



Compiler

Kapitel 4

Syntaktische Analyse

Folie: 1

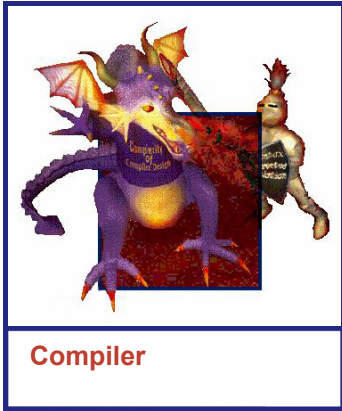
Kapitel 4

Syntaktische Analyse



Autor:
Aho et al.

© Pearson Studium 2008



Kapitel 4
Syntaktische Analyse



Autor:
Aho et al.

© Pearson Studium 2008

Übersicht

Symboltabelle

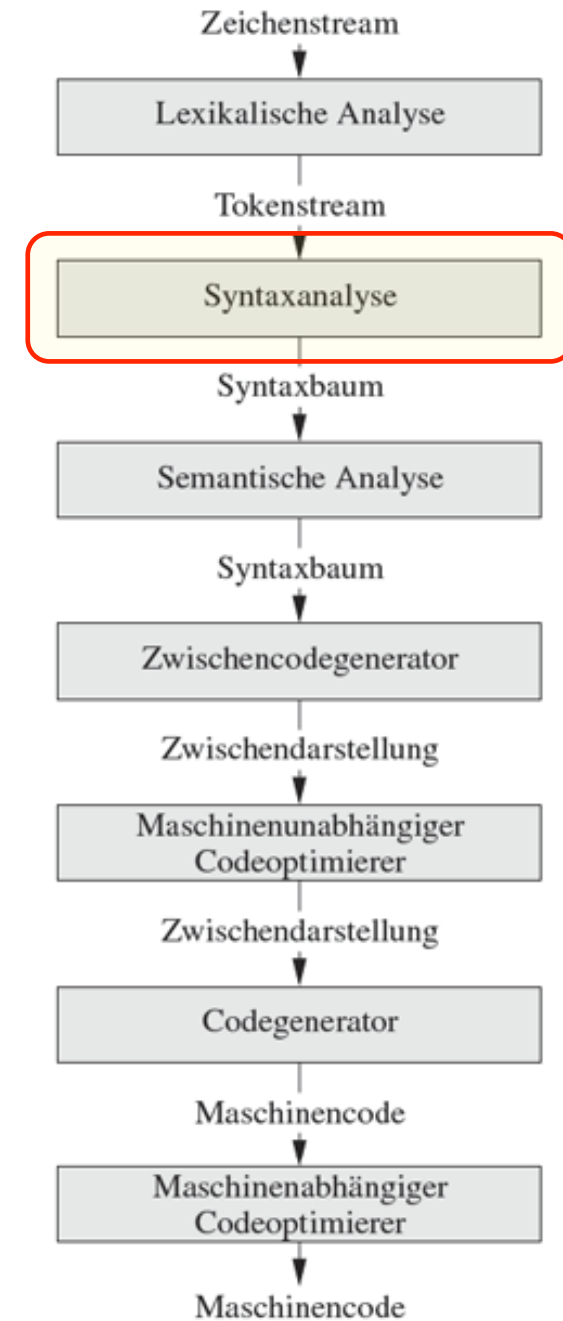


Abbildung 1.6: Phasen eines Compilers



Compiler

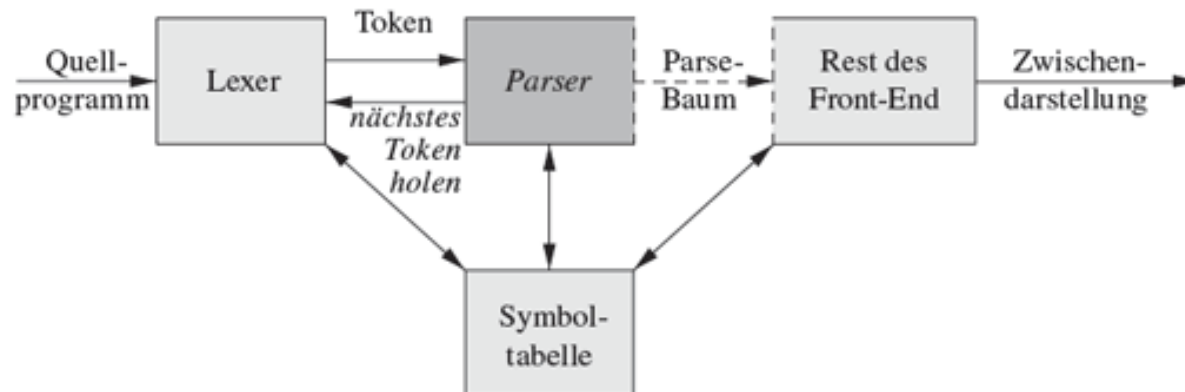
Kapitel 4

Syntaktische Analyse

Übersicht

1	position	...
2	initial	...
3	rate	...

SYMBOLTABELLE



position = initial + rate * 60

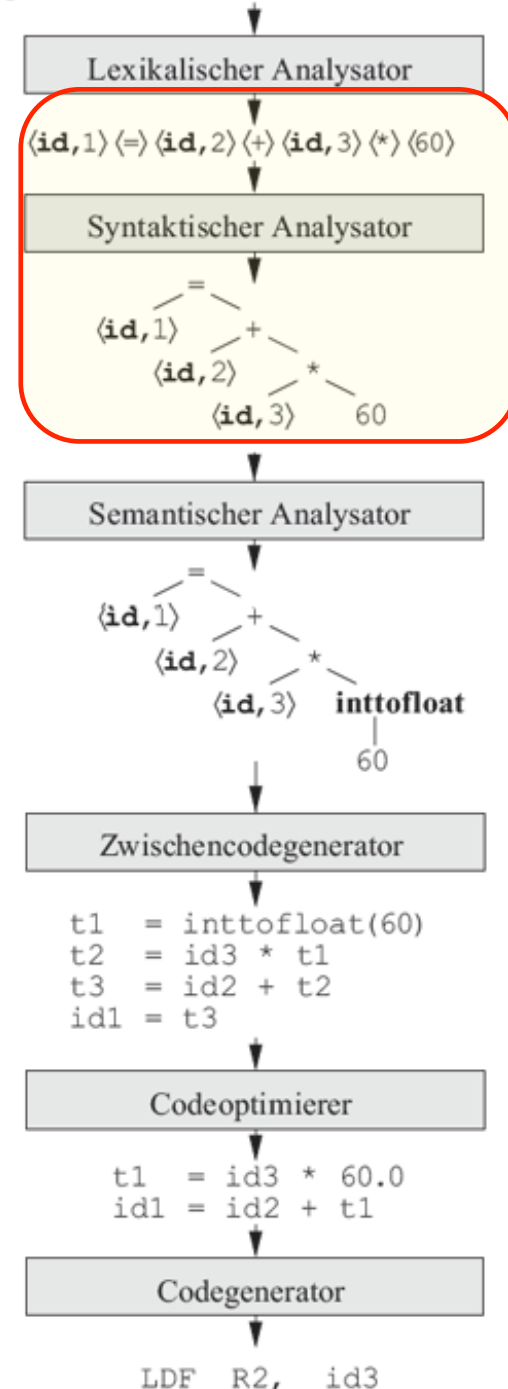


Abbildung 4.1: Stellung des Parsers im Compilermodell



Compiler

Kapitel 4

Syntaktische Analyse

4



Autor:
Aho et al.

© Pearson Studium 2008

Syntax: Definition

syn-tax: the way in which words are put together to form phrases, clauses, or sentences.

– *Webster's Dictionary*

Die Syntax (griechisch σύνταξις ['syntaksis] - *die Zusammenstellung*) behandelt die Muster und Regeln, nach denen Wörter zu größeren funktionellen Einheiten wie Phrasen (*Teilsätze*) und Sätzen zusammengestellt und Beziehungen wie Teil-Ganzes, Abhängigkeit etc. zwischen diesen formuliert werden (*Satzbau*).

– *Wikipedia*



Compiler

Kapitel 4

Syntaktische Analyse

Folie: 5

Natürliche Sprache

this is some text without spaces and punctuation marks which is therefore quite difficult to read by humans lexical analysis will break this text up into words while the parsing phase will extract the grammatical structure of the text

this is some text without spaces and punctuation marks which is therefore quite difficult to read by humans lexical analysis will break this text up into words while the parsing phase will extract the grammatical structure of the text

This **is** some **text** without **spaces** and **punctuation-marks** which **is** therefore quite difficult to **read** by **humans**. **Lexical-analysis** will **break** this text up into **words** while the **parsing-phase** will **extract** the **grammatical structure** of the **text**.



Compiler

Kapitel 4

Syntaktische Analyse

Folie: 6

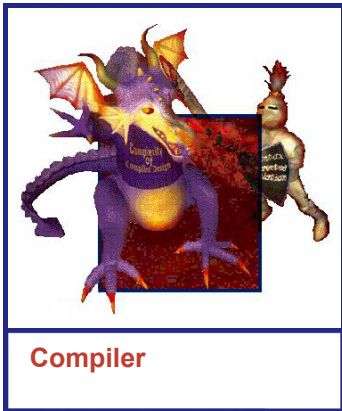
PEARSON
Studium **it**
informatik

Autor:
Aho et al.

© Pearson Studium 2008

Wie soll die Syntax einer Programmiersprache beschrieben werden?

- Beispiel:
 - If – Then – Else Anweisung



Compiler

Kapitel 4

Syntaktische Analyse

Folie: 7



Autor:
Aho et al.

© Pearson Studium 2008

Deutsche Grammatik

1. **Satzbauplan – Hauptsatz** Der 1. Satzbauplan hat die Reihenfolge:

Subjekt – finitives Prädikat – indirektes Objekt – direktes Objekt – Adverbien – Prädikatrest

Die Satzverneinung steht vor dem Prädikatrest.

Beispiel:

„der Verkäufer – hatte – seinem Kunden – das Buch – gestern – in seinem Laden – (nicht) – gegeben.“

Quelle: Wikipedia.de



Compiler

Kapitel 4

Syntaktische Analyse

10

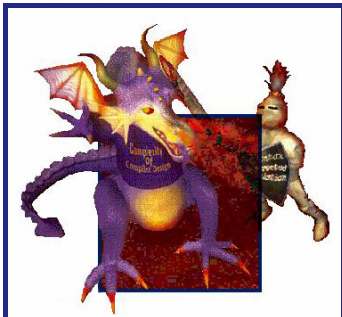


Autor:
Aho et al.

© Pearson Studium 2008

Syntax Analyse: Übersicht

- Beschreibung der syntaktischen Struktur von Programmen
 - kontext freie Grammatiken (kfG)
 - beschreiben: Zuweisungen, Tests, ..., Programme
- Erkennen der syntaktischen Struktur (“parsing”)
 - Top-Down parsing (LL(1))
 - Bottom-Up Shift/reduce parsing (LR(1))
- yacc/sablecc/... Werkzeuge



Compiler

Kapitel 4

Syntaktische Analyse

11

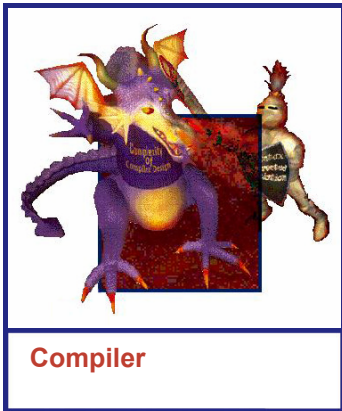
PEARSON
Studium **it**
informatik

Autor:
Aho et al.

© Pearson Studium 2008

Warum studieren wir Parsing?

- Essenziell für die semantische Analyse
- Viele andere Anwendungen:
 - Natural Language Understanding
 - theoretische Informatik
 - ...



Kapitel 4

Syntaktische Analyse

Folie: 12



Autor:
Aho et al.

© Pearson Studium 2008

Wiederholung: Formale Sprachen

- **Alphabet** Σ : endliche Menge von Symbolen
 - $\{0,1\}$, $\{a,b,c,\dots,z\}$, Ascii, ...
- **String** (Wort): **endliche Folge** von Symbolen $\in \Sigma$
 - 101, helloworld, if a=0 then a:= b
- **Sprache** = **abzählbare Menge** von Strings
 - Primzahlen im Binärformat, English, Java Programme
 - Lexeme für ein Token !!
Identifizier : $\{a,b,\dots,foo,\dots,x_3,\dots\}$



Compiler

Kapitel 4

Syntaktische Analyse

13

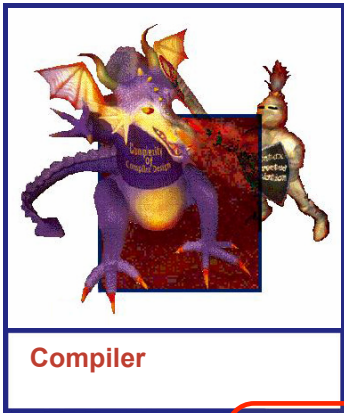
PEARSON
Studium **it**
informatik

Autor:
Aho et al.

© Pearson Studium 2008

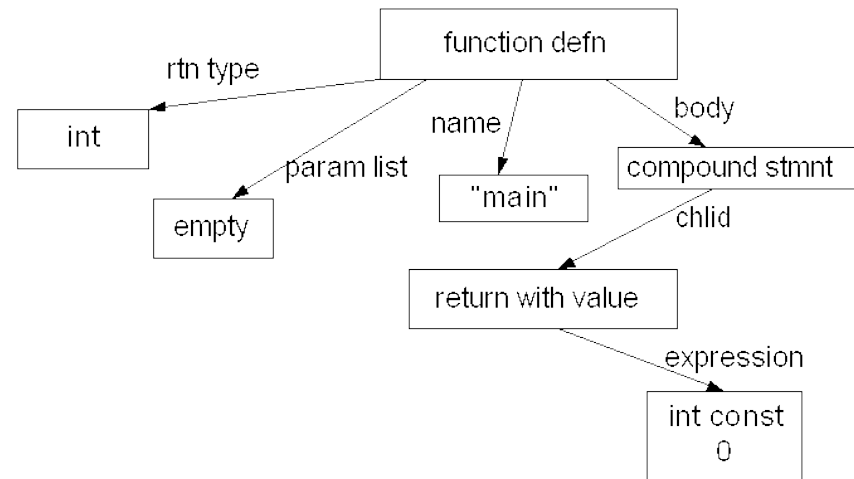
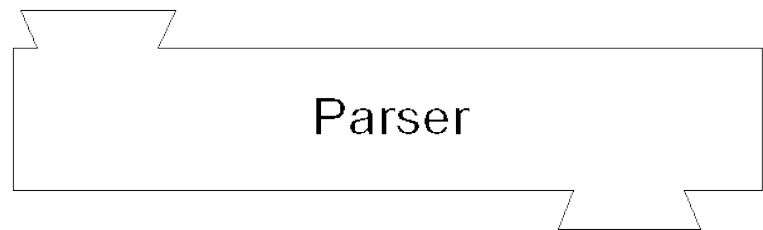
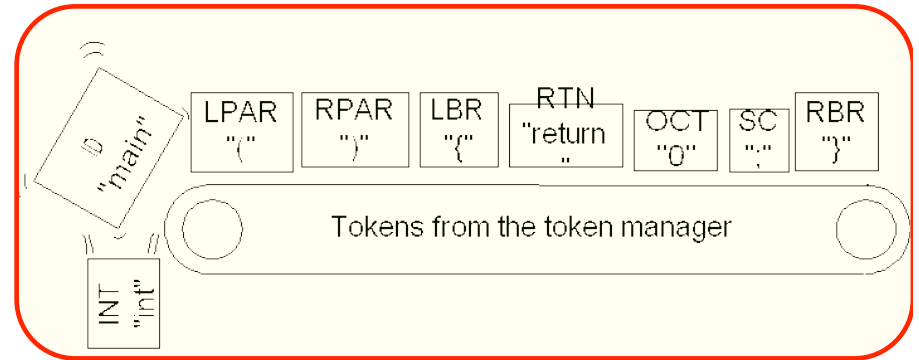
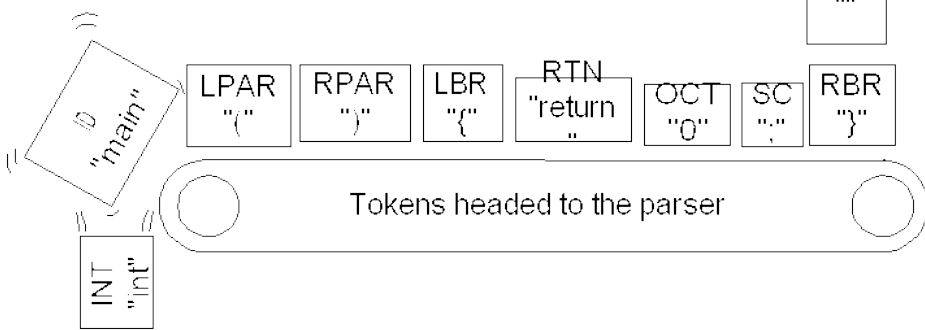
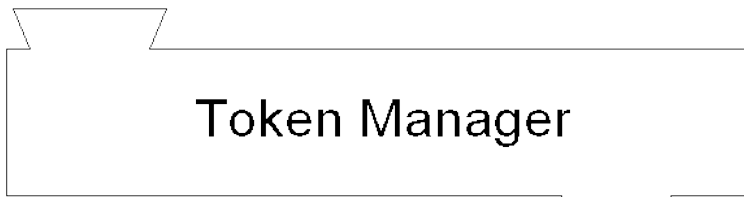
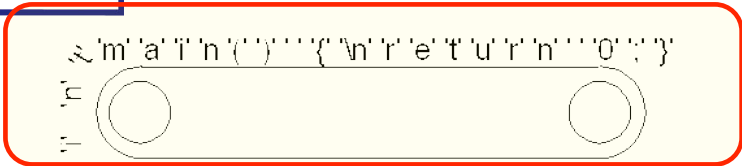
Anwendung der formalen Sprachen

- Beim Lexing:
 - Zuordnung zu **Token Klassen**
 - $L(\text{num}) = \{“0”, “1”, “2”, “3”, \dots, “10”, “11”, \dots\}$
 - $L(\text{id}) = \{“a”, “b”, \dots, “a1”, “a2”, \dots, “aa”, “ab”, \dots\}$
 - $\Sigma = \text{Ascii oder Unicode}$
- Beim Parsing:
 - Zuordnung zu **grammatischen Konstrukten**
 - $L(\text{assignment}) = \{ “id = id”, “id = num”, “id = id + id”, “id = id + num”, “id = num + id”, \dots \}$
 - $\Sigma = \text{Token (oder Ascii/Unicode wenn kein Lexing)}$



Compiler

Alphabete





Compiler

Kapitel 4

Syntaktische Analyse

15

Beschreibung der Sprachen I

```
if n == ((x+2)*3) then
    return 0
else
    ....
```

Aufzählung

- Beispiel: Arithmetische Ausdrücke über natürliche Zahlen mit **+**, *****, **(,)**:

$\{0, 1, 2, 3, \dots, 0+0, 0+1, 0+2, \dots, 1+0, 1+1, 1+2, \dots,$
 $0*0, 0*1, 0*2, \dots, 0+0+0, 0+0+1, \dots$
 $0+0*0, 0+0*1, 0+0*2, \dots$
 $0*0+0, 0*0+1, \dots, 1*0+0, 1*0+1, \dots$
 $(0+0), (0+1), (0+2), \dots, (1+0), (1+1), (1+2), \dots,$
 $(0*0), (0*1), (0*2), \dots, (0+0)+0, (0+0)+1, \dots\}$

Können wir reguläre Ausdrücke verwenden ?



Compiler

Kapitel 4

Syntaktische Analyse

16

Beschreibung der Sprachen II

Reguläre Ausdrücke: a , ε , $M|N$, MN , M^*

– Beispiel: Arithmetische Ausdrücke über natürliche Zahlen mit $+$, $*$, $($, $)$:

– $\text{Nat} = 0 \mid [1-9][0-9]^*$ $\{0,1,2,3,\dots\}$

– $\text{Ex} = \text{Nat} ((\text{"+"} \mid \text{"*"}) \text{Nat})^*$

$\{0,\dots,0+0,\dots,0*0,\dots\}$

– Mit Klammern ?



Compiler

Kapitel 4

Syntaktische Analyse

PEARSON
Studium **it**
informatik

Autor:
Aho et al.

© Pearson Studium 2008

Quiz: Ein einfacheres Problem (or *Who wants to be a millionaire?*)

- Beschreiben Sie die Sprache $\{\text{"ab"}, \text{"aabb"}, \text{"aaabbb"}, \dots\}$ mit einem regulären Ausdruck
 - **Unmöglich!**
 - Das Gleiche gilt für:
 - Korrekt geklammerte Ausdrücke $((\dots) \dots)$
 - Korrekt geschachtelte Schleifen,...
- ⇒ Reguläre Ausdrücke reichen zum Parsen nicht aus



Compiler

Kapitel 4

Syntaktische Analyse

18

PEARSON
Studium **it**
informatik

Autor:
Aho et al.

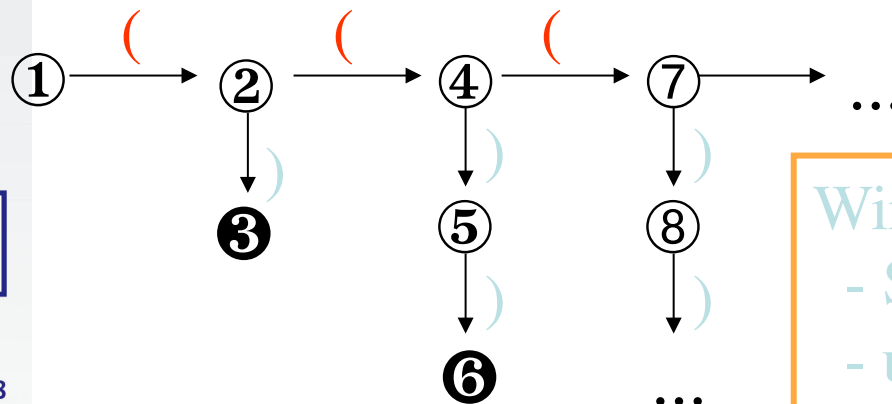
© Pearson Studium 2008

Erklärung

Beschreibung von $\{“()”, “(())”, “((()))”, \dots\}$
mit einem regulären Ausdruck:
unmöglich!

Warum:

- Regulärer Ausdruck \rightarrow endlicher DFA
- DFA: Zustände haben keinen Speicher



Wir brauchen
- Speicher oder
- unendlich viele Zustände



Compiler

Kapitel 4

Syntaktische Analyse

Klammern Problem

- $x := (a+b) * 2;$ ✓
- $x := (a+b*2;$ ✗
- `if x>2 then return x*x else
return 0;` ✓
- `if x>2 then return x*x if
y>2;` ✗
- `else return 0 then return
x*x;` ✗

⇒ kontextfreie Grammatiken (Rekursion)



Compiler

Kapitel 4

Syntaktische Analyse

20

Bausteine von kontextfreien Grammatiken

N = Menge von **Nichtterminalen**

T = Menge von **Terminalen**

Startsymbol $S \in N$

Menge **P** von **Produktionen** der Form:

$$\begin{aligned} & - S_0 \rightarrow S_1 \dots S_k \quad \text{mit} \\ & \quad S_0 \in N, S_i \in T \cup N, k \geq 0 \end{aligned}$$

Notation:

$$- S_0 \rightarrow \alpha_1 \mid \dots \mid \alpha_k \text{ bezeichnet } \{S_0 \rightarrow \alpha_i \mid 1 \leq i \leq k\}$$



Compiler

Kapitel 4

Syntaktische Analyse

21

PEARSON
Studium **it**
informatik

Autor:
Aho et al.

© Pearson Studium 2008

Beschreibung der Sprachen IV

Beispiel: Arithmetische Ausdrücke über
Bezeichner **id** mit **+**, *****, **(**, **)**:

Version 1: $N=\{E\}$, $S=E$, $T = \{\mathbf{id}, \mathbf{+}, \mathbf{*}, \mathbf{(}, \mathbf{)}\}$

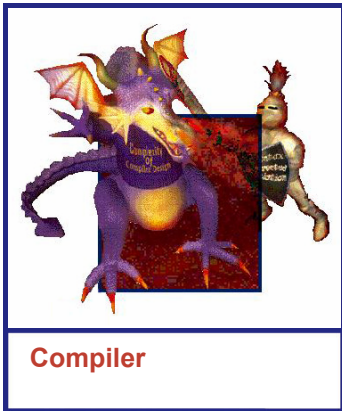
$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

Version 2: $N=\{E,T,F\}$, $S=E$, $T = \{\mathbf{id}, \mathbf{+}, \mathbf{*}, \mathbf{(}, \mathbf{)}\}$

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid \mathbf{id}$$



Kapitel 4

Syntaktische Analyse

Folie: 25



Autor:
Aho et al.

© Pearson Studium 2008

Wo ist die kfG? Deutsche Grammatik

1. **Satzbauplan – Hauptsatz** Der 1. Satzbauplan hat die Reihenfolge:

**Subjekt – finitives Prädikat – indirektes Objekt
– direktes Objekt – Adverbien – Prädikatrest**

Die Satzverneinung steht vor dem Prädikatrest.

Beispiel:

**„der Verkäufer – hatte – seinem Kunden – das
Buch – gestern – in seinem Laden – (nicht)
– gegeben.“**

Quelle: Wikipedia.de



Compiler

Kapitel 4

Syntaktische Analyse

26

Ableitungsschritt

1. Sei $A \rightarrow \gamma$ eine Produktion
2. Sei $\alpha A \beta$ ein String über $N \cup T$
3. Dann schreiben wir:

$$\alpha A \beta \Rightarrow \alpha \gamma \beta$$

Beispiel:

$$E \rightarrow E + E \mid E * E \mid (E) \mid \text{id}$$

$$E \Rightarrow (E) \Rightarrow (\text{id})$$



Compiler

Kapitel 4

Syntaktische Analyse

27

Ableitungen und Sprache

1. $\alpha \overset{*}{\Rightarrow} \alpha$ für jedes α

2. Wenn $\alpha \overset{*}{\Rightarrow} \beta$ und $\beta \Rightarrow \gamma$, dann $\alpha \overset{*}{\Rightarrow} \gamma$

1. Von Grammatik G **erzeugte Sprache**:
 $L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$

2. Zwei Grammatiken mit derselben Sprache: werden **äquivalent** genannt



Compiler

Kapitel 4

Syntaktische Analyse

28

PEARSON
Studium **it**
informatik

Autor:
Aho et al.

© Pearson Studium 2008

Die Chomsky Hierarchie

Type 3: Reguläre Sprachen

- reguläre Ausdrücke, endliche Automaten

Type 2: kontextfreie Sprachen

- kfG's, Kellerautomaten

Type 1: kontextsensitive Sprachen

- CSG's ($AB \rightarrow AC$, $CB \rightarrow CD$), LBA's

Type 0: rekursive aufzählbare Sprachen

- Turingmaschine

$\{a^n b^n c^n | n > 1\}$: Typ 1 nicht Typ 2,
 $\{a^n b^n | n > 1\}$: Typ 2 nicht Typ 3



Compiler

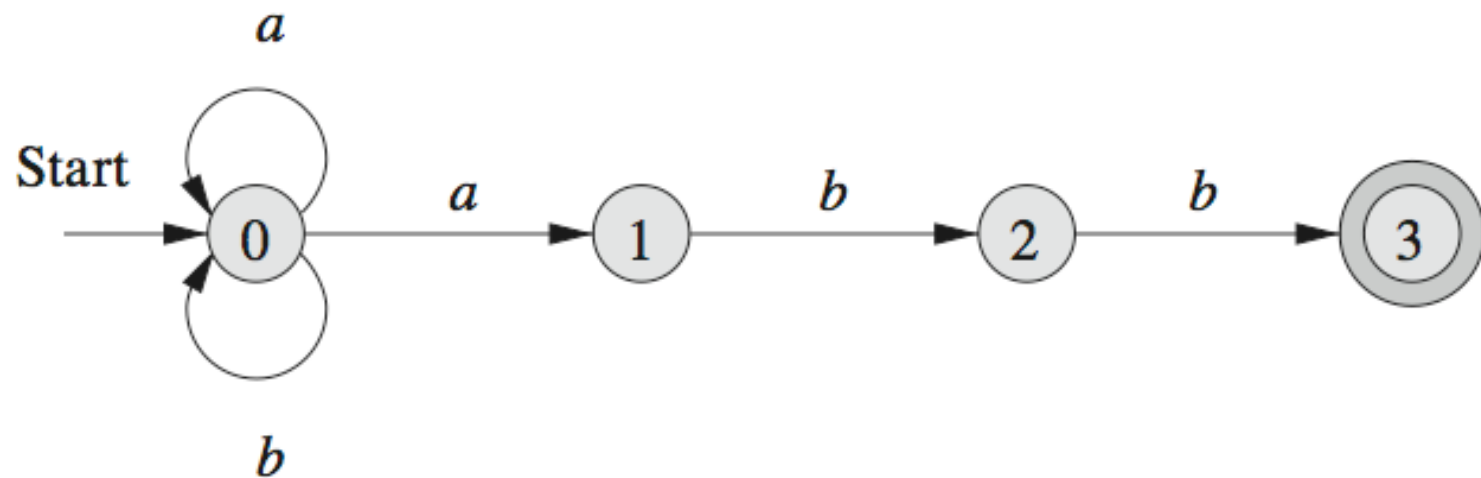
Kapitel 4

Syntaktische Analyse

Folie: 29

Reguläre Ausdrücke \rightarrow kfG

- $(a|b)^*abb$



$$A_0 \rightarrow aA_0 \mid bA_0 \mid aA_1$$

$$A_1 \rightarrow bA_2$$

$$A_2 \rightarrow bA_3$$

$$A_3 \rightarrow \epsilon$$



Compiler

Kapitel 4

Syntaktische Analyse

Quiz

$Ex \rightarrow id \mid (Ex) \mid Ex + Ex \mid Ex * Ex$

$T = \{id, (,), +, *\}, N = \{Ex\}, S = Ex$

Finden Sie (falls möglich) von Ex
aus Ableitungen für:

– $id+id*id$

– $(id+ (+id))$



Lösung

$$Ex \rightarrow id \mid (Ex) \mid Ex + Ex \mid Ex * Ex$$

Ex

Ex + Ex

id + Ex

id + Ex * Ex

id + id* Ex

id + id* id

Ex

Ex + Ex

Ex + Ex * Ex

Ex + Ex * id

Ex + id* id

id + id* id

■ Ex

■ Ex * Ex

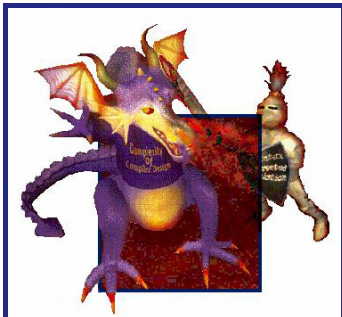
■ Ex + Ex * Ex

■ id + Ex * Ex

■ id + id* Ex

■ id + id* id

alle Strings:
Satzformen
 der
 Grammatik



Compiler

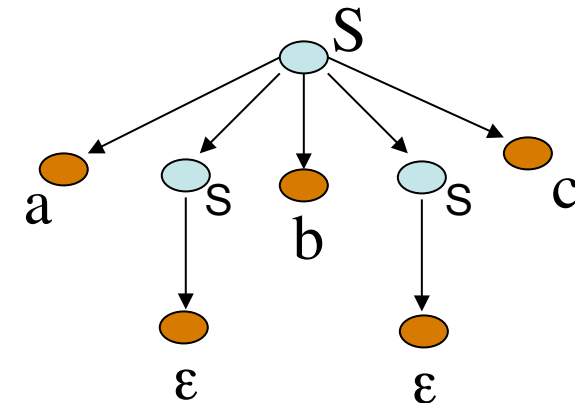
Kapitel 4

Syntaktische Analyse

32

Parsebäume

$S \rightarrow aSbSc \mid \epsilon$



Blätter:

- Terminale oder Nichtterminale
- Grenze: erzeugte Satzform

Innere Knoten:

- Nichtterminale
- Kinder: ordered, obtained by using a production rule

Wurzel: Startsymbol S



Compiler

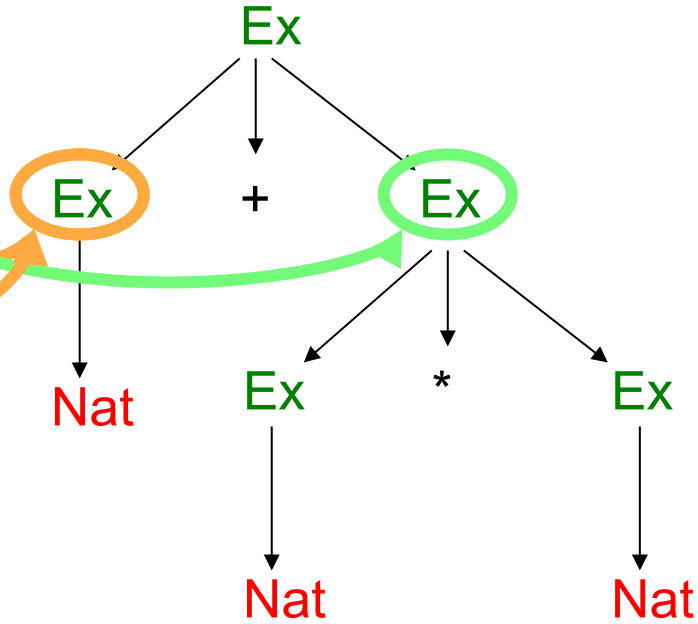
$Ex \rightarrow Nat \mid (Ex) \mid Ex + Ex \mid Ex * Ex$

Parsebäume

Ex

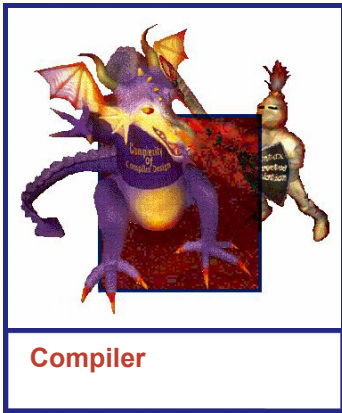
Kap
Syr

- Ex
 - Ex + Ex
 - Ex + Ex * Ex
 - Ex + Ex * Nat
 - Ex + Nat * Nat
 - Nat + Nat * Nat
- Rechtsableitun
g:



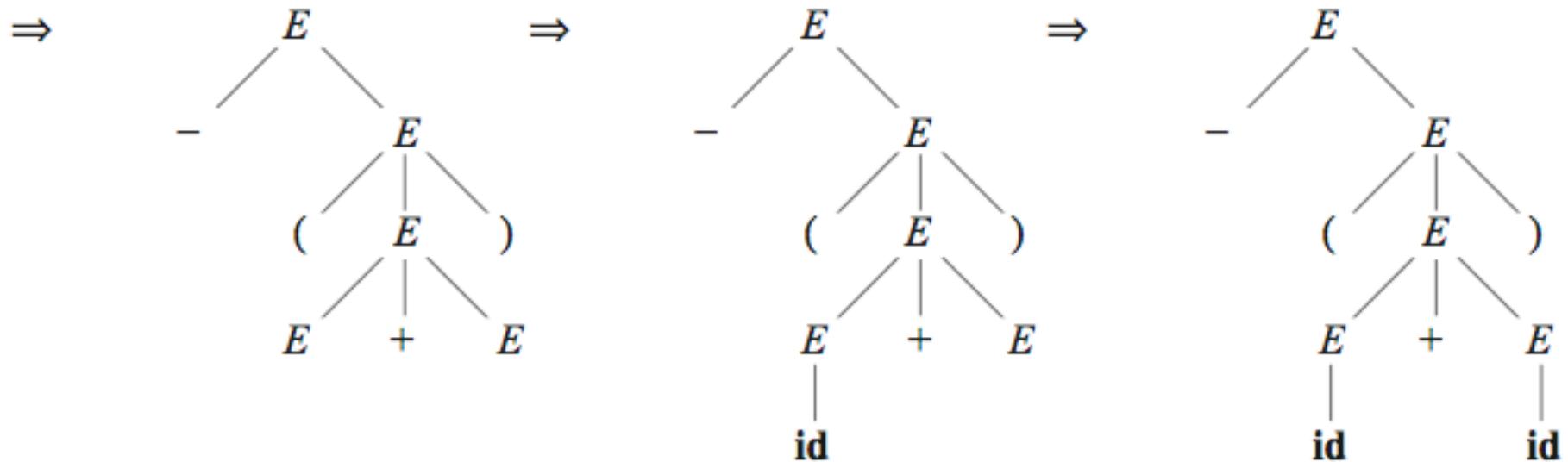
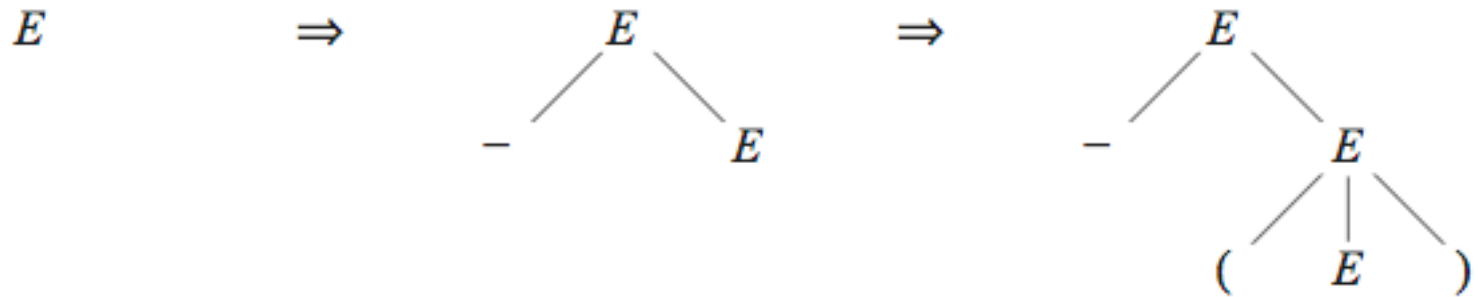
Blätter: erzeugter String

Nat + Nat * Nat



Folge von Parse-Bäume

$$E \rightarrow E + E \mid E * E \mid -E \mid (E) \mid id$$





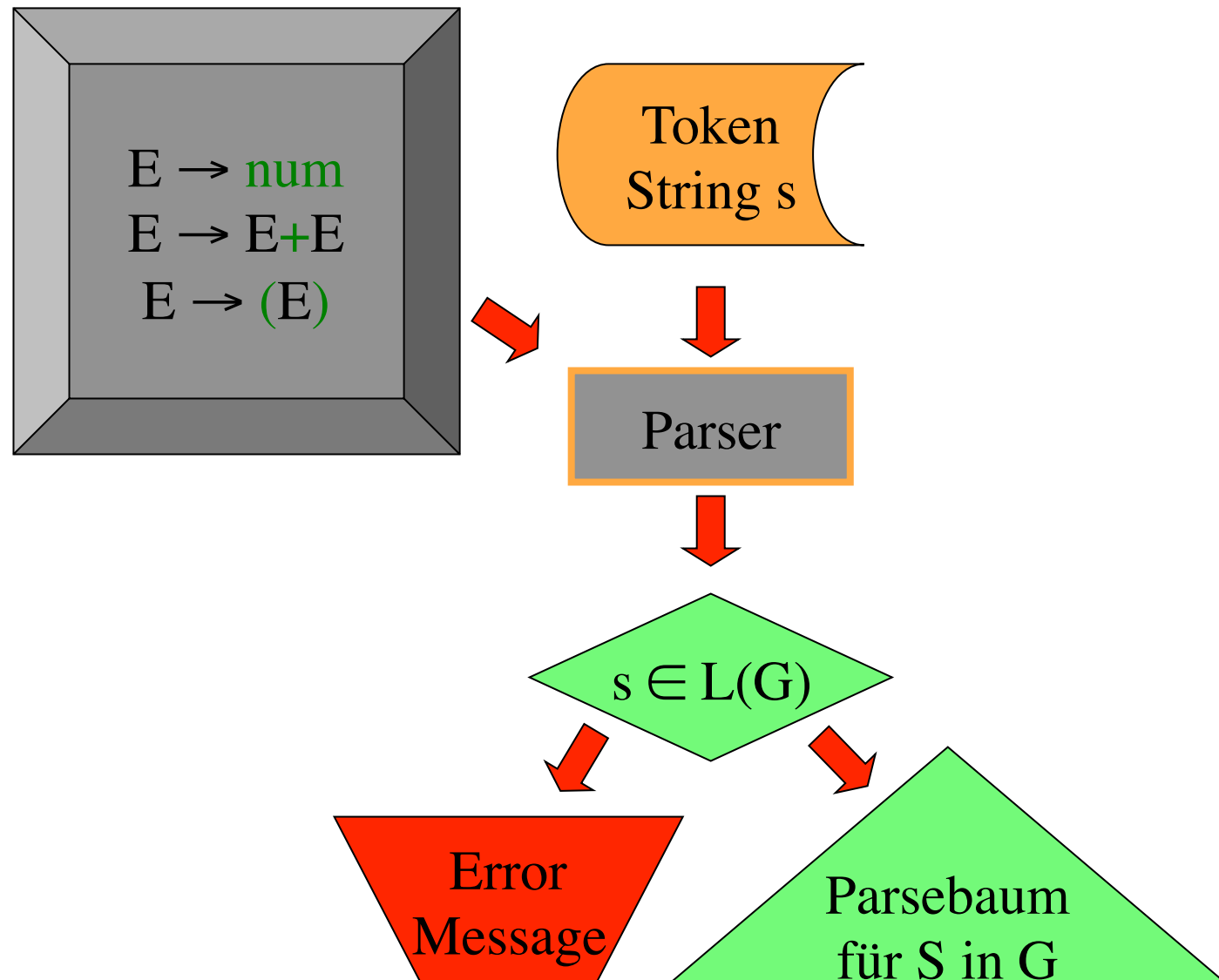
Compiler

Kapitel 4

Syntaktische Analyse

35

Was ist Parsing





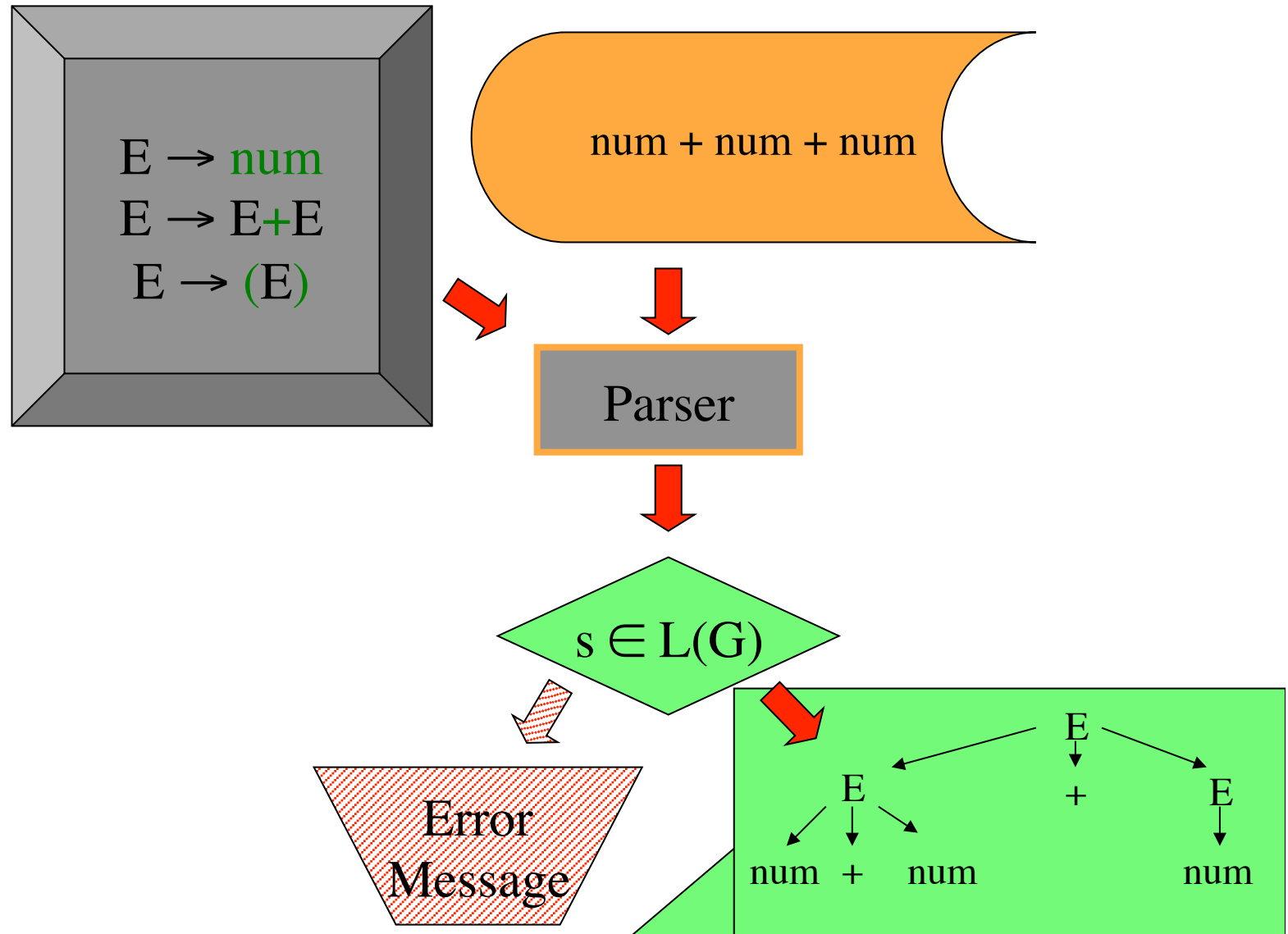
Compiler

Kapitel 4

Syntaktische Analyse

36

Was ist Parsing II





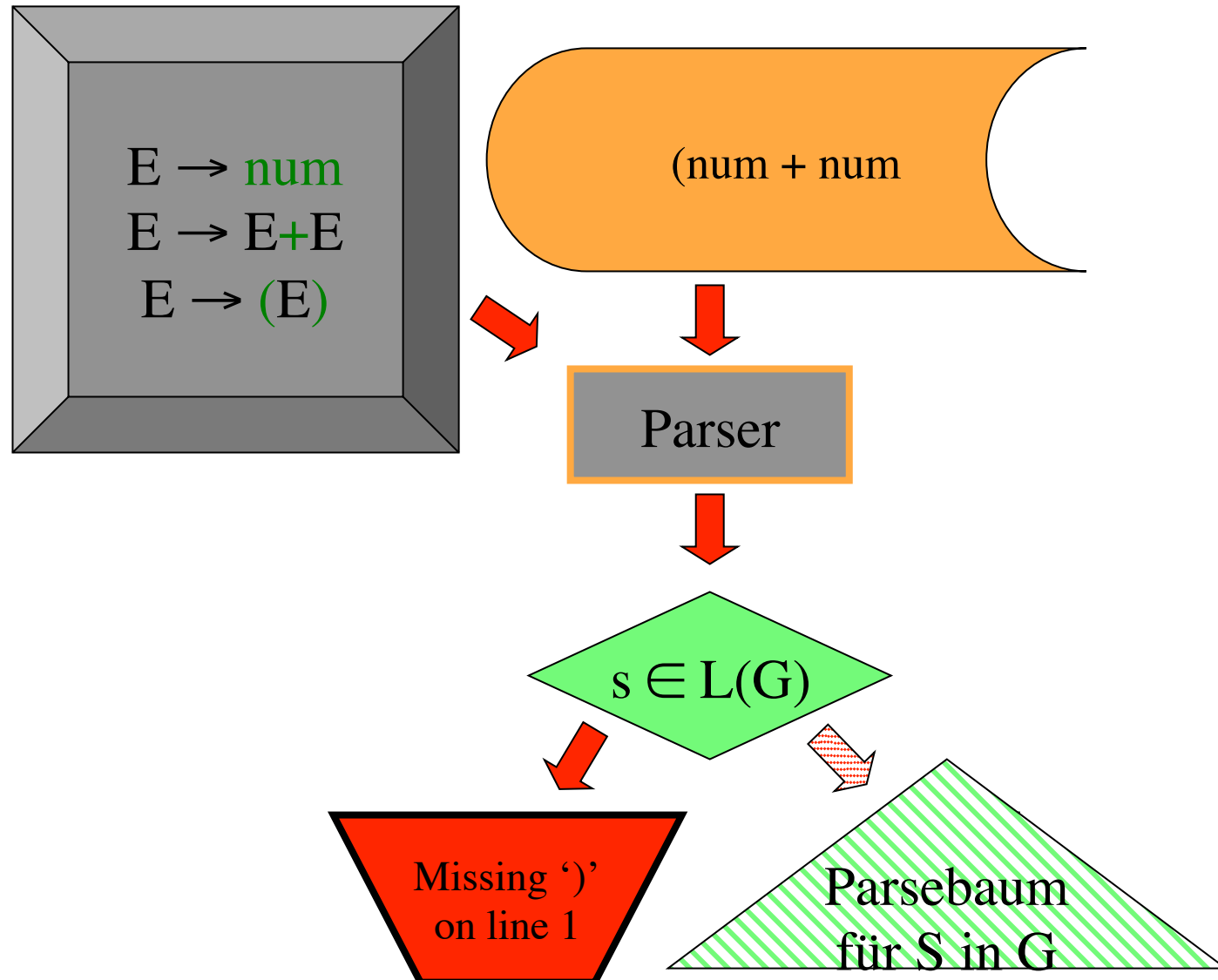
Compiler

Kapitel 4

Syntaktische Analyse

37

Was ist Parsing III





Compiler

Kapitel 4

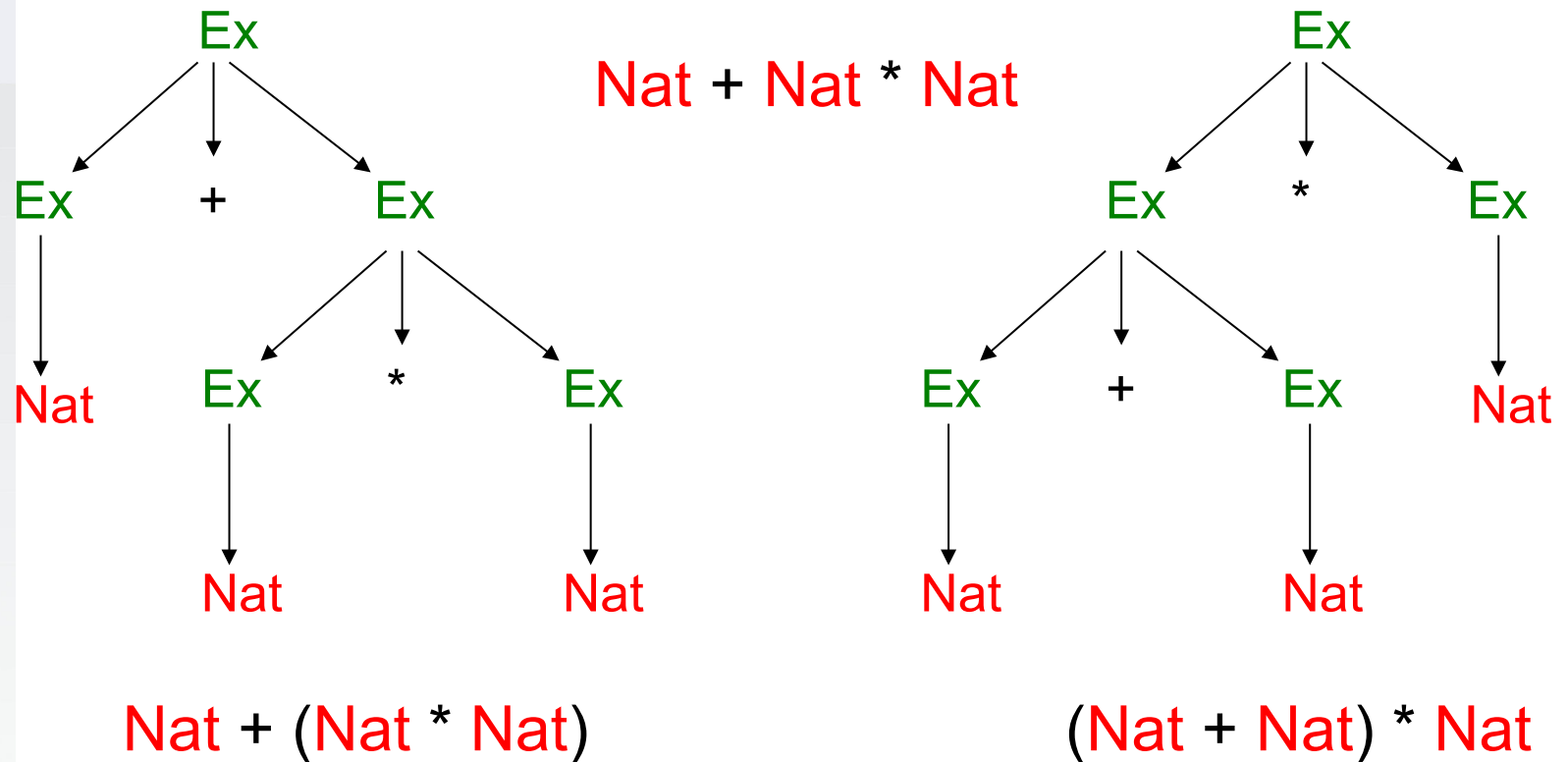
Syntaktische Analyse

38

$Ex \rightarrow Nat \mid (Ex) \mid Ex + Ex \mid Ex * Ex$

Mehrdeutige Grammatiken

mehrdeutig: falls es für einen String ≥ 2
verschiedene Parsebäume gibt:





Compiler

Kapitel 4

Syntaktische Analyse

Folie: 39

Eliminieren von Mehrdeutigkeiten

- $S \rightarrow ab \mid aT$
- $T \rightarrow b \mid bTc$



Compiler

Kapitel 4

Syntaktische Analyse

Folie: 40

Eliminieren von Mehrdeutigkeiten

```
stmt  →  if expr then stmt  
        |  if expr then stmt else stmt  
        |  other
```

if E_1 then if E_2 then S_1 else S_2



Compiler

Kapitel 4

Syntaktische Analyse

Folie: 41

PEARSON
Studium **it**
informatik

Autor:
Aho et al.

© Pearson Studium 2008

Eliminieren von Mehrdeutigkeiten II

$stmt \rightarrow$ if $expr$ then $stmt$
| if $expr$ then $stmt$ else $stmt$
| other

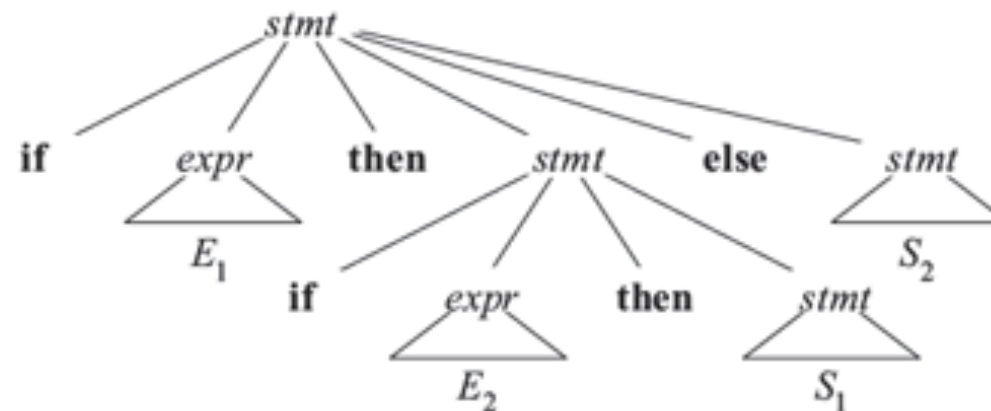
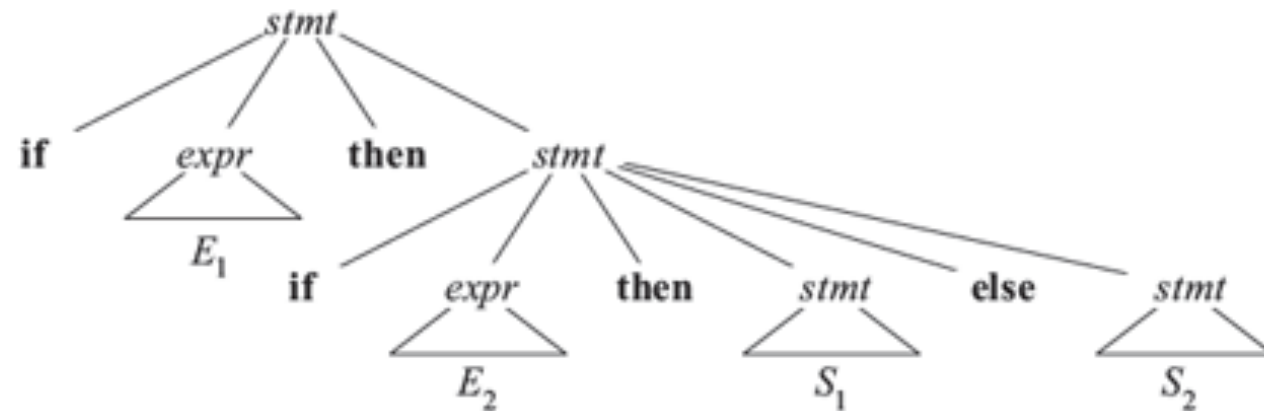


Abbildung 4.9: Zwei Parse-Bäume für einen mehrdeutigen Satz



Compiler

Kapitel 4

Syntaktische Analyse

Folie: 42

Eliminieren von Mehrdeutigkeiten

- Sprache verbessern:
 - $S \rightarrow \text{if } E \text{ then } S \text{ endif}$
 - $S \rightarrow \text{if } E \text{ then } S \text{ else } S \text{ endif}$

$\text{if } E_1 \text{ then if } E_2 \text{ then } S_1 \text{ else } S_2$

$\text{if } E_1 \text{ then if } E_2 \text{ then } S_1 \text{ endif else } S_2 \text{ endif}$



Compiler

Kapitel 4

Syntaktische Analyse

Folie: 43

Eliminieren von Mehrdeutigkeiten III

```
stmt → if expr then stmt  
      | if expr then stmt else stmt  
      | other
```

```
stmt → matched_stmt  
      | open_stmt  
matched_stmt → if expr then matched_stmt else matched_stmt  
              | other  
open_stmt → if expr then stmt  
            | if expr then matched_stmt else open_stmt
```

if E_1 then if E_2 then S_1 else S_2



Compiler

Kapitel 4

Syntaktische Analyse

44



Autor:
Aho et al.

© Pearson Studium 2008

Zusammenfassung

Parsing: Warum, Wann, Was Kontext-freie Grammatiken

- Bestandteile, Ableitungen,, Parse Bäume, Mehrdeutigkeit
- Mächtigkeit (im Vergleich zu regulären Ausdrücken)

Jetzt:

- **Wie** macht man Parsing