Kombinatorische Kategorialgrammatiken III

Vorlesung “Grammatikformalismen”

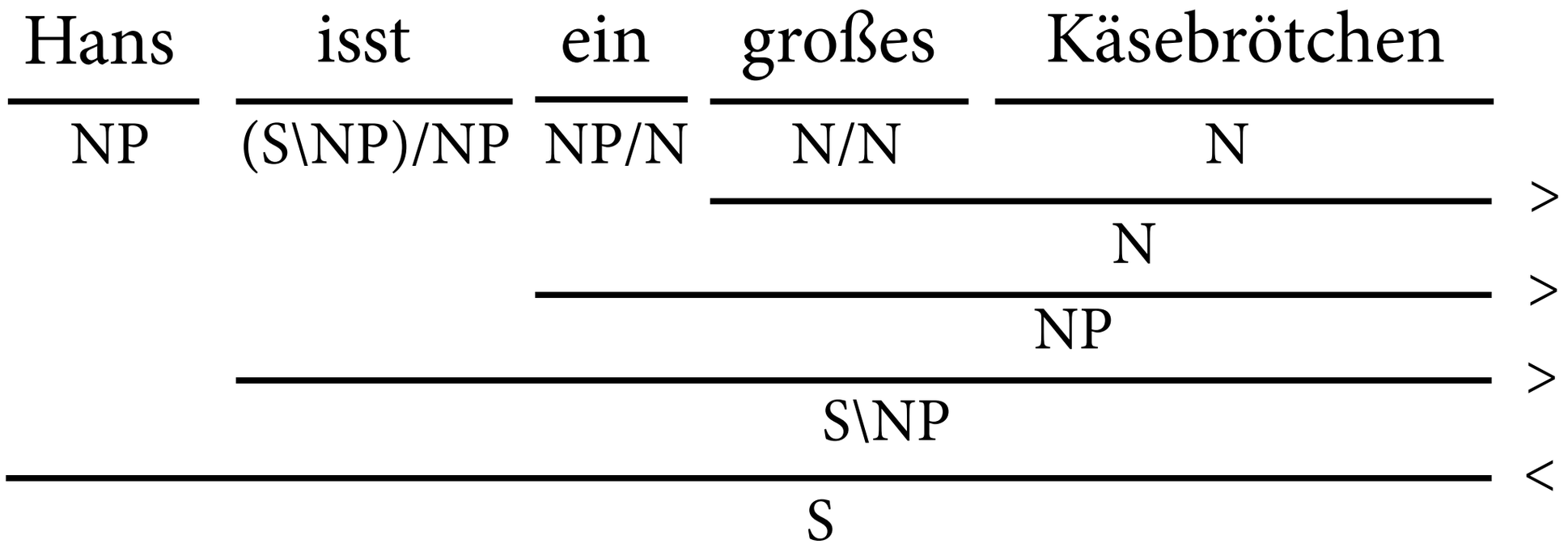
Alexander Koller

26. Mai 2015

CCG: Beispiel

Lexikon:

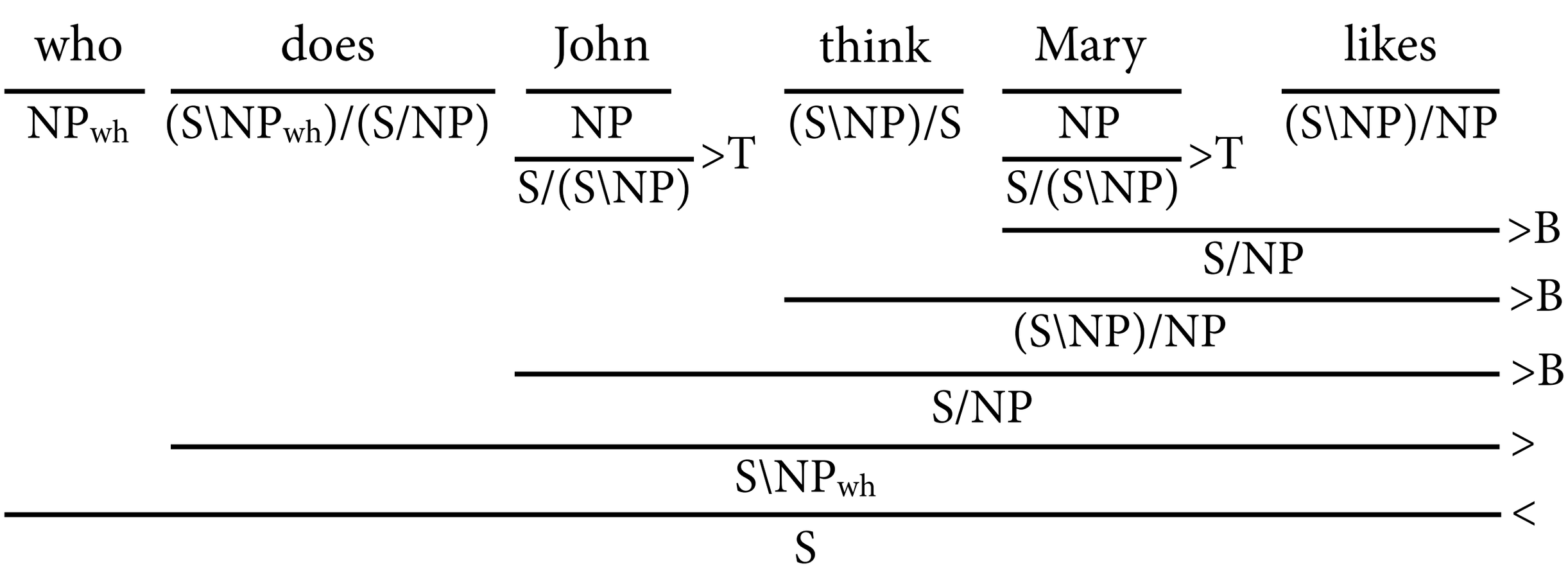
Hans: NP	ein: NP/N
isst: (S\NP)/NP	Käsebrötchen: N
großes: N/N	



CCG: Beispiel

Lexikon:

John: NP	think: (S\NP)/S
Mary: NP	likes: (S\NP)/NP
who: NP_{wh}	does: $(S\backslash NP_{wh})/(S/NP)$



Schweizerdeutsch

- Crossed composition erlaubt Modellierung von cross-serial dependencies:

das	mer	em Hans	es huus	hälfed	aastriiche
	NP_{nom}	NP_{dat}	NP_{acc}	$S_{+SUB}\backslash NP_{nom}\backslash NP_{dat}/VP$ $VP\backslash NP_{acc}$	>Bx
				$S_{+SUB}\backslash NP_{nom}\backslash NP_{dat}\backslash NP_{acc}$	<
				$S_{+SUB}\backslash NP_{nom}\backslash NP_{dat}$	<
				$S_{+SUB}\backslash NP_{nom}$	<
				S_{+SUB}	

(“VP” = Abkürzung für $S\backslash NP_{nom}$)

Themen heute

- Semantikkonstruktion und der Curry-Howard-Isomorphismus
- Parsingalgorithmus für CCG
- Probabilistisches Parsing für CCG

Semantikkonstruktion

- Ziel von Parsing ist letztlich: Semantische Repräsentationen (SRen) ausrechnen.

- Semantikkonstruktion: Berechnung von SR aus syntaktischer Analyse.

- Kompositionalität: SR von großer Konstituente aus SRen ihrer Teile berechnen.

Semantikkonstruktion in CCG

- CCG erlaubt besonders elegante Beschreibung von Semantikkonstruktion.

- SR: Lambda-Terme.
  - Wörter erhalten ihre SR aus Lexikoneintrag.
  - Jede Kombinationsregel sagt, wie die SRen systematisch kombiniert werden.
  - Schöne Korrespondenz zwischen Kategorien und Typen.

Applikation

Lexikon:

Hans: NP: *h**

isst: (S\NP)/NP: *eat'*

ein Käsebrot: NP: *kb'*

$$\frac{X/Y: f \quad Y: g}{X: f(g)} \; >$$

$$\frac{Y: g \quad X\backslash Y: f}{X: f(g)} \; <$$

$$\dfrac{\dfrac{\text{Hans}}{\text{NP: } h^*} \quad \dfrac{\dfrac{\text{isst}}{\text{(S\NP)/NP: } eat'} \quad \dfrac{\text{ein Käsebrot}}{\text{NP: } kb'}}{\text{S\NP: } eat'(kb')}\,>}{\text{S: } eat'(kb')(h^*)}\,<$$

Applikation

Lexikon:

Hans: NP: *h**

isst: (S\NP)/NP: *eat'*

ein Käsebrot: NP: *kb'*

$$\frac{\text{X/Y: } f \quad \text{Y: } g}{\text{X: } f(g)} \; >$$

$$\frac{\text{Y: } g \quad \text{X\Y: } f}{\text{X: } f(g)} \; <$$

$$\dfrac{\dfrac{\text{NP: } h^* \qquad \dfrac{\text{(S\NP)/NP: } eat'_{\langle e,\langle e,t\rangle\rangle} \qquad \text{NP: } kb'}{\text{S\NP: } eat'(kb')}\,>}{\text{S: } eat'(kb')(h^*)}\,<$$

Hans | isst | ein Käsebrot

Applikation

Lexikon:

Hans: NP: *h**

isst: (S\NP)/NP: *eat'*

ein Käsebrot: NP: *kb'*

$$\frac{\text{X/Y: } f \qquad \text{Y: } g}{\text{X: } f(g)} \; >$$

$$\frac{\text{Y: } g \qquad \text{X\Y: } f}{\text{X: } f(g)} \; <$$

$$\dfrac{\dfrac{\text{Hans}}{\text{NP: } h^*} \quad \dfrac{\dfrac{\text{isst}}{\text{(S\NP)/NP: } eat'_{\langle e,\langle e,t\rangle\rangle}} \quad \dfrac{\text{ein Käsebrot}}{\text{NP: } kb'_{e}}}{\text{S\NP: } eat'(kb')} \; >}{\text{S: } eat'(kb')(h^*)} \; <$$

Applikation

Lexikon:

Hans: NP: *h**

isst: (S\NP)/NP: *eat'*

ein Käsebrot: NP: *kb'*

$$\frac{\text{X/Y: } f \qquad \text{Y: } g}{\text{X: } f(g)} \; >$$

$$\frac{\text{Y: } g \qquad \text{X\Y: } f}{\text{X: } f(g)} \; <$$

$$\dfrac{\dfrac{\text{Hans}}{\text{NP: } h^*} \quad \dfrac{\dfrac{\text{isst}}{\text{(S\NP)/NP: } eat'_{\langle e,\langle e,t\rangle\rangle}} \quad \dfrac{\text{ein Käsebrot}}{\text{NP: } kb'_{e}}}{\text{S\NP: } eat'(kb')_{\langle e,t\rangle}} \; >}{\text{S: } eat'(kb')(h^*)} \; <$$

Applikation

Lexikon:

Hans: NP: h^*

isst: (S\NP)/NP: eat'

ein Käsebrot: NP: kb'

$$\frac{\text{X/Y: } f \quad \text{Y: } g}{\text{X: } f(g)} \; >$$

$$\frac{\text{Y: } g \quad \text{X\Y: } f}{\text{X: } f(g)} \; <$$

$$\dfrac{\dfrac{\text{Hans}}{\text{NP: } h^*_{e}} \quad \dfrac{\text{isst} \quad \dfrac{\text{ein} \quad \text{Käsebrot}}{\text{NP: } kb'_{e}}}{\text{(S\NP)/NP: } eat'_{\langle e,\langle e,t\rangle\rangle}}}{\dfrac{\text{S\NP: } eat'(kb')_{\langle e,t\rangle}}{\text{S: } eat'(kb')(h^*)}} \; \begin{matrix} > \\ < \end{matrix}$$

Applikation

Lexikon:

Hans: NP: h^*

isst: (S\NP)/NP: eat'

ein Käsebrot: NP: kb'

$$\frac{\text{X/Y: } f \quad \text{Y: } g}{\text{X: } f(g)} \; >$$

$$\frac{\text{Y: } g \quad \text{X\Y: } f}{\text{X: } f(g)} \; <$$

$$\dfrac{\dfrac{\text{Hans}}{\text{NP: } h^*_{e}} \quad \dfrac{\dfrac{\text{isst}}{\text{(S\NP)/NP: } eat'_{\langle e,\langle e,t\rangle\rangle}} \quad \dfrac{\text{ein Käsebrot}}{\text{NP: } kb'_{e}}}{\text{S\NP: } eat'(kb')_{\langle e,t\rangle}} \; >}{\text{S: } eat'(kb')(h^*)_{t}} \; <$$

Kontrolle

- Subjektkontrolle: ein syntaktisches Argument füllt zwei semantische Argumentpositionen.
  → Nichtlineare Lambdaterme im Lexikon.

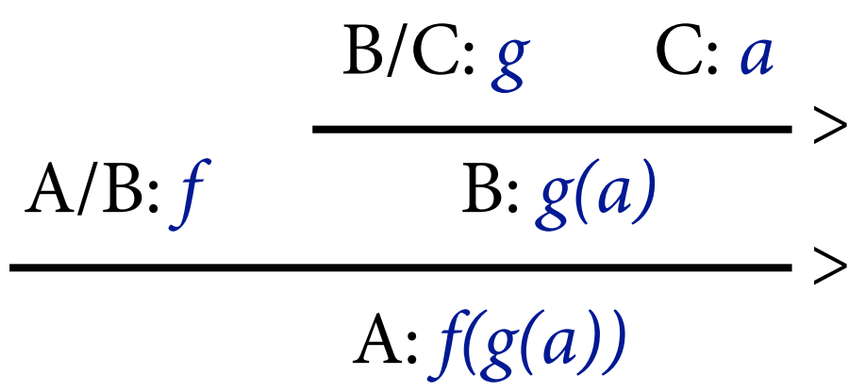
	will	singen
Hans	(S\NP)/(S\NP): $\lambda P \lambda x\ want'(P(x))(x)_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$	S\NP: $sing'_{\langle e,t\rangle}$
NP: h^*_e	S\NP: $\lambda x\ want'(sing'(x))(x)_{\langle e,t\rangle}$	
S: $want'(sing'(h^*))(h^*)_t$		

Beobachtungen

- Syntaktisches Kombinieren
  = semantisches Kombinieren.
  - z.B. Applikation = Applikation
- Kategorien ≈ Typen.
  - NP ≈ e, S ≈ t
  - A/B, A\B ≈ ⟨B, A⟩
  - Spezialfall des *Curry-Howard-Isomorphismus*.
- Diese Prinzipien auf andere Regeln fortsetzen.

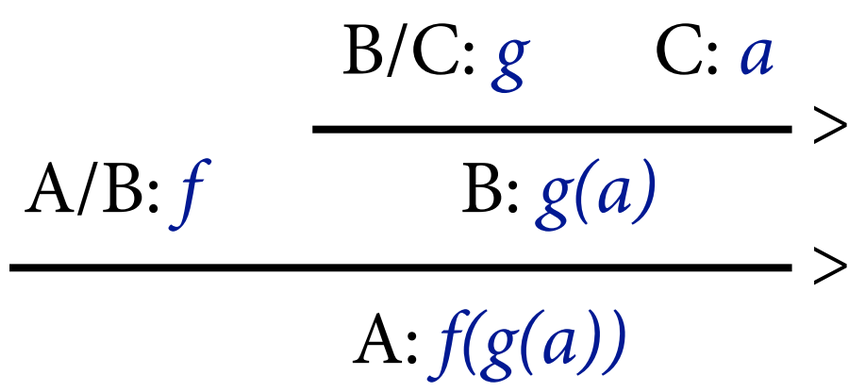
Semantik für Komposition

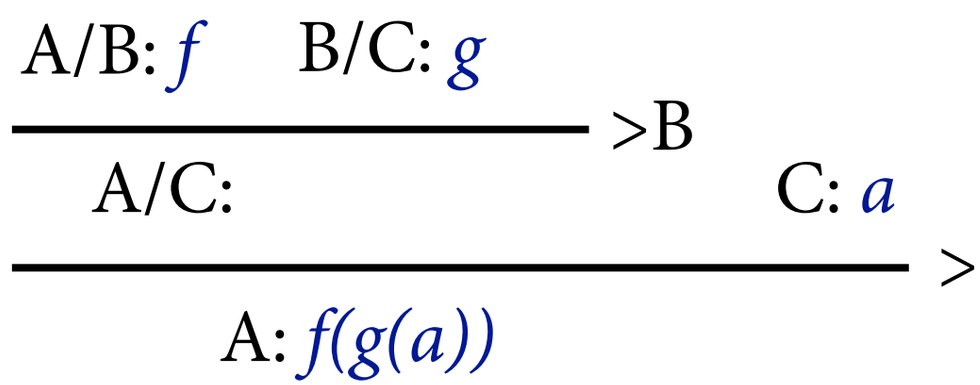
Grundprinzip: “Gleichartige” Ableitungen sollen gleiche SRen berechnen.



Semantik für Komposition

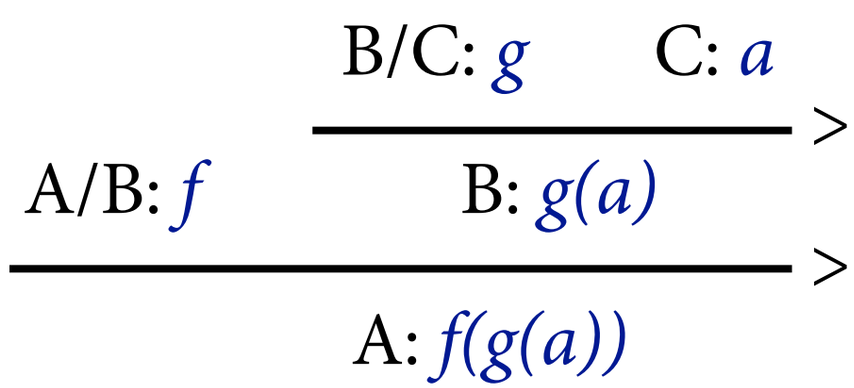
Grundprinzip: “Gleichartige” Ableitungen sollen gleiche SRen berechnen.

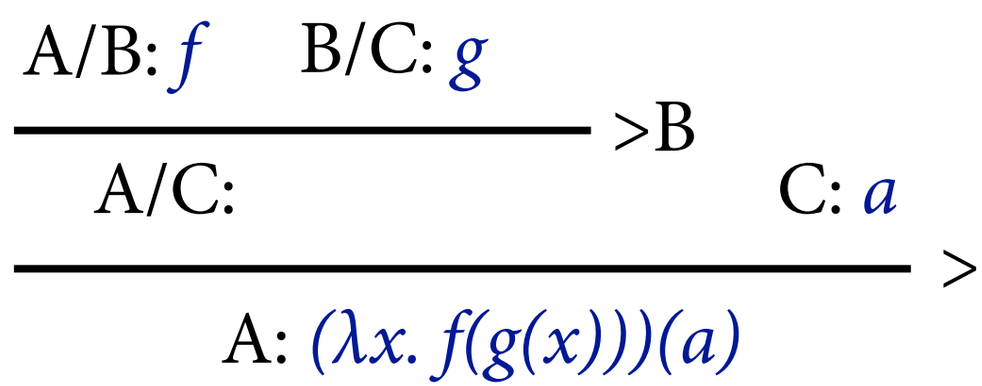




Semantik für Komposition

Grundprinzip: “Gleichartige” Ableitungen sollen gleiche SRen berechnen.

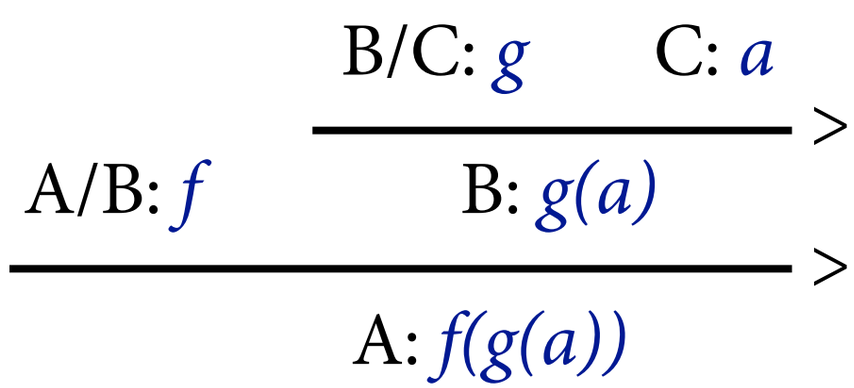




$$(\lambda x.\, f(g(x)))(a) \Rightarrow_\beta f(g(a))$$

Semantik für Komposition

Grundprinzip: “Gleichartige” Ableitungen sollen gleiche SRen berechnen.



Semantik für Type-Raising

Grundprinzip: “Gleichartige” Ableitungen sollen gleiche SRen berechnen.

$$\frac{\text{NP: } a \qquad \text{S\NP: } P}{\text{S: } P(a)} \; <$$

Semantik für Type-Raising

Grundprinzip: “Gleichartige” Ableitungen sollen gleiche SRen berechnen.

$$\frac{\text{NP: } a \qquad \text{S\NP: } P}{\text{S: } P(a)}\ <$$

$$\frac{\dfrac{\text{NP: } a}{\text{S/(S\NP): }}\ >\text{T} \qquad \text{S\NP: } P}{\text{S: } P(a)}\ >$$

Semantik für Type-Raising

Grundprinzip: “Gleichartige” Ableitungen sollen gleiche SRen berechnen.

$$\frac{\text{NP: } a \qquad \text{S\NP: } P}{\text{S: } P(a)}\ <$$

$$\frac{\dfrac{\text{NP: } a}{\text{S/(S\NP): }}\ >\text{T} \qquad \text{S\NP: } P}{\text{S: } (\lambda Q.\ Q(a))(P)}\ >$$

$$(\lambda Q.\ Q(a))(P) \Rightarrow_\beta P(a)$$

Semantik für Type-Raising

Grundprinzip: “Gleichartige” Ableitungen sollen gleiche SRen berechnen.

$$\frac{\text{NP: } a \qquad \text{S\NP: } P}{\text{S: } P(a)}\ <$$

$$\frac{\dfrac{\text{NP: } a}{\text{S/(S\NP): } \lambda Q.\ Q(a)}\ >\text{T} \qquad \text{S\NP: } P}{\text{S: } (\lambda Q.\ Q(a))(P)}\ >$$

$$(\lambda Q.\ Q(a))(P) \Rightarrow_\beta P(a)$$

Type-Raising und Komposition

$$\frac{\text{X: } a}{\text{Y/(Y\X): } \lambda P.P(a)}\,{>}\text{T} \qquad \frac{\text{X/Y: } f \qquad \text{Y/Z: } g}{\text{X/Z: } \lambda x.f(g(x))}\,{>}\text{B} \qquad \frac{\text{X/Y: } f \qquad \text{Y\Z: } g}{\text{X\Z: } \lambda x.f(g(x))}\,{>}\text{Bx}$$

$$\frac{\text{X: } a}{\text{Y\(Y/X): } \lambda P.P(a)}\,{<}\text{T} \qquad \frac{\text{Y\Z: } g \qquad \text{X\Y: } f}{\text{X\Z: } \lambda x.f(g(x))}\,{<}\text{B} \qquad \frac{\text{Y/Z: } g \qquad \text{X\Y: } f}{\text{X/Z: } \lambda x.f(g(x))}\,{<}\text{Bx}$$

$$\dfrac{\dfrac{\dfrac{\text{Hans}}{\text{NP: } h^*}}{\text{S/(S\NP): } \lambda P.P(h^*)}\,{>}\text{T} \qquad \dfrac{\text{isst}}{\text{(S\NP)/NP: } eat'}}{\text{S/NP: } \lambda x.(\lambda P.P(h^*))(eat'(x)) \Rightarrow_\beta \lambda x.eat'(x)(h^*)}\,{>}\text{B} \qquad \dfrac{\text{ein Käsebrot}}{\text{NP: } kb'}}{\text{S: } (\lambda x.eat'(x)(h^*))(kb') \Rightarrow_\beta eat'(kb')(h^*)}\,{>}$$

Einschränkungen

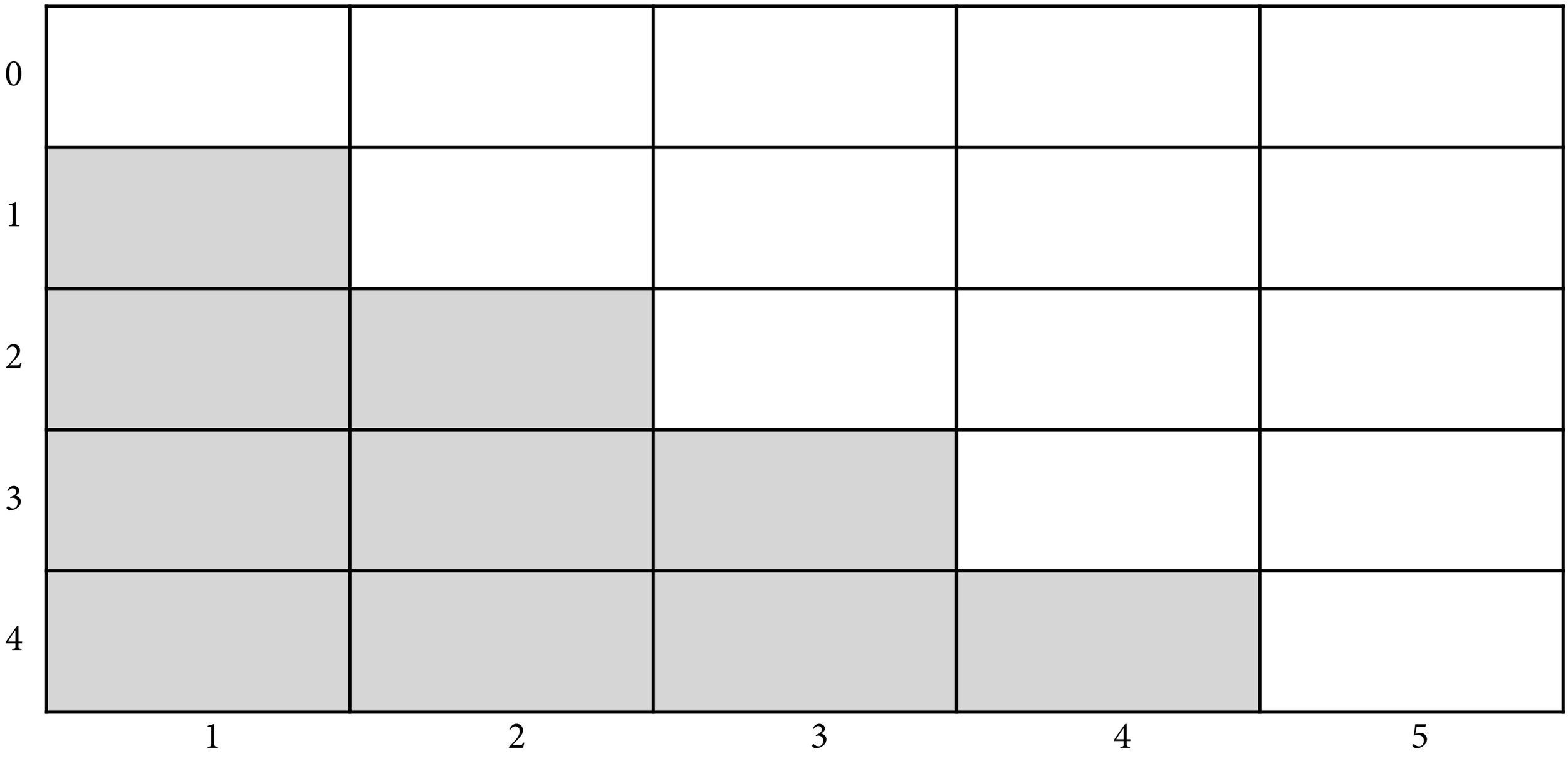
- Semantische Repräsentation von komplexen NPs?
  - “ein Käsebrot”: NP/N N ⇒ NP
  - SRen für “ein” und “Käsebrot”?
  - alternative Skopustheorie in CCG (Steedman 2012)
- Umgang mit semantischen Ambiguitäten?
  - für jede syntaktische Analyse nur eine SR
  - muss deshalb gezwungenermaßen semantisch ambigen Sätzen mehrere syntaktische Analysen zuweisen

Parsing

- Chartparsing für CCG.
- Grundlage: CKY-Parser.
  - naiver Ansatz braucht exponentielle Laufzeit
- Polynomieller Parsing-Algorithmus.

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N



CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP				
1					
2					
3					
4					

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP)				
1					
2					
3					
4					

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				
1					
2					
3					
4					

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				
1		(S\NP)/NP			
2					
3					
4					

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				
1		(S\NP)/NP			
2			NP/N		
3					
4					

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

0	NP S/(S\NP) S\(S/NP)				
1		(S\NP)/NP			
2			NP/N		
3				N/N	
4					
	1	2	3	4	5

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				
1		(S\NP)/NP			
2			NP/N		
3				N/N	
4					N

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				
1		(S\NP)/NP			
2			NP/N		
3				N/N	N
4					N

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				
1		(S\NP)/NP			
2			NP/N		NP
3				N/N	N
4					N

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				
1		(S\NP)/NP			S\NP
2			NP/N		NP
3				N/N	N
4					N

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				S
1		(S\NP)/NP			S\NP
2			NP/N		NP
3				N/N	N
4					N

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				S
1		(S\NP)/NP			S\NP
2			NP/N	NP/N	NP
3				N/N	N
4					N

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)				S
1		(S\NP)/NP		(S\NP)/N	S\NP
2			NP/N	NP/N	NP
3				N/N	N
4					N

CKY-Parsing für CCG

Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)	S/NP	S/N	S/N	S
1		(S\NP)/NP	(S\NP)/N	(S\NP)/N	S\NP
2			NP/N	NP/N	NP
3				N/N	N
4					N

CKY-Parsing für CCG

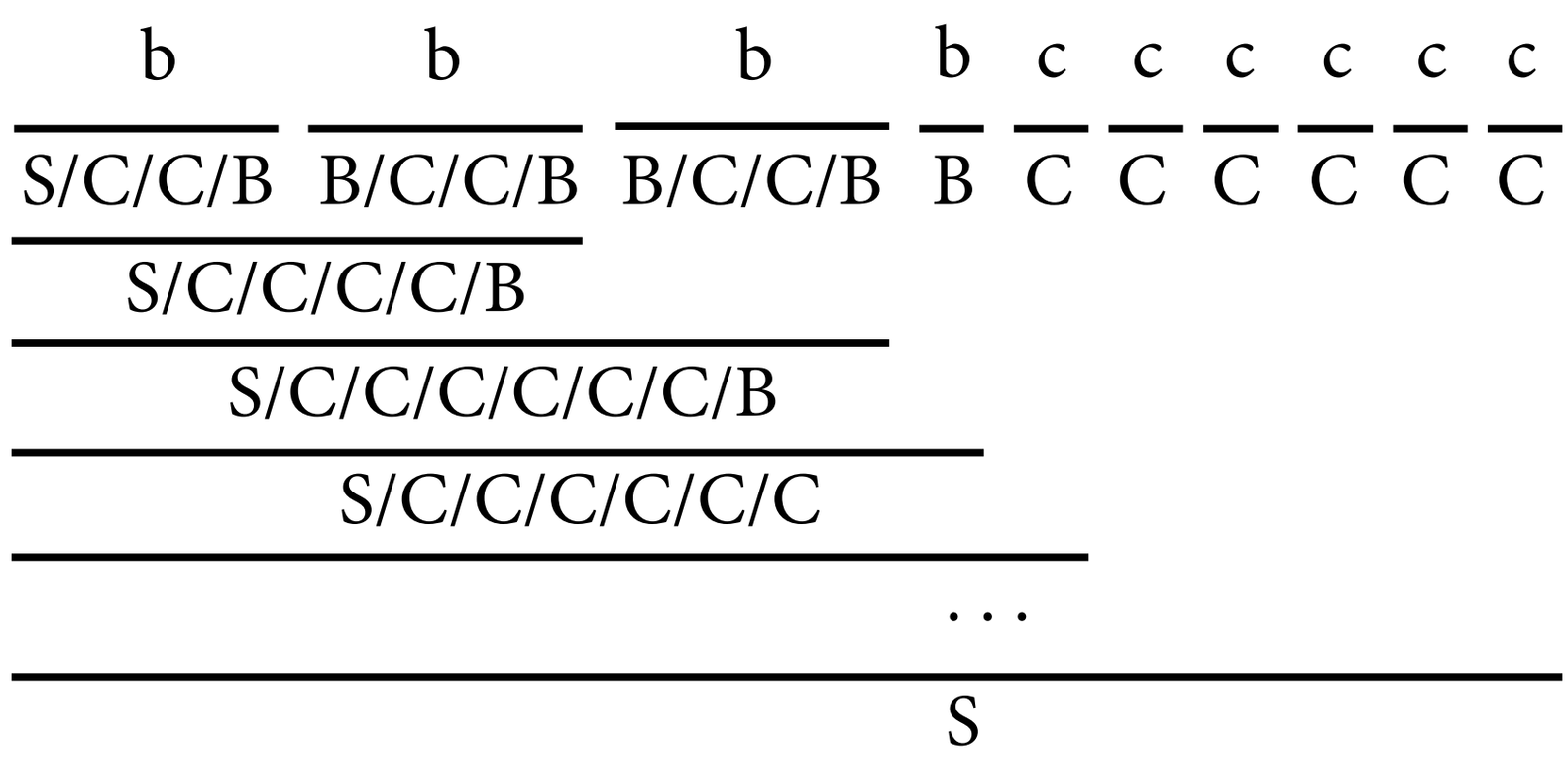
Hans	isst	ein	großes	Käsebrötchen
NP	(S\NP)/NP	NP/N	N/N	N

	1	2	3	4	5
0	NP S/(S\NP) S\(S/NP)	S/NP	S/N	S/N	S
1		(S\NP)/NP	(S\NP)/N	(S\NP)/N	S\NP
2			NP/N	NP/N	NP
3				N/N	N
4					N

Wie viele Kategorien pro Chart-Zelle?

Das Problem

- Kategorie eines Ausdrucks der Länge n kann O(n) Argumente enthalten.
  - daher exponentiell (in n) viele Kategorien möglich
  - daher worst-case-Laufzeit exponentiell



(OpenCCG-Parser geht in etwa so)

Lösung (Grundidee)

$$\frac{\dfrac{b}{S/C/C/B} \quad \dfrac{b}{B/C/C/B}}{S/C/C/C/C/B}$$

(Vijay-Shanker & Weir 1990)

Lösung (Grundidee)

$$\dfrac{\dfrac{\dfrac{b}{S/C/C/B}\quad\dfrac{b}{B/C/C/B}}{S/C/C/C/B}\quad\dfrac{b}{B/C/C/B}}{}$$

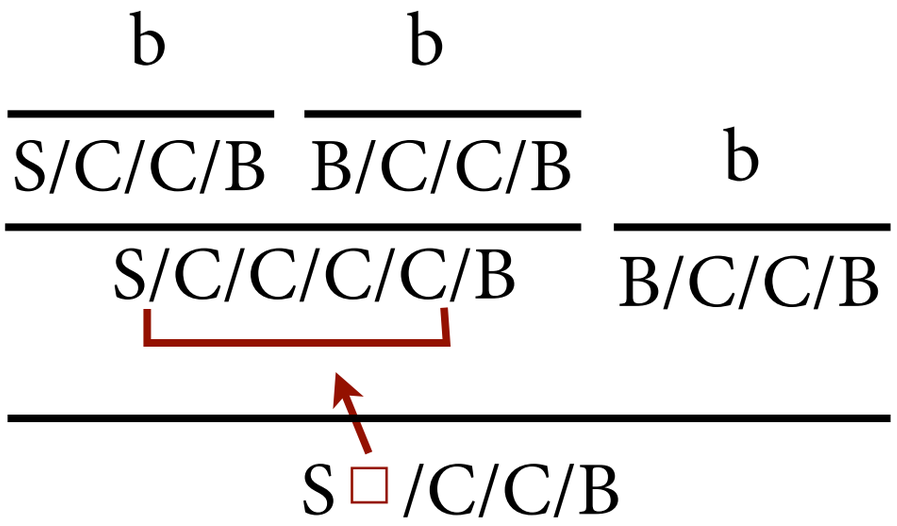
(Vijay-Shanker & Weir 1990)

Lösung (Grundidee)

$$\dfrac{\dfrac{\dfrac{b}{S/C/C/B}\quad\dfrac{b}{B/C/C/B}}{S/C/C/C/B}\quad\dfrac{b}{B/C/C/B}}{\qquad\qquad}$$

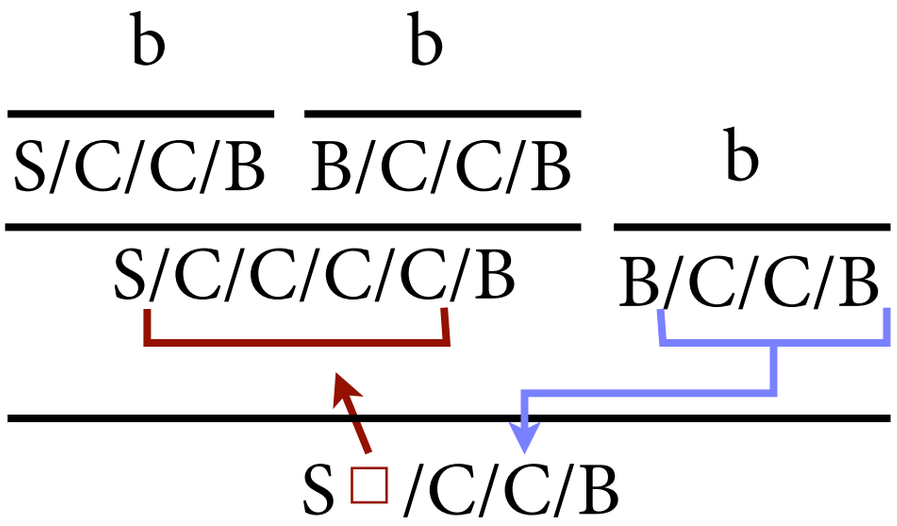
(Vijay-Shanker & Weir 1990)

Lösung (Grundidee)



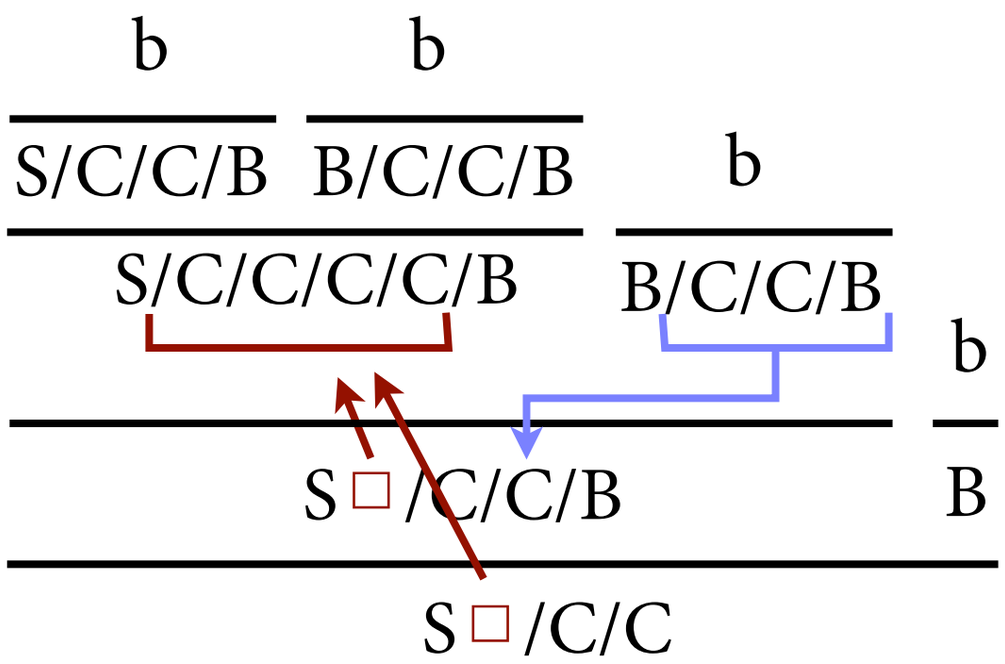
(Vijay-Shanker & Weir 1990)

Lösung (Grundidee)



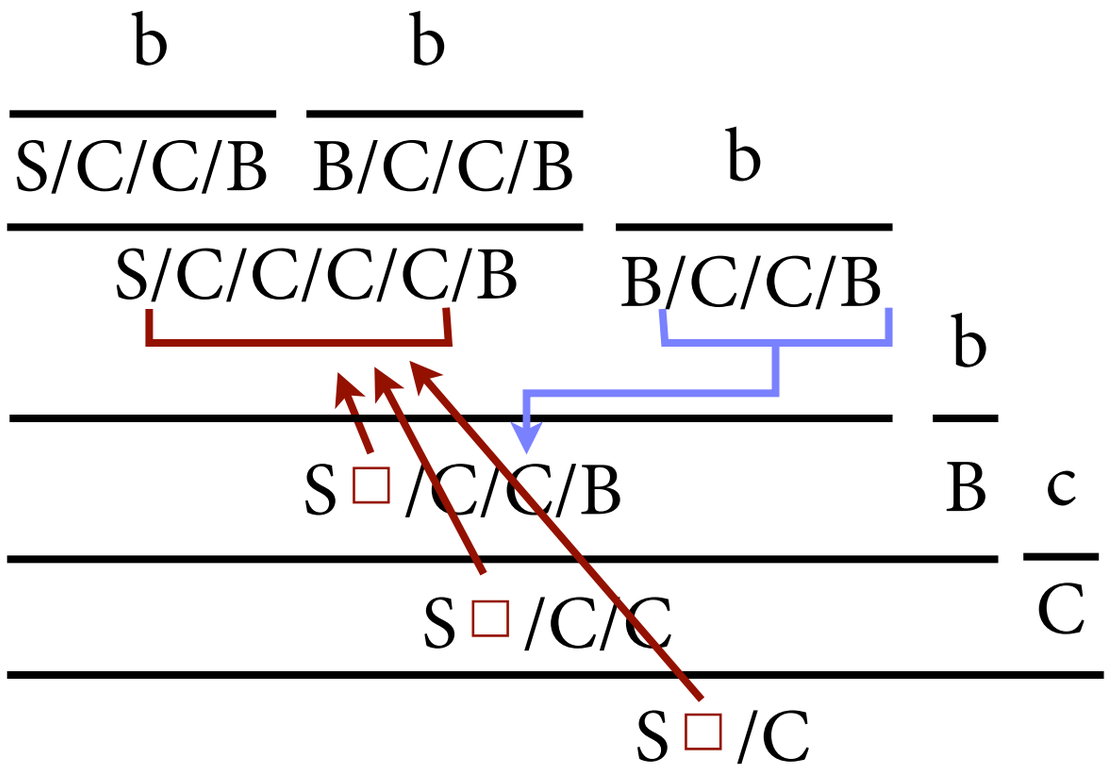
(Vijay-Shanker & Weir 1990)

Lösung (Grundidee)



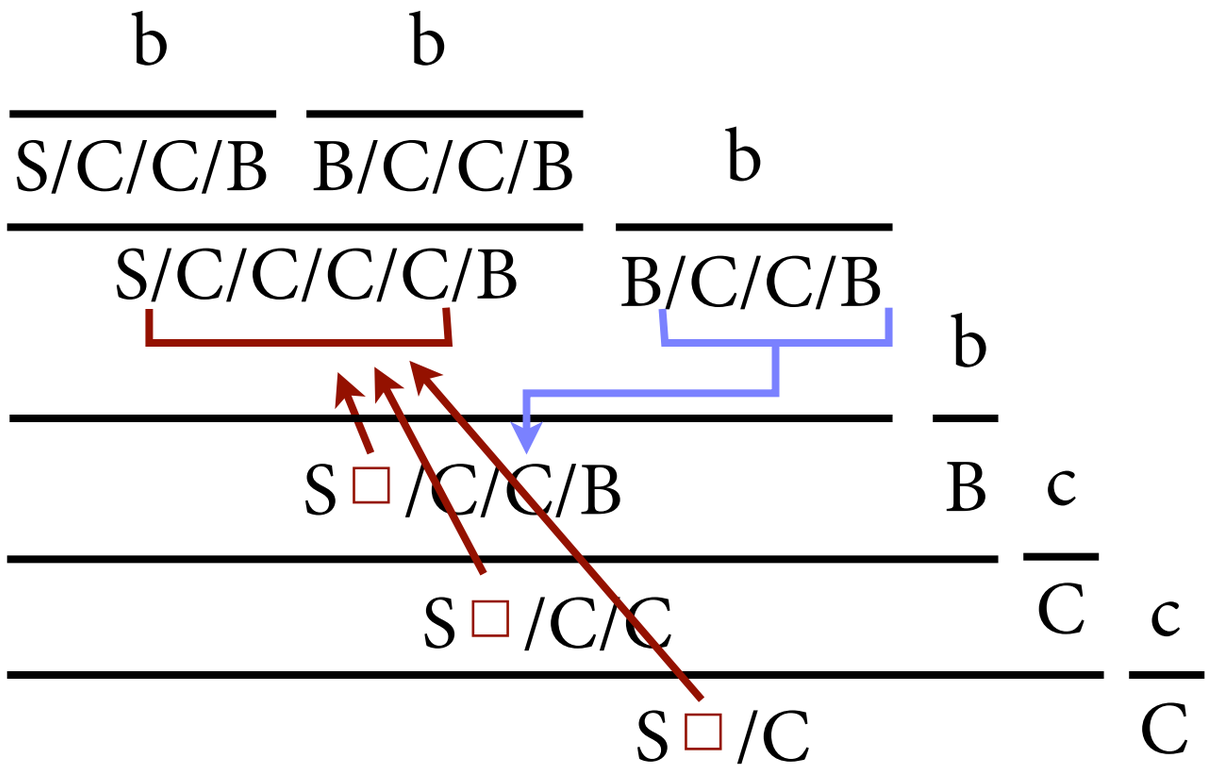
(Vijay-Shanker & Weir 1990)

Lösung (Grundidee)



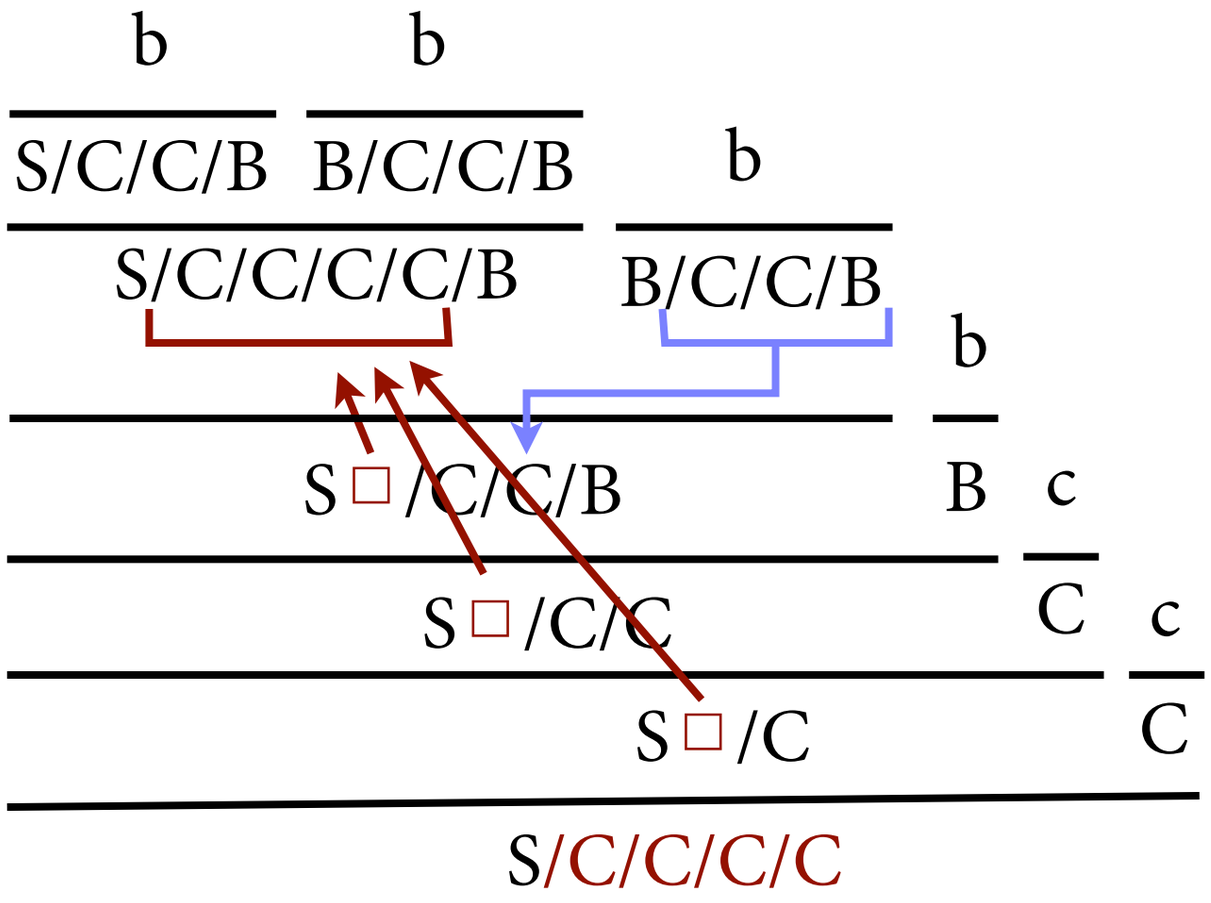
(Vijay-Shanker & Weir 1990)

Lösung (Grundidee)



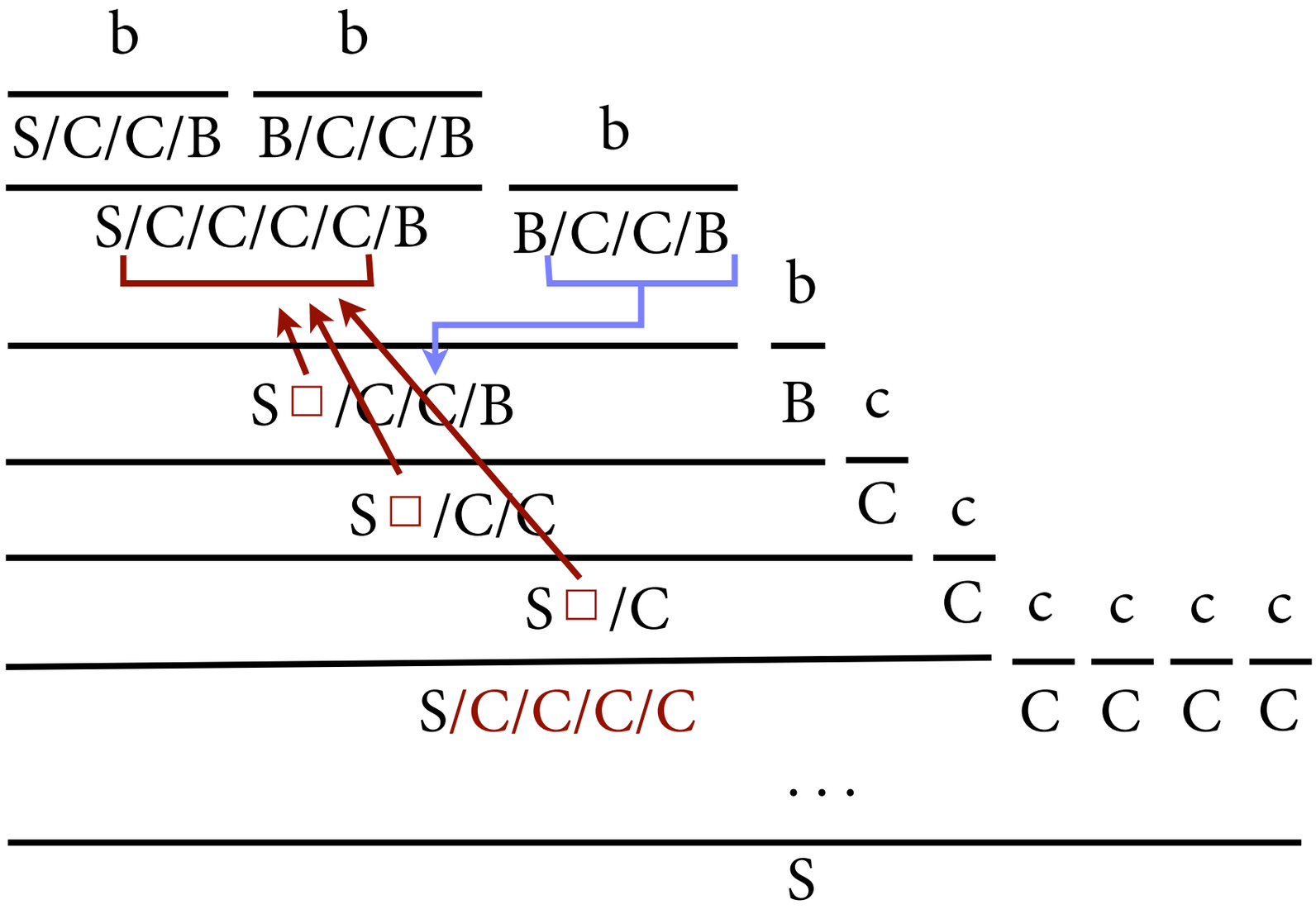
(Vijay-Shanker & Weir 1990)

Lösung (Grundidee)



(Vijay-Shanker & Weir 1990)

Lösung (Grundidee)



(Vijay-Shanker & Weir 1990)

Parsingalgorithmus

- Idee: Länge von Kategorien in Items jetzt beschränkt.
  - k_1 = Grad der maximal erlaubten Kompositionsregel
  - k_2 = max. Anzahl der Argumente in lexikalischer Kat.
  - Längenbeschränkung: $k_1 + k_2$ Argumente

- Teuerste Regel kombiniert drei Items:
  - $S\square/C + C + S/C/C/C/C/B \Rightarrow S/C/C/C/C$

- Genaue Analyse dieser Regel ergibt Laufzeit $O(n^6)$.

(Vijay-Shanker & Weir 1990)

Statistisches Parsing

- Wichtigster statistischer Parser für CCG: C&C-Parser (Clark & Curran 2007).

- Idee:
  - Maximum-Entropy-Modell auf Dependenzinterpretation von CCG-Parses.
  - Parsingalgorithmus berechnet besten Parse sehr effizient (mit Supertagging-Methoden).
  - Training: Syntaktisch annotiertes Korpus (z.B. PTB) in CCG-Annotationen übersetzen.

Dependenz-Interpretation von CCG

- Extrahiere aus CCG-Parse Dependenz-Tupel:
  $\langle \text{isst}_2, (\text{S}\backslash\text{NP}_1)/\text{NP}_2, 2, \text{käsebrot}_5 \rangle$
  - "Käsebrot" an Position 5 füllt zweites Argument von "isst" an Position 2.

- Vorgehen:
  - Argumentpositionen lexikalisch spezifizieren:
    isst: $(\text{S}\backslash\text{NP}_1)/\text{NP}_2$
  - für jede Konstituente *Kopfwort* mitführen
  - Regelanwendung etabliert Dependenz zwischen Köpfen

Beispiel

Lexikon:

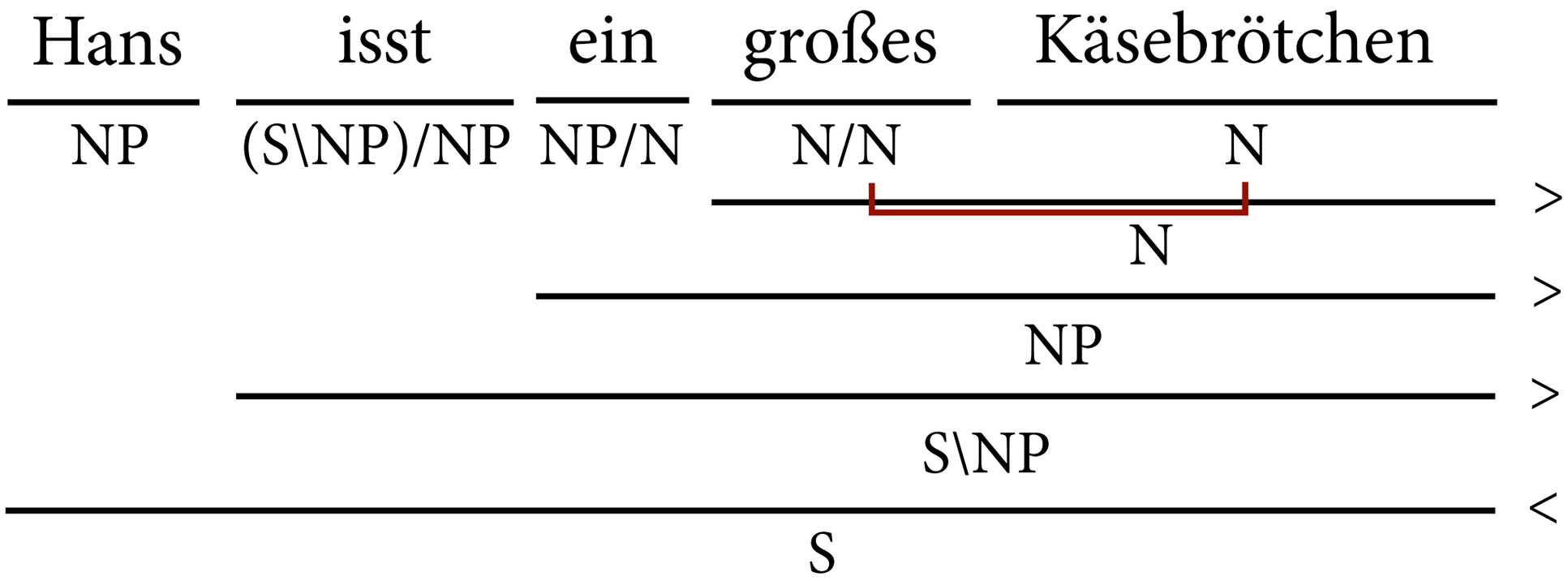
Hans: NP	ein: NP/N_1
isst: $(S\backslash NP_1)/NP_2$	Käsebrötchen: N
großes: N/N_1	

Hans	isst	ein	großes	Käsebrötchen	
NP	(S\NP)/NP	NP/N	N/N	N	
			N		>
		NP			>
	S\NP				>
S					<

Beispiel

Lexikon:

Hans: NP	ein: NP/N_1
isst: $(S\backslash NP_1)/NP_2$	Käsebrötchen: N
großes: N/N_1	

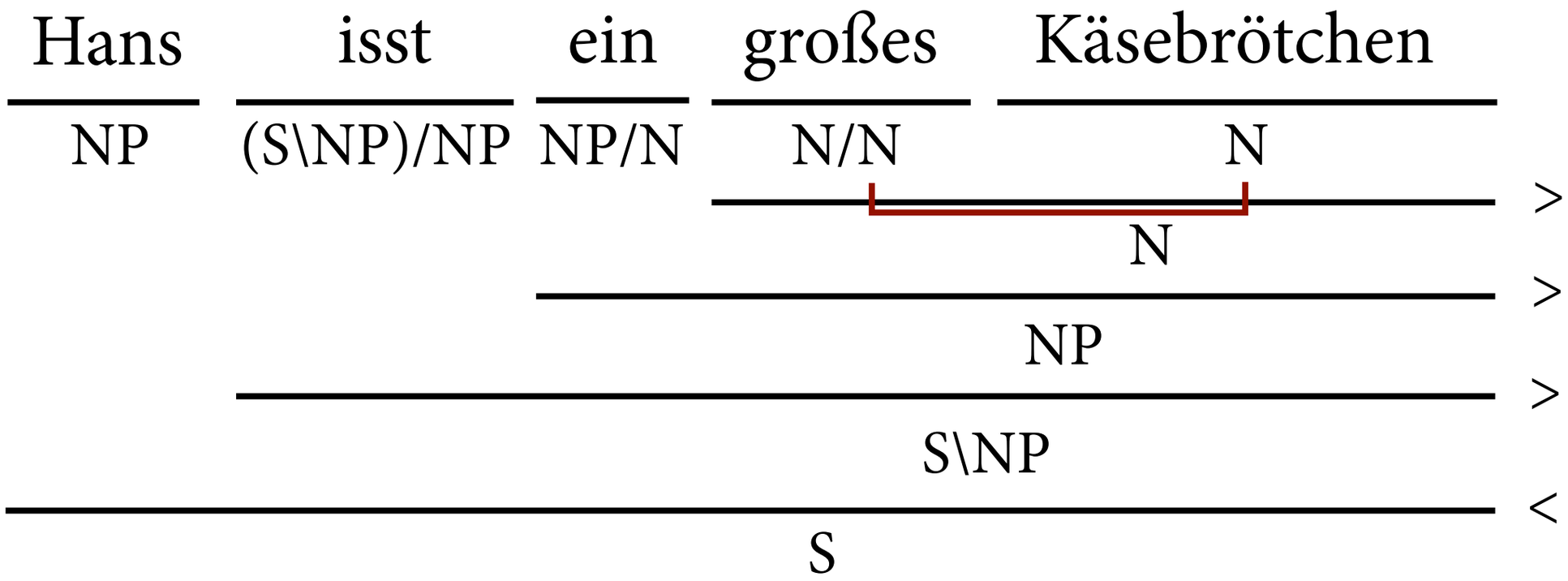


Beispiel

Lexikon:

Hans: NP	ein: NP/N_1
isst: $(S\backslash NP_1)/NP_2$	Käsebrötchen: N
großes: N/N_1	

$\langle großes_4, N/N_1, 1, KB_5 \rangle$

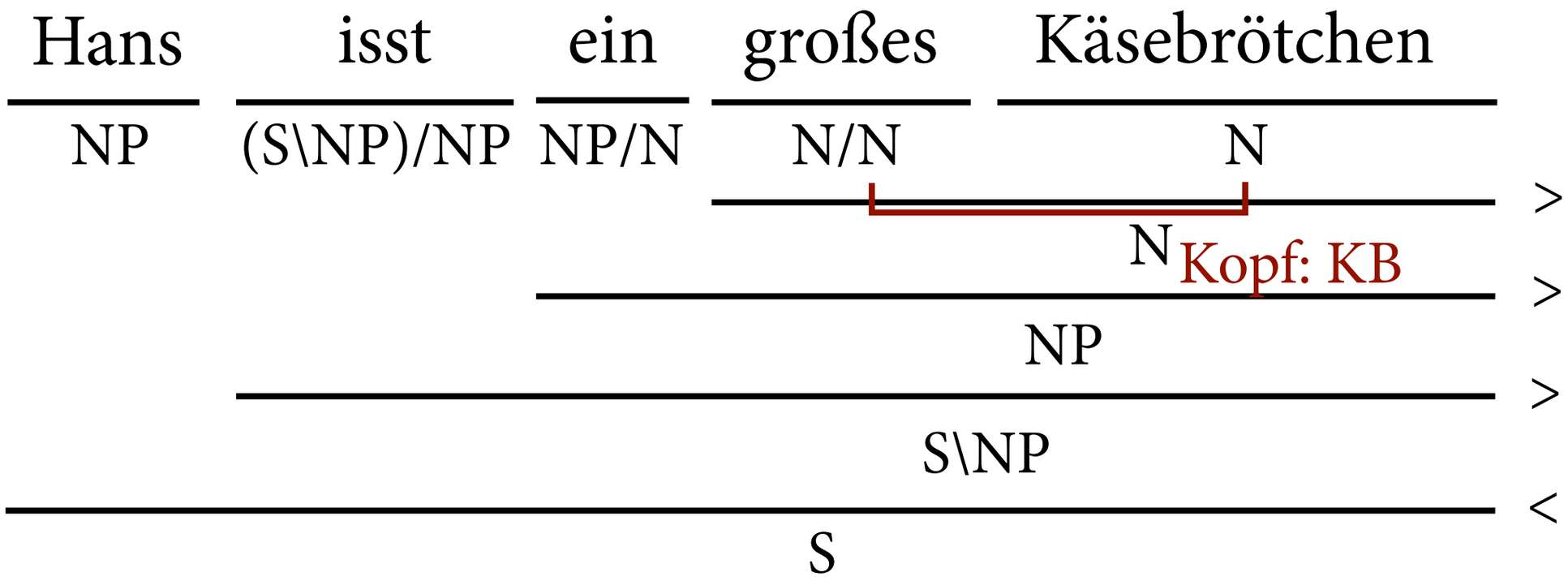


Beispiel

Lexikon:

Hans: NP	ein: NP/N_1
isst: $(S\backslash NP_1)/NP_2$	Käsebrötchen: N
großes: N/N_1	

⟨großes$_4$, N/N_1, 1, KB_5⟩

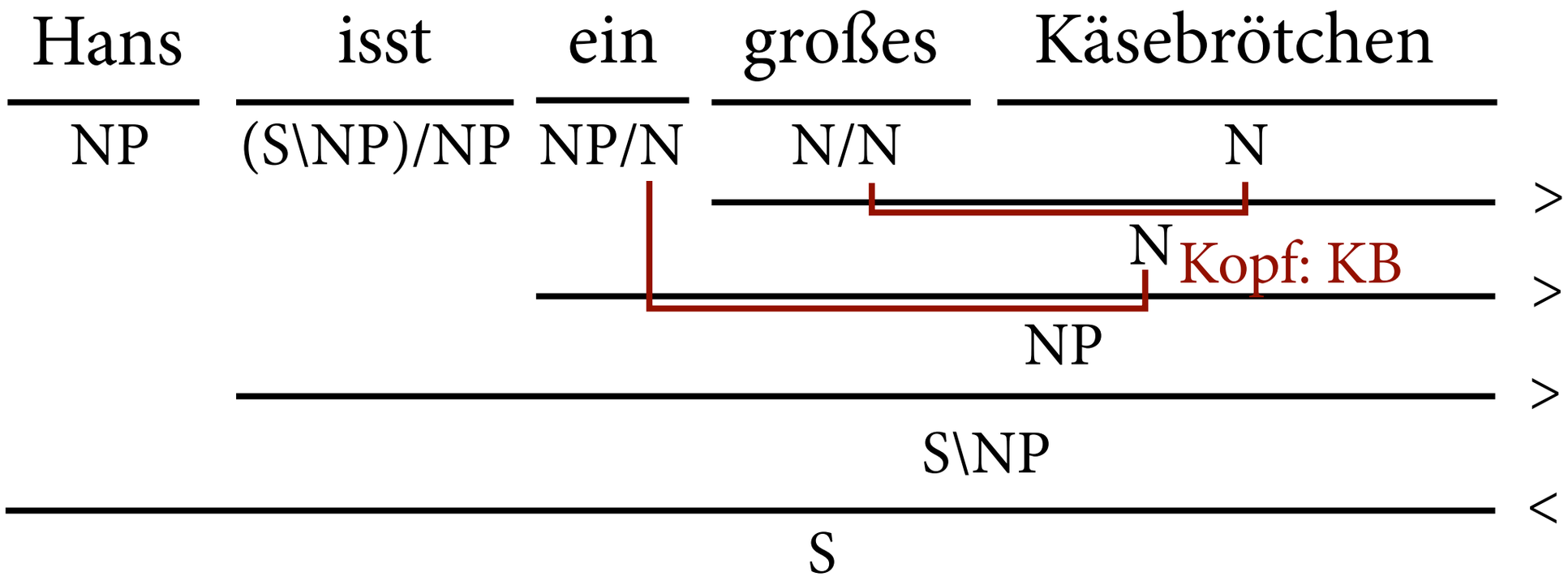


Beispiel

Lexikon:

Hans: NP	ein: NP/N_1
isst: $(S\backslash NP_1)/NP_2$	Käsebrötchen: N
großes: N/N_1	

⟨großes$_4$, N/N_1, 1, KB_5⟩



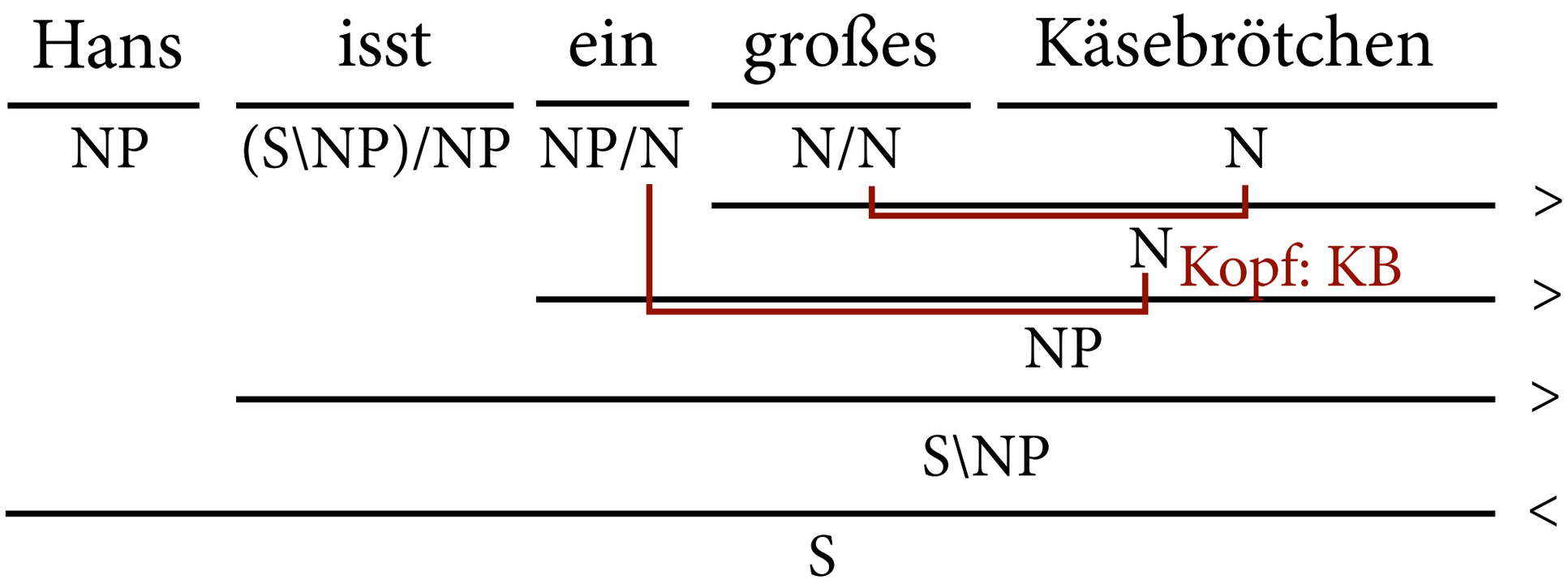
Beispiel

Lexikon:

Hans: NP	ein: NP/N_1
isst: $(S\backslash NP_1)/NP_2$	Käsebrötchen: N
großes: N/N_1	

⟨großes$_4$, N/N_1, 1, KB_5⟩

⟨ein$_3$, NP/N_1, 1, KB_5⟩



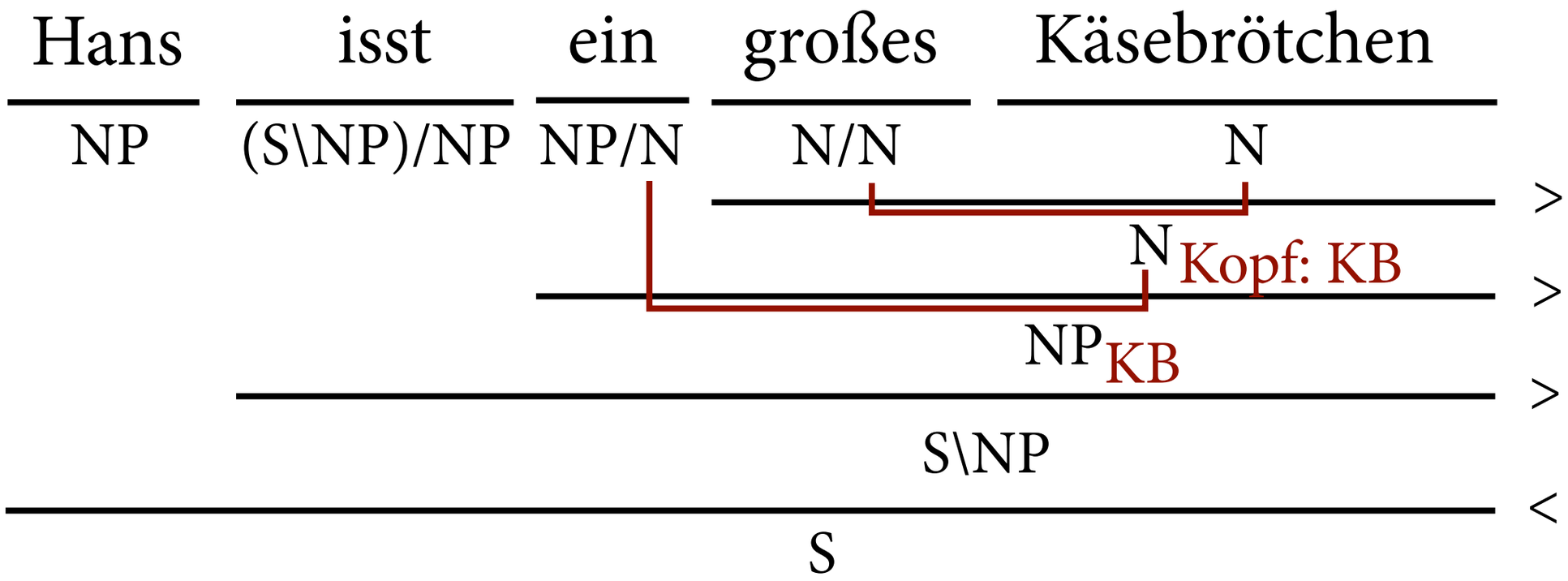
Beispiel

Lexikon:

Hans: NP	ein: NP/N_1
isst: $(S\backslash NP_1)/NP_2$	Käsebrötchen: N
großes: N/N_1	

⟨$groß es_4$, N/N_1, 1, KB_5⟩

⟨ein_3, NP/N_1, 1, KB_5⟩



Beispiel

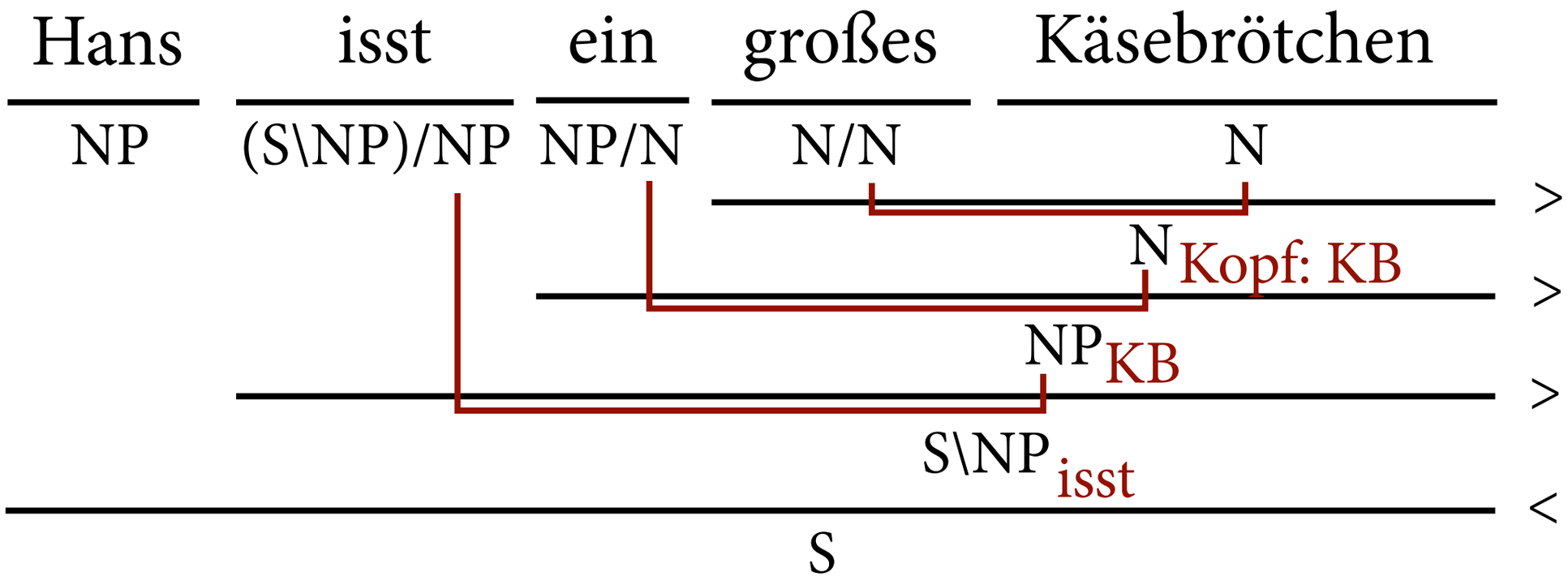
Lexikon:

Hans: NP	ein: NP/N_1
isst: $(S\backslash NP_1)/NP_2$	Käsebrötchen: N
großes: N/N_1	

⟨großes$_4$, N/N_1, 1, KB_5⟩

⟨ein$_3$, NP/N_1, 1, KB_5⟩

⟨isst$_2$, $(S\backslash NP_1)/NP_2$, 2, KB_5⟩



Beispiel

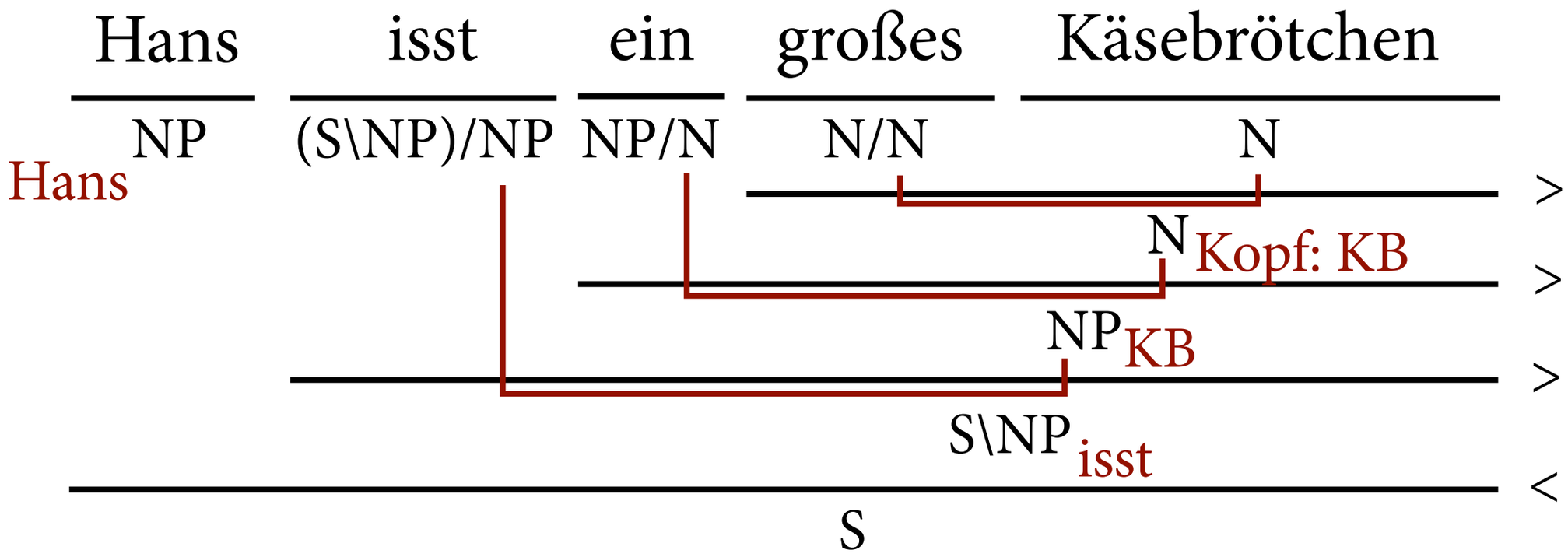
Lexikon:

Hans: NP	ein: NP/N_1
isst: $(S\backslash NP_1)/NP_2$	Käsebrötchen: N
großes: N/N_1	

$\langle großes_4, N/N_1, 1, KB_5 \rangle$

$\langle ein_3, NP/N_1, 1, KB_5 \rangle$

$\langle isst_2, (S\backslash NP_1)/NP_2, 2, KB_5 \rangle$



Beispiel

Lexikon:

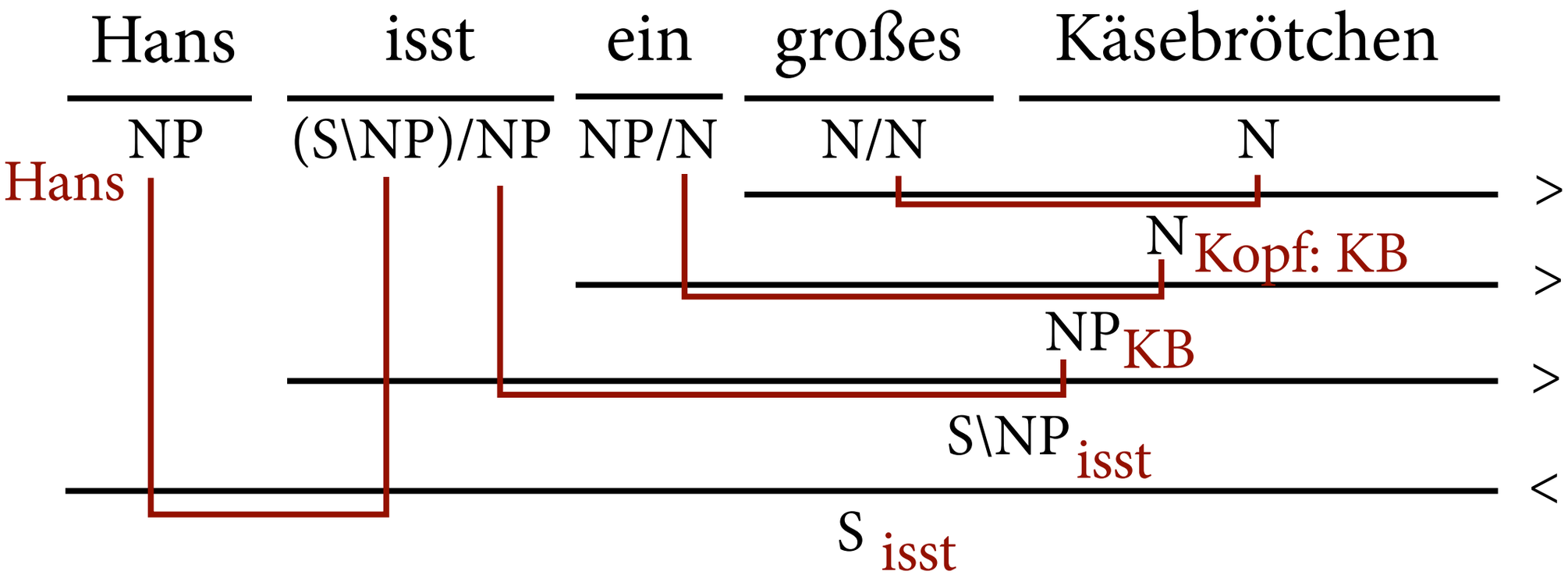
Hans: NP	ein: NP/N_1
isst: (S\NP$_1$)/NP$_2$	Käsebrötchen: N
großes: N/N_1	

⟨großes$_4$, N/N_1, 1, KB$_5$⟩

⟨ein$_3$, NP/N_1, 1, KB$_5$⟩

⟨isst$_2$, (S\NP$_1$)/NP$_2$, 2, KB$_5$⟩

⟨isst$_2$, (S\NP$_1$)/NP$_2$, 1, Hans$_1$⟩



Maximum-Entropy-Modell

- Wir wollen diskriminatives W.modell
  P(t|w) für Parsebaum t gegeben String w.

- Verwende dafür Featurefunktionen f(t,w), die Passung von t für w definieren.

- Dann mit Vektor θ von Gewichten in MaxEnt-Modell zusammensetzen:

$$P(t|w) = \frac{e^{\theta \cdot f(t,w)}}{\sum_t' e^{\theta \cdot f(t',w)}}$$

Einige Features

Feature type	Example
LexCat + Word	$(S/S)/NP$ + Before
LexCat + POS	$(S/S)/NP$ + IN
RootCat	$S[dcl]$
RootCat + Word	$S[dcl]$ + was
RootCat + POS	$S[dcl]$ + VBD
Rule	$S[dcl] \rightarrow NP\ S[dcl]\backslash NP$
Rule + Word	$S[dcl] \rightarrow NP\ S[dcl]\backslash NP$ + bought
Rule + POS	$S[dcl] \rightarrow NP\ S[dcl]\backslash NP$ + VBD

Feature type	Example
Word–Word	$\langle bought, (S\backslash NP_1)/NP_2, 2, stake, (NP\backslash NP)/(S[dcl]/NP)\rangle$
Word–POS	$\langle bought, (S\backslash NP_1)/NP_2, 2, \text{NN}, (NP\backslash NP)/(S[dcl]/NP)\rangle$
POS–Word	$\langle \text{VBD}, (S\backslash NP_1)/NP_2, 2, stake, (NP\backslash NP)/(S[dcl]/NP)\rangle$
POS–POS	$\langle \text{VBD}, (S\backslash NP_1)/NP_2, 2, \text{NN}, (NP\backslash NP)/(S[dcl]/NP)\rangle$
Word + Distance(words)	$\langle bought, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 2$
Word + Distance(punct)	$\langle bought, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$
Word + Distance(verbs)	$\langle bought, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$
POS + Distance(words)	$\langle \text{VBD}, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 2$
POS + Distance(punct)	$\langle \text{VBD}, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$
POS + Distance(verbs)	$\langle \text{VBD}, (S\backslash NP_1)/NP_2, 2, (NP\backslash NP)/(S[dcl]/NP)\rangle + 0$

Effizientes Parsing

- Features sind so definiert, dass sie schon auf Parse-Items ausgewertet werden können, nicht erst auf fertigem Parsebaum.

- Damit effizientes Chartparsing möglich.

- Umgang mit lexikalischer Ambiguität: Supertagging.

Zusammenfassung

- Semantikkonstruktion
- Chartparsing
- C&C-Parser, MaxEnt-Modell