoffe, dass ich durch dieses Buch beitragen kann, die Grundlagen der Theorie der formalen Sprachen und Automaten einem möglichst großen Leserkreis zu erschließen, und wünsche Ihnen viel Spass bei der Lektüre.

Konstanz, im Juli 2000 Ulrich Hedtstück

Vorwort zur 2., überarbeiteten Auflage

Durch die Einführung des weltweiten Standards XML (eXtensible Maurkup Language) für Anwendungen im Word Wide Web hat die Entwicklung der Informatik in den letzten Jahren einen weiteren entscheidenden Impuls erhalten, der im Wesentlichen auf Kenntnissen aus der Theorie der formalen Sprachen und der Automathentheorie basiert. Mit Hilfe der Spezifikation von Datenformaten auf der Grundlage kontextfreier Grammatiken sowie den Einsatz geeigneter Parsingtechnologien können unterschiedlichste Dateninhalte zu unterschiedlichsten Zwecken in einheitlicher Weise übers Internet ausgetauscht und weiterverarbeitet werden. Der Bedeutung dieser aktuellen und kommerziell höchst relevanten Technologie wollte ich Rechnung tragen und habe einen Abschnitt über XML zum Kapitel 4 hinzugefügt, der insbesondere auch einen ersten Einblick in XML-Schemata als zukünftigen Spezifikationsstandard bietet.

Die Herausgabe einer zweiten Auflage des vorliegenden Buches gab mir zudem die Gelegenheit, einige Stellen zu optimieren sowie Druckfehler und Ungenauigkeiten zu beseitigen, die sich zu meinem Leidwesen in der ersten Auflage eingeschlichen hatten. Für entsprechende Hinweise bin ich Michael Czymmeck, Klaus Hager, Eduard Klein und Hans Werner Lang zu besonderem Dank verpflichtet. Auch für die hervorragende Zusammenarbeit mit dem Oldenbourg Verlag, insbesondere mit Frau Irmela Wedler, möchte ich mich an dieser Stelle ganz herzlich bedanken.

Konstanz, im Mai 2002 Ulrich Hedtstück

Vorwort zur 5., überarbeiteten Auflage

In der fünften Auflage habe ich neben der Optimierung vieler Formulierungen, einigen gestalterischen Verbesserungen und der Aktualisierung des Literaturverzeichnisses die Abschnitte 3.8 „Reguläre Ausdrücke“, 4.2.1 „Mehrdeutigkeiten in Programmiersprachen“, 4.10.2 „XML-Schemata“ sowie 5.4 „Komplexitätsuntersuchungen mit Hilfe von Turingmaschinen“ überarbeitet und erweitert. Auch den Übungsteil habe ich um einige neue Aufgaben und Lösungsvorschläge ergänzt.

Für die angenehme und konstruktive Zusammenarbeit danke ich dem Oldenbourg Verlag sehr herzlich.

Konstanz, im Juli 2012 Ulrich Hedtstück

Inhaltsverzeichnis

Einleitung

Die Theorie der formalen Sprachen und die Automatentheorie beschäftigen sich mit den Möglichkeiten zur formalen Beschreibung und automatischen Verarbeitung von gedanklichen Gebilden wie Wörter, Sätze oder Verfahrensanweisungen. Dabei ist das Hauptziel, eine Kommunikation zwischen Menschen und Computern zu ermöglichen.

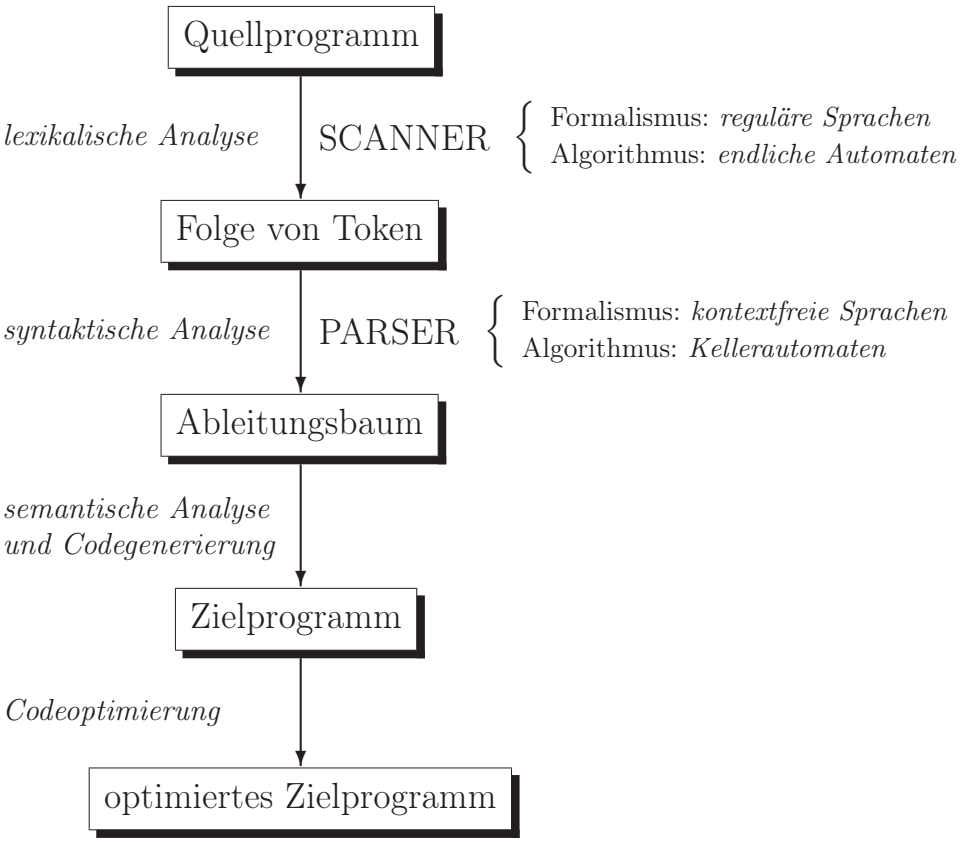
Um einen Computer dazu zu bringen, eine vorgesehene Aufgabe zu erledigen, formuliert der Benutzer geeignete Verfahrensanweisungen oder Algorithmen in einer formalen und maschinenlesbaren Sprache. Der Computer muss erkennen, ob diese Formulierungen zulässig sind, und wenn ja, stellt er sich einen internen Arbeitsplan zusammen und führt die gewünschten Aktionen aus.

Der Formulierungsaspekt dieses Prozesses ist Gegenstand der Theorie der formalen Sprachen, die verschiedene, z. T. aus der Linguistik stammende Prinzipien und Gestaltungsrahmen bereitstellt. Der Erkennungsaspekt wird in der Automatentheorie behandelt, indem geeignete Techniken und Algorithmen zur Verfügung gestellt werden, die überprüfen, ob die Eingabe für eine Weiterverarbeitung akzeptiert werden kann oder nicht. Es ist offensichtlich, dass eine starke Abhängigkeit der Erkennungsalgorithmen von den Eingabeformalismen, und umgekehrt, besteht.

Die Erstellung eines internen Arbeitsplans und der Aktionsteil für ein zulässiges Programm sind Mechanismen, die nicht mehr ausschließlich von der zugrundegelegten Sprache abhängen, sondern müssen stark Hardware-spezifisch durchgeführt werden. Deshalb interessiert uns hier nur der Teil eines Kompiliervorgangs, der die Zulässigkeit einer Eingabe als Programm überprüft.

Das folgende Schema zeigt die wesentlichen Schritte bei der Kompilierung und macht deutlich, welche Sprachtypen und Erkennungsprinzipien dabei

eine Rolle spielen. Dabei verwenden wir hier schon die entsprechenden Fachbegriffe, die in späteren Kapiteln noch ausführlich vorgestellt werden.



Die Überprüfung, ob eine Eingabe ein zu akzeptierendes Programm darstellt, wird im Wesentlichen durch den Scanner und den Parser erledigt, die in verzahntem Ablauf Stück für Stück den Eingabestring auf unterschiedlichen Sprachebenen abarbeiten. Dabei hat der Scanner die Aufgabe, die einzelnen Eingabezeichen (z. B. ASCII-Zeichen) zu so genannten Eingabesymbolen oder Tokens zusammenzufassen (Schlüsselwörter, Bezeichner, Zahlen usw.), der Parser überprüft, ob mit diesen Eingabesymbolen Programmkonstrukte wie Ausdruck, Sequenz, Schleife oder Alternative korrekt gebildet worden sind. Als Ergebnis wird ein so genannter Ableitungsbaum erstellt, aus dem nach einer semantischen Analyse schließlich der Zielcode generiert wird.

Dieselben Techniken werden nicht nur bei der Übersetzung von Programmen eingesetzt, sondern auch bei der maschinellen Verarbeitung natürlicher Sprache, etwa bei der natürlichsprachlichen Kommunikation mit Computern oder bei der maschinellen Übersetzung.

Ganz generell können diese Konzepte verwendet werden, wann immer es um die maschinelle Verarbeitung von Formalismen geht. Weitere Anwendungsbeispiele sind das Übersetzen von einem Datenformat in ein anderes (z. B. CAD-Datenformate), die Verarbeitung von Datenbankabfragen, Textverarbeitung mit Editoren oder das Erkennen von Schrift und Bildern.

Die Theorie der formalen Sprachen bzw. die Automatentheorie liefern nicht nur eine theoretische Grundlage für das Erkennen und Verarbeiten von Formalismen, sondern auch einen theoretischen Rahmen für die Untersuchung der Möglichkeiten zur Problemlösung durch den Computer. Das Prinzip der Turingmaschinen ist so grundlegend, dass damit alles algorithmische Denken des Menschen begründet werden kann. Anhand dieses theoretischen Algorithmus-Modells werden einerseits Grenzen und Möglichkeiten der automatischen Problemlösung aufgezeigt, andererseits können praktisch relevante Folgerungen für die Entwicklung effizienter, d. h. schneller Computerprogramme abgeleitet werden.

Kapitel 1

Formale Sprachen

Eine natürliche Sprache wird im Wesentlichen durch sprachliche Sätze charakterisiert, die von Menschen geäußert werden, wobei man grammatikalische Korrektheit und sinnvollen Gebrauch voraussetzt. Natürliche Sprachen sind außerordentlich komplex und können nicht vollständig durch ein allgemein gültiges Regelwerk beschrieben werden. Zwar hat man heute sehr gute Regelwerke für die grammatikalische Korrektheit (Syntax) von einzelnen Sätzen, aber die Bedeutung (Semantik) eines Satzes oder gar Textes lässt sich nur ansatzweise mit formalen Methoden erfassen.

Ein großes Problem bei natürlichen Sprachen ist z. B., dass viele Äußerungen mehrdeutig sind, und zwar nicht nur in ihrer Bedeutung, sondern auch in ihrer syntaktischen Struktur. So kann etwa in der Anweisung „Öffne die Datei mit dem Editor“ der Satzteil „mit dem Editor“ sowohl als Attribut des Objekts „Datei“ als auch als Adverbiale Bestimmung des Verbs „öffnen“ interpretiert werden.

Deshalb können natürliche Sprachen nicht für die Beschreibung von Aufgaben, die ein Computer lösen soll, verwendet werden, außer in sehr eingeschränkter Form, etwa als Steuerungsbefehle von technischen Geräten oder als Datenbankabfragesprache. Besser eignen sich formale Sprachen wie Programmiersprachen, in denen auf der Basis einer relativ kleinen Menge von syntaktischen Regeln Programme formuliert werden, die eine Semantik aufweisen, die von Rechnern weitgehend verstanden wird.

Die Theorie der formalen Sprachen bietet eine theoretische Grundlage für alle Sprachen an, die mit automatischen Verfahren verarbeitet werden, wobei

hauptsächlich syntaktische Aspekte berücksichtigt werden. Wichtige Begriffe wie Alphabet, Buchstabe oder Wort, die vom Umgang mit natürlichen Sprachen her geläufig sind, werden dabei verallgemeinert.

1.1 Buchstaben, Wörter, Sprachen

Zum Verständnis der folgenden Definitionen hilft die Veranschaulichung anhand der analogen Verwendung in der natürlichen Sprache.

Definition: (Alphabet, Wort, Sprache)

Ein *Alphabet* ist eine endliche, nichtleere Menge von *Zeichen* (auch *Symbole* oder *Buchstaben* genannt). Wir verwenden meist V als Abkürzung für Alphabete (von engl.: *vocabulary*).

Sei V ein Alphabet und $k \in \mathbb{N}$. Dabei ist $\mathbb{N} = \{0, 1, 2, ...\}$ die Menge der natürlichen Zahlen einschließlich der Null.

Eine endliche Folge $(x_1, ..., x_k)$ mit $x_i \in V$ $(i = 1, ..., k)$ heißt *Wort über* V der *Länge* k. Es gibt genau ein Wort der Länge 0, das *Leerwort*, das wir mit ε bezeichnen.

Die Länge eines Wortes x wird durch $|x|$ dargestellt.

Wir verwenden, wann immer es das Alphabet erlaubt, die Kurzschreibweise $x_1x_2...x_k$ für Wörter, indem wir die Zeichen einfach aneinanderfügen. (Beim Alphabet {1,11} wäre dies nicht möglich, denn man könnte z. B. bei dem Wort 111 nicht erkennen, aus welchen Zeichen es besteht.)

V^* bezeichnet die *Menge aller Wörter* über V.

Die Menge der *nichtleeren* Wörter über V ist

$$V^+ := V^* \setminus \{\varepsilon\}.$$

Jede (beliebige) Teilmenge von V^* wird als *Sprache* oder *formale Sprache* bezeichnet.

In diesem allgemeinen Sinn fassen wir jede Menge als Alphabet auf, sofern sie nichtleer und endlich ist, und jede Wortmenge wird als Sprache bezeichnet.