

# Theoretische Informatik I

## Einheit 3

### Kontextfreie Sprachen



1. Kontextfreie Grammatiken
2. Pushdown Automaten
3. Eigenschaften kontextfreier Sprachen

- **Was ist das einfachste Beschreibungsmodell?**
  - Analyse und Compilation muss formal beschreibbar sein
  - Generisches Modell für “alle” Programmiersprachen erforderlich
- **Typ-3 Sprachen sind einfach und effizient**
  - Beschreibung durch Grammatiken oder reguläre Ausdrücke
  - Beschreibung umwandelbar in endlichen Automaten
  - Erkennung von Wörtern der Sprache in “Echtzeit”
- **Aber Programmiersprachen sind nicht regulär**
  - Die meisten Programmstrukturen enthalten Schachtelungen wie Blöcke, if-then-else, arithmetische Ausdrücke, Klammersausdrücke, ...
  - Korrekte Klammersausdrücke und Schachtelungen sind nicht regulär



**Syntaxanalyse und Compilation von Programmiersprachen braucht mehr als reguläre Sprachen**

## ● Programmiersprachen

- **Compiler** kann kontextfreie Grammatiken effizient verarbeiten
- **Parser** kann aus kontextfreier Grammatik **automatisch erzeugt** werden
  - Standard Unix tool **YACC** unterstützt schnellen Compilerentwurf

## ● Markup Sprachen

- **HTML**: Formatierung von Dokumenten mit Links zu Programmaufrufen
- **XML**: Einheitliche Beschreibung der Semantik von Dokumenten

Beide Sprachen erfordern die Mächtigkeit kontextfreier Grammatiken

Mehr in HMU §5.3

## ● Wichtige Fragen

- **Analyse**: Systematische Rekonstruktion des Syntaxbaums
  - Ist das immer **eindeutig** möglich?
- **Compilation**: Zuweisung von Objectcode an Wörter der Sprache
- **Maschinenmodell**: effektive generische Erkennungsverfahren

# Theoretische Informatik I

## Einheit 3.1

### Kontextfreie Grammatiken



1. Grammatiken und Ableitungen
2. Ableitungsbäume
3. Mehrdeutigkeiten

# RÜCKBLICK: KONTEXTFREIE GRAMMATIKEN

- Eine **kontextfreie Grammatik (kfG)** ist ein **4-Tupel  $G = (V, T, P, S)$**  mit
  - $T$  endliches **Terminalalphabet**
  - $V$  endliches **Hilfsalphabet** mit  $V \cap T = \emptyset$
  - $P \subseteq V \times \Gamma^*$  endliche Menge der **Produktionen** (wobei  $\Gamma = V \cup T$ )
  - $S \in V$  **Startsymbol**

Die übliche Schreibweise für Produktionen  $(A, r) \in P$  ist  **$A \rightarrow r$**

Eine kompakte Notation für  $A \rightarrow r_1, A \rightarrow r_2 \dots A \rightarrow r_n$  ist  **$A \rightarrow r_1 | r_2 | \dots | r_n$**

- **Ableitbarkeit in einer kontextfreien Grammatik**

–  **$w \rightarrow z$**   $\equiv \exists x, y \in \Gamma^*. \exists A \rightarrow r \in P. w = x A y \wedge z = x r y$

–  **$w \xrightarrow{*} z$**   $\equiv \exists n \in \mathbb{N}. w \xrightarrow{n} z$

wobei  **$w \xrightarrow{0} z$**   $\equiv w = z$  und  **$w \xrightarrow{n+1} z$**   $\equiv \exists u \in \Gamma^*. w \rightarrow u \wedge u \xrightarrow{n} z$

- **Von  $G$  erzeugte Sprache:**

$$L(G) \equiv \{w \in T^* \mid S \xrightarrow{*} w\}$$

# GRAMMATIK FÜR GESCHACHTELTE KLAMMERAUSDRÜCKE

$$G_5 = ( \{S\}, \{ (, ) \}, \{ S \rightarrow (S), S \rightarrow \epsilon \}, S )$$

$$\text{Zeige: } L(G_5) = \{ ({}^k)^k \mid k \in \mathbb{N} \}$$

# GRAMMATIK FÜR GESCHACHELTE KLAMMERAUSDRÜCKE

$$G_5 = ( \{S\}, \{ (, ) \}, \{ S \rightarrow (S), S \rightarrow \epsilon \}, S )$$

**Zeige:**  $L(G_5) = \{ ({}^k) {}^k \mid k \in \mathbb{N} \}$

• **Beweis durch Induktion über Länge der Ableitung**

–  $\forall k \in \mathbb{N}. \forall w \in \{ (, ) \}^*. S \xrightarrow{k+1} w \Leftrightarrow w = ({}^k) {}^k$

# GRAMMATIK FÜR GESCHACHELTE KLAMMERAUSDRÜCKE

$$G_5 = ( \{S\}, \{ (, ) \}, \{ S \rightarrow (S), S \rightarrow \epsilon \}, S )$$

$$\text{Zeige: } L(G_5) = \{ ({}^k) {}^k \mid k \in \mathbb{N} \}$$

- **Beweis durch Induktion über Länge der Ableitung**

- $\forall k \in \mathbb{N}. \forall w \in \{ (, ) \}^*. S \xrightarrow{k+1} w \Leftrightarrow w = ({}^k) {}^k$

- **Basisfall**

- $S \xrightarrow{1} w \Leftrightarrow (S \rightarrow w) \in P \Leftrightarrow w = \epsilon \Leftrightarrow w = ({}^0) {}^0$

✓



# GRAMMATIK FÜR GESCHACHELTE KLAMMERAUSDRÜCKE

$$G_5 = ( \{S\}, \{ (, ) \}, \{ S \rightarrow (S), S \rightarrow \epsilon \}, S )$$

**Zeige:**  $L(G_5) = \{ ({}^k) {}^k \mid k \in \mathbb{N} \}$

• **Beweis durch Induktion über Länge der Ableitung**

–  $\forall k \in \mathbb{N}. \forall w \in \{ (, ) \}^*. S \xrightarrow{k+1} w \Leftrightarrow w = ({}^k) {}^k$

• **Basisfall**

–  $S \xrightarrow{1} w \Leftrightarrow (S \rightarrow w) \in P \Leftrightarrow w = \epsilon \Leftrightarrow w = ({}^0) {}^0$  ✓

• **Induktionsschritt**

– Es gelte  $\forall v \in \{ (, ) \}^*. S \xrightarrow{k+1} v \Leftrightarrow v = ({}^k) {}^k$

–  $S \xrightarrow{k+2} w \Leftrightarrow S \rightarrow (S) \xrightarrow{k+1} w$

$\Leftrightarrow \exists v \in \{ (, ) \}^*. S \xrightarrow{k+1} v \wedge w = (v)$

$\Leftrightarrow \exists v \in \{ (, ) \}^*. v = ({}^k) {}^k \wedge w = (v)$  (Annahme)

$\Leftrightarrow w = ({}^{k+1}) {}^{k+1}$  ✓

$$\{ ({}^k) {}^k \mid k \in \mathbb{N} \} \in \mathcal{L}_2 - \mathcal{L}_3$$

# KONTEXTFREIE GRAMMATIK FÜR PALINDROME

Wähle  $G_6 = (\{S\}, \{0, 1\}, P, S)$

mit  $P = \{S \rightarrow \epsilon, S \rightarrow 0, S \rightarrow 1, S \rightarrow 0S0, S \rightarrow 1S1\}$

Zeige:  $L(G_6) = \{w \in \{0, 1\}^* \mid w = w^R\}$

# KONTEXTFREIE GRAMMATIK FÜR PALINDROME

Wähle  $G_6 = (\{S\}, \{0, 1\}, P, S)$

mit  $P = \{S \rightarrow \epsilon, S \rightarrow 0, S \rightarrow 1, S \rightarrow 0S0, S \rightarrow 1S1\}$

Zeige:  $L(G_6) = \{w \in \{0, 1\}^* \mid w = w^R\}$

• **Beweis durch Induktion über Länge der Ableitung**

–  $\forall k \in \mathbb{N}. \forall w \in \{0, 1\}^*. S \xrightarrow{k+1} w \Leftrightarrow w = w^R \wedge |w| \in \{2k, 2k+1\}$

# KONTEXTFREIE GRAMMATIK FÜR PALINDROME

Wähle  $G_6 = (\{S\}, \{0, 1\}, P, S)$

mit  $P = \{S \rightarrow \epsilon, S \rightarrow 0, S \rightarrow 1, S \rightarrow 0S0, S \rightarrow 1S1\}$

Zeige:  $L(G_6) = \{w \in \{0, 1\}^* \mid w = w^R\}$

- **Beweis durch Induktion über Länge der Ableitung**

- $\forall k \in \mathbb{N}. \forall w \in \{0, 1\}^*. S \xrightarrow{k+1} w \Leftrightarrow w = w^R \wedge |w| \in \{2k, 2k+1\}$

- **Basisfall**

- $S \xrightarrow{1} w \Leftrightarrow (S \rightarrow w) \in P \Leftrightarrow w \in \{0, 1, \epsilon\} \Leftrightarrow w = w^R \wedge |w| \in \{0, 1\}$  ✓

# KONTEXTFREIE GRAMMATIK FÜR PALINDROME

Wähle  $G_6 = (\{S\}, \{0, 1\}, P, S)$

mit  $P = \{S \rightarrow \epsilon, S \rightarrow 0, S \rightarrow 1, S \rightarrow 0S0, S \rightarrow 1S1\}$

Zeige:  $L(G_6) = \{w \in \{0, 1\}^* \mid w = w^R\}$

- **Beweise durch Induktion über Länge der Ableitung**

- $\forall k \in \mathbb{N}. \forall w \in \{0, 1\}^*. S \xrightarrow{k+1} w \Leftrightarrow w = w^R \wedge |w| \in \{2k, 2k+1\}$

- **Basisfall**

- $S \xrightarrow{1} w \Leftrightarrow (S \rightarrow w) \in P \Leftrightarrow w \in \{0, 1, \epsilon\} \Leftrightarrow w = w^R \wedge |w| \in \{0, 1\}$  ✓

- **Induktionsschritt**

- Es gelte  $\forall v \in \{0, 1\}^*. S \xrightarrow{k+1} v \Leftrightarrow v = v^R \wedge |v| \in \{2k, 2k+1\}$

- $S \xrightarrow{k+2} w \Leftrightarrow S \rightarrow 0S0 \xrightarrow{k+1} w \vee S \rightarrow 1S1 \xrightarrow{k+1} w$   
 $\Leftrightarrow \exists v \in \{0, 1\}^*. S \xrightarrow{k+1} v \wedge (w = 0v0 \vee w = 1v1)$   
 $\Leftrightarrow \exists v \in \{0, 1\}^*. v = v^R \wedge |v| \in \{2k, 2k+1\} \wedge (w = 0v0 \vee w = 1v1)$   
 $\Leftrightarrow w = w^R \wedge |w| \in \{2k+2, 2k+3\}$  ✓

# GRAMMATIK FÜR ARITHMETISCHE AUSDRÜCKE

- **Ausdrücke über Operatoren  $+$  und  $*$**

- **Bezeichner** (*I*dentifier):

- Buchstabe gefolgt von Buchstaben/Ziffern
- Buchstaben *a*, *b*, *c*, Ziffern *0*, *1*

- **Ausdrücke** (*E*xpressions):

- Schachtelung mit  $+$ ,  $*$  und Klammern

- $G_7 = (\{E, I\}, \{a, b, c, 0, 1, +, *, (, )\}, P, E)$

mit  $P = \{ E \rightarrow I \mid E + E \mid E * E \mid (E)$

$I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1 \}$

**Kann man die Struktur eines arithmetischen Ausdrucks mit  $G_7$  immer rekonstruieren?**

## Rekonstruierbare Auswahl von Produktionen

- **Beliebige Ableitung**

$$\begin{aligned} E &\longrightarrow E * E \longrightarrow I * E \longrightarrow I * (E) \\ &\longrightarrow I * (E + E) \longrightarrow I * (I + E) \longrightarrow I * (I + I) \longrightarrow I * (a + I) \\ &\longrightarrow I * (a + I0) \longrightarrow I * (a + I00) \longrightarrow I * (a + b00) \longrightarrow a * (a + b00) \end{aligned}$$

## Rekonstruierbare Auswahl von Produktionen

- **Beliebige Ableitung**

$$\begin{aligned} E &\longrightarrow E * E \longrightarrow I * E \longrightarrow I * (E) \\ &\longrightarrow I * (E + E) \longrightarrow I * (I + E) \longrightarrow I * (I + I) \longrightarrow I * (a + I) \\ &\longrightarrow I * (a + I0) \longrightarrow I * (a + I00) \longrightarrow I * (a + b00) \longrightarrow a * (a + b00) \end{aligned}$$

- **Linksseitige Ableitung**  $w \longrightarrow_L z$

– In  $w$  wird die am weitesten links stehende Variable ersetzt

$$E \longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E$$



## Rekonstruierbare Auswahl von Produktionen

- **Beliebige Ableitung**

$$\begin{aligned} E &\longrightarrow E * E \longrightarrow I * E \longrightarrow I * (E) \\ &\longrightarrow I * (E + E) \longrightarrow I * (I + E) \longrightarrow I * (I + I) \longrightarrow I * (a + I) \\ &\longrightarrow I * (a + I0) \longrightarrow I * (a + I00) \longrightarrow I * (a + b00) \longrightarrow a * (a + b00) \end{aligned}$$

- **Linksseitige Ableitung**  $w \longrightarrow_L z$

– In  $w$  wird die am weitesten links stehende Variable ersetzt

$$\begin{aligned} E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\ &\longrightarrow_L a * (E + E) \longrightarrow_L a * (I + E) \longrightarrow_L a * (a + E) \longrightarrow_L a * (a + I) \\ &\longrightarrow_L a * (a + I0) \longrightarrow_L a * (a + I00) \longrightarrow_L a * (a + b00) \end{aligned}$$

## Rekonstruierbare Auswahl von Produktionen

- **Beliebige Ableitung**

$$\begin{aligned}
 E &\longrightarrow E * E \longrightarrow I * E \longrightarrow I * (E) \\
 &\longrightarrow I * (E + E) \longrightarrow I * (I + E) \longrightarrow I * (I + I) \longrightarrow I * (a + I) \\
 &\longrightarrow I * (a + I0) \longrightarrow I * (a + I00) \longrightarrow I * (a + b00) \longrightarrow a * (a + b00)
 \end{aligned}$$

- **Linksseitige Ableitung**  $w \longrightarrow_L z$

– In  $w$  wird die am weitesten links stehende Variable ersetzt

$$\begin{aligned}
 E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\
 &\longrightarrow_L a * (E + E) \longrightarrow_L a * (I + E) \longrightarrow_L a * (a + E) \longrightarrow_L a * (a + I) \\
 &\longrightarrow_L a * (a + I0) \longrightarrow_L a * (a + I00) \longrightarrow_L a * (a + b00)
 \end{aligned}$$

- **Rechtsseitige Ableitung**  $w \longrightarrow_R z$

– In  $w$  wird die am weitesten rechts stehende Variable ersetzt

$$\begin{aligned}
 E &\longrightarrow_R E * E \longrightarrow_R E * (E) \longrightarrow_R E * (E + E) \longrightarrow_R E * (E + I) \\
 &\longrightarrow_R E * (E + I0) \longrightarrow_R E * (E + I00) \longrightarrow_R E * (E + b00) \\
 &\longrightarrow_R E * (I + b00) \longrightarrow_R E * (a + b00) \longrightarrow_R I * (a + b00) \longrightarrow_R a * (a + b00)
 \end{aligned}$$

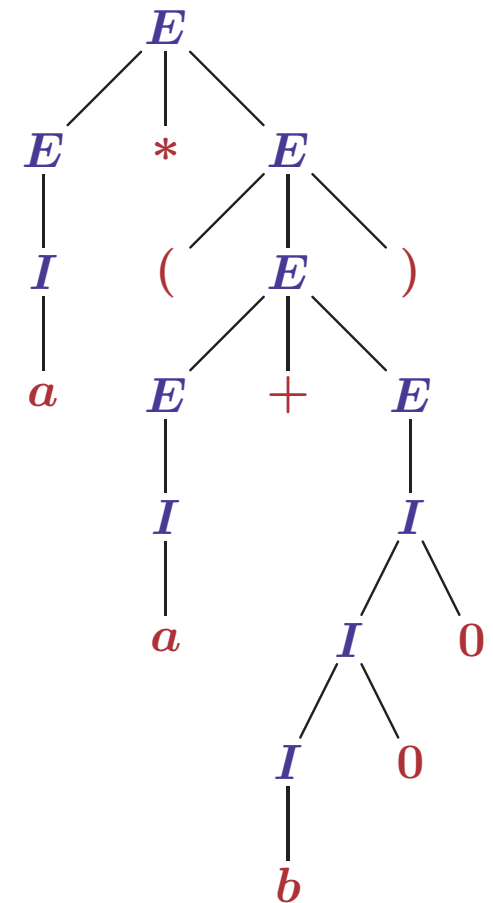
# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$



# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

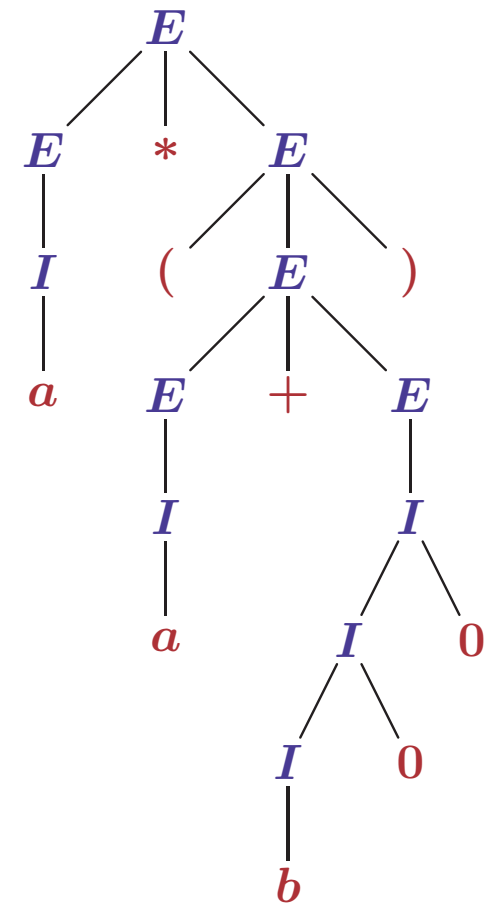
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts



# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

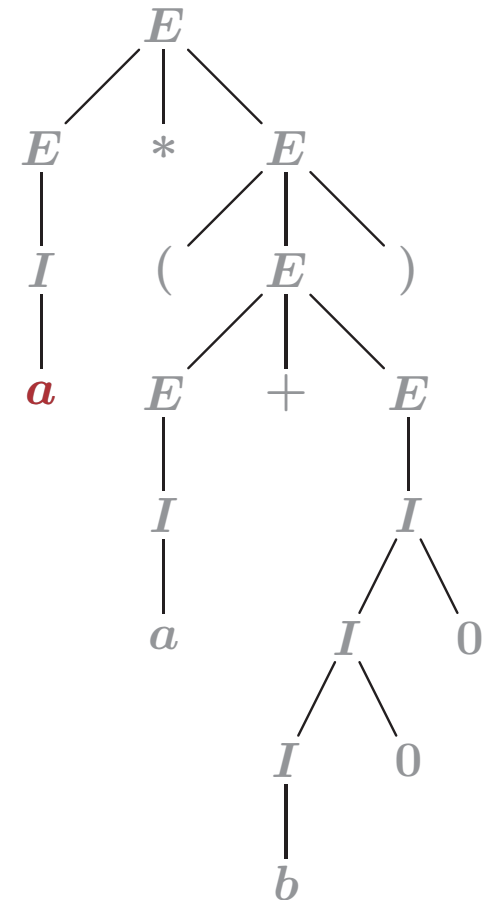
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts
- $a$



# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

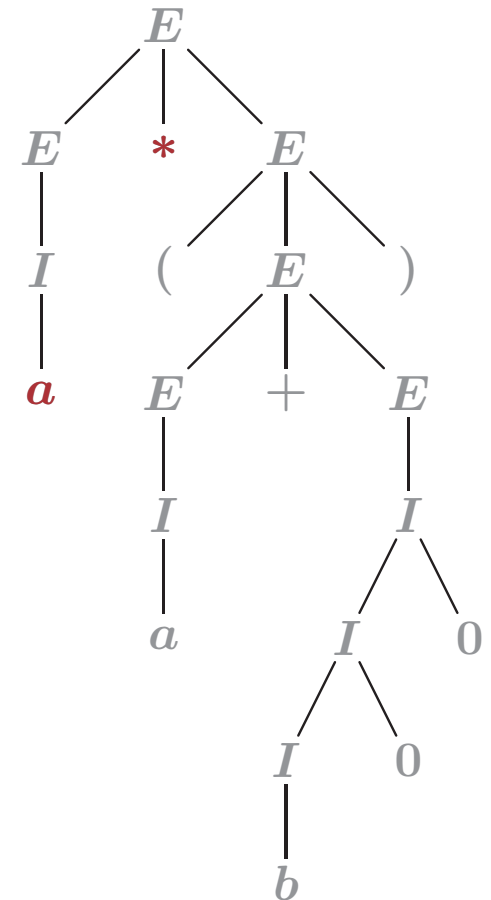
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts
- $a*$



# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

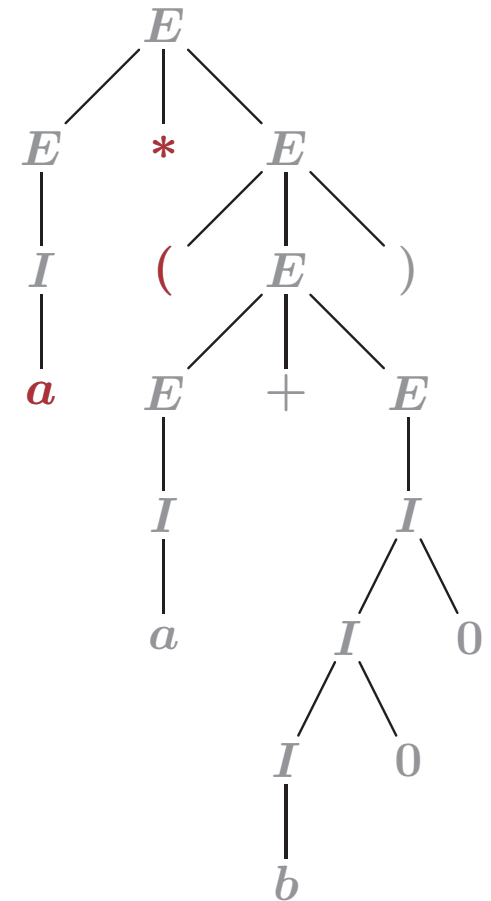
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts
- $a * ($



# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

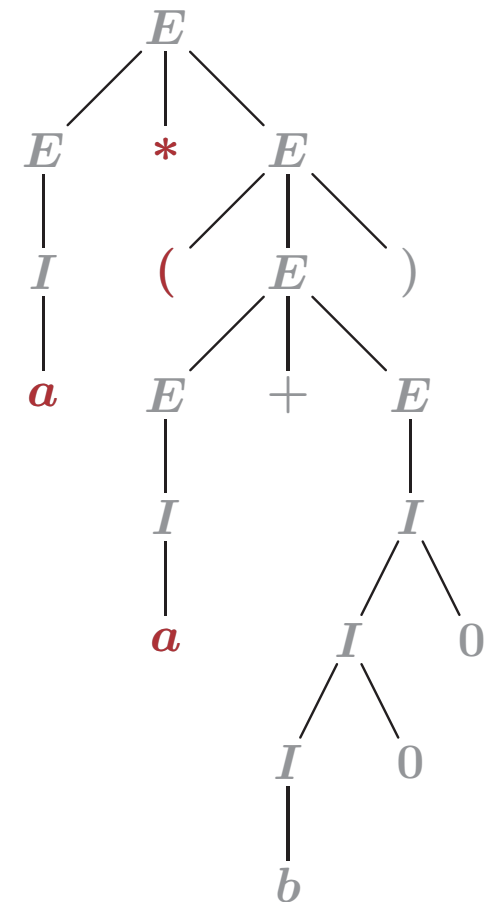
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts
- $a * (a$





# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

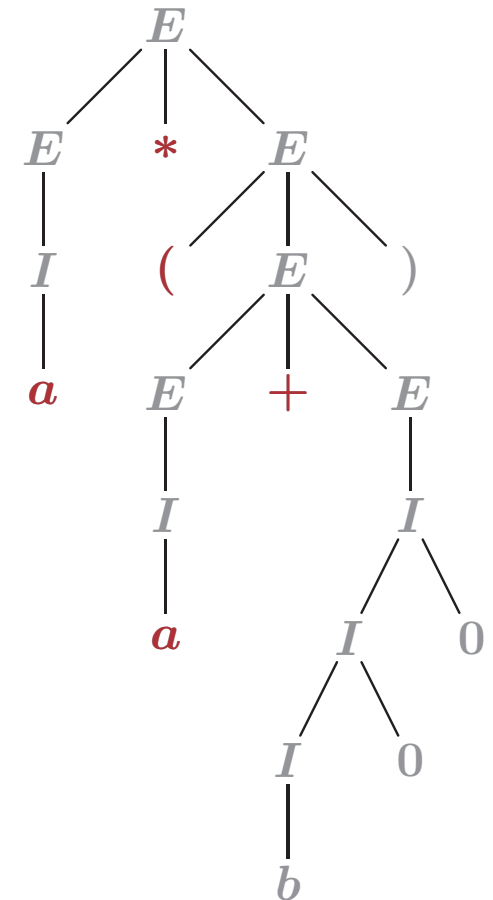
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts
- $a * (a +$



# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

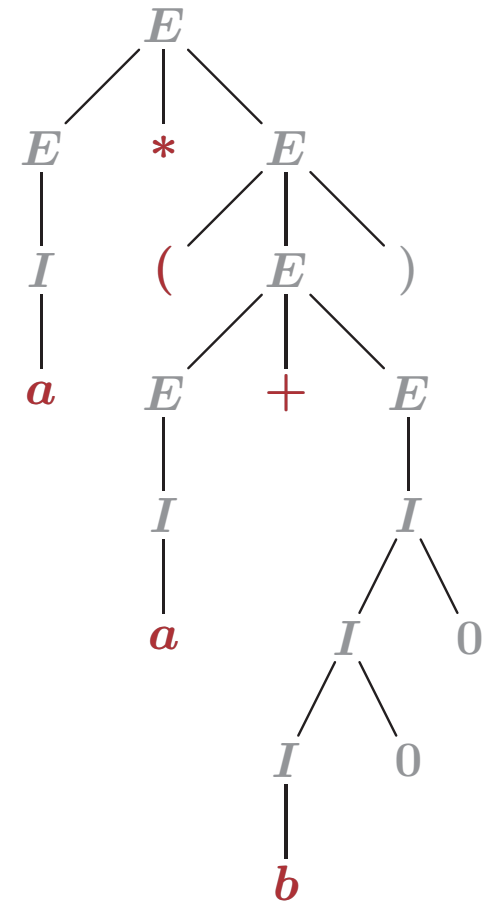
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts
- $a * (a + b$



# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

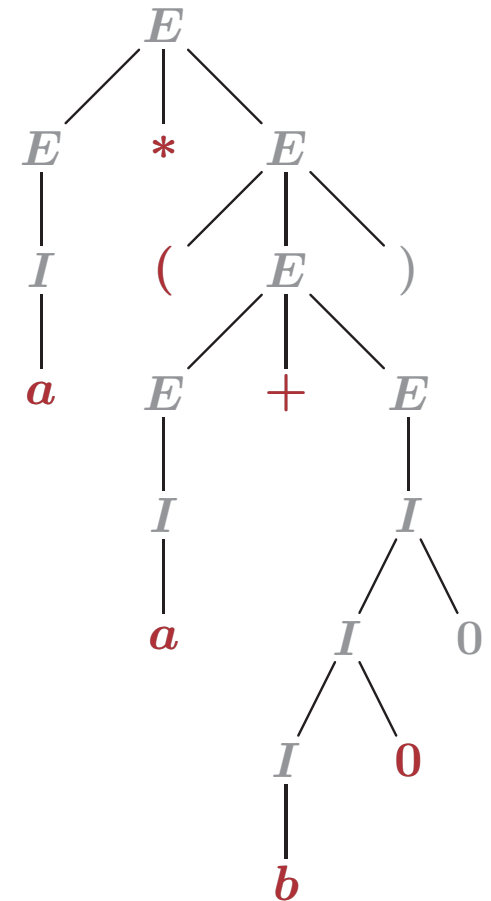
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts
- $a * (a + b0$



# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

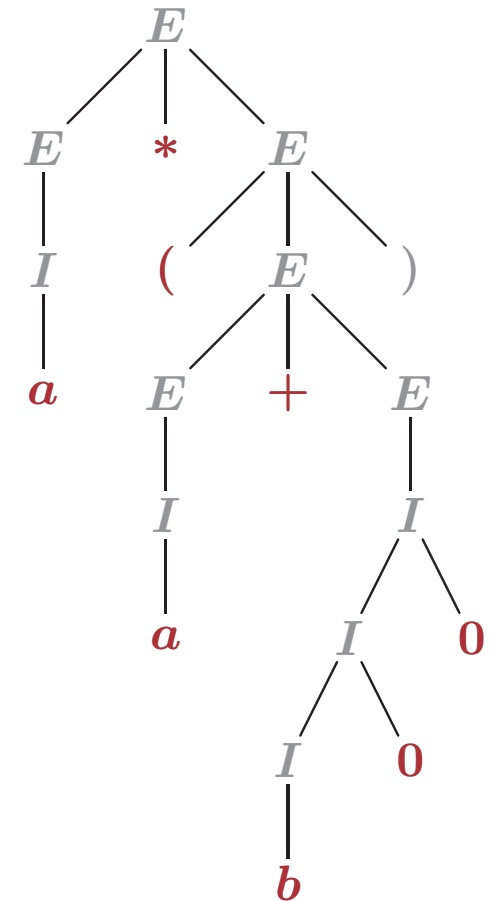
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts
- $a * (a + b00$



# ABLEITUNGSBÄUME (PARSEBÄUME)

## Baumdarstellung von Ableitungen

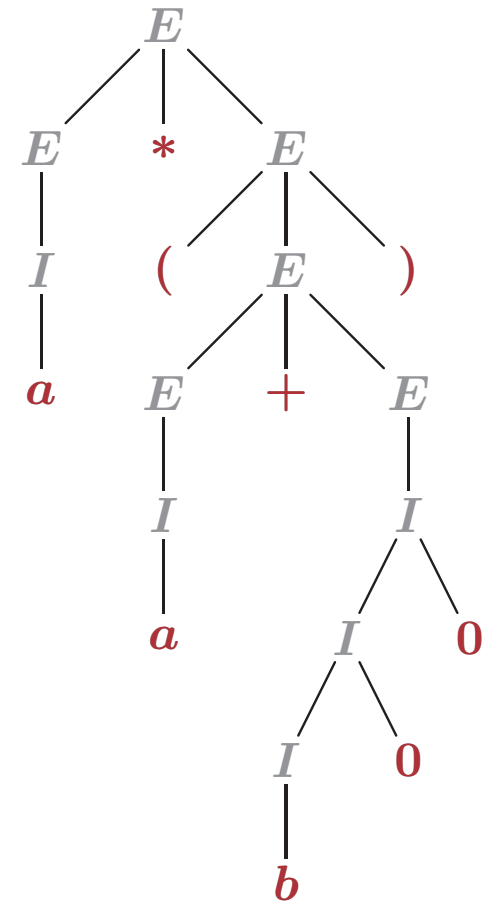
- **Ableitungsbäume sind markierte Bäume**

(Einheit 1, Anhang)

- Innere Knoten mit Variablen  $A \in V$  markiert
- Wurzel markiert mit Startsymbol
- Blätter markiert mit Terminalsymbolen  $a \in T$  oder mit  $\epsilon$
- Hat ein innerer Knoten Markierung  $A$  und Nachfolger mit Markierungen  $v_1 \dots v_n$ , so ist  $A \rightarrow v_1 \dots v_n \in P$

- **Blätter repräsentieren Terminalwörter**

- Auslesen durch Tiefensuche von links nach rechts
- $a * (a + b00)$



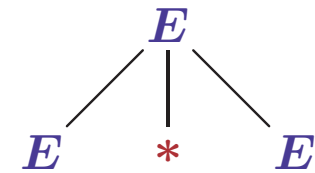
- **Baum repräsentiert Ableitungen**
  - Rekursive Erzeugung beginnend mit Wurzel
  - Vorrang für tiefe linke Knoten ergibt Linksableitung

# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$E \longrightarrow_L E * E$$

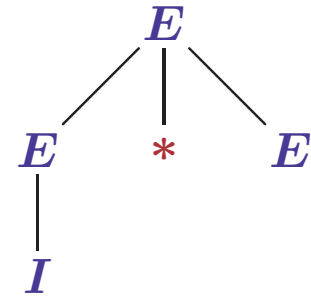


# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$E \xrightarrow{\gamma_L} E * E \xrightarrow{\gamma_L} I * E$$



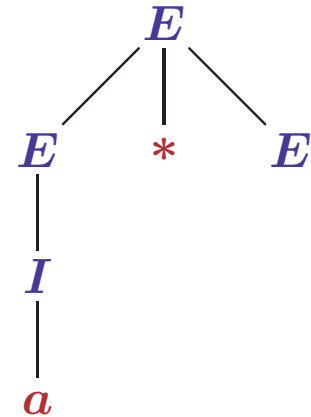


# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$E \xrightarrow{\gamma_L} E * E \xrightarrow{\gamma_L} I * E \xrightarrow{\gamma_L} a * E$$

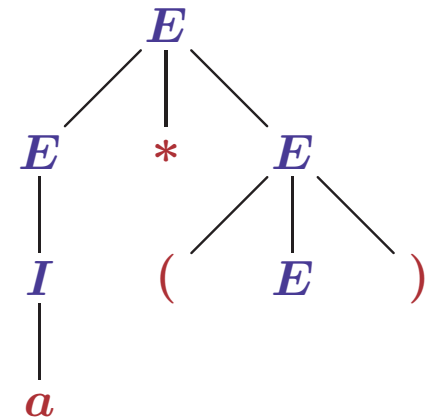


# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$E \xrightarrow{\gamma_L} E * E \xrightarrow{\gamma_L} I * E \xrightarrow{\gamma_L} a * E \xrightarrow{\gamma_L} a * (E)$$

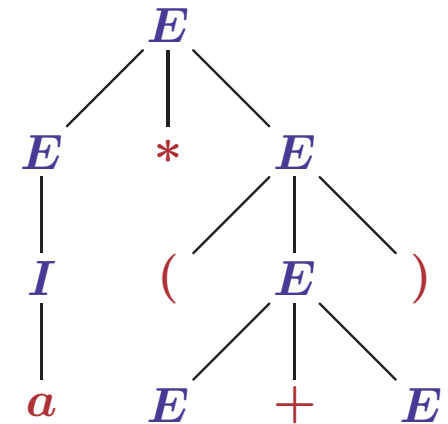


# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$\begin{aligned} E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\ &\longrightarrow_L a * (E + E) \end{aligned}$$

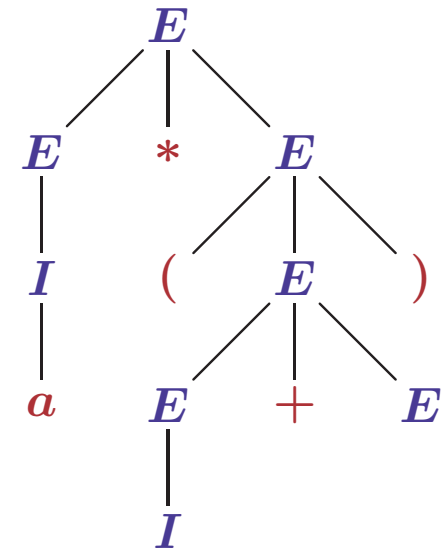


# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$\begin{aligned} E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\ &\longrightarrow_L a * (E + E) \longrightarrow_L a * (I + E) \end{aligned}$$

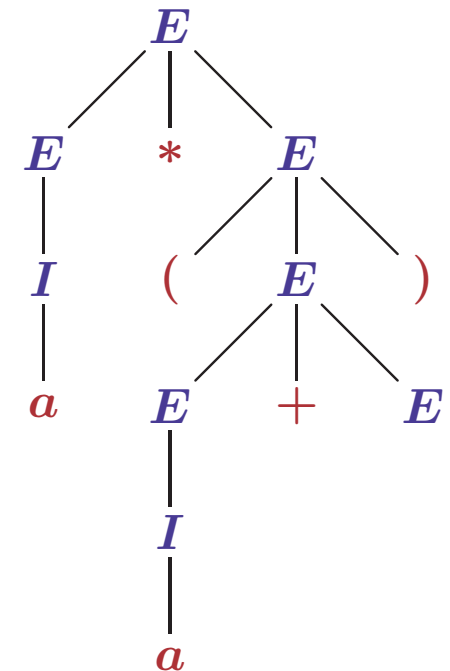


# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$\begin{aligned} E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\ &\longrightarrow_L a * (E + E) \longrightarrow_L a * (I + E) \longrightarrow_L a * (a + E) \end{aligned}$$

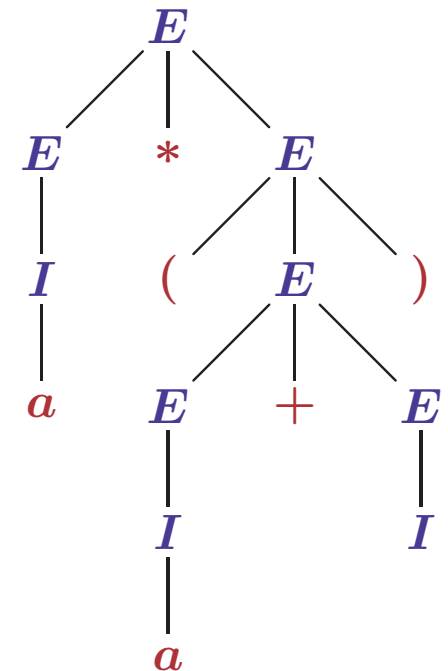


# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$\begin{aligned} E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\ &\longrightarrow_L a * (E + E) \longrightarrow_L a * (I + E) \longrightarrow_L a * (a + E) \\ &\longrightarrow_L a * (a + I) \end{aligned}$$

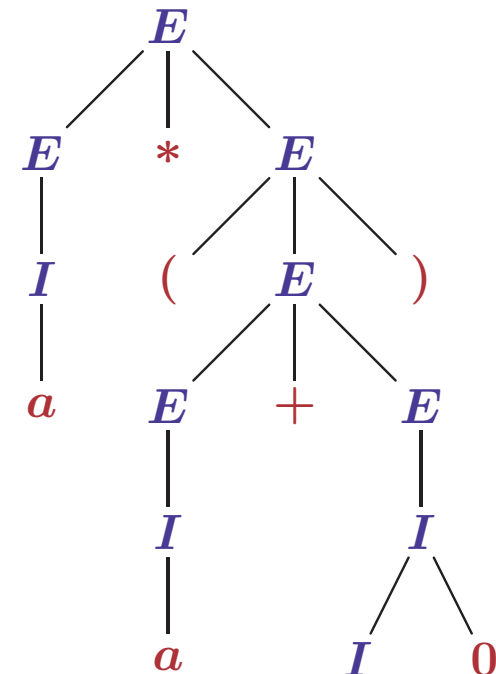


# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$\begin{aligned} E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\ &\longrightarrow_L a * (E + E) \longrightarrow_L a * (I + E) \longrightarrow_L a * (a + E) \\ &\longrightarrow_L a * (a + I) \longrightarrow_L a * (a + I0) \end{aligned}$$

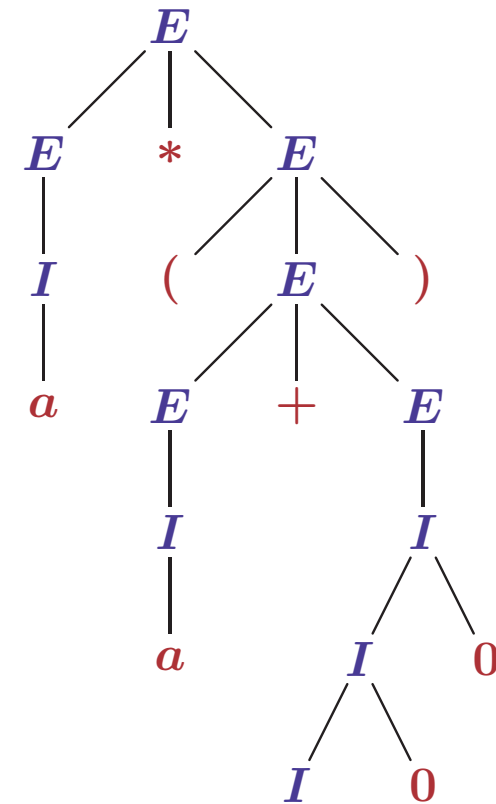


# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

- **Baum repräsentiert Ableitungen**

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$\begin{aligned}
 E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\
 &\longrightarrow_L a * (E + E) \longrightarrow_L a * (I + E) \longrightarrow_L a * (a + E) \\
 &\longrightarrow_L a * (a + I) \longrightarrow_L a * (a + I0) \longrightarrow_L a * (a + I00)
 \end{aligned}$$





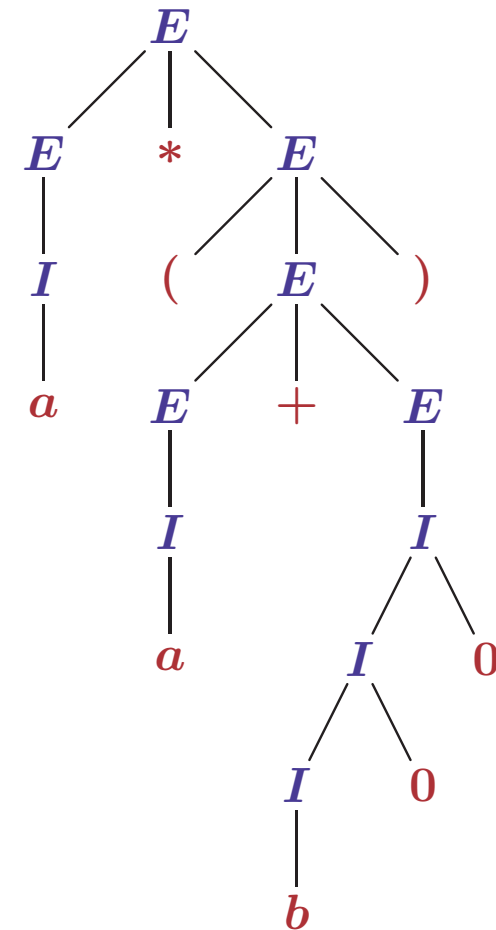
# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

## ● Baum repräsentiert Ableitungen

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$\begin{aligned}
 E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\
 &\longrightarrow_L a * (E + E) \longrightarrow_L a * (I + E) \longrightarrow_L a * (a + E) \\
 &\longrightarrow_L a * (a + I) \longrightarrow_L a * (a + I 0) \longrightarrow_L a * (a + I 0 0) \\
 &\longrightarrow_L a * (a + b 0 0)
 \end{aligned}$$

- Vorrang für tiefe rechte Knoten ergibt Rechtsableitung



# ABLEITUNGSBÄUME REPRÄSENTIEREN ABLEITUNGEN

## ● Baum repräsentiert Ableitungen

- Rekursive Erzeugung beginnend mit Wurzel
- Vorrang für tiefe linke Knoten ergibt Linksableitung

$$\begin{aligned}
 E &\longrightarrow_L E * E \longrightarrow_L I * E \longrightarrow_L a * E \longrightarrow_L a * (E) \\
 &\longrightarrow_L a * (E + E) \longrightarrow_L a * (I + E) \longrightarrow_L a * (a + E) \\
 &\longrightarrow_L a * (a + I) \longrightarrow_L a * (a + I 0) \longrightarrow_L a * (a + I 0 0) \\
 &\longrightarrow_L a * (a + b 0 0)
 \end{aligned}$$

- Vorrang für tiefe rechte Knoten ergibt Rechtsableitung

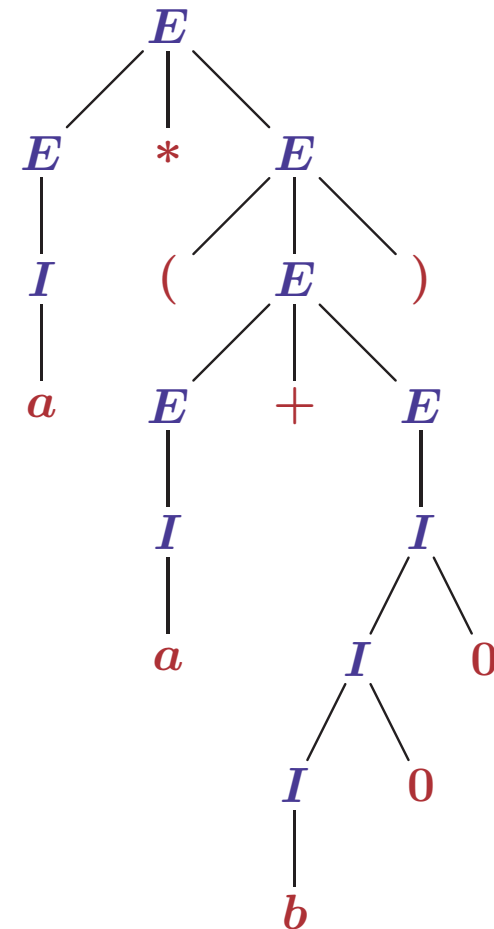
Es gilt:  $S \xrightarrow{*} w$

$\Leftrightarrow$  es gibt einen Ableitungsbaum mit Blattmarkierung  $w$

$\Rightarrow$  : Konstruiere Baum induktiv aus Linksableitung von  $w$

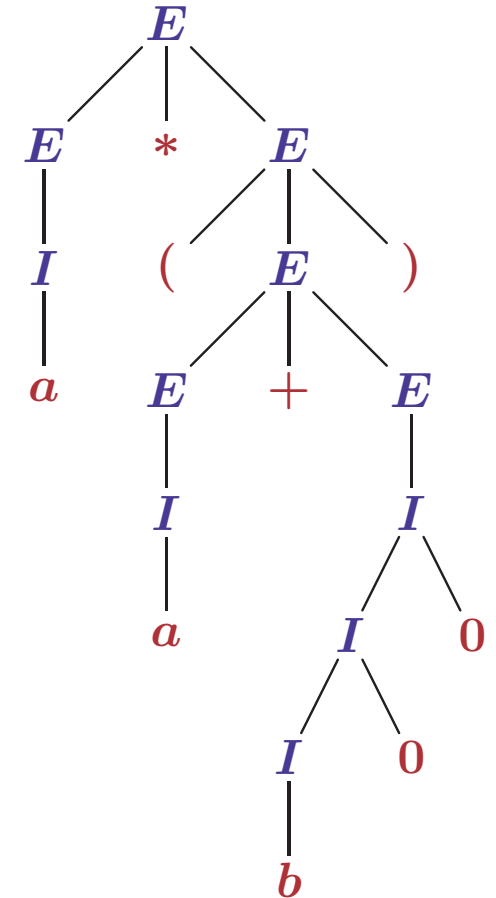
$\Leftarrow$  : Extrahiere Linksableitung von  $w$  induktiv aus Baum

Details in HMU §5.2



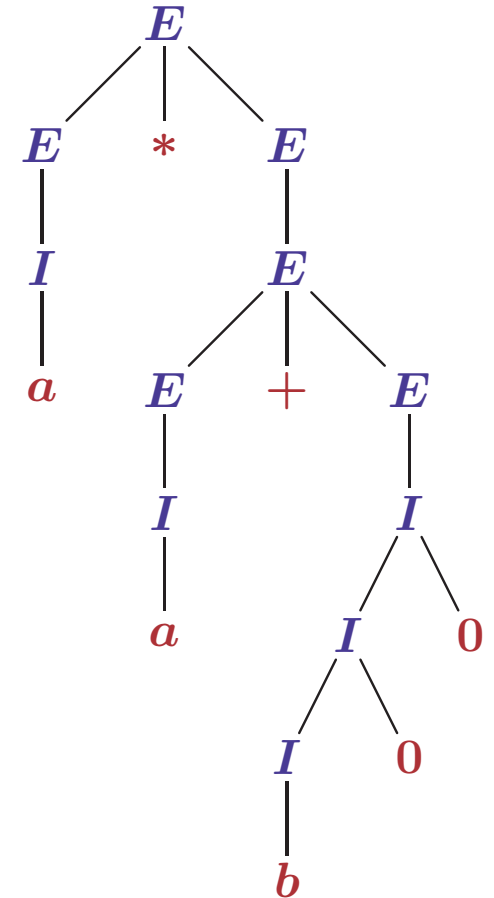
- **Parser erzeugen Syntaxbaum**

- Rekonstruktion des Ableitungsbaumes aus dem Wort
- Entferne Blätter mit Klammern



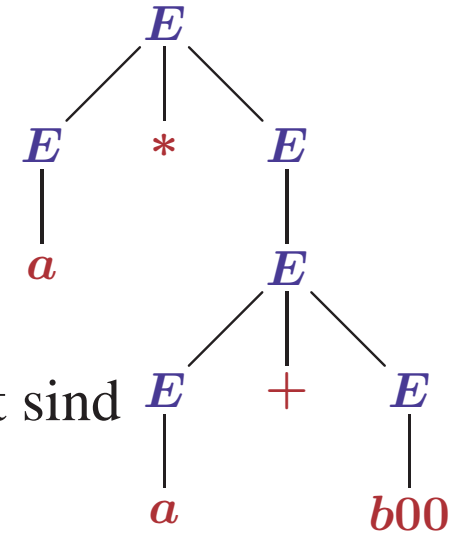
- **Parser erzeugen Syntaxbaum**

- Rekonstruktion des Ableitungsbaumes aus dem Wort
- Entferne Blätter mit Klammern
- Gruppieren Identifier zu lexikalischen Einheiten



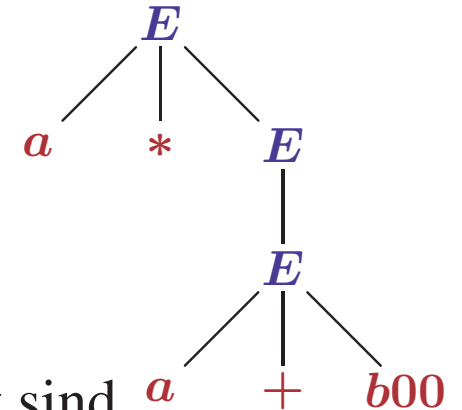
- **Parser erzeugen Syntaxbaum**

- Rekonstruktion des Ableitungsbaumes aus dem Wort
- Entferne Blätter mit Klammern
- Gruppierere Identifier zu lexikalischen Einheiten
- Entferne Vorgänger von Identifiern, die mit  $E$  markiert sind



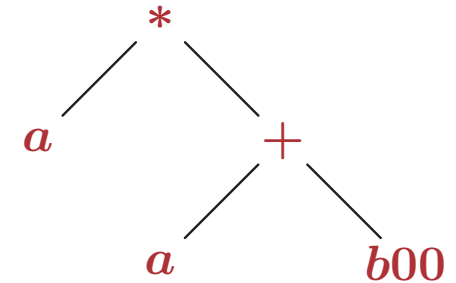
- **Parser erzeugen Syntaxbaum**

- Rekonstruktion des Ableitungsbaumes aus dem Wort
- Entferne Blätter mit Klammern
- Gruppierere Identifier zu lexikalischen Einheiten
- Entferne Vorgänger von Identifiern, die mit  $E$  markiert sind
- Entferne  $E$ -Vorgänger von  $+/*$



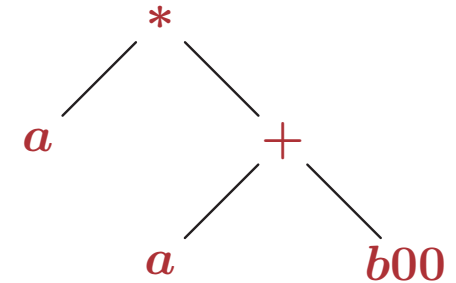
- **Parser erzeugen Syntaxbaum**

- Rekonstruktion des Ableitungsbaumes aus dem Wort
- Entferne Blätter mit Klammern
- Gruppierere Identifier zu lexikalischen Einheiten
- Entferne Vorgänger von Identifiern, die mit  $E$  markiert sind
- Entferne  $E$ -Vorgänger von  $+/*$



## ● Parser erzeugen **Syntaxbaum**

- Rekonstruktion des Ableitungsbaumes aus dem Wort
- Entferne Blätter mit Klammern
- Gruppiere Identifier zu lexikalischen Einheiten
- Entferne Vorgänger von Identifiern, die mit  $E$  markiert sind
- Entferne  $E$ -Vorgänger von  $+/*$



## ● Compiler erzeugt **Maschinencode**

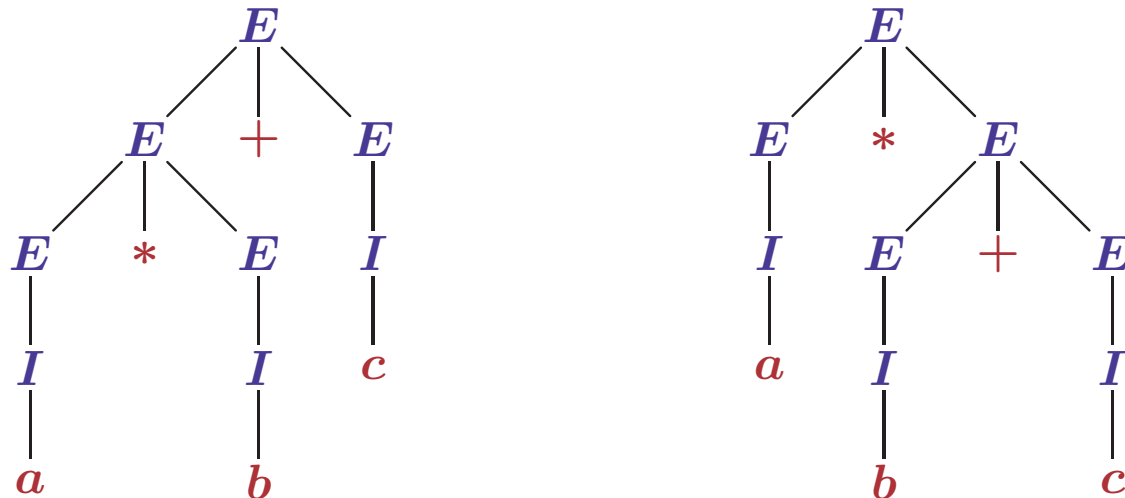
- Übersetze Identifier in Registernamen
- $+(x, y)$ :
  - Bestimme Wert von  $y$  und lege ihn im Register ab
  - Bestimme Wert von  $x$
  - Addiere Registerwert
- Coderzeugung für  $*(x, y)$  analog

Details in Vossen/Witt §7.2

**Nur möglich, wenn Ableitungsbaum eindeutig ist**



# WANN IST DER ABLEITUNGSBAUM EINDEUTIG?



- Das Wort  $a * b + c$  hat zwei Ableitungen in  $G_7$

$$\begin{aligned} E &\longrightarrow E + E \longrightarrow E * E + E \longrightarrow I * E + E \longrightarrow a * E + E \longrightarrow a * I + E \\ &\longrightarrow a * b + E \longrightarrow a * b + I \longrightarrow a * b + c \end{aligned}$$

$$\begin{aligned} E &\longrightarrow E * E \longrightarrow I * E \longrightarrow a * E \longrightarrow a * E + E \longrightarrow a * I + E \\ &\longrightarrow a * b + E \longrightarrow a * b + I \longrightarrow a * b + c \end{aligned}$$

Beide Ableitungen sind Linksableitungen aber verschieden

- Die Grammatik  $G_7$  ist **mehrdeutig**
  - Wörter der Sprache können nicht eindeutig analysiert werden

# MEHRDEUTIGKEIT

- **Eindeutige Grammatik**  $G = (V, T, P, S)$ 
  - Jedes Wort  $w \in L(G)$  hat genau einen Ableitungsbaum
  - Andernfalls ist  $G$  **mehrdeutig**  
(ein  $w \in L(G)$  hat mindestens zwei verschiedene Ableitungsbäume)
  - $G_7$  ist mehrdeutig
- **Eindeutige Sprache**  $L$ 
  - Es gibt eine eindeutige Grammatik  $G$  mit  $L = L(G)$
  - Die Sprache von  $G_7$  ist eindeutig ↪ nächste Folie
- **Inhärent mehrdeutige Sprache**  $L$ 
  - Eine eindeutige Grammatik für  $L$  kann nicht angegeben werden
  - $\{0^i 1^j 2^k \mid i=j \vee j=k\}$  ist inhärent mehrdeutig ↪ Beweisskizze folgt

**Programmiersprachen müssen eindeutig sein**

# AUFLÖSUNG VON MEHRDEUTIGKEITEN

- Was fehlt bei  $G_7 = (\{E, I\}, \{a, b, c, 0, 1, +, *, (, )\}, P, E)$   
mit  $P = \{E \rightarrow I \mid E + E \mid E * E \mid (E),$   
 $I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1\}$  ?

# AUFLÖSUNG VON MEHRDEUTIGKEITEN

- Was fehlt bei  $G_7 = (\{E, I\}, \{a, b, c, 0, 1, +, *, (, )\}, P, E)$   
mit  $P = \{E \rightarrow I \mid E + E \mid E * E \mid (E),$   
 $I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1\}$  ?

$G_7$  beinhaltet nicht die üblichen Konventionen für  $*$  und  $+$

$*$  bindet stärker als  $+$

$*$  und  $+$  werden als linkssassoziativ angesehen

Alle anderen Lesarten benötigen Klammern

# AUFLÖSUNG VON MEHRDEUTIGKEITEN

- Was fehlt bei  $G_7 = (\{E, I\}, \{a, b, c, 0, 1, +, *, (, )\}, P, E)$  mit  $P = \{E \rightarrow I \mid E+E \mid E * E \mid (E), I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1\}$  ?

$G_7$  beinhaltet nicht die üblichen Konventionen für  $*$  und  $+$

$*$  bindet stärker als  $+$

$*$  und  $+$  werden als linkssassoziativ angesehen

Alle anderen Lesarten benötigen Klammern

- **Prioritätsregeln können Eindeutigkeit erzeugen**

– Niedrigste Priorität  $+$  steht linkssassoziativ und außen  $\mapsto$  **T**erme

– Höhere Priorität  $*$  steht linkssassoziativ und innen  $\mapsto$  **F**aktoren

– Faktoren können Bezeichner oder Ausdrücke in Klammern sein

Setze  $G'_7 = (\{E, T, F, I\}, \{a, b, c, 0, 1, +, *, (, )\}, P', E)$

mit  $P' = \{E \rightarrow T \mid E+T, T \rightarrow F \mid T * F, F \rightarrow I \mid (E)$

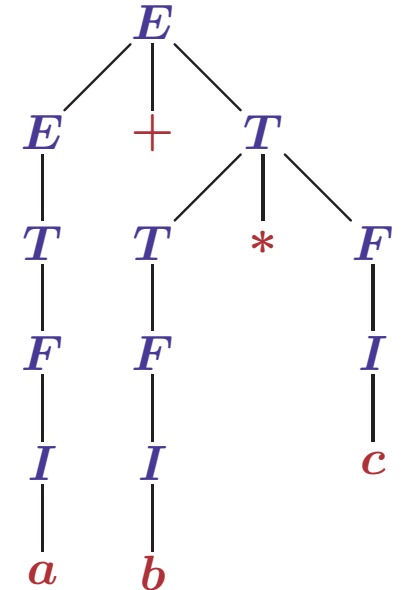
$I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1\}$

$G'_7$  ist äquivalent zu  $G_7$  und eindeutig

# BEGRÜNDUNG DER EINDEUTIGKEIT VON $G'_7$

$$P' = \{ E \rightarrow T \mid E+T, T \rightarrow F \mid T*F, F \rightarrow I \mid (E) \\ I \rightarrow a \mid b \mid c \mid Ia \mid Ib \mid Ic \mid I0 \mid I1 \}$$

- **Jeder Ausdruck muss aus einer Termfolge bestehen**
  - Termfolge muss von rechts nach links erzeugt werden
  - Terme haben keine Ausdrücke als direkte Teile
  - Es gibt nur einen Parsebaum für  $t_1 + t_2 + \dots + t_k$
- **Jeder Term muss aus einer Faktorenfolge bestehen**
  - Faktorenfolge muss von rechts nach links erzeugt werden
  - Faktoren haben keine Terme als direkte Teile
  - Es gibt nur einen Parsebaum für  $f_1 * f_2 * \dots * f_n$
- **Jeder Faktor ist Bezeichner oder geklammerter Ausdruck**



Einziger Ableitungsbaum  
für  $a + b * c$

$L = \{0^i 1^j 2^k \mid i=j \vee j=k\}$  IST INHÄRENT MEHRDEUTIG

- **$L$  ist Vereinigung zweier kontextfreier Sprachen**

- $L_1 = \{0^i 1^i 2^k \mid i, k \in \mathbb{N}\}$

- $G_1 = (\{S_1, A\}, \{0, 1, 2\}, \{S_1 \rightarrow S_1 2, S_1 \rightarrow A, A \rightarrow 0 A 1, A \rightarrow \epsilon\}, S_1)$

- $L_2 = \{0^i 1^j 2^j \mid i, j \in \mathbb{N}\}$

- $G_2 = (\{S_2, B\}, \{0, 1, 2\}, \{S_2 \rightarrow 0 S_2, S_2 \rightarrow B, B \rightarrow 1 B 2, B \rightarrow \epsilon\}, S_2)$

- Wörter von  $L_1$  und  $L_2$  haben verschiedene Ableitungsbäume

$L = \{0^i 1^j 2^k \mid i=j \vee j=k\}$  IST INHÄRENT MEHRDEUTIG

- **$L$  ist Vereinigung zweier kontextfreier Sprachen**
  - $L_1 = \{0^i 1^i 2^k \mid i, k \in \mathbb{N}\}$   
 $G_1 = (\{S_1, A\}, \{0, 1, 2\}, \{S_1 \rightarrow S_1 2, S_1 \rightarrow A, A \rightarrow 0 A 1, A \rightarrow \epsilon\}, S_1)$
  - $L_2 = \{0^i 1^j 2^j \mid i, j \in \mathbb{N}\}$   
 $G_2 = (\{S_2, B\}, \{0, 1, 2\}, \{S_2 \rightarrow 0 S_2, S_2 \rightarrow B, B \rightarrow 1 B 2, B \rightarrow \epsilon\}, S_2)$
  - Wörter von  $L_1$  und  $L_2$  haben verschiedene Ableitungsbäume
- **Manche Wörter von  $L$  gehören zu  $L_1 \cap L_2$** 
  - Wörter aus  $L_1 \cap L_2$  haben eine Ableitung in  $G_1$  und eine in  $G_2$
  - Wörter aus  $L$ , die zu  $L_1 \cap L_2$  gehören, haben keine eindeutige Ableitung



$L = \{0^i 1^j 2^k \mid i=j \vee j=k\}$  IST INHÄRENT MEHRDEUTIG

- **$L$  ist Vereinigung zweier kontextfreier Sprachen**

- $L_1 = \{0^i 1^i 2^k \mid i, k \in \mathbb{N}\}$

- $G_1 = (\{S_1, A\}, \{0, 1, 2\}, \{S_1 \rightarrow S_1 2, S_1 \rightarrow A, A \rightarrow 0 A 1, A \rightarrow \epsilon\}, S_1)$

- $L_2 = \{0^i 1^j 2^j \mid i, j \in \mathbb{N}\}$

- $G_2 = (\{S_2, B\}, \{0, 1, 2\}, \{S_2 \rightarrow 0 S_2, S_2 \rightarrow B, B \rightarrow 1 B 2, B \rightarrow \epsilon\}, S_2)$

- Wörter von  $L_1$  und  $L_2$  haben verschiedene Ableitungsbäume

- **Manche Wörter von  $L$  gehören zu  $L_1 \cap L_2$**

- Wörter aus  $L_1 \cap L_2$  haben eine Ableitung in  $G_1$  und eine in  $G_2$

- Wörter aus  $L$ , die zu  $L_1 \cap L_2$  gehören, haben keine eindeutige Ableitung

- **$L_1 \cap L_2 = \{0^i 1^i 2^i \mid i \in \mathbb{N}\}$  ist selbst nicht kontextfrei**

Beweis in §3.3

- Man kann keine einheitliche kfG zur Beschreibung von  $L_1 \cap L_2$  angeben

- Damit läßt sich auch keine bessere (eindeutige) kfG für  $L$  angeben

Intuitives Argument. Präziser Beweis benötigt verallgemeinertes Pumping-Lemma (Wegener §6.7)

## ● **Beschreibungsform für Programmiersprachen**

- Äquivalente Beschreibungsformen: Details in Vossen/Witt §7.3, 7.4

**Backus-Naur Form**, **Syntaxdiagramme**, Definitionsgleichungen, ...

## ● **Compiler benötigt Ableitungsbaum**

- Rekonstruktion nur möglich für **eindeutige Grammatiken**
- Mehrdeutige Grammatiken können evtl. eindeutig gemacht werden
- Manche kontextfreie Sprachen haben **keine eindeutige Grammatik**

## ● **Noch zu klärende Fragen**

- Welches **Maschinenmodell** erkennt genau die kontextfreien Sprachen?
- Wie kann man den Ableitungsbaum rekonstruieren? (**Syntaxanalyse**)
- Welche **Abschlusseigenschaften** gelten für kontextfreie Sprachen?
- Welche Spracheigenschaften lassen sich **nicht kontextfrei** beschreiben?