Übersetzer

Vorlesung WS 1997/98
Uwe Kastens

Ziele:

- grundlegende **Verfahren der Sprachimplementierung** erlernen,

- den Einsatz von **generierenden Werkzeugen** und von **Standardverfahren** kennen und schätzen lernen,

- Übersetzerkonstruktion als Beispiel für das Zusammenwirken von grundlegenden **Algorithmen**, **Theorien** und von **Software-Engineering-Methoden** bei **präzise definierten Aufgaben** erkennen,

- Anwendbarkeit der Verfahren auch für **Sprachen jenseits der Programmiersprachen** erkennen.

Inhalt

1. Übersetzeraufgaben und -struktur

Aufgabenzerlegung mit Bezug zu Sprachdefinitionen,
Symbole, Syntax, statische und dynamische Semantik

2. Übersetzerstruktur, Schnittstellen

zentrale Datenstrukturen
Interpretierer, Debugger

3. Lexikalische Analyse

Spezifikation und Implementierung endlicher Automaten,
Generatoren dafür
Module zur Speicherung und Codierung von Symbolen

4. Syntaktische Analyse

Ziel- und quellorientierte Verfahren,
Rekursiver Abstieg, LR-Verfahren, Generatoren dafür,
Grammatikklassen, Fehlerbehandlung

5. Semantische Analyse

Bezeichneridentifikation, Typprüfung,
attributierte Grammatiken, Generatoren dafür

6. Optimierende Programmtransformation

Datenflußanalyse spezifiziert durch Gleichungssysteme,
Lösungsalgorithmen, Transformationen

7. Code-Erzeugung

Typische Eigenschaften von Zielprozessoren,
Verfahren zur Registerzuteilung
Optimierende Code-Auswahl, Generatoren dafür

8. Code-Parallelisierung

parallele Funktionseinheiten, Pipelines, Abhängigkeitsgraphen

Literatur

U. Kastens:
Übersetzerbau, Handbuch der Informatik 3.3,
Oldenbourg, 1990
(Hörerschein für Rabatt beim Kauf)

W. M. Waite, L. R. Carter:
An Introduction to Compiler Construction,
Harper Collins, New York, 1993

W. M. Waite, G. Goos:
Compiler Construction,
Springer-Verlag, 1983

R. Wilhelm, D. Maurer:
Übersetzerbau - Theorie, Konstruktion, Generierung,
Springer-Verlag, 1992

A. Aho, R. Sethi, J. D. Ullman:
Compilers - Principles, Techniques and Tools,
Addison-Wesley, 1986

A. W. Appel:
Modern Compiler Implementation in C,
Cambridge University Press, 1997
auch in Java und in ML

Was übersetzt ein Übersetzer?

Ein **Übersetzer** transformiert korrekte Sätze einer **Quellsprache** in **gleichbedeutende** Sätze einer **Zielsprache**.

Quellsprache:	**Zielsprache:**
Programmiersprache	**Maschinensprache**
C++	Sparc-Code
Programmiersprache	**Abstrakte Maschine**
Java	Java-Byte-Code
Programmiersprache	**Programmiersprache**
Source-to-Source	
C++	C
Anwendungssprache	**Anwendungssprache**
LaTeX	HTML
DB-Sprache (SQL)	DB-Systemaufrufe

Ein **Interpretierer** führt Sätze einer Quellsprache aus.

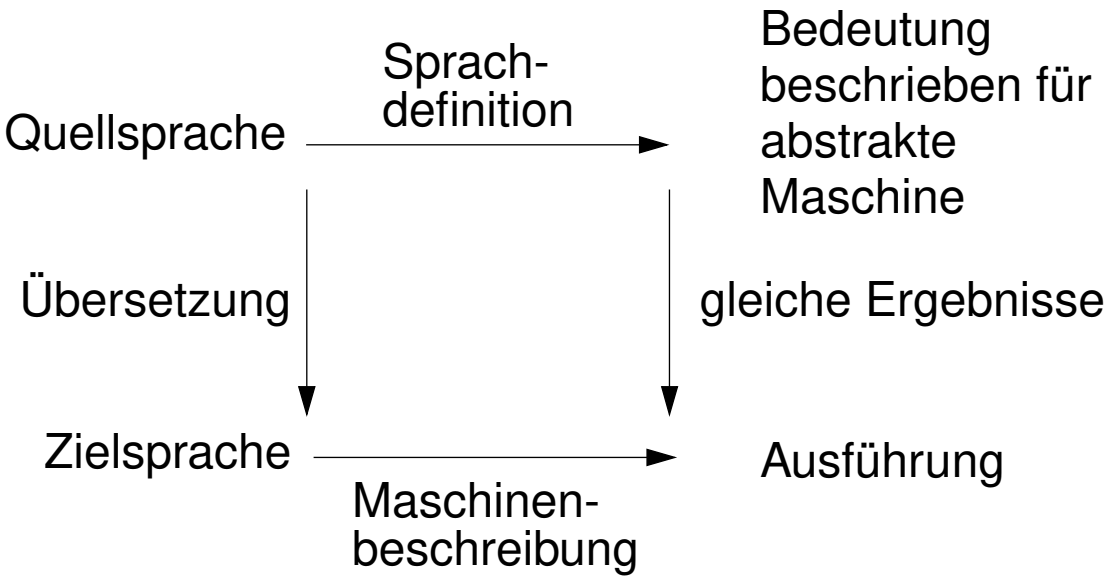
Lisp

Shell-Kommandosprache

Java-Byte-Code

Bedeutungserhaltende Übersetzung

Ein **Übersetzer** transformiert **korrekte Sätze** einer **Quellsprache** in **gleichbedeutende** Sätze einer **Zielsprache**.



Die **Bedeutung** ist nur für **korrekte** Programme definiert.
Aufgabe: Fehlerbehandlung.

Der Übersetzer ermittelt **statische** Programmeigenschaften zur **Übersetzungszeit,** z. B. **übersetzbar**.

Dynamische Eigenschaften werden zur **Laufzeit** des Programms ermittelt, z. B. **ausführbar**.

Sprachdefinition - Übersetzeraufgaben

- **Notation der Grundsymbole**

(Wortsymbole, Bezeichner, Literale)

formal definiert: reguläre Ausdrücke

Aufgabe: lexikalische Analyse

- **syntaktische Struktur**

formal definiert: kontextfreie Grammatik

Aufgabe: syntaktische Analyse

- **statische Semantik**

(Definition von Namen für Objekte, Typregeln)

meist verbal definiert

Aufgaben: semantische Analyse, Transformation

- **dynamische Semantik**

(Bedeutung, Wirkung von Sprachkonstrukten)

meist verbal definiert, in Termen einer abstrakten Sprachmaschine

Aufgaben: Transformation, Code-Generierung

- **Definition der Zielsprache (-maschine)**

Aufgaben: Transformation, Code-Generierung, Assemblierung

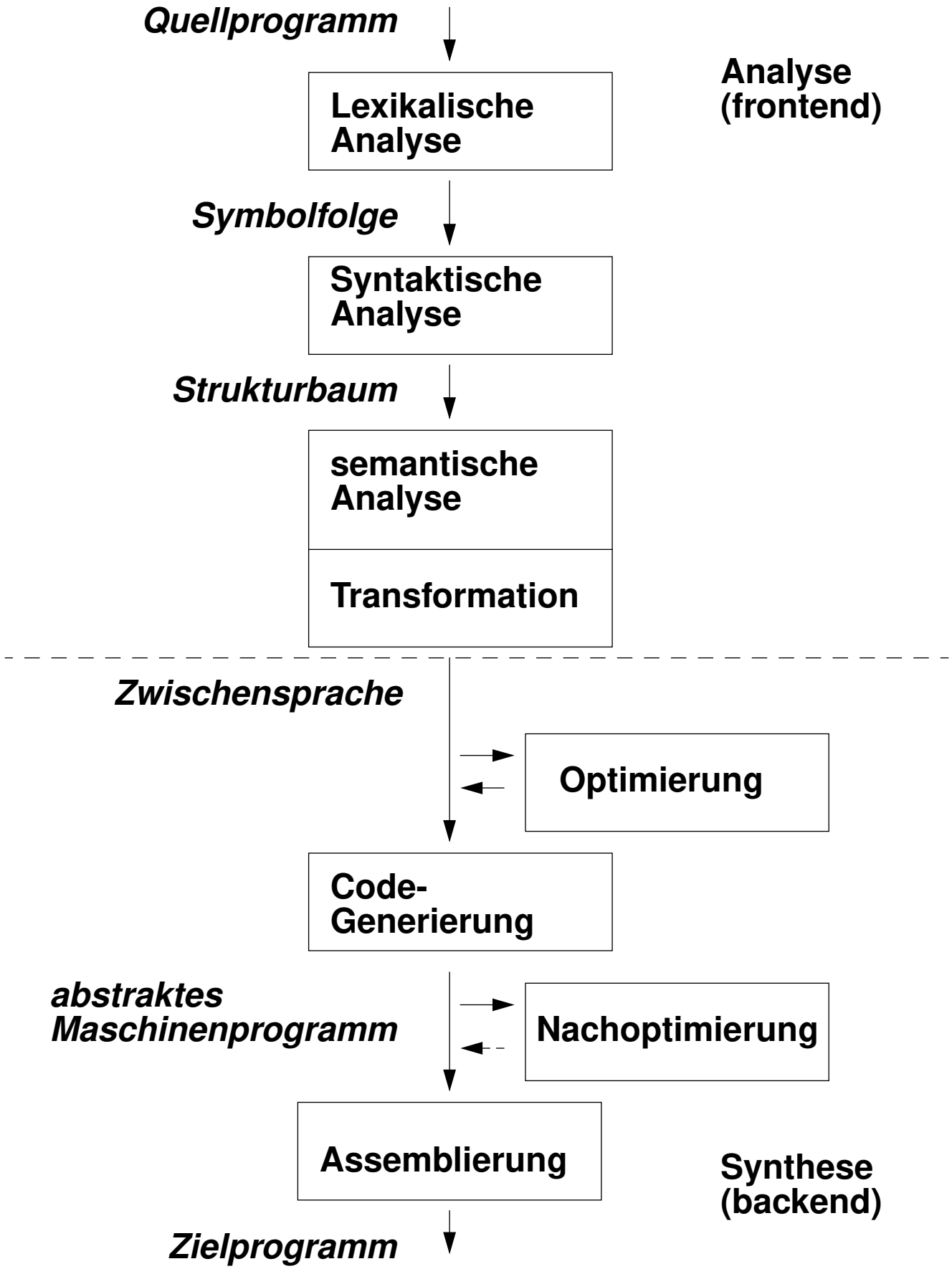
Übersetzeraufgaben

Strukturierung	Lexikalische Analyse	Symbol-erkennung Konversion
	Syntaktische Analyse	Zerteilung Baumaufbau
Translation	Semantische Analyse	Bezeichner-analyse Typ-analyse
	Transformation	Daten-abbildung Operations-abbildung
Kodierung	Code-Generierung	Ausführungs-reihenfolge Register-zuteilung Instruktions-Auswahl
	Assemblierung	Instruktions-kodierung Interne Adressierung Externe Adressierung

Zielmaschine - Übersetzeraufgaben

Eigenschaften von Zielprozessoren	**Aufgaben der Kodierung**
Instruktionssatz RISC - CISC	Instruktionsauswahl
Instruktionsformat 2-, 3-Adress	Instruktionskodierung Registerzuteilung
Register Klassen, Anzahl,	Registerzuteilung Ausführungsreihenfolge
Speicheradressierung Ausrichtung, Adressbildung	Datenabbildung interne Adressierung Instruktionsauswahl Instruktionskodierung
Instruktionsausführung Parallelverarbeitung	Ausführungsreihenfolge

Übersetzerstruktur und Schnittstellen



Übersetzerqualitäten

- **Korrektheit:**

  richtige Programme richtig übersetzen,

  falsche Programme zurückweisen

- **Effizienz:**

  Speicher- und Zeitbedarf des Übersetzers

- **Code-Effizienz:**

  Speicher- und Zeitbedarf des Ziel-Codes

  Aufgabe: Optimierung

- **Benutzerfreundlichkeit:**

  Aufgabe: Fehlerbehandlung

- **Robustheit:**

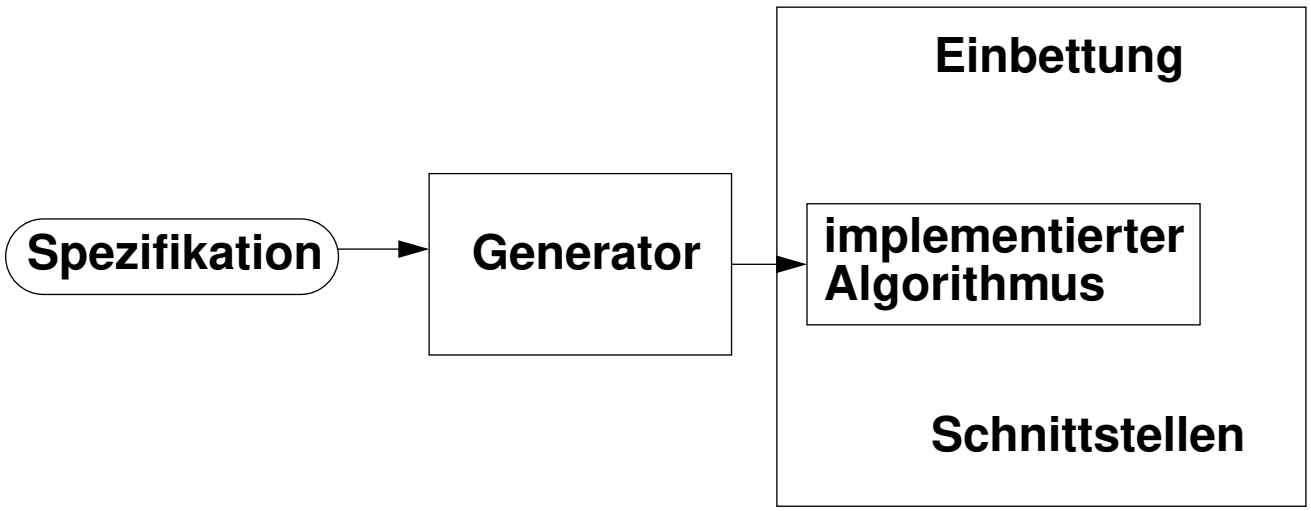
  jede Eingabe wird bewältigt

Strategien zur Übersetzerkonstruktion

- **Sprachdefinition exakt befolgen**
- **generierende Werkzeuge einsetzen**
- **Standard-Module verwenden**
- **Standard-Methoden anwenden**
- **Validieren (Test-Suite)**
- **Verifizieren**

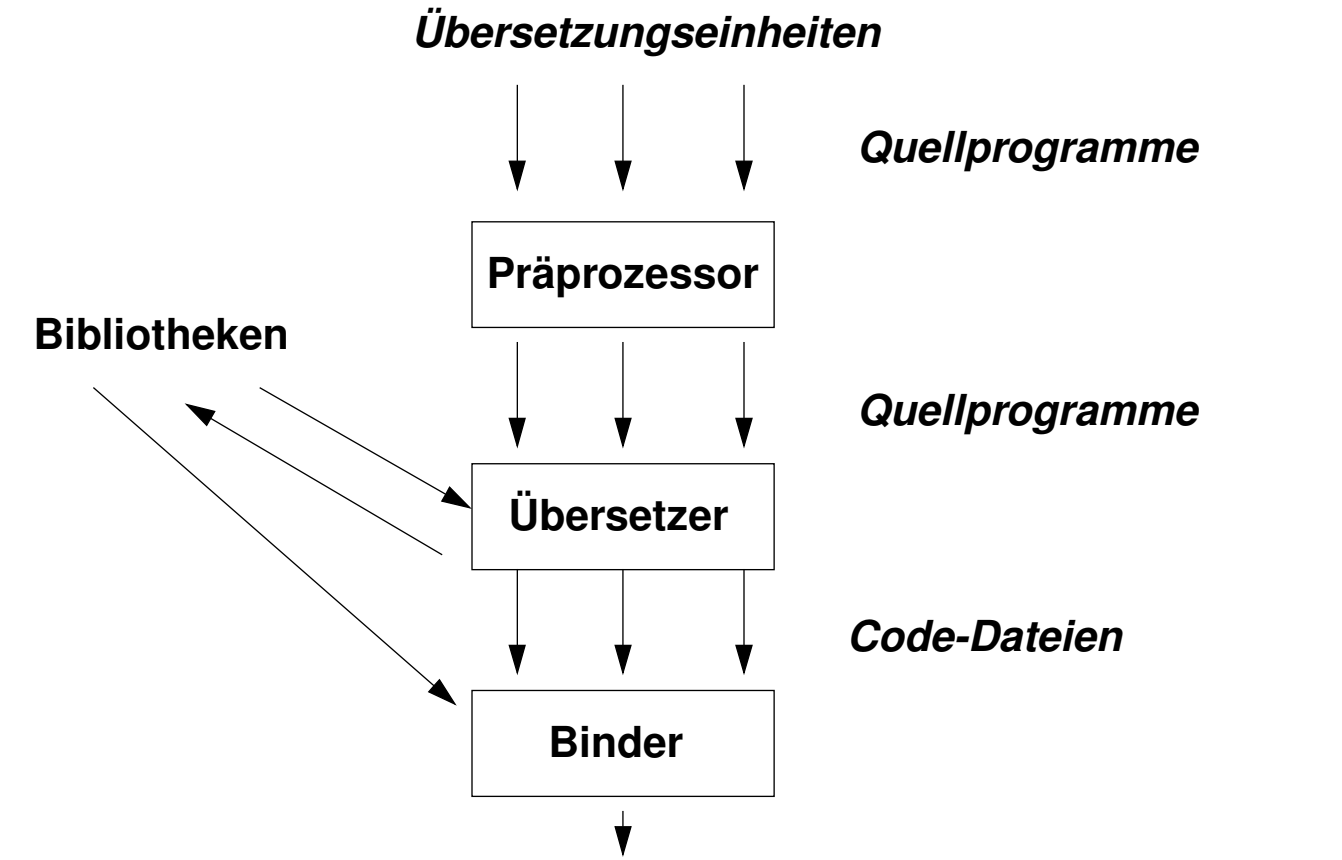
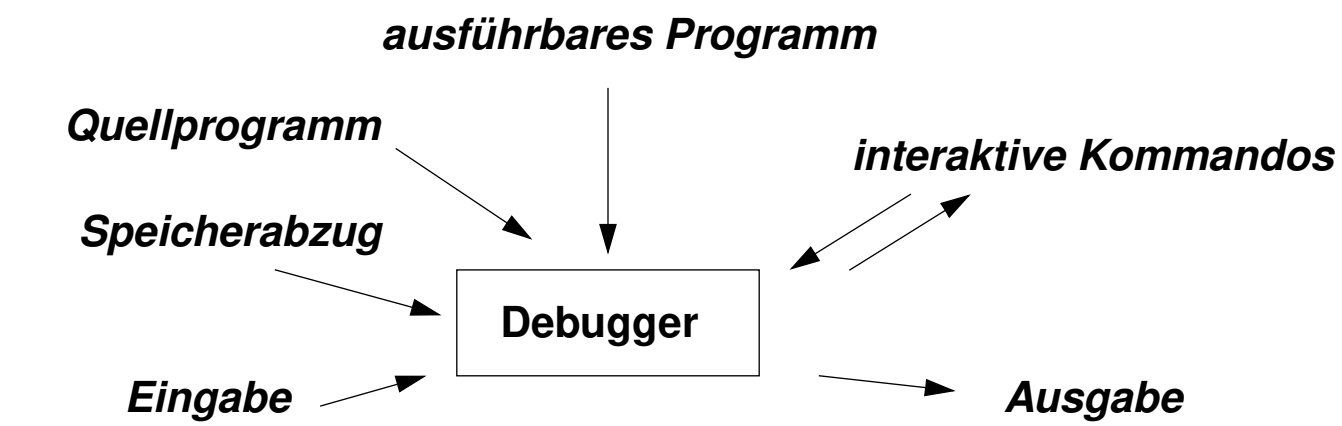
Generatoren

Schema:



typische Generatoraufgaben:

reg. Ausdrücke	Scanner-Generator	endl. Automat
kontextfreie Gram.	Parser-Generator	Kellerautomat
attributierte Gram.	Attributauswerter-Generator	Baumdurchlauf Berechnungen
Code-Muster	Code-Auswahl-Generator	Pattern-Matching

integriertes System Eli:

Umgebung des Übersetzers