1.3 Wortproblem

Def

Gegeben eine Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in \Sigma^*$, dann ist das Wortproblem zu entscheiden, ob $w \in L(G)$

Notation:

$$T_m^n(G) = \{ w \in (V \cup \Sigma)^* \mid |w| \leq n \text{ und } w \text{ lässt sich in } m \text{ Schritten aus } S \text{ ableiten} \}$$

Für Typ 1 - Gram. ist das einfach zu berechne

$$T_0^n(G) = \{S\} \qquad n \geq 1$$

$$T_{m+1}^n(G) = T_m^n(G) \cup \{ w \mid |w| \leq n \text{ und es ex. } w' \in T_m^n(G) \text{ mit } w' \Rightarrow_G w \}$$

$G = (V, \Sigma, P, S)$

$\Sigma = \{a, b\}$

$V = (S, S')$

$P = \{ S \to \varepsilon,\ S \to S',\ S' \to ab,\ S' \to aS'b \}$

$T_0^4(G) = \{S\}$

$T_1^4(G) = \{ S, \varepsilon, S' \}$

$T_2^4(G) = \{ S, \varepsilon, S', ab, aS'b \}$

$T_3^4(G) = \{ S, \varepsilon, S', ab, aabb, \overbrace{aaS'bb}^{aS'b}, \cancel{aaS'bb} \}$

$T_4^4(G) = T_5^4(G) = \dots$

Satz

Es existiert ein Algorithmus, der das Wortproblem für Typ 1 Sprachen löst.

Beweis:

Gegeben eine Typ 1 Grammatik G und ein Wort w:

1. Für $n = |w|$ bestimme das m mit
   $T^n_m(G) = T^n_{m+n}(G)$ für alle dann gilt
   $\forall i \in \mathbb{N}: T^n_m(G) = T^n_{m+i}(G)$
2. Prüfe ob $w \in T^n_m(G)$. $\square$

1.4 Syntaxbäume

Jeder Ableitung in einer Typ?-Gram. kann man einen Syntaxbaum zuordnen. Für $x \in L(G)$ sei

$S = x_0 \Rightarrow \alpha_1 \Rightarrow \dots \Rightarrow x_n = x$ die Ableitung von α.

Der Wurzel des Baumes wird S zugeordnet.

Für $i = 1, \dots, n$: Falls im i-ten Schritt die Variable A durch das Wort z ersetzt wird ($A \to z$), dann erhält der A-Knoten entsprechende Kinder im Syntaxbäume ($|z|$ Knoten), die mit den Zeichen aus z beschriftet werden.

Bsp: $G = (\{E, T, F\}, \{(,), a, +, *\}, P, E)$

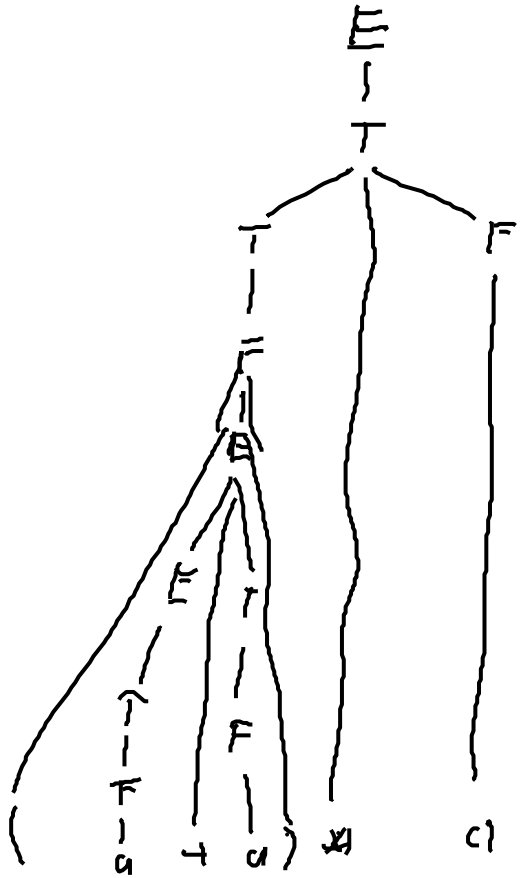
$P = \{E \to T,\ E \to E+T,\ T \to F,\ T \to T*F,\ F \to a,$

$F \to (E)\}$

$E \Rightarrow T \Rightarrow T*F \Rightarrow F*F \Rightarrow (E)*F \Rightarrow (E+T)*F \Rightarrow (T+T)*F$

$\Rightarrow (F+T)*F \Rightarrow (F+F)*F \Rightarrow (a+F)*F \Rightarrow (a+a)*F$

$\Rightarrow (a+a)*a$



Bem: Verschiedene Ableitungen können zu dem gleichen Syntaxbaum führen.

Bem: Man kann immer das am weitesten links stehende Symbol ersetzen.

⇒ Linksableitung

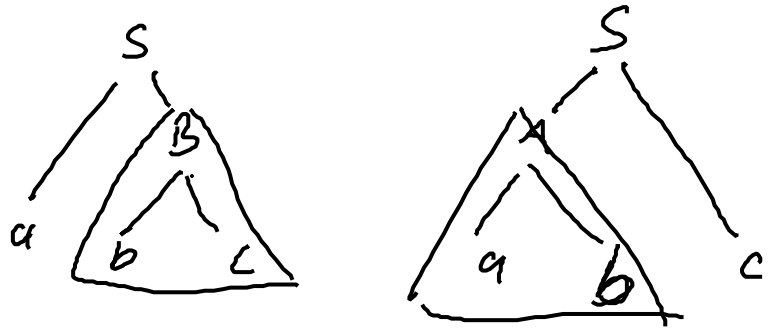
Entsprechend Rechtsableitung.

Bem: Zu einem gegebenen Wort kann es verschiedene Syntaxbäume geben

Bsp $P = \{ S \to aB,\ S \to Ac,\ A \to ab,\ B \to bc \}$

$S \Rightarrow aB \Rightarrow abc$

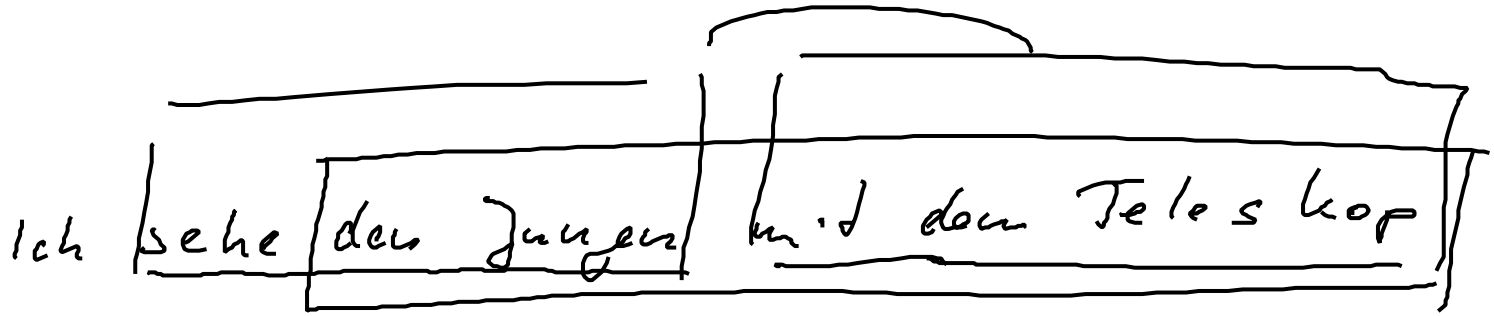
$S \Rightarrow Ac \Rightarrow abc$



$\Rightarrow$ Grammatik ist mehrdeutig

1. Syntaxbaum kann uns was über die Bedeutung sagen

2. Sprachen können inhärent mehrdeutig sein.



Def

Eine Sprache A heißt *inhärent mehrdeutig*, wenn keine Grammatik G mit $L(G) = A$ existiert, die nicht mehrdeutig ist.

Bsp $L = \{ a^i b^j c^k \mid i = j \text{ oder } j = k \}$

$S \to I \quad | \quad S \to K$

$I \to AC \quad | \quad A \to aAb \quad | \quad A \to ab$

$C \to cC \quad | \quad C \to c$

$K \to A'B'$

$A' \to aA' \quad | \quad A' \to a$

$B' \to bB'c \quad | \quad B' \to bc$

1. § BNF

Kompakte Beschreibung von Typ 2 - Spr., gem. für Programmiersprache (erstmals eingesetzt ALGOL 60):

Falls

$$A \to \alpha_1, \quad A \to \alpha_2, \quad \ldots, \quad A \to \alpha_n$$

dann schreibt man

$$A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$$

EBNF

Falls

$$A \to \alpha\gamma$$

$$A \to \alpha\beta\gamma$$

dann

$$A \to \alpha [\beta] \gamma$$

Falls

$$A \to \alpha\gamma$$

$$A \to \alpha B \gamma$$

$$B \to \beta$$

$$B \to \beta B$$

dann

$$A \to \alpha \{\beta\} \gamma$$

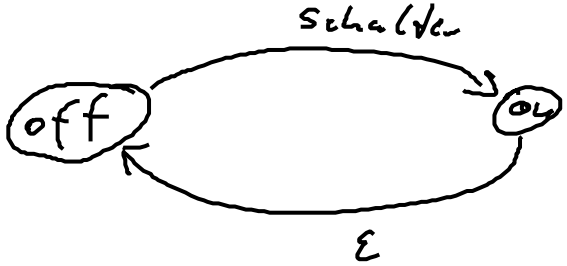
Bei BNF benutzt man "::=" statt "→"

1.6 Reguläre Sprachen (Typ 3)

1.6.1 Endliche Automaten

Endliche von Zuständen, endl. Eingabealphabet, endlich viele Zustandsübergänge

Bsp "Useless Machine"

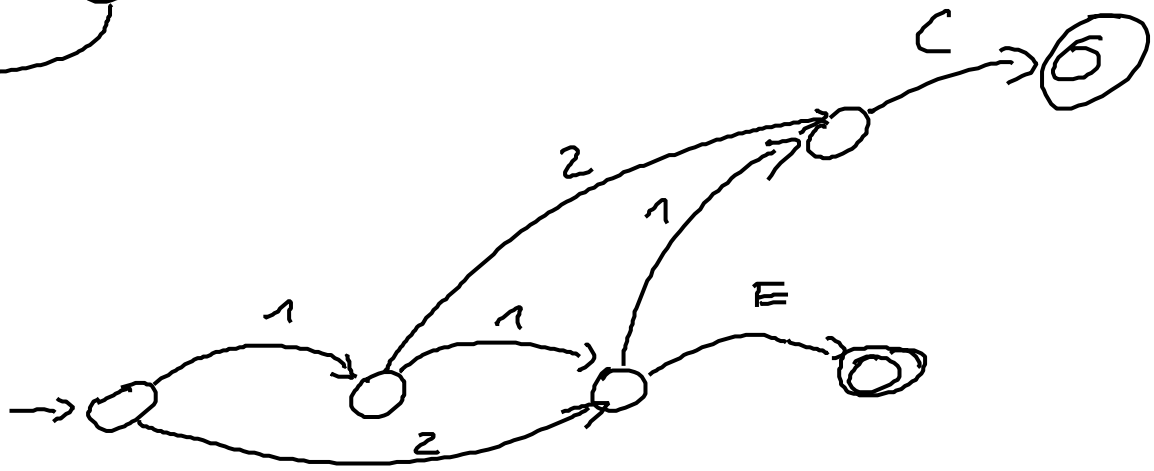


Bsp "Kaffeeautomat"

1 und 2 Euro-Stücke

Cappuccino: 3 Euro

Espresso : 2 Euro



Def

Ein deterministischer endlicher Automat (DEA, engl DFA)

ist ein 5-Tupel

$$M = (Z, \Sigma, \delta, z_0, E)$$

wobei:

- Z : endl. Zustandsmenge
- Σ : Eingabealphabet (endl.) mit $Z \cap \Sigma = \emptyset$
- $\delta : Z \times \Sigma \to Z$ Transitions- oder Überführungsfkt.
- $z_0 \in Z$ Anfangszustand
- $E \subseteq Z$ Endzustände

Bsp:

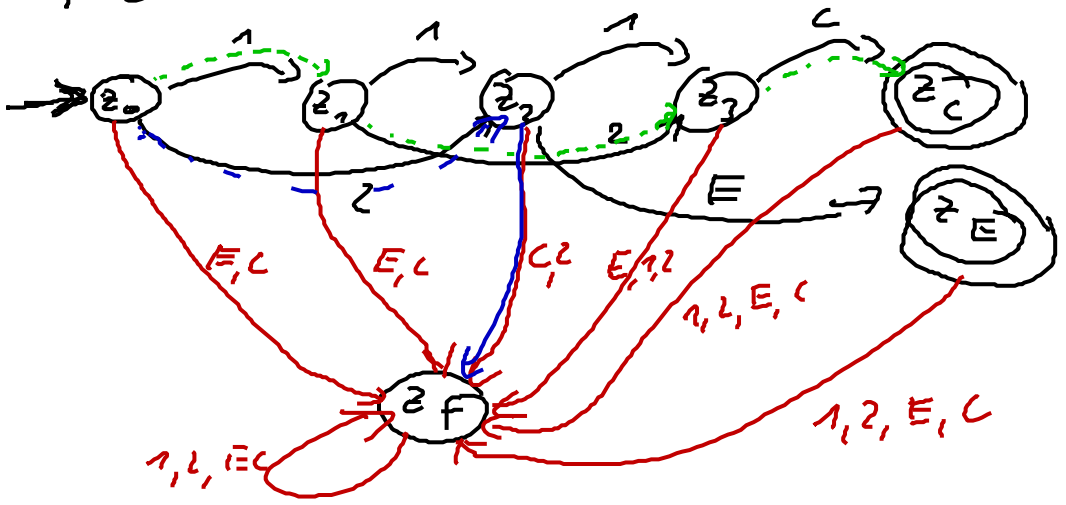
$Z = \{ z_0, z_1, z_2, z_3, z_E, z_C, z_f \}$

$\Sigma = \{ 1, 2, C, E \}$

z_0

$E = \{ z_E, z_C \}$

δ gegeben durch Zustandsgraphen:

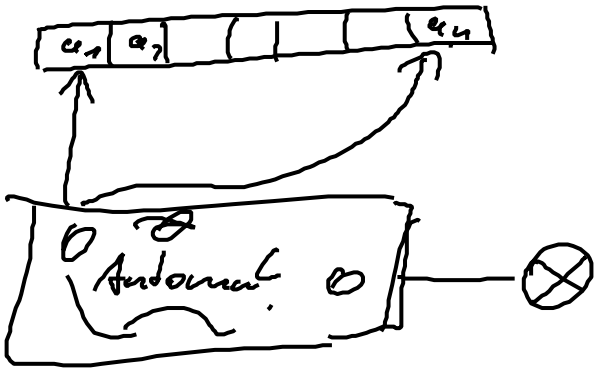


1 2 C

2 2 E

Automaten erkennen/akzeptieren Wörter

Eingabeband



wird ausgegeben, wenn wir letztes Zeichen lesen

Akzeptanz Signal

Def

Zu gegebenem DFA $M = (Z, \Sigma, \delta, z_0, E)$ def. wir

$\hat{\delta}: Z \times \Sigma^* \to Z$

$\hat{\delta}(z, \varepsilon) = z$

$\hat{\delta}(z, aw) = \hat{\delta}(\underbrace{\delta(z,a)}_{z'}, w)$ $\quad a \in \Sigma, w \in \Sigma^*$

Bsp

$$\hat{\delta}(z_0, 12c) = \hat{\delta}(\delta(z_0, 1), 2c)$$

$$= \hat{\delta}(z_1, 2c)$$

$$= \hat{\delta}(\delta(z_1, 2), c)$$

$$= \hat{\delta}(z_3, c)$$

$$= \hat{\delta}(\delta(z_3, c), \varepsilon)$$

$$= \hat{\delta}(z_c, \varepsilon) = z_c$$

Def

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA. Dann ist die von M akzeptierte Sprache:

$$T(M) = \{ w \in \Sigma^* \mid \hat{\delta}(z_0, w) \in E \}$$

$$\boxed{31_{Oct} = 25_{Dec}}$$