

## Übersicht

- Kontextfreie Grammatik
- Syntaktische Struktur eines Programms
  - Syntaktische Struktur
  - Kanonische Ableitungen
- Kellerautomat
- Item-Kellerautomat
  - Item-Kellerautomat
  - Kellerautomat mit Ausgabe
  - Linksparser und Rechtsparser
- FIRST und FOLLOW
  - FIRST und FOLLOW
  - FIRST
  - FOLLOW
  - Rekursive Gleichungssysteme
- FIRST<sub>1</sub> und FOLLOW<sub>1</sub>

## Lernziele

- Die wichtigsten theoretischen Aspekte von kontextfreien Grammatiken und Kellerautomaten und deren Zusammenhänge wiedergeben können
- Den Item-Kellerautomat als Basis für einen Parser beschreiben können
- Die Grundidee von FIRST- und FOLLOW-Mengen sowie deren wichtigste Eigenschaften skizzieren können
- Beschreiben können, wie man FIRST<sub>1</sub>- und FOLLOW<sub>1</sub>-Mengen berechnet

## Aufgabe der syntaktischen Analyse

- Erkennung und Darstellung der *syntaktischen Struktur* eines Programms  
(gegeben als Folge von Symbolen – vom Scanner erzeugt)
  - Beschreibung der syntaktischen Struktur:  
(deterministisch analysierbare) *kontextfreie Grammatik* ☞
  - Erkennung der syntaktischen Struktur:  
*Kellerautomat* ☞
- Erkennung und Behandlung von *Syntaxfehlern*

## Theoretische Grundlage (für die syntaktische Analyse)

- Äquivalenz zwischen kontextfreien Grammatiken und (nicht-determin.) Kellerautomaten

## Klassen von Syntax-Analyseverfahren

Für die syntaktische Analyse  
von Programmiersprachen

- *Top-Down-Analysatoren*  
Beginnend mit Startsymbol, Generierung von Satzformen und Bestätigung durch  
gelesene Eingabe
- *Bottom-up-Analysatoren*  
Sukzessive Reduktion der Eingabe („shift - reduce“) auf das Startsymbol

## Kontextfreie Grammatik (kurz: kfG)

- $G = (V_N, V_T, P, S)$ 
  - $V_N$ : endliche Menge von **Nichtterminalen**
  - $V_T$ : endliche Menge von **Terminalen** (wobei  $V_N \cap V_T = \emptyset$ )
  - $P \subseteq V_N \times (V_N \cup V_T)^*$ : endliche Menge von **Produktionsregeln**
  - $S \in V_N$ : **Startsymbol**

## Kontextfreie Sprache

Sprache, die durch kfG definierbar ist

## Notationelle Konventionen

- Verwendung von Zeichen
  - A, B, C, ..., X, Y, Z: Nichtterminale
  - a, b, c: Terminale
  - u, v, w, x, y, z: Terminalworte
  - $\alpha, \beta, \chi$ : beliebige Worte
- Darstellung von Produktionsregeln
  - $A \rightarrow \alpha$  für  $(A, \alpha)$
  - $A \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3$  für  $(A, \alpha_1), (A, \alpha_2), (A, \alpha_3)$



## Beispiele

- Pascal-ähnliche Grammatik

Anw → If\_Anw |  
While\_Anw |  
Repeat\_Anw |  
Proz\_Aufruf |  
Wertzuweisung

If\_Anw → **if** Bed **then** An\_Folge **else** An\_Folge **fi** |  
**if** Bed **then** An\_Folge **fi**

While\_Anw → **while** Bed **do** An\_Folge **od**

Repeat\_Anw → **repeat** An\_Folge **until** Bed

Proz\_Aufruf → Name ( Ausdr\_Folge )

Wertzuweisung → Name := Ausdr

An\_Folge → Anw | An\_Folge; Anw

Ausdr\_Folge → Ausdr | Ausdr\_Folge, Ausdr

*Beschreiben dieselbe Sprache,  
haben aber unterschiedliche Eigenschaften*

- Arithmetische Ausdrücke (vereinfacht)

- $G_0 = (\{E, T, F\}, \{+, *, (, ), \text{id}\}, \{E \rightarrow E+T \mid T, T \rightarrow T * F \mid F, F \rightarrow (E) \mid \text{id}\}, E)$
- $G_1 = (\{E\}, \{+, *, (, ), \text{id}\}, \{E \rightarrow E+E \mid E * E \mid (E) \mid \text{id}\}, E)$

## Wichtige Begriffe

- Sei  $G = (V_N, V_T, P, S)$  kfG,  $V = V_N \cup V_T$

- Produziert direkt

- $\varphi$  **produziert direkt**  $\psi$  **gemäß G**, i.Z.  $\varphi \Rightarrow_G \psi$ , (bzw.  $\varphi \Rightarrow \psi$ , falls G klar ist), wenn es  $\sigma, \tau, \alpha \in V^*$  und  $A \in V_N$  gibt, so dass  $\varphi = \sigma A \tau$  und  $\psi = \sigma \alpha \tau$  und  $A \rightarrow \alpha \in P$

*Intuitiv:*

Vorkommen von linker Seite einer Produktion in  $\varphi$  wird durch entsprechende rechte Seite ersetzt

- Produziert

- $\varphi$  **produziert**  $\psi$  **gemäß G**, i.Z.  $\varphi \Rightarrow_G^* \psi$ , (bzw.  $\varphi \Rightarrow^* \psi$ , falls G klar ist), oder  $\psi$  **ist** aus  $\varphi$  **gemäß G ableitbar**, wenn es  $\varphi_0, \varphi_1, \dots, \varphi_n \in V^*$  ( $n \geq 0$ ) gibt, so dass  $\varphi = \varphi_0$ ,  $\psi = \varphi_n$  und  $\varphi_i \Rightarrow_G \varphi_{i+1}$  für  $0 \leq i < n$

$\Rightarrow_G^*$  ist reflexiv-transitive Hülle von  $\Rightarrow_G$

- Ableitung

- Ist  $\psi$  aus  $\varphi$  ableitbar (mit  $\varphi_i$ ,  $0 \leq i \leq n$ , wie oben), dann heißt  $\varphi_0, \varphi_1, \dots, \varphi_n$  **Ableitung** von  $\psi$  aus  $\varphi$

## Beispiel

- Sei

- $G_0 = (\{E, T, F\}, \{+, *, (, ), \text{id}\}, \{E \rightarrow E+T \mid T, T \rightarrow T*F \mid F, F \rightarrow (E) \mid \text{id}\}, E)$
  - $G_1 = (\{E\}, \{+, *, (, ), \text{id}\}, \{E \rightarrow E+E \mid E*E \mid (E) \mid \text{id}\}, E)$

- Dann gilt

- $E \Rightarrow_{G_1} E+E \Rightarrow_{G_1} E*E+E \Rightarrow_{G_1} \text{id}*E+E \Rightarrow_{G_1} \text{id}*E+\text{id} \Rightarrow_{G_1} \text{id}*\text{id}+\text{id}$
  - $E \Rightarrow_{G_0} E+T \Rightarrow_{G_0} T+T \Rightarrow_{G_0} T*F+T \Rightarrow_{G_0} T*\text{id}+T \Rightarrow_{G_0} F*\text{id}+T \Rightarrow_{G_0} F*\text{id}+F \Rightarrow_{G_0} \text{id}*\text{id}+F \Rightarrow_{G_0} \text{id}*\text{id}+\text{id}$

## Weitere Begriffe

- Sei  $G = (V_N, V_T, P, S)$  kfG,  $V = V_N \cup V_T$ 
  - **Definierte Sprache:**  $L(G) =_{\text{def}} \{u \in V_T^* \mid S \Rightarrow_G^* u\}$
  - **Satz:**  $x \in L(G)$
  - **Satzform:**  $\alpha \in V^*$  mit  $S \Rightarrow_G^* \alpha$

## Beispiele

- Sätze
  - $\text{id*id+id} \in L(G_0)$
  - $\text{id*id+id} \in L(G_1)$
- Satzformen
  - Von  $G_0$ :  $E, E+T, T*\text{id}+T, \dots$
  - Von  $G_1$ :  $E, E+E, \text{id}*E+E, \dots$

## Eigenschaft

Alternative Definition  
für „kontextfrei“

- Für beliebige kfG  $G = (V_N, V_T, P, S)$  und (beliebige)  $\varphi, \psi \in V^*$  gilt:  
$$\varphi = \varphi_1\varphi_2 \dots \varphi_n \wedge \varphi \Rightarrow^* \psi \Leftrightarrow \exists \psi_1, \psi_2, \dots, \psi_n: \psi = \psi_1\psi_2 \dots \psi_n \wedge \forall_{i=1..n} \varphi_i \Rightarrow^* \psi_i$$

## Spezielle Nichtterminale

- Unproduktive und unerreichbare Nichtterminale A
  - **Unerreichbar**: Es gibt keine Worte  $\alpha, \beta$  mit  $S \Rightarrow^* \alpha A \beta$
  - **Unproduktiv**: Es gibt kein (terminales) Wort  $u$  mit  $A \Rightarrow^* u$
- **Reduzierte kfG** G
  - G enthält weder unerreichbare noch unproduktive Nichtterminale

*Im folgenden stets vorausgesetzt  
(wenn nicht explizit anders vorgegeben)*

## Offensichtlich

- Elimination unproduktiver und unerreichbarer Nichtterminale ändert die von der Grammatik definierte Sprache nicht

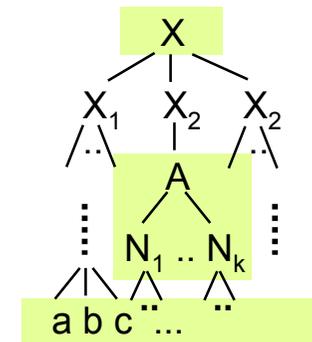
## Beispiel

- Geg.:  $G'_1 = (\{E, X, Y\}, \{+, *, (, ), \text{id}\}, \{E \rightarrow E+E \mid E * E \mid (E) \mid \text{id} \mid X, Y \rightarrow (\text{id})\}, E)$
- Dann
  - Unproduktiv: X (keine Regel mit X links !)
  - Unerreichbar: Y (keine Regel mit Y rechts !)
  - Zu  $G'_1$  gehörige reduzierte Grammatik:  $G_1$

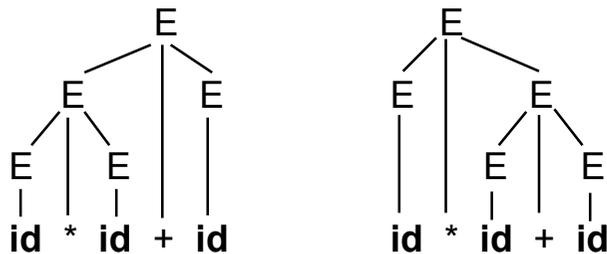


## Syntaxbaum

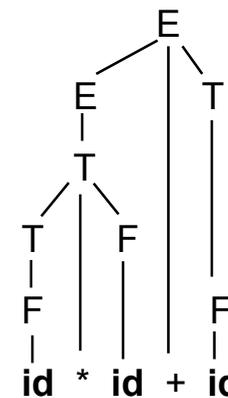
- Seien
  - $G = (V_N, V_T, P, S)$  kfG,  $V = V_N \cup V_T$
  - $B =$  geordneter Baum: Blattmarkierungen  $\in V_T \cup \{\epsilon\}$ ; (innere) Knotenmarkierungen  $\in V_N$
- $B$  heißt **Syntaxbaum** (syn.: Strukturbaum) für  $x \in V_T^*$  und  $X \in V_N$  gemäß  $G$ , wenn gilt:
  - (a) Wurzel ist markiert mit  $X$
  - (b) „Blattwort“ von  $B$  (= Blattmarkierungen v.l.n.r. konkateniert) ist  $x$
  - (c) Ist  $n$  innerer Knoten markiert mit  $A \in V_N$ , so gilt
    - Kinder sind markiert (v.l.n.r.) mit  $N_1, N_2, \dots, N_k \in V$  und  $A \rightarrow N_1 N_2 \dots N_k \in P$
    - Einziges Kind ist markiert mit  $\epsilon$  und  $A \rightarrow \epsilon \in P$
- $B$  heißt **Syntaxbaum** für  $x \in V_T^*$ , falls  $X = S$



## Beispiel (Syntaxbäume für $id * id + id$ )



Syntaxbäume gemäß  $G_1$



Syntaxbaum gemäß  $G_0$

$$G_1 = (\{E\} \{+, *, (, ), id\}, \{E \rightarrow E+E \mid E * E \mid (E) \mid id\}, E) \quad G_0 = (\{E, T, F\}, \{+, *, (, ), id\}, \{E \rightarrow E+T \mid T, T \rightarrow T * F \mid F, F \rightarrow (E) \mid id\}, E)$$

## Weitere Begriffe

- Mehrdeutig
  - **Mehrdeutiger Satz**  $x \in L(G)$ :  $x$  hat mehr als einen Syntaxbaum
  - **Mehrdeutige kfG**  $G$ :  $G$  hat (mindestens) einen mehrdeutigen Satz
- **Eindeutige kfG**  $G$ :  $G$  ist nicht mehrdeutig

*Bemerkungen und Definitionen für Sätze  
gelten sinngemäß auch für Satzformen*

## Beispiele

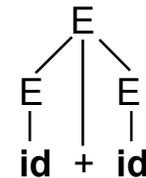
- $G_1$  ist mehrdeutig, da der Satz **id \* id + id** mehrdeutig ist (2 Syntaxbäume!)
- $G_0$  ist eindeutig



## Bemerkungen (über beliebige Sätze x)

- 1. x hat mindestens eine Ableitung (folgt unmittelbar aus Definition von Satz)
  - Beispiel: **id + id** hat gemäß  $G_1$  zwei Ableitungen
    - $E \Rightarrow E+E \Rightarrow \mathbf{id+E} \Rightarrow \mathbf{id+id}$
    - $E \Rightarrow E+E \Rightarrow E+\mathbf{id} \Rightarrow \mathbf{id+id}$

- 2. Zu jeder Ableitung für x gehört ein Syntaxbaum für x
  - Beispiel:



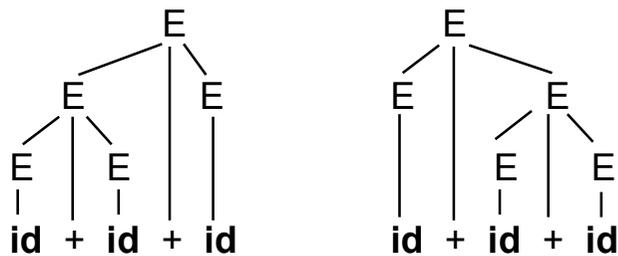
- 3. x besitzt mindestens einen Syntaxbaum (folgt unmittelbar aus 1 und 2)
- 4. Zu jedem Syntaxbaum für x gibt es mindestens eine Ableitung für x

## Es gilt

- Syntaxbaum abstrahiert von der Reihenfolge der Anwendungen der Produktionen (d.h. evtl. mehrere Ableitungen für einen Syntaxbaum)
- Umgekehrt auch möglich: mehrere Syntaxbäume für eine Ableitung

## Beispiel

- Ableitung
  - $E \Rightarrow_{G_1} E+E \Rightarrow_{G_1} E+E+E \Rightarrow_{G_1} \mathbf{id+E+E} \Rightarrow_{G_1} \mathbf{id+id+E} \Rightarrow_{G_1} \mathbf{id+id+id}$
- Syntaxbäume



## Kanonische Ableitungen und Satzformen

- Sei  $\varphi_1, \varphi_2, \dots, \varphi_n$  eine Ableitung von  $\varphi = \varphi_n$  aus  $\varphi_1 = S$ 
  - **Linksableitung** (i.Z.  $S \Rightarrow_{lm}^* \varphi$ ): beim Schritt von  $\varphi_i$  nach  $\varphi_{i+1}$  wird jeweils das in  $\varphi_i$  am weitesten links stehende Nichtterminal ersetzt
  - **Rechtsableitung** (i.Z.  $S \Rightarrow_{rm}^* \varphi$ ): beim Schritt von  $\varphi_i$  nach  $\varphi_{i+1}$  wird jeweils das in  $\varphi_i$  am weitesten rechts stehende Nichtterminal ersetzt
  - **Links(Rechts-)satzform**: Satzform, die in einer Links(Rechts-)ableitung auftritt



## Weitere Bemerkungen

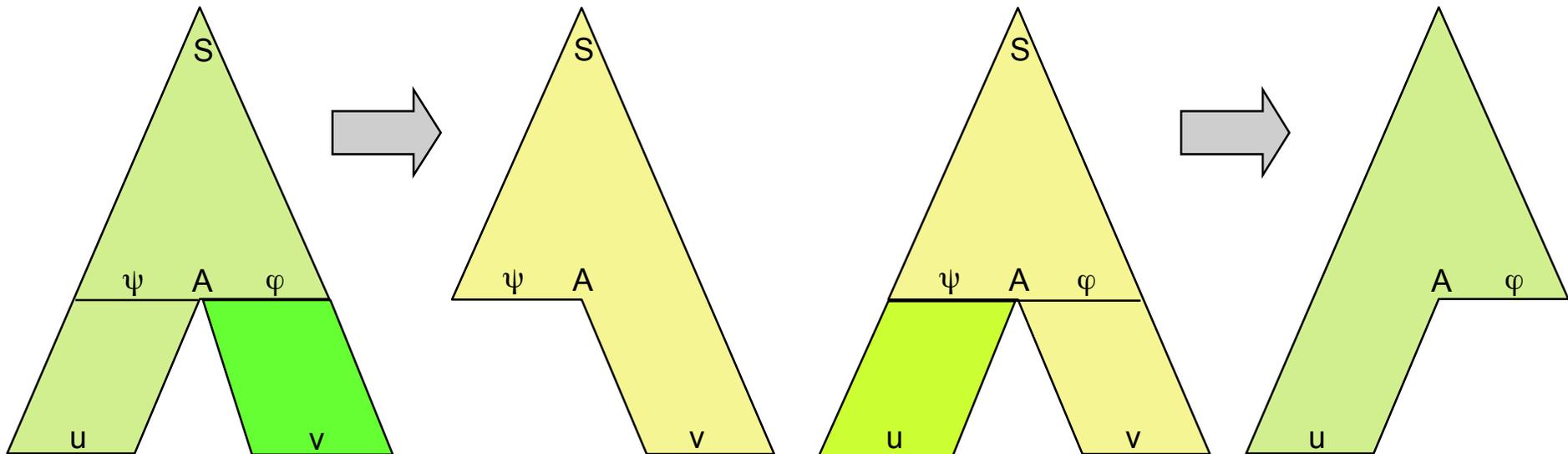
- 5. Zu jedem Satz gibt es mindestens eine Links- und eine Rechtsableitung
  - Beispiel: betrachtet kfG  $G_1$  und  $\mathbf{id * id + id}$
  - Linksableitungen
    - $E \Rightarrow_{lm} E+E \Rightarrow_{lm} E^*E+E \Rightarrow_{lm} \mathbf{id^*E+E} \Rightarrow_{lm} \mathbf{id^*id+E} \Rightarrow_{lm} \mathbf{id^*id+id}$
    - $E \Rightarrow_{lm} E^*E \Rightarrow_{lm} \mathbf{id^*E} \Rightarrow_{lm} \mathbf{id^*E+E} \Rightarrow_{lm} \mathbf{id^*id+E} \Rightarrow_{lm} \mathbf{id^*id+id}$
  - Rechtsableitungen
    - $E \Rightarrow_{rm} E+E \Rightarrow_{rm} E+\mathbf{id} \Rightarrow_{rm} E^*E+\mathbf{id} \Rightarrow_{rm} E^*\mathbf{id+id} \Rightarrow_{rm} \mathbf{id^*id+id}$
    - $E \Rightarrow_{rm} E^*E \Rightarrow_{rm} E^*E+E \Rightarrow_{rm} E^*E+\mathbf{id} \Rightarrow_{rm} E^*\mathbf{id+id} \Rightarrow_{rm} \mathbf{id^*id+id}$
- 6. Zu jedem eindeutigen Satz gibt es genau eine Links- und eine Rechtsableitung
  - Beispiel: betrachtet kfG  $G_1$  und  $\mathbf{id + id}$ 
    - Linksableitung:  $E \Rightarrow_{lm} E+E \Rightarrow_{lm} \mathbf{id+E} \Rightarrow_{lm} \mathbf{id+id}$
    - Rechtsableitung:  $E \Rightarrow_{rm} E+E \Rightarrow_{rm} E+ \mathbf{id} \Rightarrow_{rm} \mathbf{id+id}$

Es gilt:  
Eindeutig = 1 Syntaxbaum



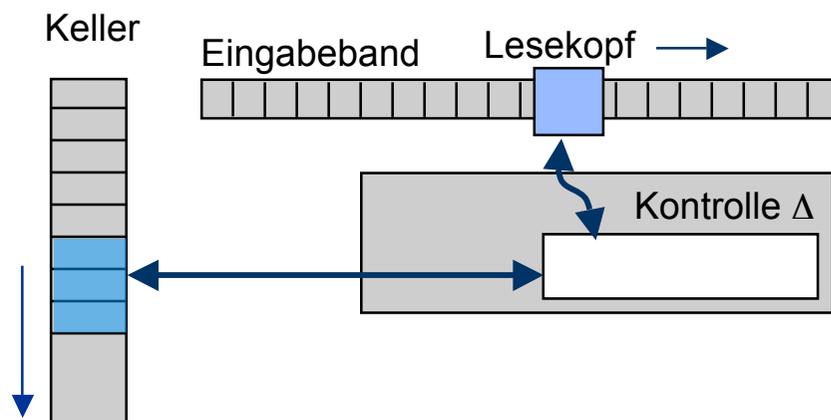
## Zusammenhang zwischen Links- und Rechtsableitungen

- Zu jeder Linksableitung gibt es eine korrespondierende Rechtsableitung und umgekehrt
  - Wenn  $S \Rightarrow_{lm}^* uA\varphi$ , dann gibt es  $\psi$ , mit  $\psi \Rightarrow^* u$ , so dass für alle  $v$  mit  $\varphi \Rightarrow^* v$  gilt:  $S \Rightarrow_{rm}^* \psi Av$
  - Wenn  $S \Rightarrow_{rm}^* \psi Av$ , dann gibt es  $\varphi$ , mit  $\varphi \Rightarrow^* v$ , so daß für alle  $u$  mit  $\psi \Rightarrow^* u$  gilt:  $S \Rightarrow_{lm}^* uA\varphi$



## Kellerautomat (intuitiv)

- (mathematische) Maschine zum Erkennen kontextfreier Sprachen
- Charakterisiert durch
  - *Keller* mit unbegrenzter Speicherfähigkeit
  - *Eingabeband*
  - *Lesekopf* zum zeichenweisen Lesen des Eingabebandes (von links nach rechts)
  - Kontrolle
    - Zustandsübergänge abhängig von Eingabezeichen und obersten Kellersymbolen
    - Lesekopf wird evtl. (um ein Zeichen) verschoben
    - nur oberste Kellersymbole werden verändert



## Kellerautomat (formal)

- Tupel  $(V, Q, \delta, q_0, F)$  wobei
  - $V$ : (endliches) **Eingabealphabet**
  - $Q$ : endliche Menge von **Zuständen**
  - $q_0 \in Q$ : **Anfangszustand**
  - $F \subseteq Q$ : Menge von **Endzuständen**
  - $\delta: Q^+ \times (V \cup \{\varepsilon\}) \rightarrow \mathcal{F}(Q^*)$ : **Übergangsrelation**

$\mathcal{F}(Q^*)$  ist die Menge der endlichen Teilmengen von  $Q^*$

*Intuitiv: Zu jedem Kellerinhalt kann es mehrere Folgeinhalte gemäß  $\delta$  geben*

(dargestellt als endliche partielle Funktion  $\delta$  in die endlichen Teilmengen von  $Q^*$ )

## Charakterisierung

- Kellerinhalt = Folge von Zuständen
- Oberstes Kellersymbol = Aktueller Zustand
- Mehrfach nichtdeterministisches Verhalten
  - Mehrere Fortsetzungen gemäß  $\delta$  ———  $\delta$  ist Relation
  - Suffixe ( $\phi \in \delta(\gamma, a), \psi \in \delta(\gamma', a)$  mit  $\phi \neq \psi$  und  $\gamma$  Suffix von  $\gamma'$ )
  - $\varepsilon$ -Übergänge ———  $\varepsilon$  ist spezielles Suffix

## Beispiel

- Kellerautomat zur Erkennung korrekter (vollständig geklammerter) Klammerfolgen
  - $(\{[, ]\}, \{A, K, E\}, \{(A, [) \mapsto \{AK\}, (K, [) \mapsto \{KK\}, (KK, [) \mapsto \{K\}, (AK, [) \mapsto \{E\}\}, A, \{E\})$

## Konfiguration, akzeptierte Sprache

- Sei  $P = (V, Q, \delta, q_0, F)$  ein Kellerautomat und  $v, w \in V^*$  mit  $v$  Suffix von  $w$
- Konfigurationen von  $P$ 
  - **Konfiguration:** Paar  $(\gamma, v) \in Q^+ \times V^*$
  - **Anfangskonfiguration:**  $(q_0, w)$
  - **Endkonfiguration:**  $(q, \varepsilon)$  mit  $q \in F$
- **Übergangsrelationen** zwischen Konfigurationen ( $\gamma_2, \gamma_3 \in Q^*, \gamma_1 \in Q^+, a \in V \cup \{\varepsilon\}$ )
  - $\vdash_P: (\gamma_1\gamma_2, av) \vdash_P (\gamma_1\gamma_3, v)$ , wenn  $\gamma_3 \in \delta(\gamma_2, a)$
  - $\vdash_P^*$ : reflexiv-transitive Hülle von  $\vdash_P$
- **Akzeptiertes Wort, akzeptierte Sprache**  $L(P)$ 
  - $w \in V^*$  wird von  $P$  akzeptiert  $\Leftrightarrow (q_0, w) \vdash_P^* (q, \varepsilon)$  mit  $q \in F$
  - $L(P) =_{\text{def}} \{w \in V^* \mid w \text{ wird von } P \text{ akzeptiert}\}$

Nichtleere Zustandsfolge im Keller, Resteingabe

**Beachte:**

- für  $\delta$  nur oberste Kellerelemente relevant
- $\vdash_P$  berücksichtigt vollständige Kellerinhalte

**Beispiel** (Klammerfolgen mit  $\delta = \{(A, []) \mapsto \{AK\}, (K, []) \mapsto \{KK\}, (KK, ]) \mapsto \{K\}, (AK, ]) \mapsto \{E\}\}$ )

- Konfigurationen ( $v \in V^*, v \neq \varepsilon$ )
  - $\{(A, v), (AK^*K, [v]), (AK^+K, ]v), (AK, ]), (E, \varepsilon)\}$
- Folgekonfigurationen (Verarbeitung von  $[[[]]]$ )
  - $(A, [[[]]]) \vdash_P (AK, [[]]) \vdash_P (AKK, [])] \vdash_P (AKKK, ])] \vdash_P (AKK, ])] \vdash_P (AK, [])] \vdash_P (AKK, ])] \vdash_P (AK, ])] \vdash_P (E, \varepsilon)$

## Deterministischer Kellerautomat

- Sei  $P = (V, Q, \delta, q_0, F)$  ein Kellerautomat
- $P$  heißt **deterministischer Kellerautomat**, wenn gilt
  - $|\delta(\gamma, a)| \leq 1$  für alle  $\gamma \in Q^+$ ,  $a \in V \cup \{\varepsilon\}$  und
  - Ist  $\delta(\gamma, a)$  definiert für  $\gamma \in Q^+$ ,  $a \in V$ , dann ist
    - $\delta(\gamma', a)$  undefiniert, falls  $\gamma'$  echtes Suffix von  $\gamma$  ist
    - $\delta(\gamma', \varepsilon)$  undefiniert, falls  $\gamma'$  Suffix von  $\gamma$  ist

Mit anderen Worten:  
 $\delta$  ist partielle Funktion

Umfasst  $\delta(\gamma, \varepsilon)$  undefiniert

## Beispiel

- Kellerautomat zur Klammererkennung ist deterministisch
- Es war  $\delta = \{(A, [) \mapsto \{AK\}, (K, [) \mapsto \{KK\}, (KK, [) \mapsto \{K\}, (AK, [) \mapsto \{E\}\}$
- Es gilt
  - $|\delta(\gamma, a)| \leq 1$  für alle  $\gamma \in \{A, K, E\}^+$ ,  $a \in \{[, ]\} \cup \{\varepsilon\}$
  - $\delta(A, [)$  definiert:  $\delta(A, \varepsilon)$  undefiniert
  - $\delta(K, [)$  definiert:  $\delta(K, \varepsilon)$  undefiniert
  - $\delta(KK, [)$  definiert:  $\delta(K, [)$ ,  $\delta(K, \varepsilon)$ ,  $\delta(KK, \varepsilon)$  undefiniert
  - $\delta(AK, [)$  definiert:  $\delta(K, [)$ ,  $\delta(K, \varepsilon)$ ,  $\delta(AK, \varepsilon)$  undefiniert

## Kontextfreies Item

- Sei  $G = (V_N, V_T, P, S)$  kfG,  $A \rightarrow \alpha\beta \in P$ 
  - **Item:**  $[A \rightarrow \alpha.\beta]$
  - **Vollständiges Item:**  $[A \rightarrow \alpha\beta.]$
  - **Start-Item:**  $[A \rightarrow .\alpha\beta]$

Immer erreichbar durch Hinzufügen von  $S' \rightarrow .S$  für  $S' \in V_N$

## Item-Kellerautomat (intuitiv)

- Zustandsmenge:  $It_G =$  Menge aller Items der Grammatik
- Aktueller Zustand: gerade bearbeitetes Item (repräsentiert Analyse-Situation)
- Endzustände: vollständige Items  $S \rightarrow \alpha.$  (falls  $S$  in keiner Produktion rechts vorkommt)

## Item-Kellerautomat einer kfG

- Sei  $G = (V_N, V_T, P, S)$  kfG,  $S' \in V_N$
- **Item-Kellerautomat  $K_G$ :** Kellerautomat  $(V_T, It_G, \delta, [S' \rightarrow .S], \{[S' \rightarrow S.]\})$

### Übergänge von $\delta$ :

- (E)  $\delta([X \rightarrow \beta.Y\gamma], \epsilon) = \{[X \rightarrow \beta.Y\gamma] [Y \rightarrow \alpha.] \mid Y \rightarrow \alpha \in P\}$     Expansion
- (L)  $\delta([X \rightarrow \beta.a\gamma], a) = \{[X \rightarrow \beta a.\gamma]\}$     Lesen
- (R)  $\delta([X \rightarrow \beta.Y\gamma] [Y \rightarrow \alpha.], \epsilon) = \{[X \rightarrow \beta Y.\gamma]\}$     Reduktion

Ist i.a. nicht-deterministisch

- Es gilt:  $L(G) = L(K_G)$

**Beispiel:** Geg.:  $G_0' = (\{S, E, T, F\}, \{+, *, (, ), id\}, \{S \rightarrow E, E \rightarrow E+T \mid T, T \rightarrow T * F \mid F, F \rightarrow (E) \mid id\}, S)$

**Übergangsrelation**  $\delta$  von  $K_{G_0'}$

Oberes Kellerende	Eingabe	neues oberes Kellerende
[S → .E]	ε	[S → .E] [E → .E+T]
[S → .E]	ε	[S → .E] [E → .T]
[E → .E+T]	ε	[E → .E+T] [E → .E+T]
[E → .E+T]	ε	[E → .E+T] [E → .T]
[E → E+.T]	ε	[E → E+.T] [T → .T * F]
[E → E+.T]	ε	[E → E+.T] [T → .F]
[E → .T]	ε	[E → .T] [T → .T * F]
[E → .T]	ε	[E → .T] [T → .F]
[T → .T * F]	ε	[T → .T * F] [T → .T * F]
[T → .T * F]	ε	[T → .T * F] [T → .F]
[T → T *.F]	ε	[T → T *.F] [F → .(E)]
[T → T *.F]	ε	[T → T *.F] [F → .id]
[T → .F]	ε	[T → .F] [F → .id]
[T → .F]	ε	[T → .F] [F → .id]
[F → .(E)]	ε	[F → .(E)] [E → .E+T]
[F → .(E)]	ε	[F → .(E)] [E → .T]
[E → E.+T]	+	[E → E+.T]
[T → T.*F]	*	[T → T *.F]
F → .id	id	F → id.

Oberes Kellerende	Eingabe	neues oberes Kellerende
[F → .(E)]	(	[F → .(E)]
[F → (E.)]	)	[F → (E).]
[S → .E] [E → E+T.]	ε	[S → E.]
[S → .E] [E → T.]	ε	[S → E.]
[E → .E+T] [E → E+T.]	ε	[E → E.+T]
[E → .E+T] [E → T.]	ε	[E → E.+T]
[E → E+.T] [T → T * F.]	ε	[E → E+T.]
[E → E+.T] [T → F.]	ε	[E → E+T.]
[E → .T] [T → T * F.]	ε	[E → T.]
[E → .T] [T → F.]	ε	[E → T.]
[T → .T * F] [T → T * F.]	ε	[T → T *.F]
[T → .T * F] [T → F.]	ε	[T → T *.F]
[T → T *.F] [F → (E).]	ε	[T → T * F.]
[T → T *.F] [F → id.]	ε	[T → T * F.]
[T → .F] [F → (E).]	ε	[T → F.]
[T → .F] [F → id.]	ε	[T → F.]
[F → .(E)] [E → E+T.]	ε	[F → (E.)]
[F → .(E)] [E → T.]	ε	[F → (E.)]

Kellerinhalt	Resteingabe
[S → .E]	id + id * id
[S → .E] [E → .E+T]	id + id * id
[S → .E] [E → .E+T] [E → .T]	id + id * id
[S → .E] [E → .E+T] [E → .T] [T → .F]	id + id * id
[S → .E] [E → .E+T] [E → .T] [T → .F] [F → .id]	id + id * id
[S → .E] [E → .E+T] [E → .T] [T → .F] [F → id.]	+ id * id
[S → .E] [E → .E+T] [E → .T] [T → F.]	+ id * id
[S → .E] [E → .E+T] [E → T.]	+ id * id
[S → .E] [E → E.+T]	+ id * id
[S → .E] [E → E+T.]	id * id
[S → .E] [E → E+T.] [T → .T*F]	id * id
[S → .E] [E → E+T.] [T → .T*F] [T → .F]	id * id
[S → .E] [E → E+T.] [T → .T*F] [T → .F] [F → .id]	id * id
[S → .E] [E → E+T.] [T → .T*F] [T → .F] [F → id.]	* id
[S → .E] [E → E+T.] [T → .T*F] [T → F.]	* id
[S → .E] [E → E+T.] [T → T*.F]	* id
[S → .E] [E → E+T.] [T → T*.F]	id
[S → .E] [E → E+T.] [T → T*.F] [F → .id]	id
[S → .E] [E → E+T.] [T → T*.F] [F → id.]	
[S → .E] [E → E+T.] [T → T*.F.]	
[S → .E] [E → E+T.]	
[S → E.]	

## Arbeitsweise des Kellerautomaten

	Expansion
	Lesen
	Reduktion

## Situation

- Bisher: nur Akzeptoren (die das „Wortproblem“ lösen)
- Für Übersetzung wichtig: syntaktische Struktur akzeptierter Wörter

## Kellerautomat mit Ausgabe

- **Kellerautomat mit Ausgabe:**  $K = (V, Q, O, \delta, q_0, F)$  wobei
  - $V, Q, q_0, F$ : wie bisher definiert
  - $O$ : endliches Ausgabealphabet
  - $\delta$ : endliche Relation zwischen  $Q^+ \times (V \cup \{\varepsilon\})$  und  $Q^* \times (O \cup \{\varepsilon\})$
- **Konfiguration:** Element aus  $Q^+ \times V^* \times O^*$

## Spezielle Item-Kellerautomaten (mit Ausgabe)

- Ausgabealphabet: Produktionen (d.h.  $O = P$ )
- **Linksparser:** Ausgabe nur bei (E)-Übergängen
- **Rechtsparser:** Ausgabe nur bei (R)-Übergängen
- **Konfiguration:** Element aus  $It_G^+ \times V_T^* \times P^*$





## Linksparser

- Sei  $G = (V_N, V_T, P, S)$  kfG
- **Linksparser** (entspricht Linksableitung)
  - Item-Kellerautomat mit Ausgabe  $K_G^l = (V_T, It_G, P, \delta_l, [S' \rightarrow .S], \{[S' \rightarrow S.]\})$
  - (E)-Übergänge:  $\delta_l([X \rightarrow \beta.Y\gamma], \varepsilon) = \{([X \rightarrow \beta.Y\gamma] [Y \rightarrow .\alpha], Y \rightarrow \alpha) \mid Y \rightarrow \alpha \in P\}$
  - (L) und (R)-Übergänge: keine Ausgabe
- Ausgabe-Übergang
  - $(\rho[X \rightarrow \beta.Y\gamma], w, o) \vdash_{K_G^l} (\rho[X \rightarrow \beta.Y\gamma] [Y \rightarrow .\alpha], w, o(Y \rightarrow \alpha))$

## Rechtsparser

- Sei  $G = (V_N, V_T, P, S)$  kfG
- **Rechtsparser** (entspricht Rechtsableitung)
  - Item-Kellerautomat mit Ausgabe  $K_G^r = (V_T, It_G, P, \delta_r, [S' \rightarrow .S], \{[S' \rightarrow S.]\})$
  - (R)-Übergänge:  $\delta_r([X \rightarrow \beta.Y\gamma] [Y \rightarrow \alpha.], \varepsilon) = \{([X \rightarrow \beta.Y.\gamma], Y \rightarrow \alpha)\}$
  - (E) und (L)-Übergänge: keine Ausgabe
- Ausgabe-Übergang
  - $(\rho[X \rightarrow \beta.Y\gamma] [Y \rightarrow \alpha.], w, o) \vdash_{K_G^r} (\rho[X \rightarrow \beta.Y.\gamma], w, o(Y \rightarrow \alpha))$

### Unterschied:

*Linksparser hat weniger Information;  
Rechtsparser gibt angewandte Produktion aus,  
wenn rechte Seite vollständig erkannt*

## Ziel

- (E)-Übergänge  $\delta([X \rightarrow \beta.Y\gamma], \epsilon) = \{[X \rightarrow \beta.Y\gamma] [Y \rightarrow \cdot\alpha] \mid Y \rightarrow \alpha \in P\}$   
des Item-Kellerautomaten deterministisch machen
- Dazu benötigt
  - Anfänge der von einem Nichtterminal (oben:  $Y$ ) produzierbaren Worte / **FIRST**
  - Anfänge von Worten (oben:  $\gamma$ ), die auf ein Nichtterminal folgen können (falls  $Y \epsilon$  produziert) / **FOLLOW**

## Hilfsdefinitionen

- Sei  $V$  Alphabet,  $w = a_1 \dots a_n$ ,  $a_i \in V$ ; außerdem:  $V^{\leq k} =_{\text{def}} \bigcup_{i=0..k} V^i$ ;  $V_{\#}^{\leq k} =_{\text{def}} V^{\leq k} \cup V^{\leq k-1}\{\#\}$
- **k-Präfix** von  $w$ :  $k: w =_{\text{def}} a_1 \dots a_{\min(n,k)}$  Somit:  $k: \epsilon = \epsilon$
- **k-Konkatenation** (auf Wörtern):  $\oplus_k =_{\text{def}} V^* \times V^* \rightarrow V^{\leq k}$  definiert durch:  $u \oplus_k v =_{\text{def}} k: uv$
- **k-Konkatenation** (auf Wortmengen):  $L_1 \oplus_k L_2 =_{\text{def}} \{u \oplus_k v \mid u \in L_1, v \in L_2\}$  mit  $L_1, L_2 \subseteq V^*$

## Beispiel

- Sei  $V = \{a, b, c\}$ ,  $L_1 = \{aabc, ab, c\}$ ,  $L_2 = \{bcbb, bbc\}$ ; dann gilt
  - $3: aabc = aab$ ,  $3: ab = ab$ ,  $2: bcbb = bc$ ,  $2: bbc = bb$
  - $aabc \oplus_5 bcbb = aabcb$ ,  $c \oplus_5 bbc = cbcc$
  - $L_1 \oplus_5 L_2 = \{aabcb, abccb, abbbc, cbcbb, cbcc\}$

## FIRST<sub>k</sub> und FOLLOW<sub>k</sub>

- Sei  $G = (V_N, V_T, P, S)$  kfG,  $V = V_N \cup V_T$
- **FIRST<sub>k</sub>** (auf Worten):  $\text{FIRST}_k: V^* \rightarrow \mathbb{P}(V_T^{\leq k})$  mit  $\text{FIRST}_k(\alpha) =_{\text{def}} \{ \beta \mid \alpha \Rightarrow^* \beta \}$
- **FOLLOW<sub>k</sub>** (auf Worten):  $\text{FOLLOW}_k: V^* \rightarrow \mathbb{P}(V_T^{\leq k})$  mit  $\text{FOLLOW}_k(\alpha) =_{\text{def}} \{ w \mid S \Rightarrow^* \beta \alpha \gamma \text{ und } w \in \text{FIRST}_k(\gamma) \}$
- FIRST<sub>k</sub> (auf Wortmengen):  
 $\text{FIRST}_k(L) =_{\text{def}} \bigcup_{\alpha \in L} \text{FIRST}_k(\alpha)$  für  $L \subseteq V^*$

Mengen von Worten  
über  $V_T$  mit Länge  $\leq k$

Somit:  $\text{FIRST}_k(\varepsilon) = \{\varepsilon\}$

Mengen von Worten über  $V_T$   
mit Länge  $\leq k$ , evtl. durch #  
abgeschlossen; siehe oben  
(„Hilfsdefinitionen“)

## Beispiel

- Für  $P = \{S \rightarrow AcBd, A \rightarrow a \mid aA \mid \varepsilon, B \rightarrow b \mid bB \mid \varepsilon\}$  gilt:
  - $\text{FIRST}_2(A) = \{a, aa, \varepsilon\}$
  - $\text{FOLLOW}_3(A) = \{cbb, cbd, cd\}$

## Wichtige Eigenschaften

- Seien  $L \subseteq V^*$ ,  $k \geq 1$ ;  $X_i \in V^*$ ,  $u_i \in V_T^*$ ; Dann gelten
  - $L \oplus_k \{\varepsilon\} = \{\varepsilon\} \oplus_k L = \text{FIRST}_k(L)$
  - $\text{FIRST}_k(X_1 \dots X_n) = \text{FIRST}_k(X_1) \oplus_k \dots \oplus_k \text{FIRST}_k(X_n)$
  - $\text{FIRST}_k(u_1) \oplus_k \dots \oplus_k \text{FIRST}_k(u_n) = \{u_1\} \oplus_k \dots \oplus_k \{u_n\}$

## Berechnung von FIRST und FOLLOW

- Rekursiv über die Produktionen der Grammatik

## Notationelle Konventionen (für eine Produktion $p$ )

- $n_p$  Anzahl Vorkommen von Nichtterminalen rechts in Produktion  $p$
- $p[i]$   $i$ -tes Nichtterminal in  $p$  ( $0 \leq i \leq n_p$ )
- $p[0]$  linke Seite von  $p$
- $(p, i)$  Vorkommen eines Nichtterminals in  $p$  an Position  $i$

## Beispiel

- Sei  $p = E \rightarrow E+T$ ; dann
  - $n_p = 2$
  - $p[1] = E, p[2] = T$
  - $p[0] = E$
  - Vorkommen von  $E$ :  $(p,0), (p,1)$

## Außerdem vorausgesetzt

- Erweiterte Grammatiken (d.h.  $S$  tritt nicht auf einer rechten Seite auf)
- Reduzierte Grammatiken (wie bisher immer)

## Herleitung der Berechnung von FIRST

- Es war  $FIRST_k(\alpha) =_{\text{def}} \{ k: u \mid \alpha \Rightarrow^* u \}$
- Damit (für  $X \in V_N$ )

$$FIRST_k(X) =_{\text{def}} \{ k: u \mid X \Rightarrow^* u \}$$

$$= [ \text{für } X \rightarrow u_0 X_1 u_1 X_2 \dots X_{n_p} u_{n_p} \text{ mit } n_p \geq 1 ]$$

$$\bigcup_{\{p: p[0]=X\}} FIRST_k(u_0 X_1 u_1 X_2 \dots X_{n_p} u_{n_p})$$

$\{p: p[0]=X\}$  = Menge aller Produktionen, bei denen X links vorkommt

$$= [ \text{Zerlegungseigenschaft von } FIRST_k ]$$

$$\bigcup_{\{p: p[0]=X\}} FIRST_k(u_0) \oplus_k FIRST_k(X_1) \oplus_k FIRST_k(u_1) \oplus_k \dots \oplus_k FIRST_k(X_{n_p}) \oplus_k FIRST_k(u_{n_p})$$

$$= [ \text{Eigenschaft von } FIRST_k \text{ für Terminalzeichenreihen} ]$$

$$\bigcup_{\{p: p[0]=X\}} \{u_0\} \oplus_k FIRST_k(X_1) \oplus_k \{u_1\} \oplus_k \dots \oplus_k FIRST_k(X_{n_p}) \oplus_k \{u_{n_p}\}$$

## Rekursives Gleichungssystem für $FIRST_k$

- $Fi_k(X) =_{\text{def}} \bigcup_{\{p: p[0]=X\}} \{u_0\} \oplus_k Fi_k(X_1) \oplus_k \{u_1\} \oplus_k \dots \oplus_k Fi_k(X_{n_p}) \oplus_k \{u_{n_p}\}$  für alle  $X \in V_N$   
falls  $p = X_0 \rightarrow u_0 X_1 u_1 X_2 \dots X_{n_p} u_{n_p}$  mit  $n_p \geq 1$

Falls es für X Produktionen beider Typen gibt, ist  $Fi_k(X)$  durch die Vereinigung der beiden rechten Seiten definiert

- $Fi_k(X) =_{\text{def}} \bigcup_{\{p: p[0]=X\}} \{k: u\}$ , falls  $p = X \rightarrow u$

Offensichtlicher Spezialfall

## Beispiel

- Sei

- $G_2 = (\{S, E, E', T, T', F\}, \{+, *, (, ), \text{id}\}, P, S)$  mit
- $P = \{S \rightarrow E, E \rightarrow TE', E' \rightarrow +E, E' \rightarrow \varepsilon, T \rightarrow FT', T' \rightarrow *T, T' \rightarrow \varepsilon, F \rightarrow (E), F \rightarrow \text{id}\}$
- Wobei gilt:  $L(G_2) = L(G_0) = L(G_1)$

- Es war

- $Fi_k(X) =_{\text{def}} \bigcup_{\{p: p[0]=X\}} \{u_0\} \oplus_k Fi_k(X_1) \oplus_k \{u_1\} \oplus_k \dots \oplus_k Fi_k(X_{n_p}) \oplus_k \{u_{n_p}\}$   
falls  $p = X \rightarrow u_0 X_1 u_1 X_2 \dots X_{n_p} u_{n_p}$
- $Fi_k(X) =_{\text{def}} \bigcup_{\{p: p[0]=X\}} \{k: u\},$  falls  $p = X \rightarrow u$

- Man erhält (als rekursives Gleichungssystem)

- $Fi_k(S) = Fi_k(E)$
- $Fi_k(E) = Fi_k(T) \oplus_k Fi_k(E')$
- $Fi_k(E') = (\{+\} \oplus_k Fi_k(E)) \cup \{\varepsilon\}$
- $Fi_k(T) = Fi_k(F) \oplus_k Fi_k(T')$
- $Fi_k(T') = (\{*\} \oplus_k Fi_k(T)) \cup \{\varepsilon\}$
- $Fi_k(F) = (\{( \} \oplus_k Fi_k(E) \oplus_k \{ \}) \cup \{\text{id}\}$

## Rekursives Gleichungssystem für FOLLOW<sub>k</sub>

- $Fo_k(S) =_{\text{def}} \{\#\}$
- $Fo_k(X) =_{\text{def}} \bigcup_{\{p: p[i]=X, 1 \leq i \leq n_p\}} \{u_i\} \oplus_k Fi_k(X_{i+1}) \oplus_k \{u_{i+1}\} \oplus_k \dots \oplus_k Fi_k(X_{n_p}) \oplus_k \{u_{n_p}\} \oplus_k Fo_k(X_0)$   
für alle  $X \in V_N \setminus \{S\}$ , falls  $p = X_0 \rightarrow u_0 X_1 u_1 X_2 \dots X_{n_p} u_{n_p}$

Alle Produktionen, bei denen X rechts vorkommt

## Beispiel

- Sei wieder
  - $G_2 = (\{S, E, E', T, T', F\}, \{+, *, (, ), \text{id}\}, P, S)$  mit
  - $P = \{S \rightarrow E, E \rightarrow TE', E' \rightarrow +E, E' \rightarrow \varepsilon, T \rightarrow FT', T' \rightarrow *T, T' \rightarrow \varepsilon, F \rightarrow (E), F \rightarrow \text{id}\}$
- Man erhält (als rekursives Gleichungssystem)
  - $Fo_k(S) = \{\#\}$
  - $Fo_k(E) = Fo_k(S) \cup Fo_k(E') \cup (\{\}) \oplus_k Fo_k(F)$
  - $Fo_k(E') = Fo_k(E)$
  - $Fo_k(T) = Fo_k(T') \cup (Fi_k(E') \oplus_k Fo_k(E))$
  - $Fo_k(T') = Fo_k(T)$
  - $Fo_k(F) = Fi_k(T') \oplus_k Fo_k(T)$



## Fragestellungen (bezüglich der Gleichungssysteme für FIRST und FOLLOW)

- Gibt es überhaupt Lösungen ?
- Falls mehrere Lösungen, welche ?
- Wie berechnet man Lösungen ?

## Antworten

- Gibt *Verbandstheorie*
- Voraussetzungen
  - Endliche „Bereiche“ mit partiellen Ordnungen (Teilmengen von  $V_T^{\leq k}$ , per Inklusion geordnet)
  - Kleinste / größte Elemente  $\perp$  bzw.  $\top$  ( $\emptyset$  und  $V_T^{\leq k}$ )
  - Kleinste obere / größte untere Schranke für je 2 Elemente ( $\cup$  und  $\cap$ )
  - Monotonie der beteiligten Funktionen ( $\cup$  und  $\cap$  monoton bzgl. Inklusion)
- Unter diesen Voraussetzungen: *Fixpunktsätze*
  - Knaster-Tarski: Kriterien für die Existenz von Fixpunkten
  - Kleene: Algorithmus zur Berechnung des kleinsten Fixpunkts

## Basis für weiteres Vorgehen

d.h. nur Einzelzeichen oder  $\varepsilon$

- Für Sprachen  $L_1, L_2 \subseteq V^{\leq 1}$  gilt (gemäß Definitionen  $V^{\leq 1}$  und  $\oplus_1$ )
  - $L_1 \oplus_1 L_2 =$ 
    - $\emptyset$ , falls  $L_1 = \emptyset \vee L_2 = \emptyset$
    - $L_1$ , falls  $L_2 \neq \emptyset \wedge \varepsilon \notin L_1$
    - $(L_1 \setminus \{\varepsilon\}) \cup L_2$ , falls  $L_2 \neq \emptyset \wedge \varepsilon \in L_1$

## Beispiel

- Seien  $L_1 = \{a, b\}$ ,  $L_2 = \{a, b, \varepsilon\}$ ,  $L_3 = \{c, d\}$ ; dann gilt
  - $L_1 \oplus_1 L_3 = \{a, b\} = L_1$
  - $L_2 \oplus_1 L_3 = \{a, b, c, d\} = (L_2 \setminus \{\varepsilon\}) \cup L_3$

## Neue Form für FIRST<sub>1</sub>

- Aufspaltung der Berechnung von FIRST<sub>1</sub>( $\alpha$ ) in 2 Phasen
  - Bestimmung, ob  $\alpha \varepsilon$  produziert (mit **eps** )
  - Berechnung des  $\varepsilon$ -freien Rests (mit  **$\varepsilon$ -ffi** )
- FIRST<sub>1</sub>( $\alpha$ ) =<sub>def</sub>
  - $\varepsilon\text{-ffi}(\alpha) \cup \{\varepsilon\}$ , falls **eps**( $\alpha$ )
  - $\varepsilon\text{-ffi}(\alpha)$ , sonst

## 1. Phase

- $\epsilon$ -produktiv
  - Ein Nichtterminal  $X$  ist  **$\epsilon$ -produktiv**, wenn die Produktionen für  $X$  von einer der beiden Formen sind
    - $X \rightarrow \epsilon$
    - $X \rightarrow Y_1 \dots Y_n, Y_i \in V_N$  und alle  $Y_i$   $\epsilon$ -produktiv
- Rekursives Gleichungssystem (für  $\alpha \in V^*$  und alle  $X \in V_N$ )
  - $\text{eps}(\epsilon) =_{\text{def}} \mathbf{true}$
  - $\text{eps}(a) =_{\text{def}} \mathbf{false}$ , falls  $a \in V_T$
  - $\text{eps}(X) =_{\text{def}} \mathbf{true}$ , falls  $X \rightarrow \epsilon$
  - $\text{eps}(X) =_{\text{def}} \bigvee_{\{p: p[0]=X\}} \text{eps}(X_1) \wedge \dots \wedge \text{eps}(X_{n_p})$ , sonst

*Falls  $X$  mehrere Alternativen hat, müssen die jeweiligen Fälle mit  $\vee$  verknüpft werden*

## Beispiel

- Sei wieder
  - $G_2 = (\{S, E, E', T, T', F\}, \{+, *, (, ), \mathbf{id}\}, P, S)$  mit
  - $P = \{S \rightarrow E, E \rightarrow TE', E' \rightarrow +E, E' \rightarrow \epsilon, T \rightarrow FT', T' \rightarrow *T, T' \rightarrow \epsilon, F \rightarrow (E), F \rightarrow \mathbf{id}\}$
- Man erhält
  - $\text{eps}(S) = \text{eps}(E) = \text{eps}(T) = \text{eps}(F) = \mathbf{false}$
  - $\text{eps}(E') = \text{eps}(T') = \mathbf{true}$

## 2. Phase

- $\epsilon$ -freie FIRST-Funktion  **$\epsilon$ -ffi** mit  $\epsilon\text{-ffi}(\alpha) =_{\text{def}} \text{FIRST}_1(\alpha) \setminus \{\epsilon\}$ , d.h.
  - $\epsilon\text{-ffi}(\alpha) =_{\text{def}} \epsilon\text{-ffi}(1: \alpha)$ , für  $\alpha \in (V_N \cup V_T)^+$  mit  $\text{eps}(\alpha) = \text{false}$
  - $\epsilon\text{-ffi}(a) =_{\text{def}} \{a\}$ , falls  $a \in V_T$
  - $\epsilon\text{-ffi}(X) =_{\text{def}} \bigcup_{Y \in V_N \cup V_T} \{\epsilon\text{-ffi}(Y) \mid X \rightarrow \alpha Y \beta \in P \wedge \text{eps}(\alpha)\}$ , falls  $X \in V_N$

Damit gilt  
 $\epsilon\text{-ffi}(\epsilon) = \emptyset$

## Beispiel

- Sei wieder
  - $G_2 = (\{S, E, E', T, T', F\}, \{+, *, (, ), \text{id}\}, P, S)$  mit
  - $P = \{S \rightarrow E, E \rightarrow TE', E' \rightarrow +E, E' \rightarrow \epsilon, T \rightarrow FT', T' \rightarrow *T, T' \rightarrow \epsilon, F \rightarrow (E), F \rightarrow \text{id}\}$
- Es war
  - $\text{eps}(E') = \text{eps}(T') = \text{true}$   
 $\text{eps}(S) = \text{eps}(E) = \text{eps}(T) = \text{eps}(F) = \text{false}$
- Man erhält
  - $\epsilon\text{-ffi}(S) = \epsilon\text{-ffi}(E) = \epsilon\text{-ffi}(T) = \epsilon\text{-ffi}(F) = \{(, \text{id}\}$   
 $\epsilon\text{-ffi}(E') = \{+\}, \epsilon\text{-ffi}(T') = \{*\}$
- Damit
  - $\text{FIRST}_1(S) = \text{FIRST}_1(E) = \text{FIRST}_1(T) = \text{FIRST}_1(F) = \{(, \text{id}\}$   
 $\text{FIRST}_1(E') = \{+, \epsilon\}, \text{FIRST}_1(T') = \{*, \epsilon\}$

## FOLLOW<sub>1</sub>

- Zu FIRST analoge Überlegungen ergeben
  - FOLLOW<sub>1</sub>(S) =<sub>def</sub> {#}
  - FOLLOW<sub>1</sub>(X) =<sub>def</sub>  $\bigcup_{Y \in V_N} \{\varepsilon\text{-ffi}(\beta) \mid Y \rightarrow \alpha X \beta \in P\} \cup \bigcup_{Y \in V_N} \{\text{FOLLOW}_1(Y) \mid Y \rightarrow \alpha X \beta \in P \wedge \text{eps}(\beta)\}$

## Beispiel

- Sei wieder
  - $G_2 = (\{S, E, E', T, T', F\}, \{+, *, (, ), \text{id}\}, P, S)$  mit
  - $P = \{S \rightarrow E, E \rightarrow TE', E' \rightarrow +E, E' \rightarrow \varepsilon, T \rightarrow FT', T' \rightarrow *T, T' \rightarrow \varepsilon, F \rightarrow (E), F \rightarrow \text{id}\}$
- Es war
  - $\text{eps}(E') = \text{eps}(T') = \text{true}$ ,  $\text{eps}(S) = \text{eps}(E) = \text{eps}(T) = \text{eps}(F) = \text{false}$
  - $\varepsilon\text{-ffi}(S) = \varepsilon\text{-ffi}(E) = \varepsilon\text{-ffi}(T) = \varepsilon\text{-ffi}(F) = \{(, \text{id}\}$ ,  $\varepsilon\text{-ffi}(E') = \{+\}$ ,  $\varepsilon\text{-ffi}(T') = \{*\}$
- Man erhält
  - FOLLOW<sub>1</sub>(S) = {#}
  - FOLLOW<sub>1</sub>(E') = FOLLOW<sub>1</sub>(E) = {}  $\cup$  FOLLOW<sub>1</sub>(S)  $\cup$  FOLLOW<sub>1</sub>(E') = {}, #}
  - FOLLOW<sub>1</sub>(T') = FOLLOW<sub>1</sub>(T) =  $\varepsilon\text{-ffi}(E') \cup$  FOLLOW<sub>1</sub>(E)  $\cup$  FOLLOW<sub>1</sub>(T') = {+}  $\cup$  {}, #} = {+, ), #}
  - FOLLOW<sub>1</sub>(F) =  $\varepsilon\text{-ffi}(T') \cup$  FOLLOW<sub>1</sub>(T) = {\*}  $\cup$  {+, ), #} = {\*, +, ), #}