Head Grammars MHG

- Head Grammars (Pollard 1984)
- *wrapping* Operation
- *headed* Strings
- Regeln der Form $A \rightarrow f(\alpha_1, ..., \alpha_n)$
- α ist Nonterminal oder headed String

HG

- Regeln der Form $A \to f(\alpha_1, ..., \alpha_n)$
- α ist Nonterminal oder headed String
- $f \in \{LC_i, LL_i, LR_i, RC_i, RL_i, RR_i; i \geq 1\}$
- $LC1(u_1 a_1^* u_2, v_1 b_1^* v_2) = u_1 a_1^* u_2, v_1 b_1 v_2$
- $LC2(u_1 a_1^* u_2, v_1 b_1^* v_2) = u_1 a_1 u_2, v_1 b_1^* v_2$
- $LL2(u_1 a_1^* u_2, v_1 b_1^* v_2) = u_1 a_1 v_1 b_1^* v_2 u_2$
- Normalform: $LC_1, LC_2, LL_1, LL_2, LR_1, LR_2$

HG: Beispiel

- $S \to LL2(S_1, f^*)$
- $S \to LL2(S_2, f^*)$
- $S_1 \to LC2(a^*, S_3, d)$
- $S_2 \to g^*h$
- $S_3 \to LL2(S_1, b^*c)$
- $S_3 \to LL2(S_2, b^*c)$
- Erzeugte Sprache:
- $L_1 = \{a^n g b^n f c^n h d^n | n \geq 0\}$

Modified Head Grammars

- (Vijay-Shanker 1988)

- *split* Strings

- Spaltstelle markiert durch $\uparrow$: $w_1 \uparrow w_2$

- Drei Operationen

- $W(w_1 \uparrow w_2, u_1 \uparrow u_2) = w_1 u_1 \uparrow u_2 w_2$

- $C1(w_1 \uparrow w_2, u_1 \uparrow u_2) = w_1 \uparrow w_2 u_1 u_2$

- $C2(w_1 \uparrow w_2, u_1 \uparrow u_2) = w_1 w_2 u_1 \uparrow u_2$

MHG: Beispiel

- $S \to W(S_1, f\uparrow)$
- $S \to W(S_2, f\uparrow)$
- $S_1 \to C2(a\uparrow, S_3, d)$
- $S_2 \to g\uparrow h$
- $S_3 \to W(S_1, b\uparrow c)$
- $S_3 \to W(S_2, b\uparrow c)$
- Erzeugte Sprache:
- $L_1 = \{a^n g b^n f c^n h d^n | n \geq 0\}$

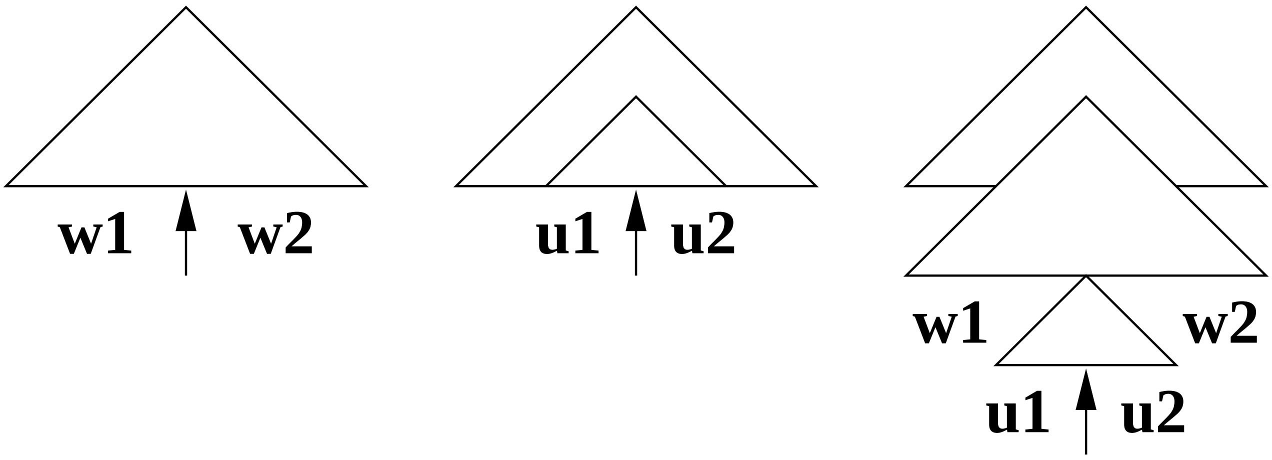
HG, MHG und TAG

- $HL \subset MHL$
- $MHL \subset TAL$
- $TAL \subset MHL$
- ?? $MHL \subset HL$

MHG und TAG

- $TAL \subset MHL$
- Konversion von TAG G zu MHG G'
- $MHL \subset TAL$
- Konversion von MHG G zu TAG G'

MHG und TAG



Adjunktion und Spaltpunkt

$$W(w_1 \uparrow w_2, u_1 \uparrow u_2) = w_1 u_1 \uparrow u_2 w_2$$

(Linear) Indexed Grammars

- Indexed Grammars (Aho 1968, Gazdar 1985)

- Regeln der Form:

- $X[..] \rightarrow X_1[i,..] \ldots X_n[i,..]$ (push)

- $X[i,..] \rightarrow X_1[..] \ldots X_n[..]$ (pop)

- $X[..] \rightarrow a$

LIG

- Linear Indexed Grammars
- Regeln der Form:
- $X[..] \rightarrow X_1[] \dots X_j[i,..] \dots X_n[]$ (push)
- $X[i,..] \rightarrow X_1[] \dots X_j[..] \dots X_n[]$ (pop)
- $X[..] \rightarrow a$

LIG: Beispiel

- $S[..] \to aS[x, ..]d$
- $S[..] \to T[..]$
- $T[i, ..] \to bT[..]c$
- $T[] \to \epsilon$
- Erzeugte Sprache:
- $L = \{a^n b^n c^n d^n | n \geq 0\}$

LIG und TAG

- $LIL \subset TAL$

- Zu LIG G konstruiere Automaten (EPDA) M'

- $TAL \subset LIL$

- Zu TAG G konstruiere MHG G' und dazu LIG G''

Combinatory Categorial Grammars (CCG)

- Kategorialgrammatiken (Ajdukiewicz 1935)

- CCG (Steedman 1985)

- G = (T,N,S,f,R) ist CCG:

- T ist endliche Menge von Terminalsymbolen

- N ist endliche Menge von Nichtterminalsymbolen

- S ist Startsymbol aus N

- f (Lexikon) ist Funktion von T nach C(N)

CCG

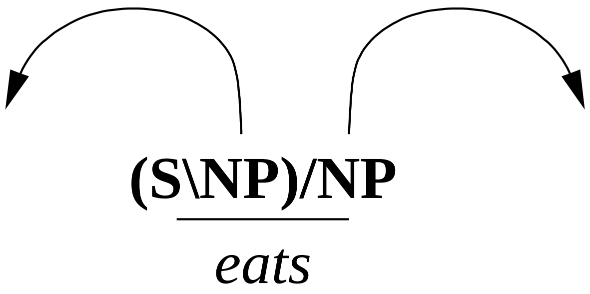
- G = (T,N,S,f,R) ist CCG:
- C (Categories):
- N $\subset$ C(N)
- $c_1, c_2 \in C(N)$, dann
- $(c_1 \backslash c_2) \in C(N)$ und
- $(c_1 / c_2) \in C(N)$.
- R ist eine endliche Menge von Kombinatorregeln

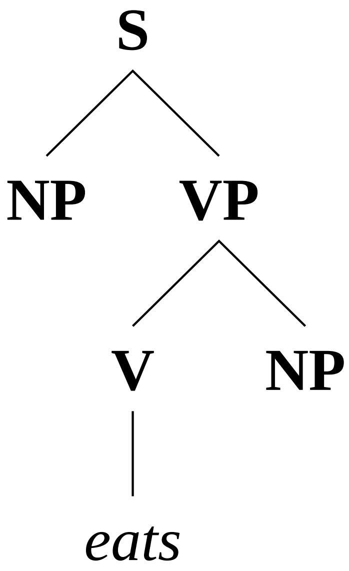
CCG

- Vier Typen von Kombinatorregeln:
- (1) forward application: $(x/y)y \to x$
- (2) backward application: $y(x\backslash y) \to x$
- (3) generalized forward composition:
- $(x/y)(...(y|z_1)|...|z_n) \to (...(x|z_1)|...|z_n)$
- (4) generalized backward composition:
- $(...(y|z_1)|...|z_n)(x\backslash y) \to (...(x|z_1)|...|z_n)$

CCG: Beispiel

Relation CCG TAG





$$\begin{aligned} \mathbf{V} &= \mathbf{(S\backslash NP)/NP} \\ \mathbf{VP} &= \mathbf{(S\backslash NP)} \end{aligned}$$

Relation CCG TAG

- $CCL \subset TAL$:

- via $CCL \subset LIL$ (direkte Konstruktion)

- $TAL \subset CCL$:

- direkte Konstruktion

Generalized Context-Free Grammars

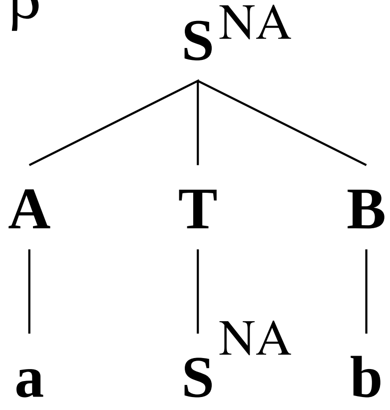
- (Pollard 1984)
- G = (V,S,F,P)
- V: Variablen, S: Startsymbol
- F: Funktionen, P: Produktionen
- Produktionen: $A \to f(A_1, ..., A_n)$
- $n \geq 0, f \in F, A, A_1, ..., A_n \in V$

Linear Context-Free Rewriting Systems

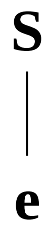
- Linear kontextfreie Ersetzungssysteme
- (Weir, Joshi 1987)
- Interpretationsfunktion:
- Tupel $(x_1, ..., x_n)$ statt Terminale
- $A \to f(A_1, ..., A_n)$
- $I(A) = I(f)(I(A_1), ..., I(A_n))$

LCFRS

β

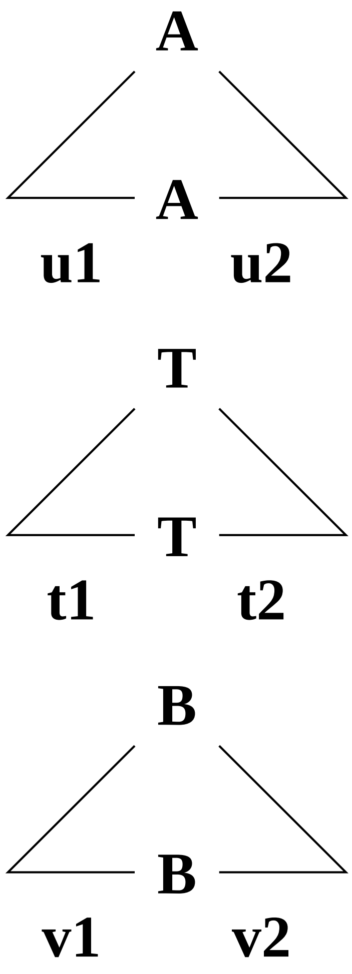


α



S -> α(S)

β1

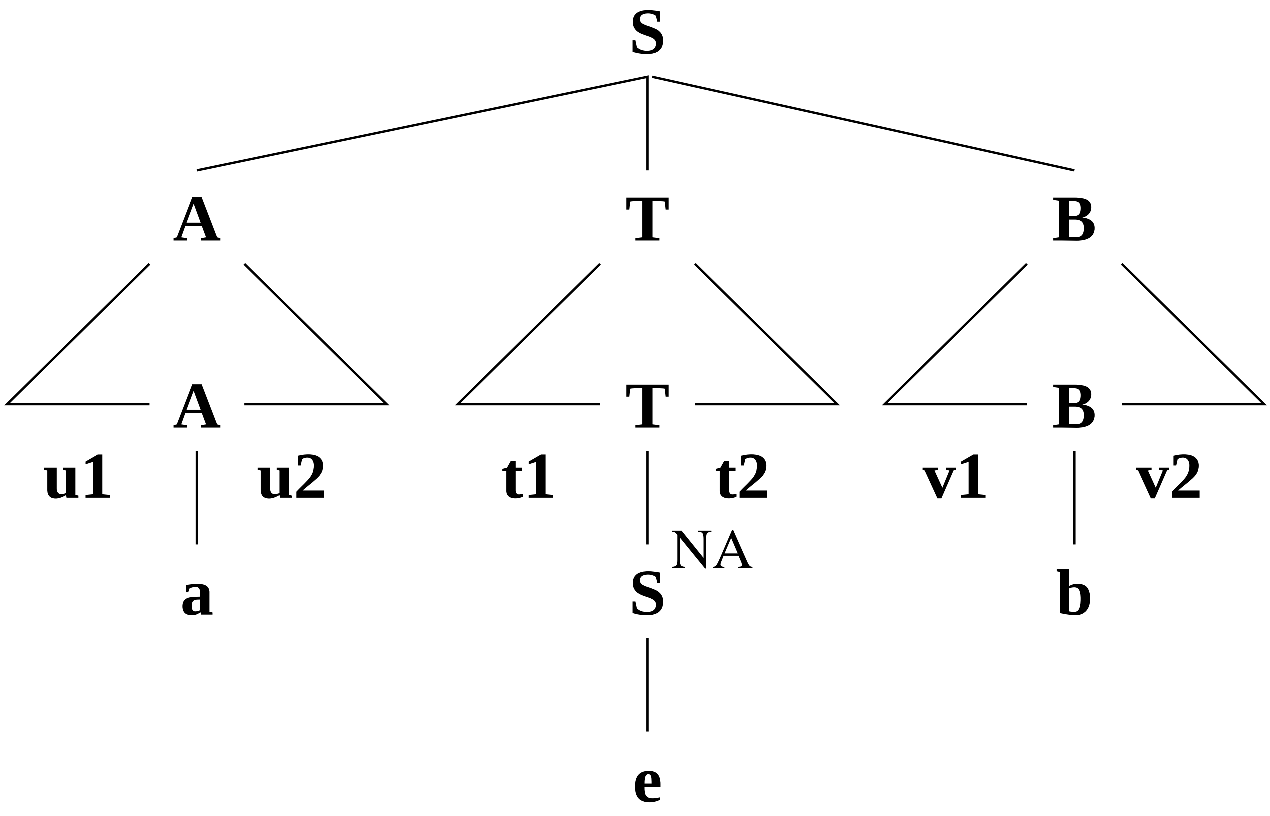


S -> β (S,A,T,B,S)

I(β1)=(u1 , u2)

I(β)=(u1 a u2 t1 , t2 v1 b v2)

LCFRS – Ableitung



- $S \vdash \alpha(S)$
  $\vdash \alpha(\beta(S, A, T, B, S))$
  $\vdash \alpha(\beta(S, \beta_1(A, A), \beta_2(T, T), \beta_3(B, B), S))$

- $I(\alpha(\beta(S, \beta_1(A, A), \beta_2(T, T), \beta_3(B, B), S)))$
  $= I(\alpha)(I(\beta)(\epsilon, I(\beta_1), I(\beta_2), I(\beta_3), \epsilon))$
  $= I(\alpha)(I(\beta)(\epsilon, (u_1, u_2), (t_1, t_2), (v_1, v_2), \epsilon))$
  $= I(\alpha)(u_1 a u_2 t_1, t_2 v_1 b v_2)$
  $= u_1 a u_2 t_1 e t_2 v_1 b v_2$

LCFRS

- Einschränkung der Interpretationsfunktion:

- Jedes Element (x_i) wird *genau einmal* verwendet

- Kein Kopieren (Index-Grammatiken) und

- auch kein Löschen.

LCFRS

- Polynomiell parsebar:
- ATM mit logarithmischem Platzverbrauch
- ATM: Alternating Turing Machine
- MHG, LIG, CCG, TAG lassen sich als LCFRS darstellen
- MC-TAG (Multi-Component TAG) charakterisieren LCFRS