Compilerbau
Grundlagen und Anwendungen

DAS VORWORT

Geleitwort

Programmiersprachen spricht man nicht – es sind formale Systeme. Programme, die die Texte, die in Programmiersprachen formuliert sind, in Folgen von Computerinstruktionen übersetzen, nennt man Compiler. Es handelt sich dabei um komplexe Programme. Am Anfang der Compilertechnologie standen die Sprachen Fortran (1957) und Algol (1960). Die Fertigung ihrer Compiler beschäftigte große Teams von Programmierern über Jahre. Durch die Systematisierung des Compilerbaus, wie sie in diesem Buch vorgestellt wird, konnte diese Arbeit für Einzelpersonen jedoch auf wenige Monate reduziert werden. Dies war ein gewaltiger Fortschritt.

Grundlegend neue Programmiersprachen gibt es heute eher selten und völlig neue Computerarchitekturen ebenfalls. Compilerbau scheint daher eine spezielle Sparte zu sein, die wenigen Spezialisten in großen Firmen vorbehalten bleibt. Wozu also dieses Buch?

Der Grund ist einfach. Jedes Programm, jede Anwendung, ist nach Regeln aufgebaut, die spezifizieren, wie Anweisungen und Datendeklarationen auszusehen haben. Wenn diese Regeln formalisiert werden, erhöht das die Klarheit und Verständlichkeit der Anwendung. Der Schlüssel dazu beruht auf Formalisierung, also auf der Spezifikation einer Syntax für die Eingabe. Dadurch wird eine Syntaxanalyse ermöglicht, die die Grundlage des Compilerbaus darstellt. Die Syntaxanalyse und weitere in diesem Buch beschriebene Techniken sind aber nicht nur zur Verarbeitung von Programmiersprachen nützlich, sondern lassen sich auch auf viele andere Probleme anwenden, bei denen es um die systematische Verarbeitung strukturierter Eingaben geht. Diese Techniken tragen maßgeblich zur Korrektheit und zum Verständnis solcher Anwendungen bei.

Möge dieses Buch bei dieser vielversprechenden Entwicklung behilflich sein!

Prof. em. Dr. Niklaus Wirth
Zürich, im Dezember 2023[1]

1. Prof. Dr. Niklaus Wirth verstarb völlig unerwartet eine Woche nach Abfassung dieses Geleitworts im 90. Lebensjahr.

Vorwort

Compiler bauen? Das machen doch nur große Firmen wie Microsoft, Google oder Oracle. Das stimmt, aber fast alle Informatikerinnen und Informatiker scheinen irgendwann einmal den Wunsch zu verspüren, eine eigene Programmiersprache zu entwerfen, und sei es nur eine domänenspezifische Sprache oder eine Kommandosprache für spezifische Zwecke. Natürlich möchte man dann auch einen Compiler dafür schreiben. Dieser Wunsch scheitert oft daran, dass das nötige Compilerbau-Wissen fehlt oder die in einschlägigen Büchern beschriebenen Techniken zu kompliziert sind und sich vor allem mit fortgeschrittenen Themen wie Optimierung, Registerallokation oder den Details der Codeerzeugung beschäftigen.

Dabei sind die Grundlagen des Compilerbaus einfach – jeder kann sie erlernen und zu seinem Methodenrepertoire hinzufügen. Während Compiler für Programmiersprachen wie Java oder C/C++ tatsächlich nur von großen Firmen entwickelt werden, gibt es viele Aufgaben (auch außerhalb des eigentlichen Compilerbaus), die sich mithilfe elementarer Techniken einfach und elegant lösen lassen. Im Prinzip kann man diese Techniken immer dann anwenden, wenn eine strukturierte Eingabe vorliegt, die durch eine Grammatik beschrieben werden kann. Beispiele dafür sind einfache Kommandosprachen, aber auch die Verarbeitung von Konfigurationsdateien, Logdateien, Stücklisten oder Messdatenreihen.

Dieses Buch zeigt, wie es geht. Es behandelt die praxisrelevanten Grundlagen des Compilerbaus, von der lexikalischen Analyse über die Syntaxanalyse bis zur Semantikverarbeitung und zur Codeerzeugung. Weitere Themen sind die Beschreibung von Übersetzungsprozessen durch attributierte Grammatiken sowie der Einsatz eines Compilergenerators zur automatischen Erzeugung der Kernteile eines Compilers. Gerade diese letzten beiden Themen sind in der Praxis höchst relevant, obwohl man sie in vielen Compiler-Büchern nicht findet.

Zur Syntaxanalyse wird in diesem Buch der rekursive Abstieg verwendet, ein einfaches Top-down-Verfahren, das auch per Hand (d.h. ohne Werkzeuge) implementiert werden kann. Zur Abrundung wird allerdings am Ende des Buches auch die Bottom-up-Syntaxanalyse vorgestellt, die zwar mächtiger, aber auch aufwendiger ist als der rekursive Abstieg.

Techniken versteht man erst so richtig, wenn man sie auf ein konkretes Beispiel anwendet. Daher wird im Buch als durchgängiges Fallbeispiel ein Compiler

für *MicroJava* – eine einfache Java-ähnliche Programmiersprache – entwickelt, der ausführbaren Bytecode – ähnlich dem Java-Bytecode – erzeugt. Der vollständige Quellcode dieses Compilers kann von [Download] heruntergeladen und studiert werden. Als Implementierungssprache für den Compiler sowie für alle Beispiele in diesem Buch wird Java verwendet.

Als Zielmaschine des Compilers wird eine vereinfachte *Java Virtual Machine* (JVM) verwendet, nämlich die *MicroJava Virtual Machine* (μJVM), die einfach genug ist, um nicht in Details zu ersticken, aber auch realistisch genug, um damit die Techniken der Codeerzeugung zu erlernen. Die μJVM besitzt als Instruktionssatz einen Bytecode, der sich an den Bytecode der JVM anlehnt. Ein Interpreter für diesen Bytecode wird ebenfalls zur Verfügung gestellt. Durch das Studium der Codeerzeugung lernt man auch viel über die Funktionsweise eines Rechners, was ein weiterer Grund ist, sich mit Compilerbau zu beschäftigen.

Am Ende jedes Kapitels gibt es Übungsaufgaben zu den behandelten Themen. Musterlösungen dazu sind unter [Download] zu finden. Man sollte versuchen, die Übungen zu bearbeiten, weil man dadurch die behandelten Techniken besser versteht. Aber selbst wenn man nicht die Zeit findet, die mehr als 70 Übungsaufgaben selbst zu lösen, sollte man sich zumindest die Musterlösungen ansehen, weil sie zusätzliche Beispiele zu den einzelnen Kapiteln darstellen.

Das Buch entstand aus einer Compilerbau-Vorlesung, die ich seit vielen Jahren an der Johannes Kepler Universität Linz sowie an der Oxford Brookes University in England halte. Es kann als begleitende Unterlage zu einer einführenden Compilerbau-Vorlesung verwendet werden, an die sich dann eine fortgeschrittene Vorlesung mit Themen wie Optimierung oder Registerallokation anschließen kann. Die Powerpoint-Folien der diesem Buch zugrunde liegenden Vorlesung werden unter [Download] zur Verfügung gestellt. Das Buch kann aber auch zum Selbststudium verwendet werden, da es alle Techniken beschreibt, die für den Bau compilerähnlicher Werkzeuge in der Praxis benötigt werden.

Ich möchte an dieser Stelle meinem ehemaligen Lehrer und Kollegen Prof. Niklaus Wirth (ETH Zürich) für das Geleitwort zu diesem Buch danken. Er ist ein Meister des Compilerbaus, von dem ich viele Techniken übernommen habe. Ebenfalls bedanken möchte ich mich beim dpunkt.verlag für die wie immer hervorragende Begleitung dieses Buchprojekts und den ausgezeichneten Lektoratsservice.

Hanspeter Mössenböck
Linz, im Dezember 2023

[Download] *https://ssw.jku.at/CompilerBuch/*

- Vorlesungsfolien
- Quelltext des MicroJava-Compilers
- Musterlösungen zu den Übungsaufgaben
- Weitere Materialien