Grundlagen der Informatik

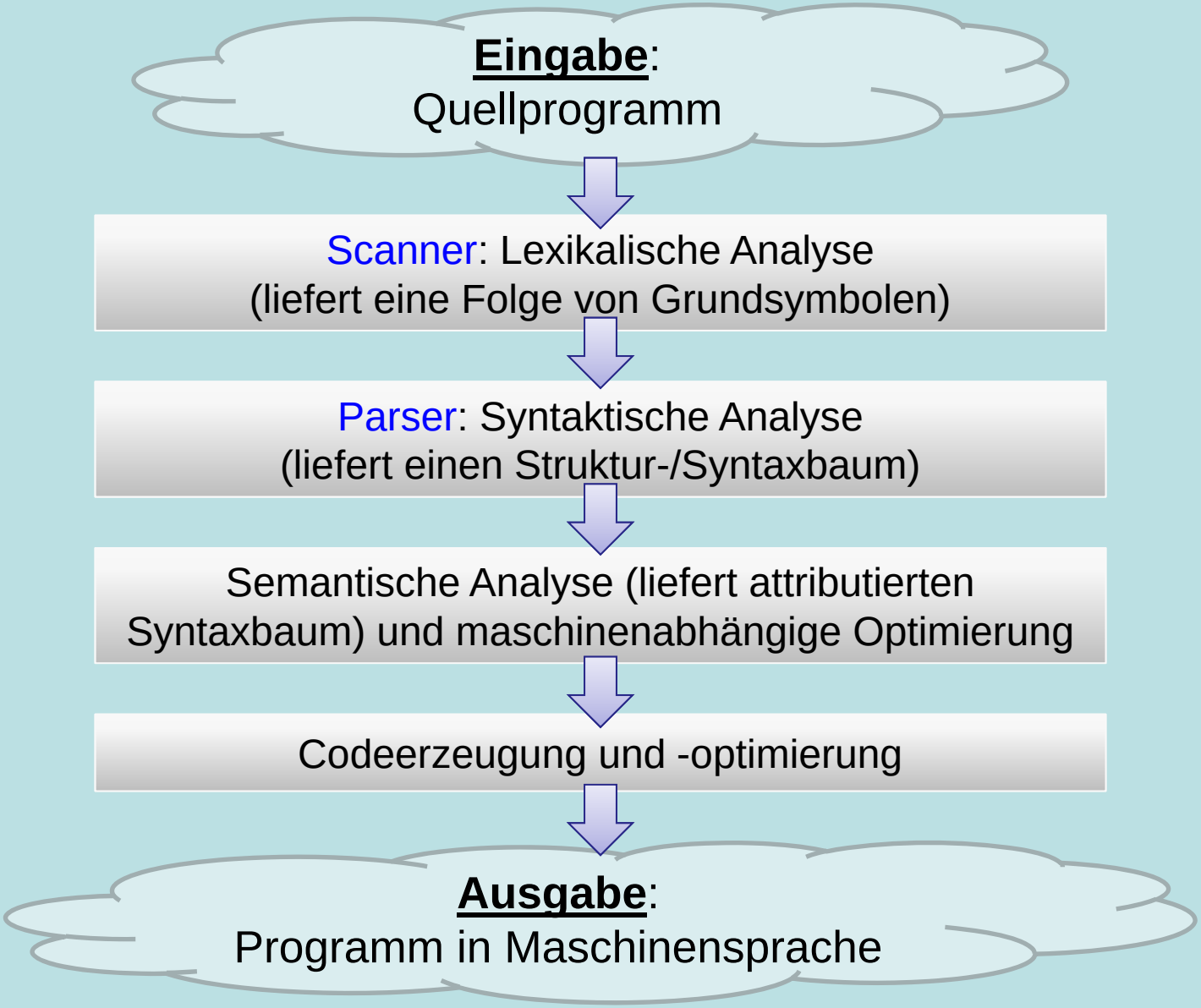
- Lexikalische und Syntaktische Analyse Forts. -

Prof. Dr. Klaus Volbert

Hochschule für angewandte Wissenschaften
Fakultät Informatik und Mathematik

Wintersemester 2010/11
Regensburg, 17./18. November 2010

Compiler


Eingabe:
Quellprogramm
Scanner: Lexikalische Analyse
(liefert eine Folge von Grundsymbolen)
Parser: Syntaktische Analyse
(liefert einen Struktur-/Syntaxbaum)
Semantische Analyse (liefert attributierten
Syntaxbaum) und maschinenabhängige Optimierung
Codeerzeugung und -optimierung
Ausgabe:
Programm in Maschinensprache

Lexikalische und Syntaktische Analyse

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Mittels formaler Sprachen soll entschieden werden, ob ein Element zu einer vorgegebenen Klasse gehört oder nicht
- Wie kann die Menge aller lexikalisch und syntaktisch korrekten C/C++- Programme beschrieben werden?
- Sei L die Menge aller syntaktisch korrekten C/C++-Programme, wie kann dann für ein Programm p entschieden werden, ob $p \in L$ (Wortproblem)?
- Sprachen können durch Grammatiken beschrieben werden
- Grammatiken geben Regeln (Produktionen) an, nach denen syntaktisch korrekte Sätze aufgebaut werden dürfen
- Anmerkung: Ein großer Teil der Arbeit eines Informatikers besteht in der Definition neuer Sprachen (z. B. Protokolle)

Begriffe

- Ein Alphabet $\sum$ ist eine endliche Menge von Zeichen ($\sum = \{0,1\}$)
- Ein Wort über einem Alphabet $\sum$ ist eine endliche Folge von Zeichen aus $\sum$, alle Wörter der Länge n: $\sum^n$
- Die Menge aller Wörter über $\sum$ ist $\sum^*$
- Das leere Wort ist ε, es gilt: $\varepsilon \in \sum^*$
- Wörter können zusammengesetzt werden (Konkatenation), es gilt: $\varepsilon u = u\varepsilon = u$, $\sum^+ = \sum\sum^* = \sum^* \backslash \{\varepsilon\}$ („mindestens einmal“)
- Die Länge eines Wortes ist die Anzahl der Zeichen, aus der sich ein Wort zusammensetzt, Notation: $|u|$, es gilt: $|\varepsilon| = 0$
- Eine Sprache L über einem Alphabet $\sum$ ist eine Menge von Wörtern über $\sum$, d.h. $L \subseteq \sum^*$
- Anmerkung:
  - Ein Programm ist ein Wort, d.h. hier weicht der Begriff von dem Wortbegriff in natürlichen Sprachen (z.B. deutsch) ab

Grammatiken zur Erzeugung von Sprachen

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Eine Grammatik G ist beschrieben durch ein 4-Tupel $G = (V, \sum, P, S)$ mit:
  - V ist ein endliches Alphabet von Variablen (Nicht-Terminale)
  - $\sum$ ist ein endliches Alphabet von Terminalen
  - $S \in V$ ist das Startsymbol
  - $P \subseteq ((V \cup \sum)^+ \times (V \cup \sum)^*)$ ist eine endliche Menge von Produktionen (Produktionen werden auch Regeln genannt)
- Ableitbarkeit
  - $b \in (V \cup \sum)^*$ ist (direkt) ableitbar aus $a \in (V \cup \sum)^+$, $a \to b$, wenn es eine Regel $(u, v) \in P$ und $\alpha, \beta \in (V \cup \sum)^*$ gibt: $a = \alpha\, u\, \beta$, $b = \alpha\, v\, \beta$
  - b ist aus a (indirekt) ableitbar, $a \overset{*}{\to} b$, wenn b durch endlich viele direkte Ableitungsschritte aus a erzeugbar ist
- Die von G erzeugte Sprache ist

$$L(G) = \{w \in \sum^* \mid S \overset{*}{\to} w\}$$

Beispiele

- $G_1 = (V, \sum, P, S)$ mit
  - $V = \{S, A, B\}$
  - $\sum = \{0,1\}$
  - $P = \{S \to ASB, A \to 0, B \to 1, S \to \varepsilon\}$
- Welche Sprache definiert G_1, also was ist $\mathrm{L} = \mathrm{L}(G_1)$?

- $G_2 = (V, \sum, P, S)$ mit
  - $V = \{S, R, L\}$
  - $\sum = \{0,1\}$
  - $P = \{S \to 01L0, 1L \to L11, 0L \to 0R, R1 \to 1R, R0 \to L0, L0 \to 0\}$
- Was ist $\mathrm{L} = \mathrm{L}(G_2)$?

Chomsky-Hierarchie

- Eine Grammatik $G = (V, \sum, P, S)$ heißt vom Typ Chomsky-0.
- G heißt kontextsensitiv (vom Typ Chomsky-1), wenn für jede Regel $a \rightarrow b$ gilt: $|a| \leq |b|$ (Ausnahme: $A \rightarrow \varepsilon$ wird zugelassen)
- G heißt kontextfrei (vom Typ Chomsky-2), wenn alle Regeln die Form $A \rightarrow b$ mit $A \in V$ haben
- G heißt regulär (vom Typ Chomsky-3), wenn alle Regeln die Form $A \rightarrow \varepsilon$ oder $A \rightarrow aB$ mit a $\in \sum$ und $A, B \in V$ haben
- Eine Sprache heißt kontextsensitiv, kontextfrei oder regulär, wenn sie durch eine entspr. Grammatik erzeugt werden kann
- Sei L_i die Menge der Typ Chomsky-i Sprachen, dann gilt:

$$L_3 \subseteq L_2 \subseteq L_1 \subseteq L_0$$

(Anmerkung: Chomsky-Hierarchie ist sogar echt)

- Welche besondere Struktur haben reguläre Sprachen?

Beispiel einer regulären Sprache

- Reguläre Grammatik über $\sum = \{0,1\}$ für

$$L_3 = \{ w \mid w \text{ enthält eine gerade Anzahl Einsen} \}$$

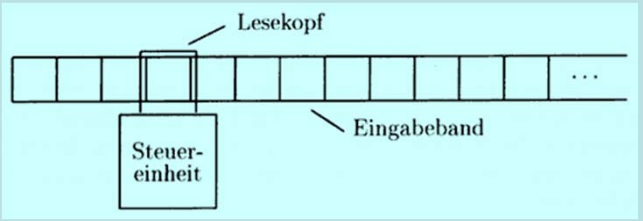
- $G_3 = (V, \sum, P, S)$ mit
  - $V = \{S, A\}$
  - $\sum = \{0,1\}$
  - $P = \{S \to 0S, S \to 1A, A \to 0A, A \to 1S, S \to \varepsilon\}$

Reguläre Ausdrücke

- Reguläre Sprachen können auch durch reguläre Ausdrücke beschrieben werden
- Reguläre Ausdrücke über $\sum$ sind wie folgt definiert:
  - $\emptyset = \{\}$ und $\{\varepsilon\}$ sind regulär
  - $\{a\}$ ist regulär für jedes $a \in \sum$
  - Sind L_1, L_2 regulär, dann auch $L_1 + L_2$, $L_1 \cdot L_2$, L_1^* mit
    - $L_1 + L_2$ Vereinigung von L_1 und L_2
    - $L_1 \cdot L_2$ Verkettung von L_1 und L_2 ($= \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$)
    - L_1^* Kleene'sche Abschluss: $\bigcup_{i \geq 0} L_1^i$ (…was ist L_1^+?)
  - Jeder reguläre Ausdruck kann durch Anwendung der oben angegebenen Regeln erzeugt werden
- Prioritäten: 1) *, 2) $\cdot$, 3) +
- Beispiel:
  - $L_{DezKonst} = (1+2+3+4+5+6+7+8+9) \cdot (0+1+2+3+4+5+6+7+8+9)^* + 0$
- Anwendung in Werkzeugen (Editoren) und Sprachen (perl, php, …)

Deterministischer endlicher Automat

- Ein deterministischer endlicher Automat (DEA/DFA) ist beschrieben durch ein 5-Tupel $\mathrm{M} = (Q, \sum, \delta, q_0, F)$ mit:
  - Q ist eine endliche, nichtleere Menge von Zuständen
  - $\sum$ ist ein endliches, nicht leeres Eingabealphabet
  - $\delta: Q \times \sum \rightarrow Q$ ist die Übergangsfunktion
  - $q_0 \in Q$ ist der Startzustand
  - $F \subseteq Q$ ist die Menge der akzeptierenden Endzustände



- Akzeptanz
  - Ein DFA akzeptiert eine Eingabe $x_1, \ldots, x_n$, wenn gilt:

$$q_0 x_1, \ldots, x_n \overset{*}{\rightarrow} x_1, \ldots, x_n q \text{ mit } q \in F$$

  - Alternativ durch induktive Fortsetzung von δ auf $Q \times \sum^* \rightarrow Q$:

$$\delta(q, \varepsilon) = q \text{ für n > 0, sonst } \delta(q, x_1, \ldots, x_n) = \delta(\delta(q, x_1, \ldots, x_{n-1}), x_n)$$

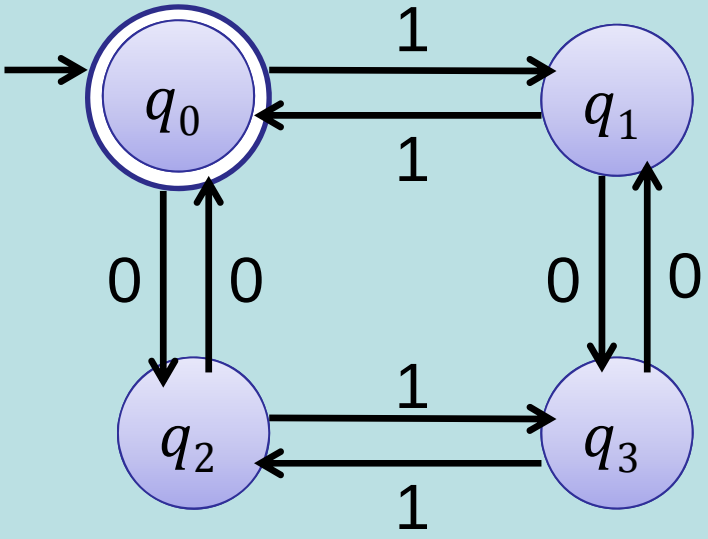
  - Damit: Ein DFA akzeptiert $\mathrm{x} = x_1, \ldots, x_n \Leftrightarrow \delta(q_0, x) \in F$

Beispiel (Automat verstehen)

- $\mathrm{M} = (Q, \sum, \delta, q_0, F)$ mit
  - $Q = \{q_0, \ldots, q_3\}$
  - $\sum = \{0,1\}$
  - $F = \{q_0\}$

δ	q_0	q_1	q_2	q_3
0	q_2	q_3	q_0	q_1
1	q_1	q_0	q_3	q_2

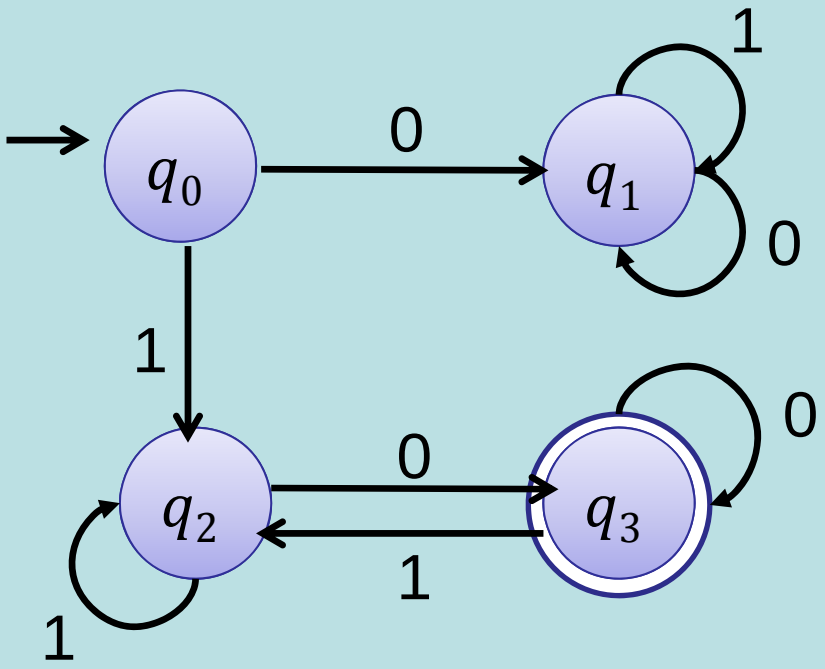
- Grafische Darstellung (Gerichteter Graph mit Knoten und Kanten):



- Welche Sprache akzeptiert der DFA M?

Beispiel (Automat entwerfen)

$$L = \{ w \mid w \text{ beginnt mit 1 und endet mit 0} \}$$



δ	q_0	q_1	q_2	q_3
0	q_1	q_1	q_3	q_3
1	q_2	q_1	q_2	q_2