Grundlagen der Informatik

- Lexikalische und Syntaktische Analyse Forts. -
- Schaltnetze und Schaltwerke -

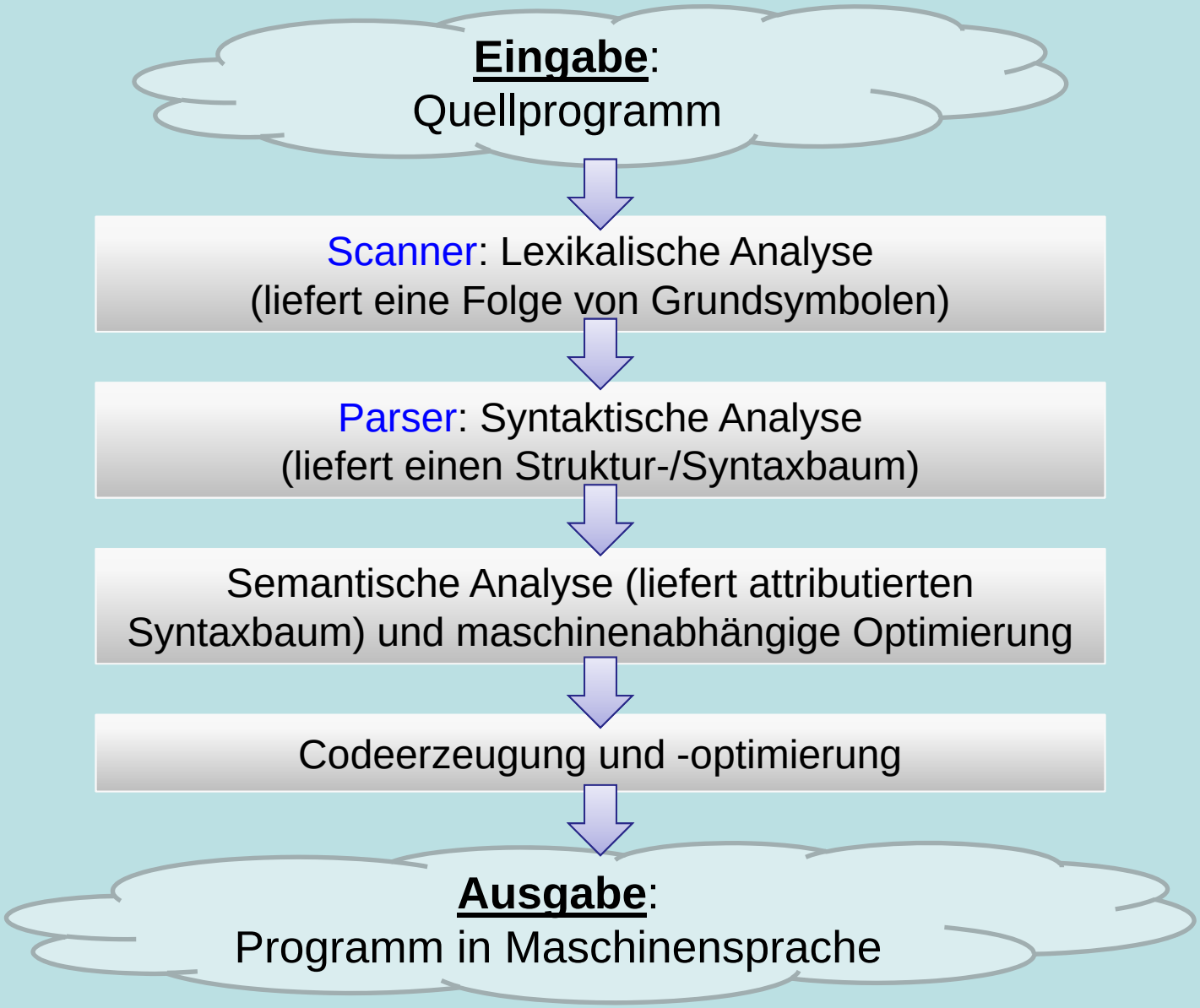
Prof. Dr. Klaus Volbert

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

Hochschule für angewandte Wissenschaften
Fakultät Informatik und Mathematik

Wintersemester 2010/11
Regensburg, 30. 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

Syntaktische Analyse (Syntaxanalyse)

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Ziele der Syntaktischen Analyse
  - Identifikation von Syntaxfehlern
  - Analyse der Struktur eines Programms
- Wesentliche Anwendungen
  - Abarbeitung arithmetischer Ausdrücke
  - Überprüfung von Klammersetzungen
- Syntaxanalyse kann mit Hilfe von kontextfreien Grammatiken durchgeführt werden
- Beispiel:
  - Abarbeitung arithmetischer Ausdrücke mit +, -, *, /, (,), Variablen, Konstanten ist kontextfrei
  - Vorsicht: Grammatik legt Ausarbeitungsreihenfolge fest und bestimmt die Mehrdeutigkeit

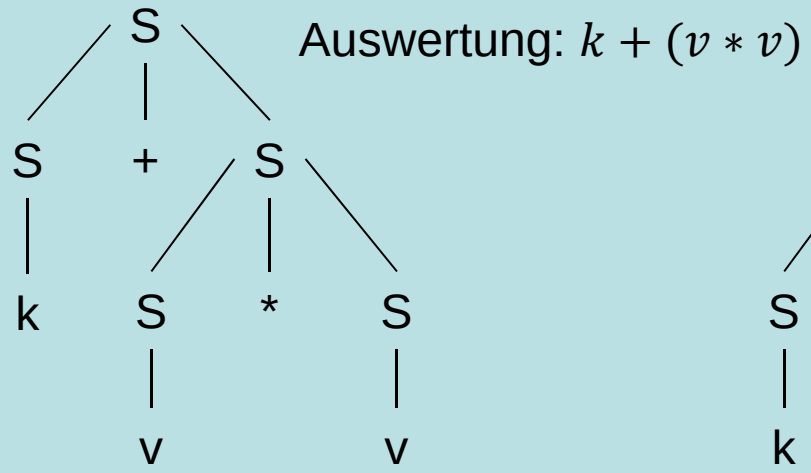
Arithmetische Ausdrücke (Grammatik 1)

- Sei $G_1 = (V, \sum, P, S)$ mit
  - $V = \{S\}$
  - $\sum = \{+,-,*,/,(,),\text{v},\text{k}\}$
  - $P = \{ S \to S + S, S \to S - S,$
    $S \to S * S, S \to S/S, S \to (S), S \to v, S \to k\}$

Grammatik ist mehrdeutig!

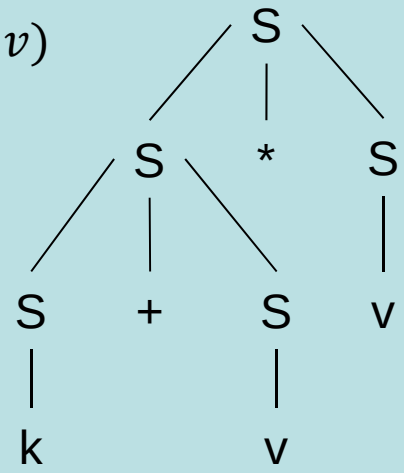
- Erzeugbares Wort: k + v * v
- Ableitungsbaum (Baum besteht aus 1 Wurzel, n Knoten, m Blättern):

$S \to S + S \to k + S * S \to k + v * v$



Auswertung: $k + (v * v)$

$S \to S * S \to S + S * v \to k + v * v$



Auswertung: $(k + v) * v$

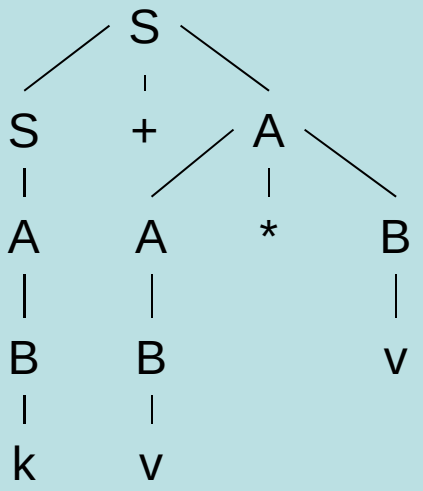
Arithmetische Ausdrücke (Grammatik 2)

- Sei $G_2 = (V, \sum, P, S)$ mit
  - $V = \{S, A, B\}$
  - $\sum = \{+,-,*,/,(,),\text{v},\text{k}\}$
  - $P = \{ S \to S + A, S \to S - A, S \to A,$
    $A \to A * B, A \to A/B, A \to B$
    $B \to (S), B \to v, B \to k\}$

Grammatik ist eindeutig!

- Erzeugbares Wort: k + v * v, Ableitungsbaum:

$$S \to S + A \to A + A * B \to B + B * v \to k + v * v$$



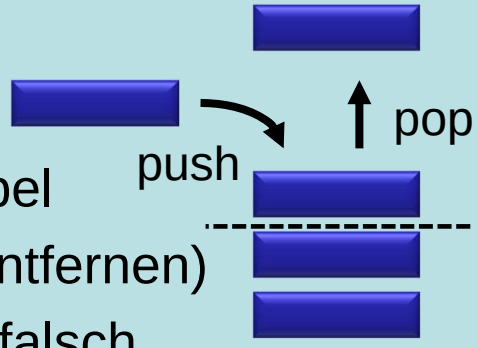
Auswertung: $k + (v * v)$

Backus-Naur-Form (BNF)

- Eine sehr häufig benutzte Darstellungsform kontextfreier Sprachen ist die Backus-Naur-Form
  - Terminale in Anführungszeichen '…' oder als Konstanten
  - Nicht-Terminale in Klammern <…>
  - Startsymbol ist ein Nicht-Terminal
  - Produktionen: statt → wird ::= verwendet
- Beispiel: (Pascal-Darstellung in BNF)
  - `<Programm>` ::= 'PROGRAM' <Bezeichner> 'BEGIN' <Satzfolge> 'END' .
  - `<Bezeichner>` ::= <Buchstabe> <Restbezeichner>
  - `<Restbezeichner>` ::= (leer) | <Buchstabe oder Ziffer> <Restbezeichner>
  - `<Buchstabe, Ziffer>` ::= <Buchstabe> | <Ziffer>
  - `<Buchstabe>` ::= A | B | C | D | … | Z | a | b | … | z
  - `<Ziffer>` ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
  - `<Satzfolge>` ::= …
  - …
- Anmerkung
  - Es gibt Erweiterungen, z.B. Erweiterte Backus-Naur-Form (EBNF)

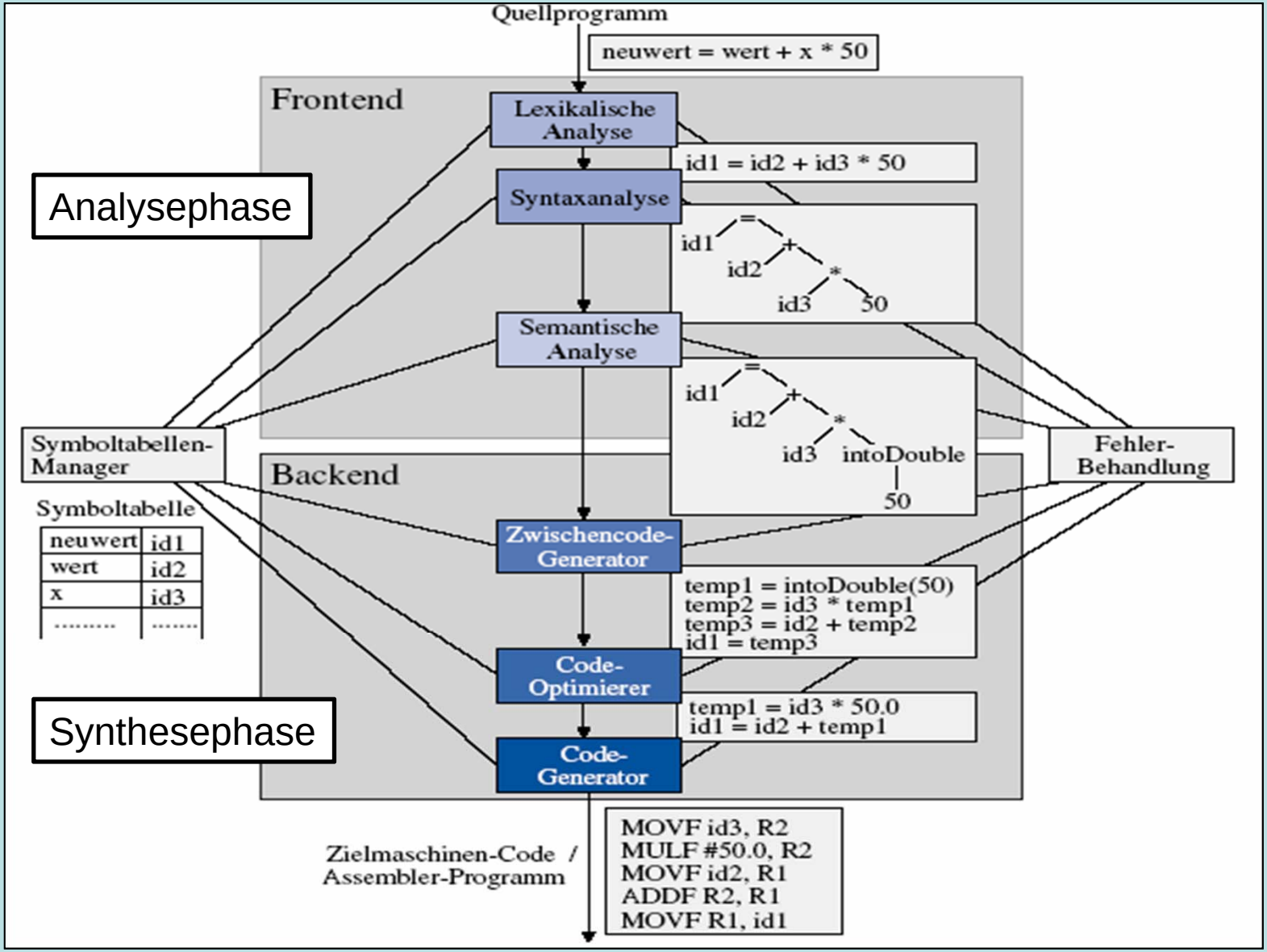
Idee eines Kellerautomaten

- Beobachtung bei Klammerausdrücken:
  Ein DFA kann nicht alle kontextfreien Sprachen erzeugen
- Es fehlt ein „Gedächtnis“, über das „mitgezählt“ werden kann, wie viele Klammern von welchem Typ bereits geöffnet sind
- Zur Lösung des Problems muss ein DFA nur um eine einfache Struktur erweitert werden: Einen Stapel (engl. Stack) oder Keller.
- Ein Stapel ist eine Datenstruktur, die eine begrenzte Menge von Objekten aufnehmen kann und folgende Operationen unterstützt (Last-In-First-Out, LIFO-Prinzip):
  - push: Legt ein Objekt auf den Stapel
  - pop: Holt u. entfernt das oberste Objekt vom Stapel
  - top: Holt das oberste Objekt vom Stapel (ohne entfernen)
  - empty: Liefert wahr, wenn der Stapel leer ist, sonst falsch



- Ein Kellerautomat ist ein DFA mit Stapel und entsprechend erweiterter Übergangsfunktion (erzeugt kontextfreie Sprachen)

Übersicht Compilerphasen



HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Schaltnetze und Schaltwerke -

Konjunktive Normalform I

- Seien $x_0, \ldots, x_{n-1}$ Boolesche Variablen
- x_i heißt positives Literal, $\overline{x_i}$heißt negatives Literal
- Eine Klausel (Disjunktionsterm) besteht aus der Disjunktion von endlich vielen Literalen und ist ein Boolescher Ausdruck der Form

$$K_t = (l_0 \vee l_1 \vee \ldots \vee l_{m-1}) \text{ mit}$$

$$l_j \in \{x_i \mid i \in \{0, \ldots, n-1\}\} \cup \{\overline{x_i} \mid i \in \{0, \ldots, n-1\}\}$$

- Ein Boolescher Ausdruck in Konjunktiver Normalform (KNF) ist eine Konjunktion von Klauseln:

$$\bigwedge_{t=1}^{s} K_t$$

- Beispiele: $(x_0 \vee x_1 \vee x_3)$, $(x_2 \vee x_5) \wedge (\overline{x_2} \vee x_3)$

Konjunktive Normalform II

- Satz: Jeder Boolesche Ausdruck kann in konjunktiver Normalform dargestellt werden
- Anmerkungen:
  - Eine KNF ist in i-KNF, wenn jede Klausel maximal i Literale enthält
  - Analog lässt sich die disjunktive Normalform (DNF) definieren
  - Boolesche Ausdrücke, die in KNF oder DNF vorliegen, können direkt in ein zweistufiges Schaltnetz überführt werden
- Erfüllbarkeit
  - Ein boolescher Ausdruck (in KNF/DNF) heißt erfüllbar, wenn es eine Belegung für $x_0, \ldots, x_{n-1}$ gibt, die den Ausdruck wahr macht (Wert: 1)
- Beispiele:
  - Sind die folgenden booleschen Ausdrücke erfüllbar?

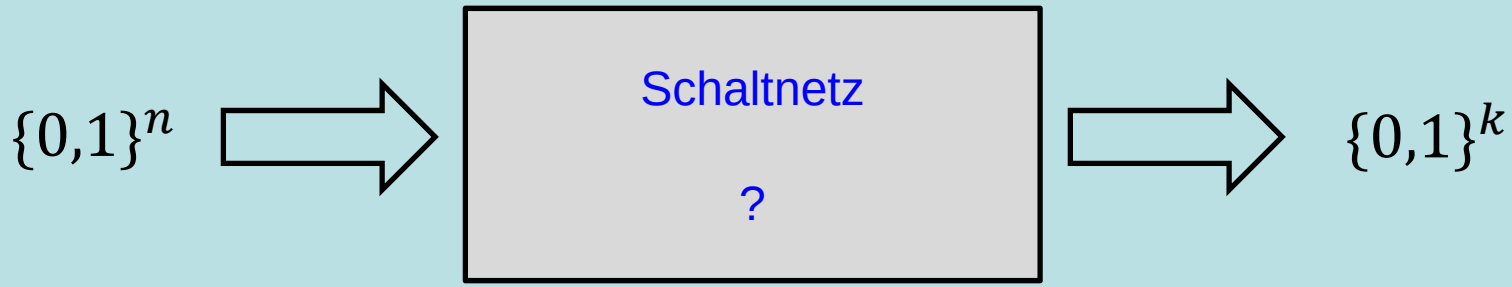
$$(x_0 \vee x_1 \vee x_3),\ (x_2 \vee x_5) \wedge (\overline{x_2} \vee x_3),\ (x_2 \vee x_5) \wedge (\overline{x_2} \vee \overline{x_5})$$

- Vorschau Erfüllbarkeitsproblem:

SAT= $\{K \mid K$ ist erfüllbarer boolescher Ausdruck in KNF$\}$, SAT ist „schwer“

Schaltnetze

- Gegeben: $f: \{0,1\}^n \rightarrow \{0,1\}^k$
- Gesucht: Schaltnetz, das die Funktion implementiert (Zusammenschaltung von Gattern)



- … oder umgekehrt: Schaltnetz gegeben, Funktion gesucht

Allgemeines Verfahren

- Eine beliebige Schaltung kann wie folgt aufgestellt werden:
  1. Aufstellen der Wahrheitstabelle zur gesuchten Schaltung
  2. Herleiten einer Normalform, hier KNF (DNF analog: 1/0, Disj./Konj. getauscht)
     - Zu allen Zeilen in der Wahrheitstabelle, denen eine 0 zugeordnet ist, wird eine Disjunktion aufgestellt
     - Dabei sind die mit 1 belegten Variablen zu negieren
     - Alle aufgestellten Disjunktionen werden dann konjunktiv verknüpft
  3. Minimierung des booleschen Ausdrucks durch Äquivalenz-umformung (Anwendung der Gesetze der booleschen Algebra)
- Entwurf von Schaltnetzen
  – Bottom-Up-Entwurf
    Sukzessive Zusammensetzung aus elementaren Bausteinen
  – Top-Down-Entwurf
    Zerlegung in wohldefinierte Teilaufgaben und anschließende Realisierung und Zusammenfügung der einzelnen Komponenten