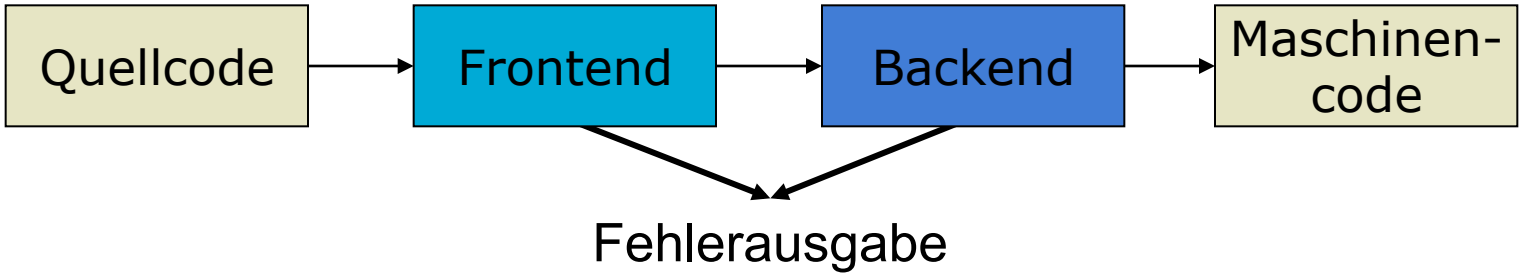
Compilerbau

Scanner
Flex

Überblick

1. Scanner
   a) Aufgaben
   b) Scannergenerator
2. Flex
3. Beispiele

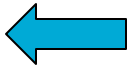
Traditioneller 2-Phasen-Compiler



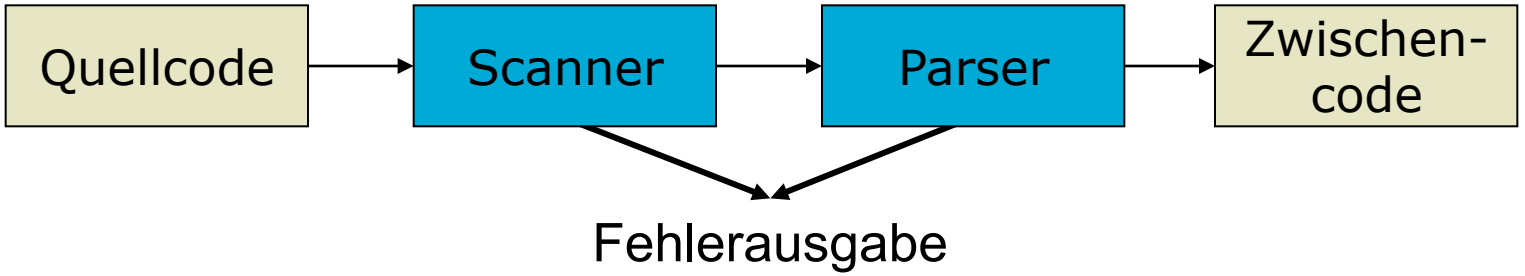
- Konsequenz
  - Frontend („Analysephase“) erzeugt vom Quellprogramm korrekten Zwischencode
  - Backend („Synthesephase“) erzeugt vom Zwischencode korrekten Maschinencode
  - Zwischenrepräsentation („Zwischencode“)

Analysephase (Frontend) (1)

- Analysephase zerlegt in 3 Phasen:

1. Lexikalische Analyse 
   - Zerlegung des Quelltextes in zusammengehörende Token
2. Syntaktische Analyse
   - Überprüfung, ob eingelesener Quellcode der Syntax (Grammatik) der Quellsprache entspricht
   - Eingabe wird in Syntaxbaum umgewandelt
3. Semantische Analyse
   - Überprüfung der statischen Semantik
   - Variablendeklaration und Zuweisungen werden geprüft

Scanner



- Aufgaben des Scanners
  - Überführung des Quellcodes in Token
    - Z. B. Zahlen, Bezeichner, Operatoren, Schlüsselwörter
  - Entfernung von Zwischenzeichen
    - Z. B. Whitespaces, Kommentare
  - Erkennung lexikalischer Fehler
    - Z. B. illegale Zeichen in Token
  - Positionsermittlung von Symbolen im Quellprogramm z. B. für Fehlerausschriften

Lexikalische Analyse

- Die von einem Scanner erkannten lexikalischen Einheiten bilden eine Reguläre Sprache
- Reguläre Sprachen können durch Reguläre Ausdrücke beschrieben werden
- Reguläre Sprachen können durch endliche Zustandsautomaten erkannt werden

Reguläre Ausdrücke
Kurznotation

Ausdruck	Bedeutung
a	Gewöhnliches Zeichen
ε	Leere Zeichenkette
s | r	Alternative
s · r	Verkettung
sr	Verkettung
s*	Wiederholung (beliebig oft, auch null)
s+	Wiederholung (einmal oder beliebig mehrfach)
s?	Optional (null oder einmal)
[a-zA-Z]	Zeichenalternative

Reguläre Ausdrücke
Beispiele

Regulärer Ausdruck	Definierte Sprache
a | b	{a, b}
(a | b)(a | b)	{aa, ab, ba, bb}
a*	{ε, a, aa, aaa, …}
(a | b)*	{ε, a, b, aa, ab, ba, bb, aaa, …}
a | a*b	{a, b, ab, aab, aaab, …}
a?b*	{ε, a, b, ab, bb, abb, …}
ab+c	{abc, abbc, abbbc, …}

Demo

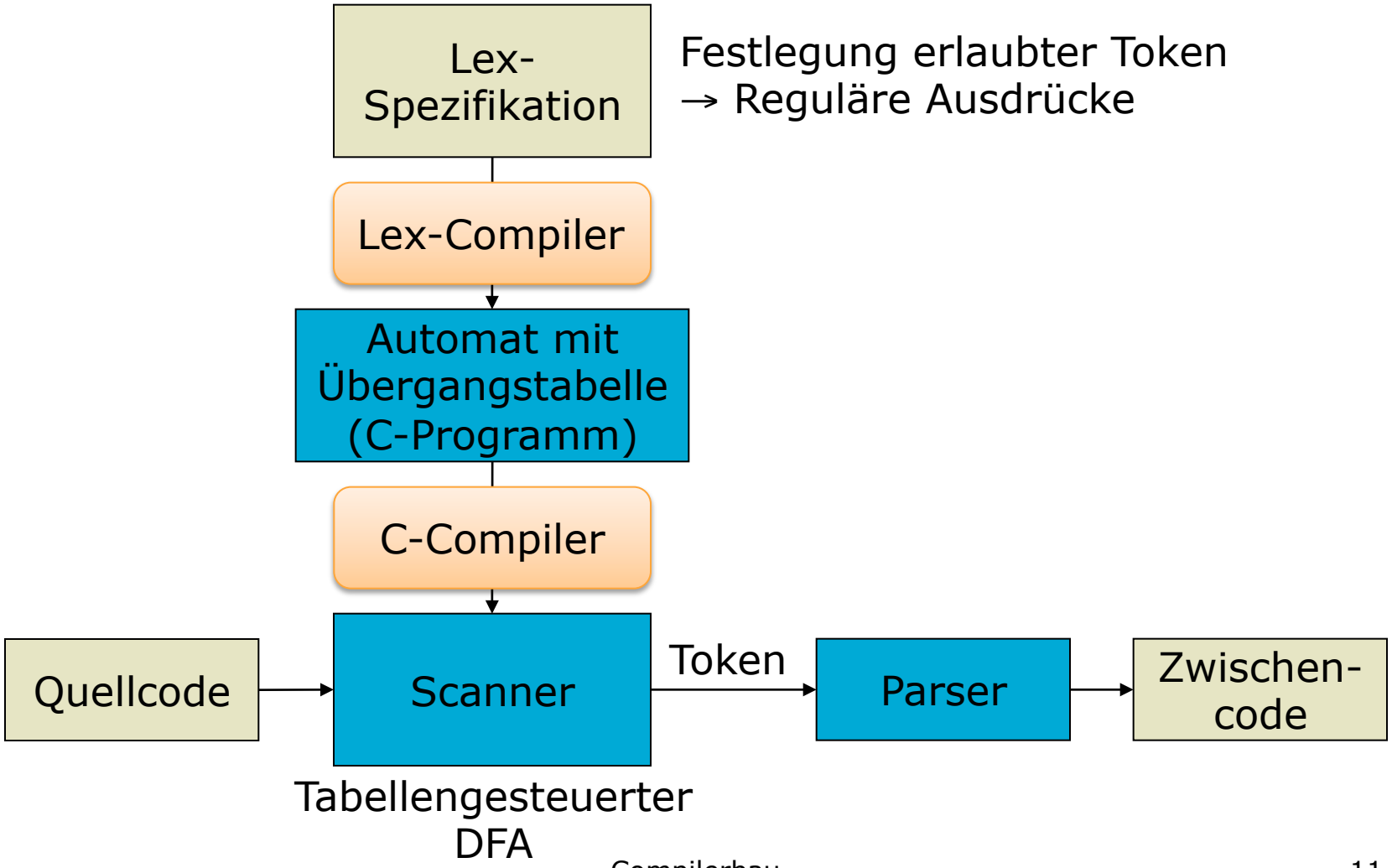
handscanner

Scannergenerator

- Konstruiert automatisch Code aus einem RA
- Schritte
  - Konstruiere aus dem RA einen DFA
  - Wende Techniken zur Zustandsminimierung an
  - Erzeuge Code für einen Scanner (tabellengetrieben oder Code direkt)
- Wichtig
  - Erzeugung einer Standardschnittstelle zum Parser

→ Lex
  - Scannergenerator für UNIX
  - Flex ist GNU-Version
    - „Fast lexical analyzer generator“

Lex
Kompilierung



Demo

flex1
flex2
flex3