Syntaktische Analyse – Teil 1

Übersicht

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 $FIRST_1$- und $FOLLOW_1$-Mengen berechnet

Syntaktische Analyse – allgemein

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 (1/5)

Kontextfreie Grammatik (kurz: kfG)

- $G = (V_N, V_T, P, S)$
  - V_N: endliche Menge von **Nichtterminalen**
  - V_T: endliche Menge von **Terminalen** (wobei $V_N \cap V_T = \varnothing$)
  - $P \subseteq V_N \times (V_N \cup V_T)^*$: 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
  - α, β, χ: beliebige Worte
- Darstellung von Produktionsregeln
  - $A \rightarrow \alpha$ für (A, α)
  - $A \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3$ für $(A, \alpha_1), (A, \alpha_2), (A, \alpha_3)$

Kontextfreie Grammatik (2/5)

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}, {+,*,(,),**id**}, {E → E+T | T, T → T*F | F, F → (E) | **id**}, E)
  - G_1 = ({E}, {+, *,(,),**id**}, {E → E+E | E*E | (E) | **id**}, E)

Kontextfreie Grammatik (3/5)

Wichtige Begriffe

- Sei $G = (V_N, V_T, P, S)$ kfG, $V = V_N \cup V_T$
- Produziert direkt
  - φ **produziert direkt** ψ **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 φ wird durch entsprechende rechte Seite ersetzt

- Produziert
  - φ **produziert** ψ **gemäß G**, i.Z. $\varphi \Rightarrow_G^* \psi$, (bzw. $\varphi \Rightarrow^* \psi$, falls G klar ist), oder ψ **ist** aus φ **gemäß G ableitbar**, wenn es $\varphi_0, \varphi_1, ..., \varphi_n \in V^*$ $(n \geq 0)$ 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 ψ aus φ ableitbar (mit φ_i, $0 \leq i \leq n$, wie oben), dann heißt $\varphi_0, \varphi_1, ..., \varphi_n$ **Ableitung** von ψ aus φ

Beispiel

- Sei
  - $G_0 = (\{E,T,F\}, \{+, *, (,), \mathbf{id}\}, \{E \rightarrow E+T \mid T,\ T \rightarrow T*F \mid F,\ F \rightarrow (E) \mid \mathbf{id}\}, E)$
  - $G_1 = (\{E\}, \{+, *, (,), \mathbf{id}\}, \{E \rightarrow E+E \mid E*E \mid (E) \mid \mathbf{id}\}, E)$
- Dann gilt
  - $E \Rightarrow_{G_1} E+E \Rightarrow_{G_1} E*E+E \Rightarrow_{G_1} \mathbf{id}*E+E \Rightarrow_{G_1} \mathbf{id}*E+\mathbf{id} \Rightarrow_{G_1} \mathbf{id}*\mathbf{id}+\mathbf{id}$
  - $E \Rightarrow_{G_0} E+T \Rightarrow_{G_0} T+T \Rightarrow_{G_0} T*F+T \Rightarrow_{G_0} T*\mathbf{id}+T \Rightarrow_{G_0} F*\mathbf{id}+T \Rightarrow_{G_0} F*\mathbf{id}+F \Rightarrow_{G_0} \mathbf{id}*\mathbf{id}+F \Rightarrow_{G_0} \mathbf{id}*\mathbf{id}+\mathbf{id}$

Kontextfreie Grammatik (4/5)

Weitere Begriffe

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

Beispiele

- Sätze
  - **id*****id**+**id** $\in L(G_0)$
  - **id*****id**+**id** $\in L(G_1)$
- Satzformen
  - Von G_0: E, E+T, T***id**+T, ...
  - Von G_1: E, E+E, **id***E+E, ...

Eigenschaft

Alternative Definition für „kontextfrei“

- Für beliebige kfG $G = (V_N, V_T, P, S)$ und (beliebige) $\varphi, \psi \in V^*$ gilt:

$$\varphi = \varphi_1\varphi_2 \ldots \varphi_n \wedge \varphi \Rightarrow^* \psi \Leftrightarrow \exists\, \psi_1, \psi_2, \ldots, \psi_n: \psi = \psi_1\psi_2 \ldots \psi_n \wedge \forall_{i=1..n}\, \varphi_i \Rightarrow^* \psi_i$$

Kontextfreie Grammatik (5/5)

Spezielle Nichtterminale

- Unproduktives und unerreichbares Nichtterminal A
  - **Unerreichbar**: Es gibt keine Worte α, β 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

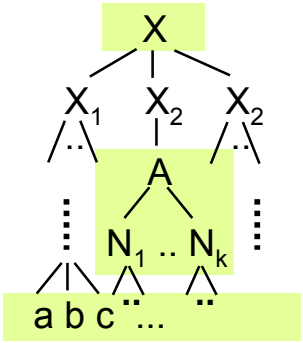
Beispiel

- Geg.: G'_1 = ({E, X, Y}, {+, *, (,), **id**}, {E → E+E | E*E | (E) | **id** | X, Y → (**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

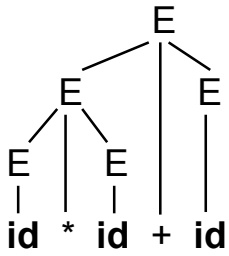
Syntaktische Struktur eines Programms (1/4)

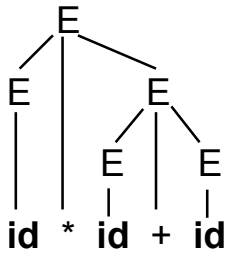
Syntaxbaum

- Seien
  - $G = (V_N, V_T, P, S)$ kfG, $V = V_N \cup V_T$
  - B = geordneter Baum: Blattmarkierungen $\in V_T \cup \{\varepsilon\}$; (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 $N_1, N_2, ..., N_k \in V$ und $A \rightarrow N_1N_2 ... N_k \in P$
    - Einziges Kind ist markiert mit ε und $A \rightarrow \varepsilon \in P$
- B heißt **Syntaxbaum** für $x \in V_T^*$, falls X = S

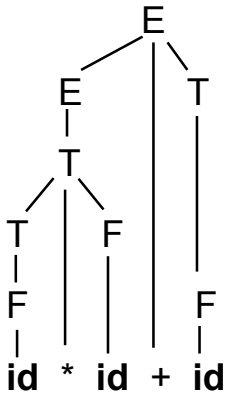


Beispiel (Syntaxbäume für **id * id + id**)





Syntaxbäume gemäß G_1



Syntaxbaum gemäß G_0

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

Syntaktische Struktur eines Programms (2/4)

Weitere Begriffe

Bemerkungen und Definitionen für Sätze gelten sinngemäß auch für Satzformen

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

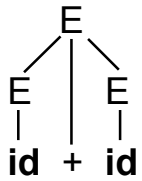
Beispiele

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

Syntaktische Struktur eines Programms (3/4)

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 ⇒ E+E ⇒ **id**+E ⇒ **id**+**id**
    - E ⇒ E+E ⇒ E+**id** ⇒ **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

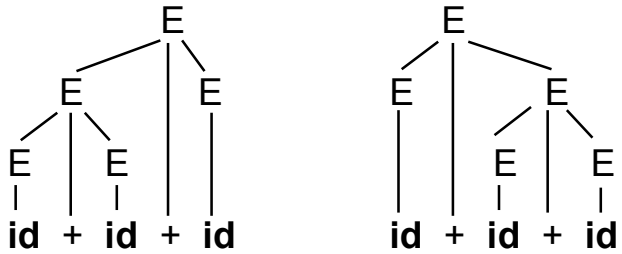
Syntaktische Struktur eines Programms (4/4)

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



Kanonische Ableitungen (1/3)

Kanonische Ableitungen und Satzformen

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

Kanonische Ableitungen (2/3)

Weitere Bemerkungen

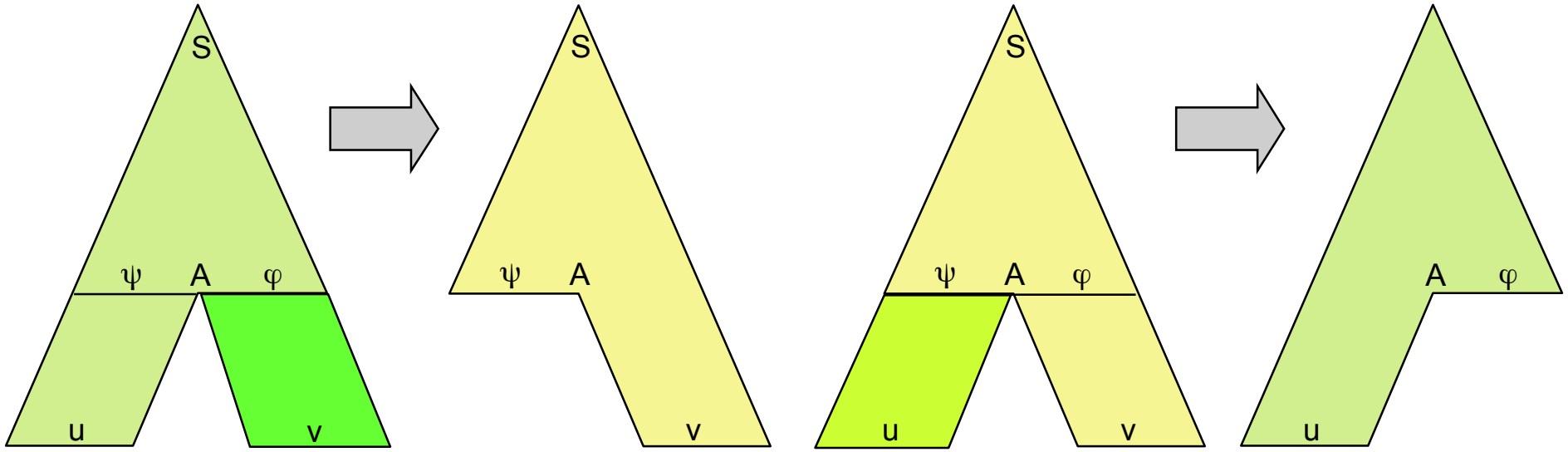
- 5. Zu jedem Satz gibt es mindestens eine Links- und eine Rechtsableitung
  - Beispiel: betrachtet kfG G_1 und **id** * **id** + **id**
  - Linksableitungen
    - $E \Rightarrow_{lm} E+E \Rightarrow_{lm} E*E+E \Rightarrow_{lm} \mathbf{id}*E+E \Rightarrow_{lm} \mathbf{id}*\mathbf{id}+E \Rightarrow_{lm}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

Es galt:
Eindeutig = 1 Syntaxbaum

Kanonische Ableitungen (3/3)

Zusammenhang zwischen Links- und Rechtsableitungen

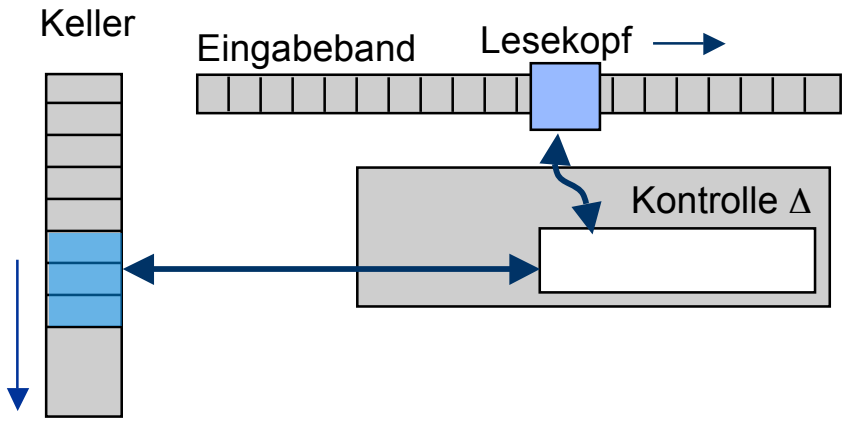
- Zu jeder Linksableitung gibt es eine korrespondierende Rechtsableitung und umgekehrt
  - Wenn $S \Rightarrow^*_{lm} uA\varphi$, dann gibt es ψ, mit $\psi \Rightarrow^* u$, so dass für alle v mit $\varphi \Rightarrow^* v$ gilt: $S \Rightarrow^*_{rm} \psi Av$
  - Wenn $S \Rightarrow^*_{rm} \psi Av$,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



Kellerautomat (1/3)

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 (2/3)

Kellerautomat (formal)

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

    (dargestellt als endliche partielle Funktion δ in die endlichen Teilmengen von Q*)

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

Intuitiv: Zu jedem Kellerinhalt kann es mehrere Folgeinhalte gemäß δ geben

Charakterisierung

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

δ *ist Relation*

ε *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})

Kellerautomat (3/3)

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, ε) mit $q \in F$

Nichtleere Zustandsfolge im Keller, Resteingabe

- **Übergangsrelationen** zwischen Konfigurationen ($\gamma_1, \gamma_3 \in Q^*$, $\gamma_2 \in Q^+$, $a \in V \cup \{\varepsilon\}$)
  - $\vdash_P$: $(\gamma_1\gamma_2, av) \vdash_P (\gamma_1\gamma_3, v)$, 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 $q \in F$
  - $L(P) =_{def} \{w \in V^* \mid w \text{ wird von P akzeptiert}\}$

***Beachte**:*
- für δ 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

Deterministischer Kellerautomat

- Sei P = (V, Q, δ, 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 γ' echtes Suffix von γ ist
    - $\delta(\gamma', \varepsilon)$ undefiniert, falls γ' Suffix von γ ist

Mit anderen Worten:
δ ist partielle Funktion

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

Beispiel

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

Item-Kellerautomat (1/3)

Kontextfreies Item

- Sei G = (V_N, V_T, P, S) kfG, A → αβ ∈ P
  - **Item**: [A → α.β]
  - **Vollständiges Item**: [A → αβ.]
  - **Start-Item**: [A → .αβ]

Item-Kellerautomat (intuitiv)

Immer erreichbar durch Hinzufügen von S‘ → .S für S‘∉V_N

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

Item-Kellerautomat einer kfG

- Sei G = (V_N, V_T, P, S) kfG, S‘∉V_N
- **Item-Kellerautomat K_G**: Kellerautomat (V_T, It_G, δ, [S‘ → .S], {[S‘ → S.]})
- Übergänge von δ:
  - (E) δ([X → β.Yγ], ε) = {[X → β.Yγ] [Y →.α] | Y → α ∈ P} Expansion
  - (L) δ([X → β.aγ], a) = {[X → βa.γ]} Lesen
  - (R) δ([X → β.Yγ] [Y → α.], ε) = {[X → βY.γ]} Reduktion
- Es gilt: L(G) = L(K_G)

Ist i.a. nicht-deterministisch

Item-Kellerautomat (2/3)

Beispiel: Geg.: G_0' = ({S,E,T,F}, {+, *,(,),**id**}, {S → E, E → E+T | T, T →T*F | F, F→ (E) | **id**}, S)

Übergangsrelation δ 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.)]

Item-Kellerautomat (3/3)

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

Kellerautomat mit Ausgabe

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
  - δ: 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 $It_G^+ \times V_T^* \times P^*$

Kellerautomat mit Ausgabe (a)

Linksparser

- Sei $G = (V_N, V_T, P, S)$ kfG
- **Linksparser** (entspricht Linksableitung)
  - Item-Kellerautomat mit Ausgabe $K^l_G = (V_T, It_G, P, \delta_l, [S' \rightarrow .S], \{[S' \rightarrow S.]\})$
  - (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
  - $(\rho[X \rightarrow \beta.Y\gamma], w, o) \vdash_{K^l_G} (\rho[X \rightarrow \beta.Y\gamma]\ [Y \rightarrow .\alpha], w, o(Y \rightarrow \alpha))$

Rechtsparser

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

- Sei $G = (V_N, V_T, P, S)$ kfG
- **Rechtsparser** (entspricht Rechtsableitung)
  - Item-Kellerautomat mit Ausgabe $K^r_G = (V_T, It_G, P, \delta_r, [S' \rightarrow .S], \{[S' \rightarrow S.]\})$
  - (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
  - $(\rho[X \rightarrow \beta.Y\gamma]\ [Y \rightarrow \alpha.], w, o) \vdash_{K^r_G} (\rho[X \rightarrow \beta Y.\gamma], w, o(Y \rightarrow \alpha))$

FIRST und FOLLOW (1/3)

Ziel

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

Hilfsdefinitionen

- Sei V Alphabet, $w = a_1...a_n$, $a_i \in V$; außerdem: $V^{\leq k} =_{def} \bigcup_{i=0..k} V^i$; $V_{\#}^{\leq k} =_{def} V^{\leq k} \cup V^{\leq k-1}\{\#\}$ / *Worte über V mit Länge ≤ k* / *# ∉ V* \
- **k-Präfix** von w: $k{:}\ w =_{def} a_1...a_{min(n,k)}$ — Somit: $k{:}\ \varepsilon = \varepsilon$
- **k-Konkatenation** (auf Wörtern): $\oplus_k =_{def} V^* \times V^* \to V^{\leq k}$ definiert durch: $u \oplus_k v =_{def} k{:}\ uv$
- **k-Konkatenation** (auf Wortmengen): $L_1 \oplus_k L_2 =_{def} \{u \oplus_k v \mid u \in L_1, v \in L_2\}$ 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 = cbbc
  - $L_1 \oplus_5 L_2$ = {aabcb, abbcb, abbbc, cbcbb, cbbc}

FIRST und FOLLOW (2/3)

$FIRST_k$ und $FOLLOW_k$

- Sei $G = (V_N, V_T, P, S)$ kfG, $V = V_N \cup V_T$
- **$FIRST_k$** (auf Worten): $FIRST_k: V^* \to \mathbb{P}(V_T^{\leq k})$ mit $FIRST_k(\alpha) =_{def} \{ k: u \mid \alpha \Rightarrow^* u \}$
- **$FOLLOW_k$** (auf Worten): $FOLLOW_k : V^* \to \mathbb{P}(V_{T\#}^{\leq k})$ mit
  $FOLLOW_k(\alpha) =_{def} \{w \mid S \Rightarrow^* \beta\alpha\gamma \text{ und } w \in FIRST_k(\gamma)\}$
- $FIRST_k$ (auf Wortmengen):
  $FIRST_k(L) =_{def} \bigcup_{\alpha \in L} FIRST_k(\alpha)$ für $L \subseteq V^*$

Mengen von Worten über V_T mit Länge ≤ k

Somit: $FIRST_k(\varepsilon) = \{\varepsilon\}$

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

Beispiel

- Für $P = \{S \to AcBd, A \to a \mid aA \mid \varepsilon, B \to b \mid bB \mid \varepsilon\}$ gilt:
  - $FIRST_2(A) = \{a, aa, \varepsilon\}$
  - $FOLLOW_3(A) = \{cbb, cbd, cd\}$

Wichtige Eigenschaften

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

FIRST und FOLLOW (3/3)

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 ($0 \le i \le n_p$)
- p[0] linke Seite von p
- (p, i) Vorkommen eines Nichtterminals in p an Position i

Beispiel

- Sei p = E → 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)

FIRST (1/2)

Herleitung der Berechnung von FIRST

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

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

$= [$ für $X \rightarrow u_0X_1u_1X_2 \ldots X_{np}u_{np}$ mit $n_p \geq 1]$

$\cup_{\{p:\ p[0]=X\}} FIRST_k(u_0X_1u_1X_2 \ldots X_{np}u_{np})$

$= [$ Zerlegungseigenschaft von $FIRST_k]$

$\cup_{\{p:\ p[0]=X\}} FIRST_k(u_0) \oplus_k FIRST_k(X_1) \oplus_k FIRST_k(u_1) \oplus_k \ldots \oplus_k FIRST_k(X_{np}) \oplus_k FIRST_k(u_{np})$

$= [$Eigenschaft von $FIRST_k$ für Terminalzeichenreihen$]$

$\cup_{\{p:\ p[0]=X\}} \{u_0\} \oplus_k FIRST_k(X_1) \oplus_k \{u_1\} \oplus_k \ldots \oplus_k FIRST_k(X_{np}) \oplus_k \{u_{np}\}$

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

Rekursives Gleichungssystem für $FIRST_k$

- $Fi_k(X) =_{def} \cup_{\{p:\ p[0]=X\}} \{u_0\} \oplus_k Fi_k(X_1) \oplus_k \{u_1\} \oplus_k \ldots \oplus_k Fi_k(X_{np}) \oplus_k \{u_{np}\}$ für alle $X \in V_N$
  falls $p = X \rightarrow u_0X_1u_1X_2 \ldots X_{np}u_{np}$ mit $n_p \geq 1$
- $Fi_k(X) =_{def} \cup_{\{p:\ p[0]=X\}} \{k: u\}$, falls $p = X \rightarrow u$

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

Offensichtlicher Spezialfall

FIRST (2/2)

Beispiel

- Sei
  - G_2 = ({S,E,E‘,T,T‘,F}, {+,*,(,),**id**}, P, S) mit
  - P = {S → E, E → TE‘, E‘ → +E, E‘ → ε, T → FT‘, T‘ → *T, T‘ → ε, F → (E), F → **id**}
  - Wobei gilt: $L(G_2) = L(G_0) = L(G_1)$
- Es war
  - $Fi_k(X) =_{def} \bigcup_{\{p:\ p[0]=X\}} \{u_0\} \oplus_k Fi_k(X_1) \oplus_k \{u_1\} \oplus_k ... \oplus_k Fi_k(X_{n_p}) \oplus_k \{u_{n_p}\}$
    falls $p = X \rightarrow u_0X_1u_1X_2 ... X_{n_p}u_{n_p}$
  - $Fi_k(X) =_{def} \bigcup_{\{p:\ p[0]=X\}} \{k: u\}$, falls $p = X \rightarrow u$
- Man erhält (als rekursives Gleichungssystem)
  - $Fi_k(S) = Fi_k(E)$
  - $Fi_k(E) = Fi_k(T) \oplus_k Fi_k(E‘)$
  - $Fi_k(E‘) = (\{+\} \oplus_k Fi_k(E)) \cup \{\varepsilon\}$
  - $Fi_k(T) = Fi_k(F) \oplus_k Fi_k(T‘)$
  - $Fi_k(T‘) = (\{*\} \oplus_k Fi_k(T)) \cup \{\varepsilon\}$
  - $Fi_k(F) = (\{(\} \oplus_k Fi_k(E) \oplus_k \{)\}) \cup \{$**id**$\}$

FOLLOW

Rekursives Gleichungssystem für $FOLLOW_k$

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

Alle Produktionen, bei denen X rechts vorkommt

Beispiel

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

Rekursive Gleichungssysteme

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. T ($\varnothing$ 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

$FIRST_1$ und $FOLLOW_1$ (1/4)

Basis für weiteres Vorgehen

d.h. nur Einzelzeichen oder ε

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

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 $FIRST_1$

- Aufspaltung der Berechnung von $FIRST_1(\alpha)$ in 2 Phasen
  - Bestimmung, ob α ε produziert (mit eps ☛)
  - Berechnung des ε-freien Rests (mit ε-ffi ☛)
- $FIRST_1(\alpha) =_{def}$
  - ε-ffi$(\alpha) \cup \{\varepsilon\}$, falls eps$(\alpha)$
  - ε-ffi(α), sonst

FIRST$_1$ und FOLLOW$_1$ (2/4)

1. Phase

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

Falls X mehrere Alternativen hat, müssen die jeweiligen Fälle mit ∨ verknüpft werden

Beispiel

- Sei wieder
  - G_2 = ({S,E,E‘,T,T‘,F}, {+,*,(,),**id**}, P, S) mit
  - P = {S → E, E → TE‘, E‘ → +E, E‘ → ε, T → FT‘, T‘ → *T, T‘ → ε, F → (E), F → **id**}
- Man erhält
  - eps(S) = eps(E) = eps(T) = eps(F) = **false**
    eps(E‘) = eps(T‘) = **true**

$FIRST_1$ und $FOLLOW_1$ (3/4)

2. Phase

- ε-freie FIRST-Funktion **ε-ffi** mit $\varepsilon\text{-ffi}(\alpha) =_{def} FIRST_1(\alpha) \setminus \{\varepsilon\}$, d.h.
  - $\varepsilon\text{-ffi}(\alpha) =_{def} \varepsilon\text{-ffi}(1{:}\ \alpha)$, für $\alpha \in (V_N \cup V_T)^+$ mit $eps(\alpha) = false$
  - $\varepsilon\text{-ffi}(a) =_{def} \{a\}$, falls $a \in V_T$
  - $\varepsilon\text{-ffi}(X) =_{def} \bigcup_{Y \in V_N \cup V_T} \{\varepsilon\text{-ffi}(Y) \mid X \rightarrow \alpha Y \beta \in P \wedge eps(\alpha)\}$, falls $X \in V_N$

Damit gilt
$\varepsilon\text{-ffi}(\varepsilon) = \varnothing$

Beispiel

- Sei wieder
  - $G_2 = (\{S,E,E',T,T',F\}, \{+,*,(,),\mathbf{id}\}, P, S)$ mit
  - $P = \{S \rightarrow E,\ E \rightarrow TE',\ E' \rightarrow +E,\ E' \rightarrow \varepsilon,\ T \rightarrow FT',\ T' \rightarrow *T,\ T' \rightarrow \varepsilon,\ F \rightarrow (E),\ F \rightarrow \mathbf{id}\}$
- Es war
  - $eps(E') = eps(T') = \mathbf{true}$
    $eps(S) = eps(E) = eps(T) = eps(F) = \mathbf{false}$
- Man erhält
  - $\varepsilon\text{-ffi}(S) = \varepsilon\text{-ffi}(E) = \varepsilon\text{-ffi}(T) = \varepsilon\text{-ffi}(F) = \{(, \mathbf{id}\}$
    $\varepsilon\text{-ffi}(E') = \{+\}$, $\varepsilon\text{-ffi}(T') = \{*\}$
- Damit
  - $FIRST_1(S) = FIRST_1(E) = FIRST_1(T) = FIRST_1(F) = \{(, \mathbf{id}\}$
    $FIRST_1(E') = \{+, \varepsilon\}$, $FIRST_1(T') = \{*, \varepsilon\}$

$FIRST_1$ und $FOLLOW_1$ (4/4)

$FOLLOW_1$

- Zu FIRST analoge Überlegungen ergeben
  - $FOLLOW_1(S) =_{def} \{\#\}$
  - $FOLLOW_1(X) =_{def} \bigcup_{Y \in V_N}\{\varepsilon\text{-ffi}(\beta) \mid Y \rightarrow \alpha X\beta \in P\} \cup$
    $\bigcup_{Y \in V_N}\{FOLLOW_1(Y) \mid Y \rightarrow \alpha X\beta \in P \wedge eps(\beta)\}$

Beispiel

- Sei wieder
  - $G_2 = (\{S,E,E',T,T',F\}, \{+,*,(,),\mathbf{id}\}, P, S)$ mit
  - $P = \{S \rightarrow E,\ E \rightarrow TE',\ E' \rightarrow +E,\ E' \rightarrow \varepsilon,\ T \rightarrow FT',\ T' \rightarrow *T,\ T' \rightarrow \varepsilon,\ F \rightarrow (E),\ F \rightarrow \mathbf{id}\}$
- Es war
  - $eps(E') = eps(T') = \mathbf{true},\ eps(S) = eps(E) = eps(T) = eps(F) = \mathbf{false}$
  - $\varepsilon\text{-ffi}(S) = \varepsilon\text{-ffi}(E) = \varepsilon\text{-ffi}(T) = \varepsilon\text{-ffi}(F) = \{(, \mathbf{id}\},\ \varepsilon\text{-ffi}(E') = \{+\},\ \varepsilon\text{-ffi}(T') = \{*\}$
- Man erhält
  - $FOLLOW_1(S) = \{\#\}$
  - $FOLLOW_1(E') = FOLLOW_1(E) = \{)\} \cup FOLLOW_1(S) \cup FOLLOW_1(E') = \{), \#\}$
  - $FOLLOW_1(T') = FOLLOW_1(T) = \varepsilon\text{-ffi}(E') \cup FOLLOW_1(E) \cup FOLLOW_1(T') = \{+\} \cup \{),\#\} = \{+,),\#\}$
  - $FOLLOW_1(F) = \varepsilon\text{-ffi}(T') \cup FOLLOW_1(T) = \{*\} \cup \{+,), \#\} = \{*, +,), \#\}$