

Übersicht

4 Syntaktische Relationen: Konstituenz

4.1 Konstituentenstruktur

4.1.1 Eigenschaften der Konstituentenstruktur

4.1.2 Konstituentenstruktur des Deutschen

4.2 Modellierung mit kontextfreier Grammatik

4.2.1 Phrasenstrukturgrammatik

4.2.2 Phrasenstrukturregeln des Deutschen

4.3 X-Bar-Phrasenstrukturschema

4.4 Adäquatheit einer CFG als Syntaxmodell

4.5 Treebanks und Grammatiken

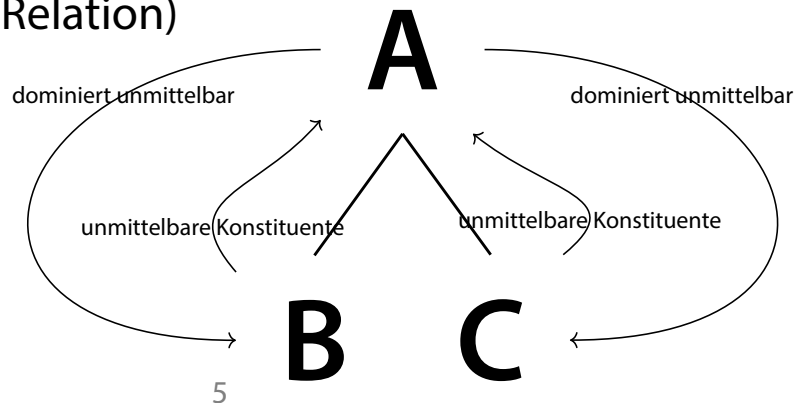
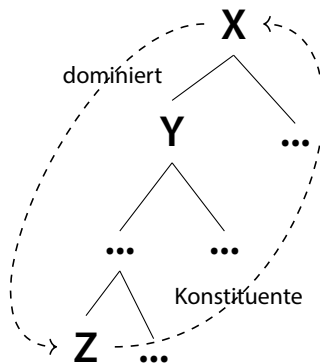
4 Syntaktische Relationen: Konstituenz

4.1 Konstituentenstruktur

4.1.1 Eigenschaften der Konstituentenstruktur

- **Konstituentenanalyse** (IC-Analyse): **Zerlegung** syntaktischer Einheit in ihre Teile (Konstituenten) und Bildung von **Konstituentenklassen** (lexikalische und syntaktische Kategorien)
 - Ermittlung über **Konstituententests**
 - Ergebnis ist eine **hierarchisch gegliederte Struktur**
- **unmittelbare Konstituenten** sind die **maximalen Konstituenten** einer Einheit (aus denen sie unmittelbar zusammengesetzt ist)

- **Konstituenz: Teil-Ganzes-Beziehung** zwischen sprachlichen Einheiten (Konstituenten)
- Beziehung der **unmittelbaren Dominanz** zwischen Einheit und ihren unmittelbaren Konstituenten
- Beziehung der **Dominanz** zwischen Einheit X und der unmittelbaren Konstituente Y; sowie zwischen X und Z, wenn Y Z dominiert (transitive Relation)



Konstituentenstruktur

- **Konstituentenstruktur:** Menge der **durch die Relation der unmittelbaren Dominanz verbundenen Konstituenten**
- durch Bezug auf **Konstituentenklassen** (lexikalische und syntaktische Kategorien als Knoten) und Abstraktion von der Wortebene ergeben sich **Konstituentenschemata**

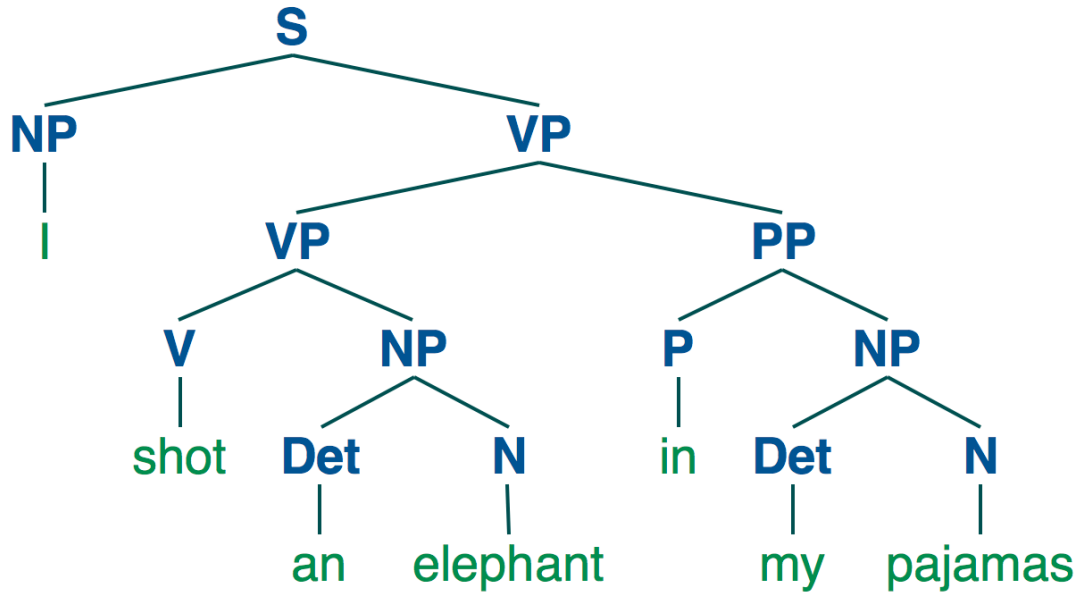


Abbildung 1: Syntaxbaum als Repräsentation der Konstituentenstruktur

Übersicht Konstituentenstruktur:

- **Elemente der Struktur (Knoten):**
 - **Wörter** → *terminale Knoten*
 - **lexikalische Kategorien** → *präterminale Knoten*
 - **syntaktische Kategorien** → *nichtterminale Knoten*
→ *Phrasen (Konstituente mit syntaktisch zusammengehörigen Subkonstituenten)*
- **Relationen der Struktur (Kanten)** → *Teil-Ganzes-Beziehung; unmittelbare Dominanz des Mutterknotens über Tochterknoten*
- **Strukturinformationen in Knoten des Syntaxbaums**

Phrasen: Kopfprinzip

- jede Phrase hat einen **Kopf (auch: Phrasenkern)**, der die **Eigenschaften der Phrase bestimmt**
- alle anderen Wörter und Phrasen in der Phrase sind zum Kopf **dependent**
- Kopf vererbt **morphosyntaktische Merkmale** an Phrase
- Kopf steuert **syntaktisches Verhalten** der Konstituente im Satz
- Kopf bestimmt die **Phrasenkategorie** (Wortart X \rightarrow Phrasenkategorie. XP)

Kopf-Perkolation

- Merkmale der Phrase werden getragen von Kopf
- Köpfe werden im Syntaxbaum nach oben weitergereicht
- **Kopf-Perkolutions-Regeln:**
 - $\text{head}(\text{NP}) = \text{head}(\text{N})$
 - $\text{head}(\text{VP}) = \text{head}(\text{V})$
 - $\text{head}(\text{S}) = \text{head}(\text{VP})$
- wichtig u.a. für **lexikalisierte Grammatiken** sowie die **Transformation einer Phrasenstruktur- in eine Dependenzgrammatik**

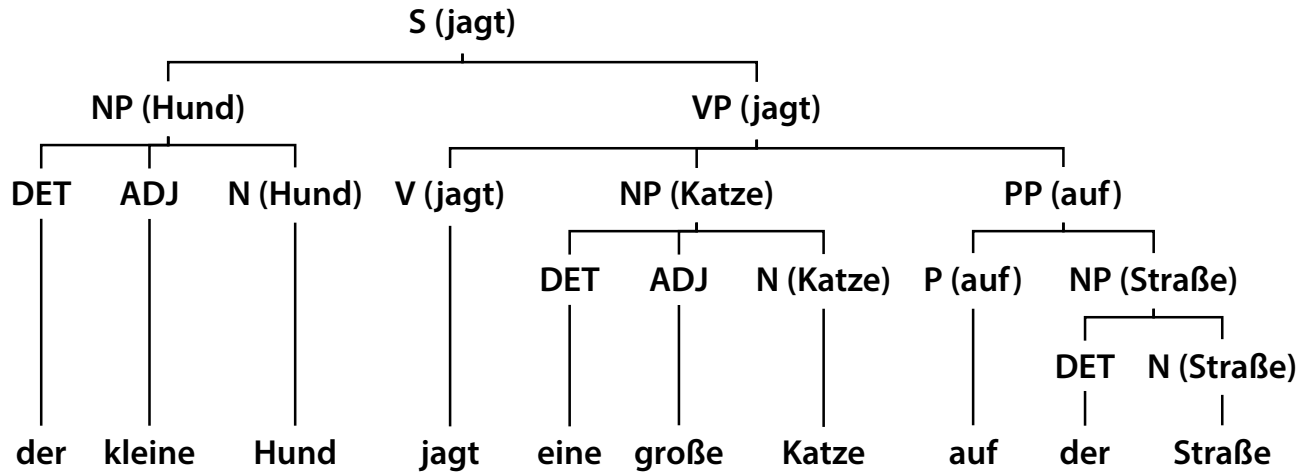


Abbildung 2: Phrasenstruktur Transitiver Satz des Deutschen mit KopfregeIn

Diskontinuierliche Konstituenten

- *long distance dependencies*
- Trennung von Teilkonstituenten einer Konstituente
- Problem für Baumdarstellung:
 - **Überkreuzung** = nicht-projektiv
- Lösung: **leere Knoten** (*empty nodes*: \emptyset , ϵ , τ , NONE)
 - *trace* (**Spur**): Konzept der Transformationsgrammatik

- **Transformationsgrammatik:**
 - Annahme: **Tiefen- und Oberflächenstruktur**
 - abstrakte vs. beobachtbare Form von Sätzen
 - z. B.: Annahme deutsche Tiefenstruktur der VP: OV (*den Hund sehen*)
 - **Transformationsregelanwendung** zur Erzeugung der Oberflächenstruktur: **läßt Spur zurück**
- im Englischen relativ begrenzt: z.B. Topikalisierung, Extraposition, **Wh-fronting**

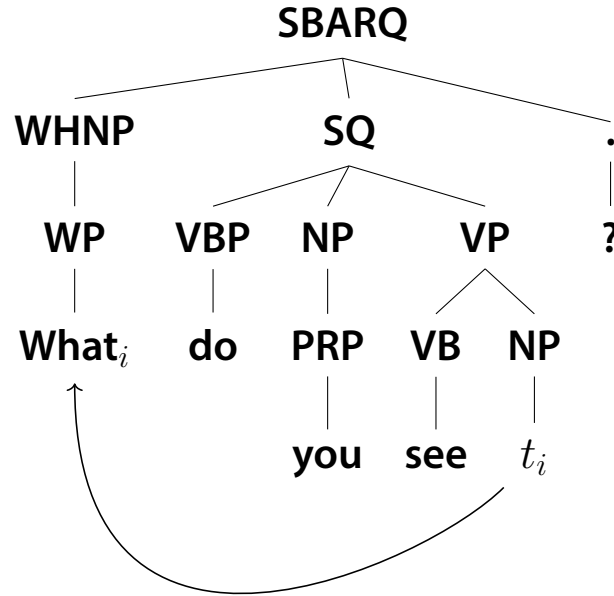


Abbildung 3: *Beispiel für Analyse long-distance dependency mit Spur (t)*

Schema Einfacher Satz

- **allgemeines Satzschema: $S = NP + VP$**
 - Ergebnis von Konstituententests (Reduktion auf Zweiwort-Satz)
 - **Subjekt-NP und Verb interdependent**, also gegenseitig abhängig (sichtbar am **Verb-Agreement**)
 - Subjekt (Satzaussage) - Prädikat (Satzgegenstand)
 - abstrahiert von linearer Ordnung: **Wortstellung sprachabhängig**

- **VP = VERB + *dependents*** (Komplemente + Adjunkte)
 - Komplemente = obligatorische (valenzgeforderte) Erweiterungen
 - Adjunkte = nicht-obligatorisch Erweiterungen, Anzahl nicht begrenzt
- **NP = NOUN + *dependents*** (nominale Erweiterungen / Modifikatoren)

4.1.2 Konstituentenstruktur des Deutschen

- Phrasenschema VP:

V + (Komplemente|Adjunkte|ADV^P)^{*}

- Phrasenschema NP:

(DET | NP_{gen}) (ADJP)^{*} N (NP_{gen}|von-PP)^{*} (PP / RELS)^{*}

→ komplexer Aufbau der Nominalphrase

→ Links-und Rechtserweiterungen

Satzgrundstruktur

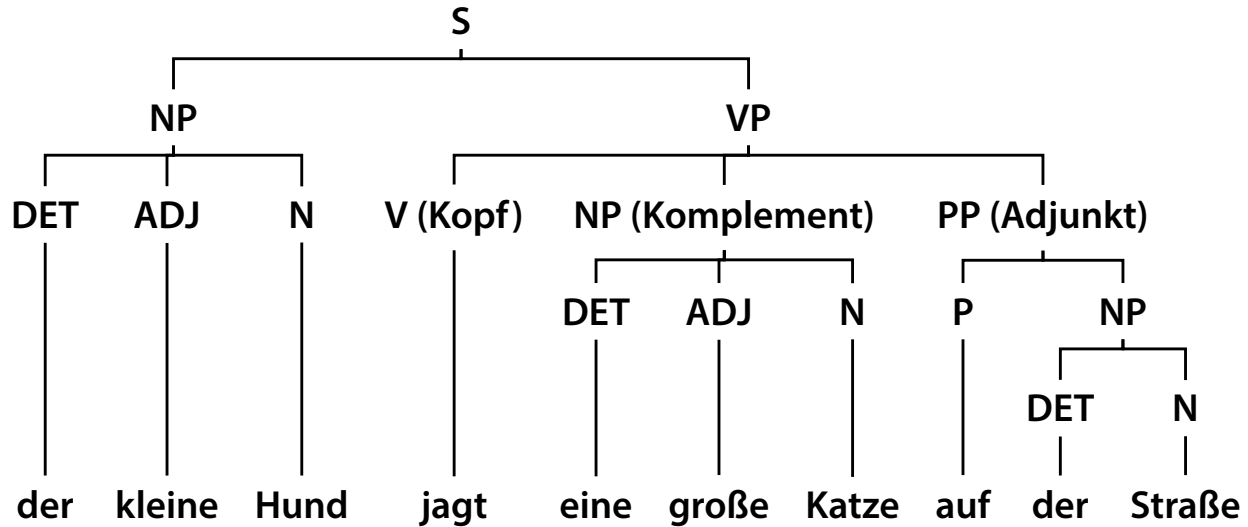


Abbildung 4: Phrasenstruktur Transativer Satz des Deutschen

NP-Struktur des Deutschen

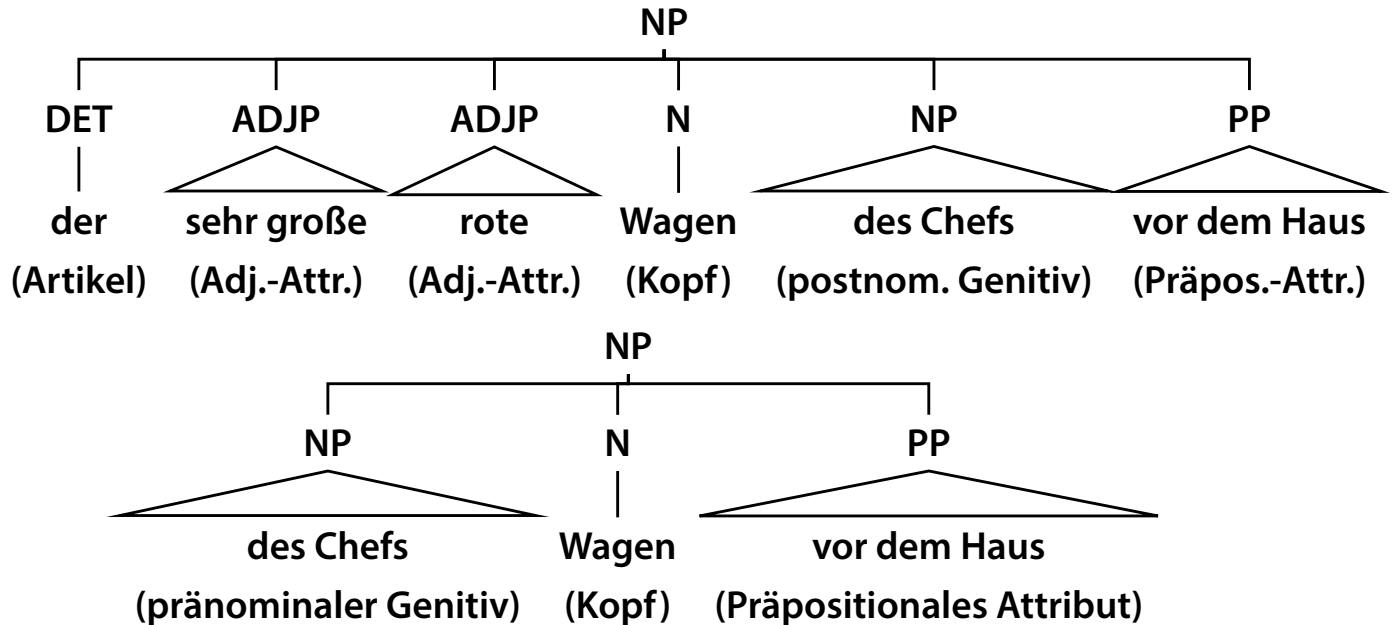


Abbildung 5: NP-Konstruktionen Deutsch (2: Genitivattribut als Determinativ-Vertreter)

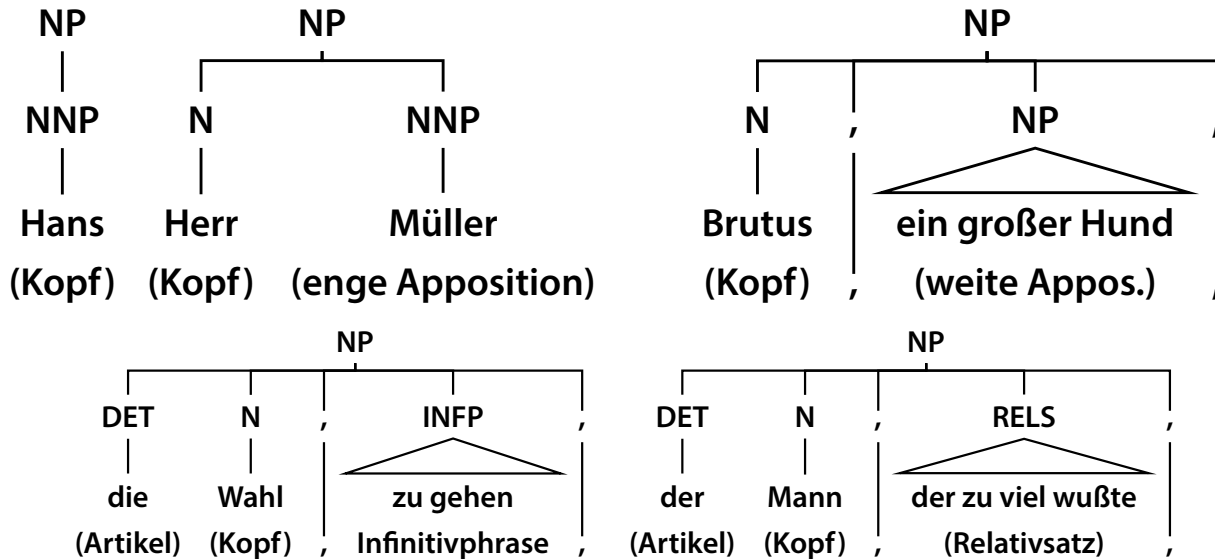


Abbildung 6: NP-Konstruktionen Deutsch

4.2 Modellierung mit kontextfreier Grammatik

4.2.1 Phrasenstrukturgrammatik

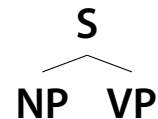
- **Syntax als Satzstrukturanalyse:**
 - Beschreibung der **syntaktischen Regeln, die den beobachtbaren Strukturen** zugrunde liegen (Grammatik)
- **Konstituenten-/Phrasenstruktur** natürlicher Sprache ist formal beschreibbar durch eine **kontextfreie Grammatik (CFG)**
 - **Phrasenstrukturgrammatik** im engeren Sinne (PSG)
 - Regeln der **Verkettung von lexikalischen und phrasalen Kategoriensymbolen** (Nicht-Terminale)

- **syntaktische Regel:** bestimmt, zu welchen **Klassen die unmittelbaren Konstituenten einer syntaktischen Kategorie** gehören: $NP \rightarrow DET N PP$
- **lexikalische Regel:** bestimmt die **Zugehörigkeit einer elementaren Konstituente (Wort) zu einer lexikalischen Kategorie:**
 $N \rightarrow Hund$

PSG-Regeln als Produktionsregeln:

- PSG-Regeln können als **Konstruktionsanweisung für Syntaxbäume** interpretiert werden: $S \rightarrow NP VP$

→ 'expandiere S zu Folge NP + VP'



- PSG-Regel definiert **Relation der unmittelbaren Dominanz zwischen Mutterknoten und Tochterknoten**

→ 'S dominiert unmittelbar NP und VP'

→ 'S dominiert vollständig die Folge NP + VP'

- PSG **erkennt durch Ableitung** Sätze als zur Sprache gehörig und weist ihnen die ihren Regeln entsprechende **Strukturbeschreibung** zu
 - Strukturbeschreibung = die auf Kategorien bezugnehmende **Konstituentenstruktur**
 - 'Die Folge NP + VP ist ein S'

Formale Grammatik:

- **besteht aus:**
 - **Startsymbol**
 - **Nichtterminalsymbole**
 - Metasymbole
 - **Terminalsymbole**
 - Alphabet, Lexikon
 - **Produktionsregeln**
 - durch Einschränkungen der Regeln ergeben sich Sprachen verschiedener Komplexität (Chomsky-Hierarchie)

Kontextfreie Grammatik:

- **CFG-Einschränkung:**
 - links nur ein Nichtterminalsymbol: $S \rightarrow NP VP$
 - Ersetzung unabhängig von Kontext (Kontextfreiheit)
- **syntaktische Regeln:** $NP \rightarrow Det N (PP)$
 - Ersetzungsregeln (linke mit rechter Seite)
 - links: syntaktische Kategorien (Phrasen/Satzknoten)
 - rechts: obligatorische und optionale Nichtterminale (syntaktische + lexikalische Kategorien)
- **Rekursion:** $NP \rightarrow Det N (PP), PP \rightarrow P NP$

- **lexikalische Regeln:**

$N \rightarrow \text{'Hund'}$

$N \rightarrow \text{'Katze'}$

$Det \rightarrow \text{'der'} \mid \text{'die'}$

→ Zuordnung lexikalische Kategorien/Wortarten (Präterminale) zu Lexemen (Terminale)

- **Wortarten (=lexikalische Kategorien):** → *Präterminale*
- **Lexeme:** → *Terminale (Alphabet)*

PSG-Regel-Konventionen:

- Regel mit fakultativen Elementen:

$NP \rightarrow (DET) N$

äquivalent zu:

$NP \rightarrow DET N \mid N$

äquivalent zu:

$NP \rightarrow DET N$

$NP \rightarrow N$



Ableitung als top-down Erzeugung (Linksableitung):

$$G = \langle \{S, NP, VP, DET, N, V\}, \{das, Tier, Ding, sieht\}, R, S \rangle$$
$$R = \{S \rightarrow NP VP, NP \rightarrow DET N, VP \rightarrow V NP,$$
$$DET \rightarrow das, N \rightarrow Tier, N \rightarrow Ding, V \rightarrow sieht\}$$

S	⇒	NP VP	(S → NP VP)
	⇒	DET N VP	(NP → DET N)
	⇒	das N VP	(DET → das)
	⇒	das Tier VP	(N → Tier)
	⇒	das Tier V NP	(VP → V NP)
	⇒	das Tier sieht NP	(V → sieht)
	⇒	das Tier sieht DET N	(NP → DET N)
	⇒	das Tier sieht das N	(DET → das)
	⇒	das Tier sieht das Ding	(N → Ding)

Auflistung 1: *NLTK: Kontextfreie Grammatik (grammar1.cfg)*

```
1 ##### Syntaktische Regeln #####
2     S → NP VP
3     NP → DET N
4     VP → V NP NP
5
6 ##### Lexikalische Regeln #####
7     DET → "der" | "die" | "das"
8     N → "Mann" | "Frau" | "Buch"
9     V → "gibt" | "schenkt"
```

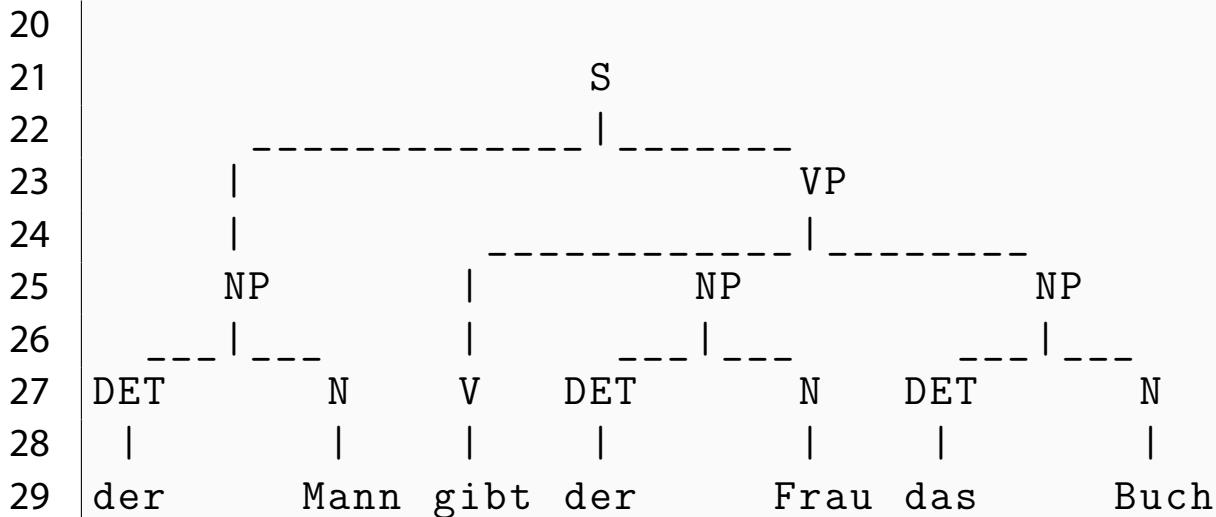

Auflistung 2: NLTK: Erkennung und Strukturwiedergabe

```
1
2 grammar = nltk.CFG.fromstring("""
3     S → NP VP
4     NP → DET N
5     DET → "der" | "die" | "das"
6     N → "Mann" | "Frau" | "Buch"
7     VP → V NP NP
8     V → "gibt" | "schenkt"
9     """)
10
11 sent = 'der Mann gibt der Frau das
12     Buch'.split()
13
14 parser = nltk.ChartParser(grammar, trace=0)
```

```

15 for tree in parser.parse(sent):
16     print(tree); tree.pretty_print()
17 #(S
18 #  (NP (DET der) (N Mann))
19 #  (VP (V gibt) (NP (DET der) (N Frau)) (NP
    (DET das) (N Buch))))

```



```
30  
31  
32 #Übergenerierung:  
33 for sentence in generate(grammar, depth=5):  
34     print(' '.join(sentence))  
35 der Mann gibt der Mann der Mann  
36 der Mann gibt der Mann der Frau  
37 der Mann gibt der Mann der Buch  
38 der Mann gibt der Mann die Mann  
39 der Mann gibt der Mann die Frau  
40 der Mann gibt der Mann die Buch  
41 der Mann gibt der Mann das Mann  
42 der Mann gibt der Mann das Frau  
43 der Mann gibt der Mann das Buch
```

Rekursive Kategorien

- **Rekursiver Aufbau natürlicher Sprachen**
 - erklärt **Nicht-Endlichkeit** natürlicher Sprachen
 - endliche Anzahl an syntaktischen Regeln, aber unbegrenzte Anzahl an bildbaren Sätzen (z. B.: durch Hinzufügung von Adjunkten)
 - modellierbar mit **rekursiven Phrasenstrukturregeln**
 - Erzeugung **verschachtelter Strukturen**
- **rekursive Kategorien:** NP, PP, Relativsätze, Komplementsätze

- **direkte Rekursion:** Ausgabe einer Regel dient als Eingabe der gleichen Regel:

NP → DET N **NP**

der Sohn der Frau des Mannes ...

- **indirekte Rekursion:** Ausgabe einer Regel dient als Eingabe einer anderen Regel:

NP → DET N **PP**

PP → P **NP**

der Mann in dem Wald in dem Land ...

neben dem Mann in dem Wald in dem Land ...

Auflistung 3: *Grammatik mit Rekursion*

- | | | |
|---|--|-------------------------------------|
| 1 | | $S \rightarrow NP$ |
| 2 | | $PP \rightarrow P \ NP$ |
| 3 | | $NP \rightarrow N \mid N \ PP$ |
| 4 | | $N \rightarrow 'Hund' \mid 'Katze'$ |
| 5 | | $P \rightarrow 'mit'$ |

Auflistung 4: NLTK: Generierung von Sätzen mit rekursiver Kategorie

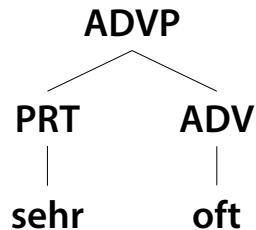
```
1 # http://www.nltk.org/howto/generate.html
2 from nltk.parse.generate import generate
3
4 grammar = nltk.CFG.fromstring("""
5     S → NP
6     PP → P NP
7     NP → N | N PP
8     N → 'Hund' | 'Katze'
9     P → 'mit'
10    """)
11 for sentence in generate(grammar, depth=8):
12     print(' '.join(sentence))
13 # Hund
14 # Katze
15 # Hund mit Hund
```

```
16 # Hund mit Katze
17 # Hund mit Hund mit Hund
18 # Hund mit Hund mit Katze
19 # Hund mit Katze mit Hund
20 # Hund mit Katze mit Katze
21 # Katze mit Hund
22 # Katze mit Katze
23 # Katze mit Hund mit Hund
24 # Katze mit Hund mit Katze
25 # Katze mit Katze mit Hund
26 # Katze mit Katze mit Katze
```


4.2.2 Phrasenstrukturregeln des Deutschen

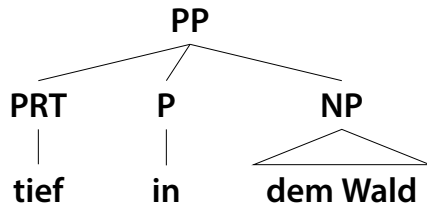
Adverbphrase ADVP, AVP:

- Phrasenschema ADVP: (PRT) ADV
- Produktionsregeln ADVP:

$$ADVP \rightarrow PRT ADV$$
$$ADVP \rightarrow ADV$$


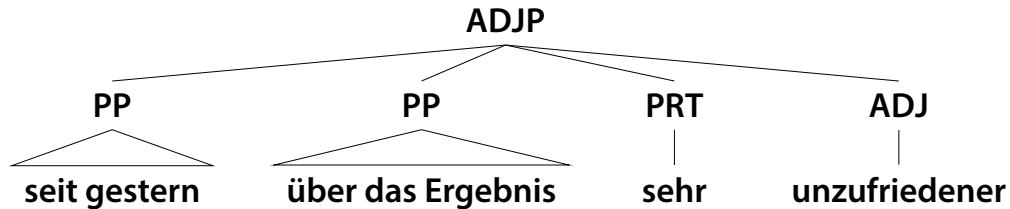
Präpositionalphrase PP:

- Phrasenschema PP: (PRT) P NP
- Produktionsregeln PP:

$$PP \rightarrow P NP$$
$$PP \rightarrow PRT P NP$$


Adjektivphrase ADJP, AP:

- Phrasenschema ADJP: **(PP)* (PRT) ADJ**



- Produktionsregeln ADJP (flacher Aufbau ohne Rekursion):

ADJP → *ADJ*

ADJP → *PRT ADJ*

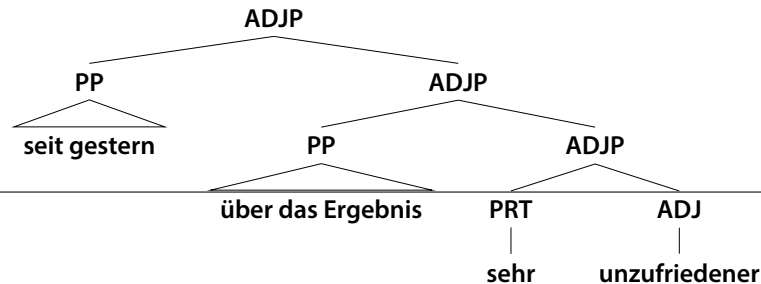
ADJP → *PP ADJ*

ADJP → *PP PRT ADJ*

ADJP → *PP PP ADJ* usw.

Adjektivphrase mit rekursivem Strukturaufbau:

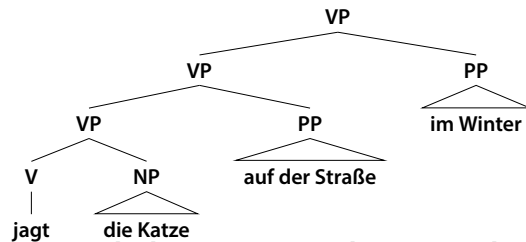
- Phrasenschema ADJP: $(PP)^* (PRT) \quad \underline{ADJ}$
- Produktionsregeln ADJP (rekursiver Aufbau):
 $ADJP \rightarrow PP \ ADJP$ (rekursive Regel)
 $ADJP \rightarrow PRT \ ADJ$
 $ADJP \rightarrow ADJ$



Verbalphrase VP:

- Phrasenschema VP (vereinfacht):

V + Komplemente + (Adjunkte)*



- Produktionsregeln VP (ohne AdvP):

$VP \rightarrow VP\ PP$ (Adjunkte)

$VP \rightarrow V\ NP$ (1 Komplement)

$VP \rightarrow V$ (ohne Komplemente, intransitiv)

Nominalphrase NP:

- Phrasenschema NP (vereinfacht):

(DET) (ADJP)* N (PP)*

- Produktionsregeln NP (flacher Aufbau ohne Rekursion):

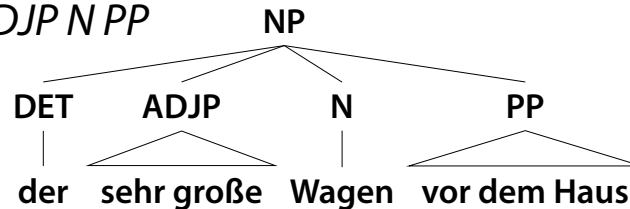
$NP \rightarrow N \mid DET N$

$NP \rightarrow ADJP N \mid DET ADJP N$

$NP \rightarrow N PP \mid DET N PP$

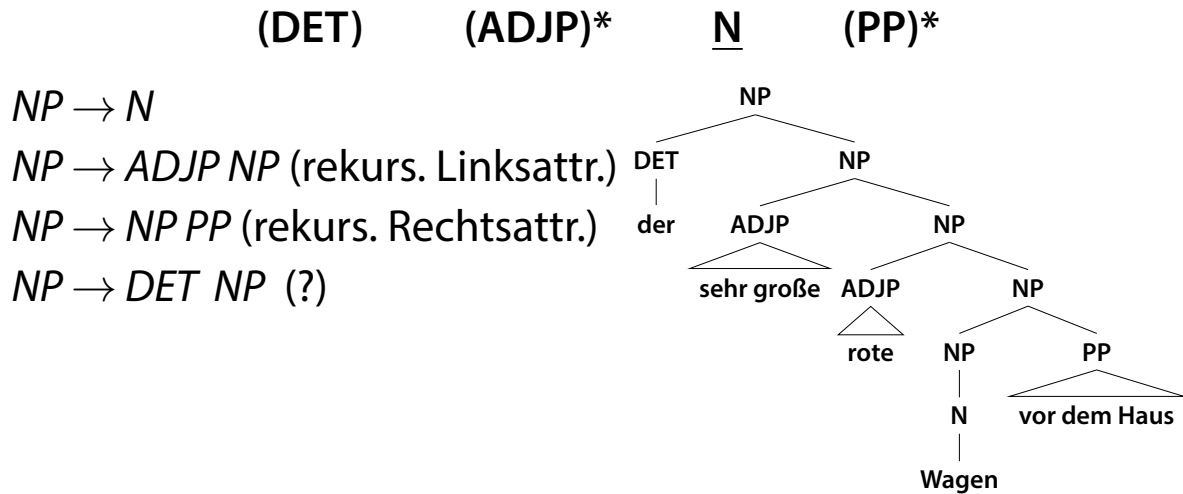
$NP \rightarrow ADJP N PP \mid DET ADJP N PP$

usw.



Nominalphrase mit rekursivem Strukturaufbau (vorläufig):

- Phrasenschema NP (vereinfacht):



Problem rekursive DET-Regel

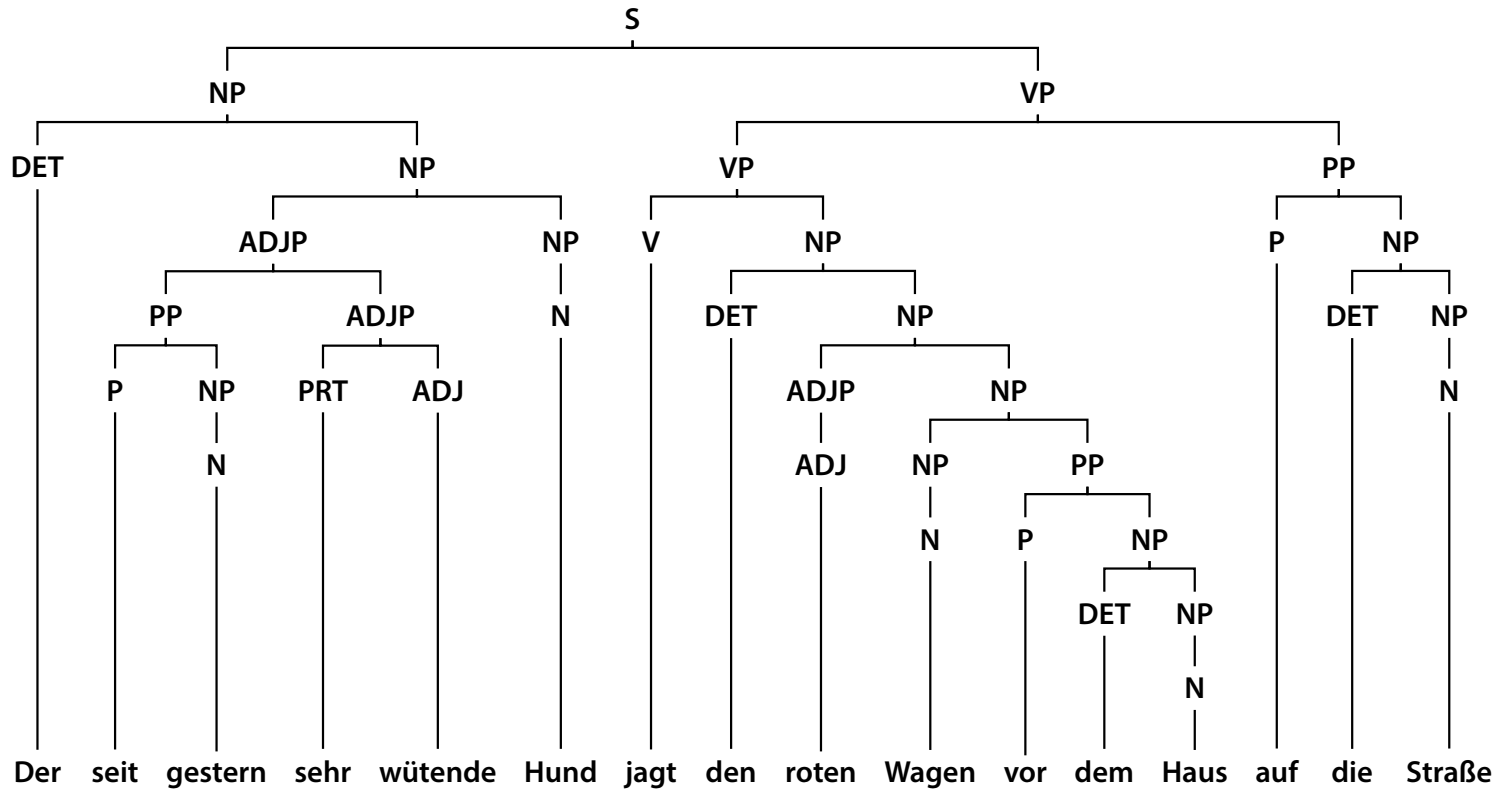
- Mit der Regel $NP \rightarrow DET\ NP$ können zwar NPs gemäß des Phrasenschemas abgeleitet werden
- die Grammatik ist aber stark **übergenerierend** und damit kein adäquates Modell des NP-internen Strukturaufbaus
 - **Übergenerierung 1**: DET rekursiv wiederholbar an erster Position (richtige Strukturposition aber falsche Anzahl)
 - **Übergenerierung 2**: DET wiederholbar an falscher Strukturposition (z. B.: ADJP DET N)
- Strukturbegrenzung notwendig (→ X-Bar)

Auflistung 5: *NLTK: Erkennung und (Über-)Generierung vorläufiger NP-Grammatik*

```
1 grammar = nltk.CFG.fromstring("""
2     NP → ADJP NP
3     NP → NP PP
4     NP → N
5     NP → DET NP
6     N → 'Nomen'
7     DET → 'das'
8     ADJP → 'schöne'
9     PP → 'darin'
10    """)
11 sent = 'das schöne schöne Nomen darin
12       darin'.split()
13 parser = nltk.ChartParser(grammar, trace=0)
14 for tree in parser.parse(sent):
15     print(tree)
```

```
15 (NP
16   (NP
17     (NP
18       (DET das)
19       (NP (ADJP schöne) (NP (ADJP schöne) (NP
20         (N Nomen))))))
21     (PP darin))
22   (PP darin))
23 for sentence in generate(grammar, depth=9):
24     print(' '.join(sentence))
25     #...
26     #das das schöne das schöne Nomen darin
27     #...
```

Satzableitung aus den Grammatikregeln (mit $S \rightarrow NP VP$)



4.3 X-Bar-Phrasenstrukturschema

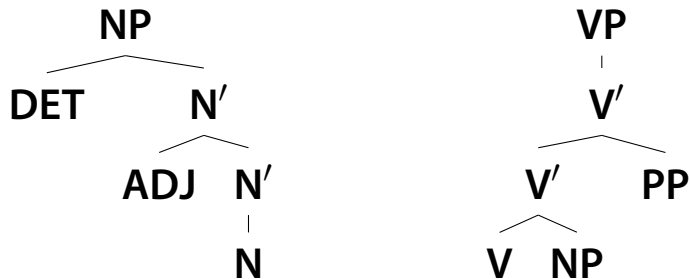
- Entwicklung durch Chomsky im Rahmen der *Government & Binding*-Theorie
- ursprüngliches Symbol: Balken (*bar*): \bar{X}
- **Beschränkung der Struktur von Phrasen**
 - **binäre Verzweigung** ($A \rightarrow B C$)
 - Einführung **phrasaler Zwischenebene** (\bar{X} oder X')
 - **gleicher Strukturaufbau für alle Phrasenarten** (X als Wortart-Variable)

Motivation für X-Bar-Schema

- bisher: **Anzahl und Art unmittelbarer Konstituenten einer Phrase nicht beschränkt**
 - Mischung aus nicht festgelegter Anzahl aus lexikalischen und phrasalen Kategorien
 - keine festgelegte Ordnung zwischen Kopf und Schwestern
- führt zu Problemen bei rekursiver Strukturdefinition (s. oben)

X-Bar-Ebene

- Einführung **phasaler Zwischenebene** (X') zwischen **Gesamtphrase** (XP, maximale Projektion) und **Kopf** (X)



- Erlaubt die Differenzierung verschiedener Arten von *dependents* in Phrase durch Strukturposition

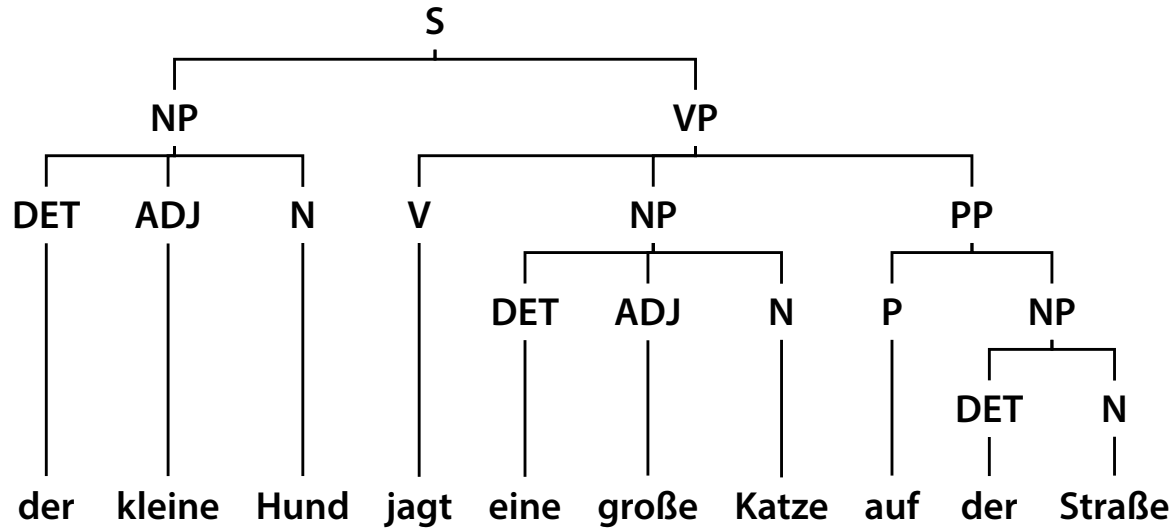


Abbildung 7: zum Vergleich: nicht-beschränkte Phrasenstruktur

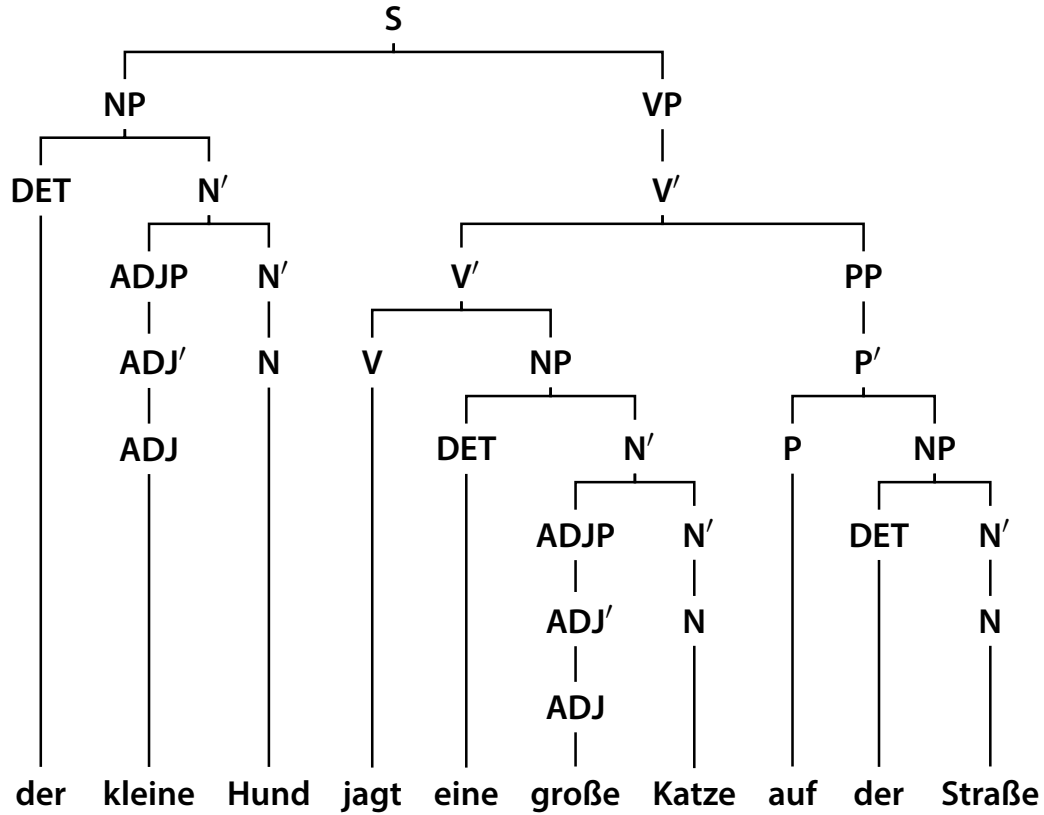


Abbildung 8: X-Bar-Analyse

- **Verallgemeinerung der verbalen Argument-Adjunkt-Distinktion:**
- **Komplement:** obligatorische (valenzgeforderte) Erweiterung, Anzahl abhängig von Valenz/Subkategorisierungsrahmen
 - direkte Verbindung mit Phrasenkopf X, bilden zusammen X'-Phrase
 - Modellierung der engen Verbindung Komplement mit Kopf
 - Deutsche NP: Genitiv-NP-Erweiterung oder von-PP

- **Adjunkt:** nicht-obligatorische Erweiterung, Anzahl nicht begrenzt
 - Verbindung mit X'-Konstituente, bilden zusammen wieder X'
 - Deutsche NP: Adjektiv-Attribut, PP-Attribut

- **Spezifizierer:** phrasenabschließende nicht-obligatorische Erweiterung, Anzahl begrenzt (nur ein Element)
 - Verbindung mit X', bilden zusammen den Abschluß der XP-Phrase
 - Deutsche NP: Determinierer, Possessivpronomen, Quantifizierer, sächsischer Genitiv (*seines Glückes Schmied*)
 - verallgemeinert: als Strukturposition (XP → (SPEC) X') im Schema für Elemente mit entsprechenden Eigenschaften
 - z. B. AUX als VP-Spezifizierer (s. unten)

Allgemeines X-Bar-Schema

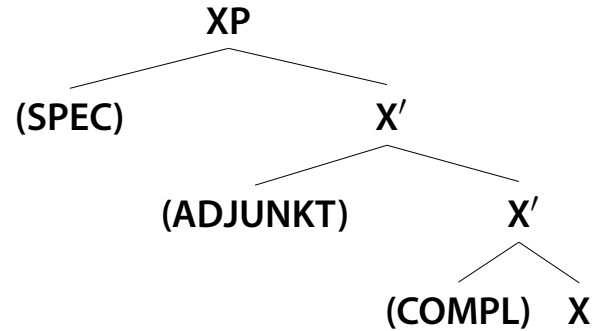


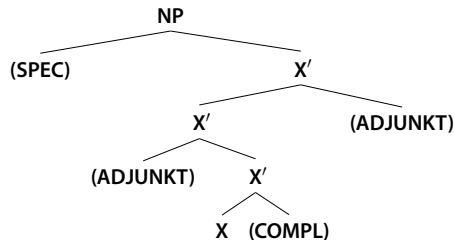
Abbildung 9: Allgemeines X-Bar-Schema

$XP \rightarrow (SPEC), X'$

$X' \rightarrow (ADJUNKT), X'$ (rekursive Regel)

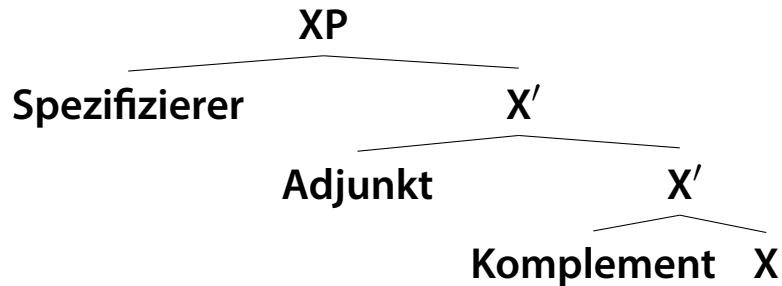
$X' \rightarrow (COMPL), X$

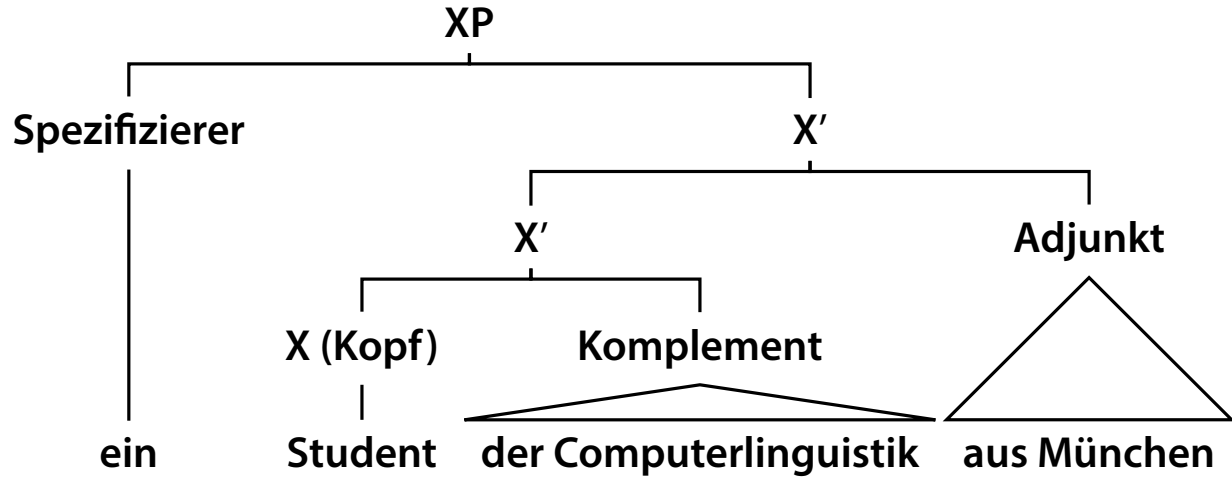
- X-Bar-Schema: **ohne implizierte lineare Struktur** (Wortstellung)
 - Einzelsprachliche Regeln
- **Links- vs. Rechtsverzweigung** (*left- vs rightbranching*)
 - Linksverzweigung: **head-final** (OV-Sprachen)
 - Rechtsverzweigung: **head-initial** (VO-Sprachen)
 - **Deutsche NP: Links- und Rechtsverzweigend**



- **Strukturpositionen im X-Bar-PSG-Schema:**

	X (Kopf)	X'	XP
Komplement	Schwester	Tochter	
Adjunkt		Schwester und Tochter	
Spezifizierer		Schwester	Tochter



Abbildung 10: *allgemeine X-Bar-Analyse am Beispiel NP*

X-Bar-CFG-Regeln:

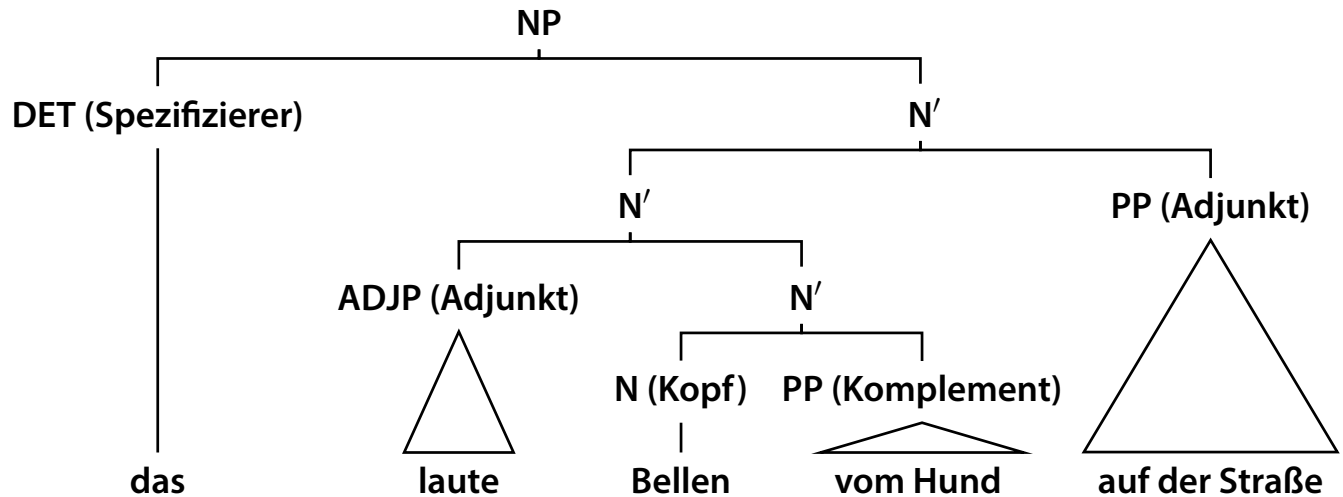


Abbildung 11: X-Bar-Analyse NP mit Komplement-Adjunkt-Spezifizierer-Distinktion

Auflistung 6: *PSG-Regeln X-Bar-Analyse NP*

- 1
- 2 #X-Bar-NP-Regeln:
- 3 NP \rightarrow DET N' #SPECIFIER
- 4 N' \rightarrow ADJP N' #ADJ-ADJUNKT
- 5 N' \rightarrow N' PP #PP-ADJUNKT
- 6 N' \rightarrow N PP #PP-KOMPLEMENT
- 7 N' \rightarrow N

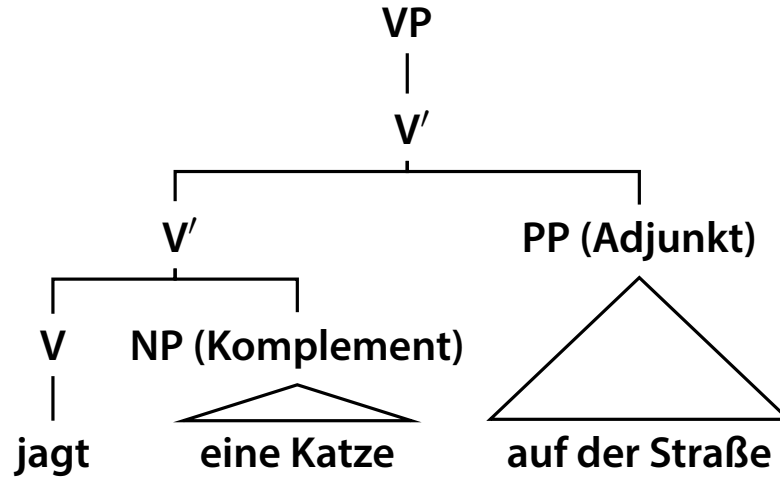


Abbildung 12: X-Bar-Analyse VP mit Komplement-Adjunkt-Distinktion

Auflistung 7: *PSG-Regeln X-Bar-Analyse VP*

- | | | | |
|---|--|-------------------|--------------------|
| 1 | | #X-Bar-VP-Regeln: | |
| 2 | | VP → V' | |
| 3 | | V' → V' PP | #V' + ADJUNKT |
| 4 | | V' → V NP | #KOPF + KOMPLEMENT |
| 5 | | V' → V #KOPF | (OHNE KOMPLEMENT) |

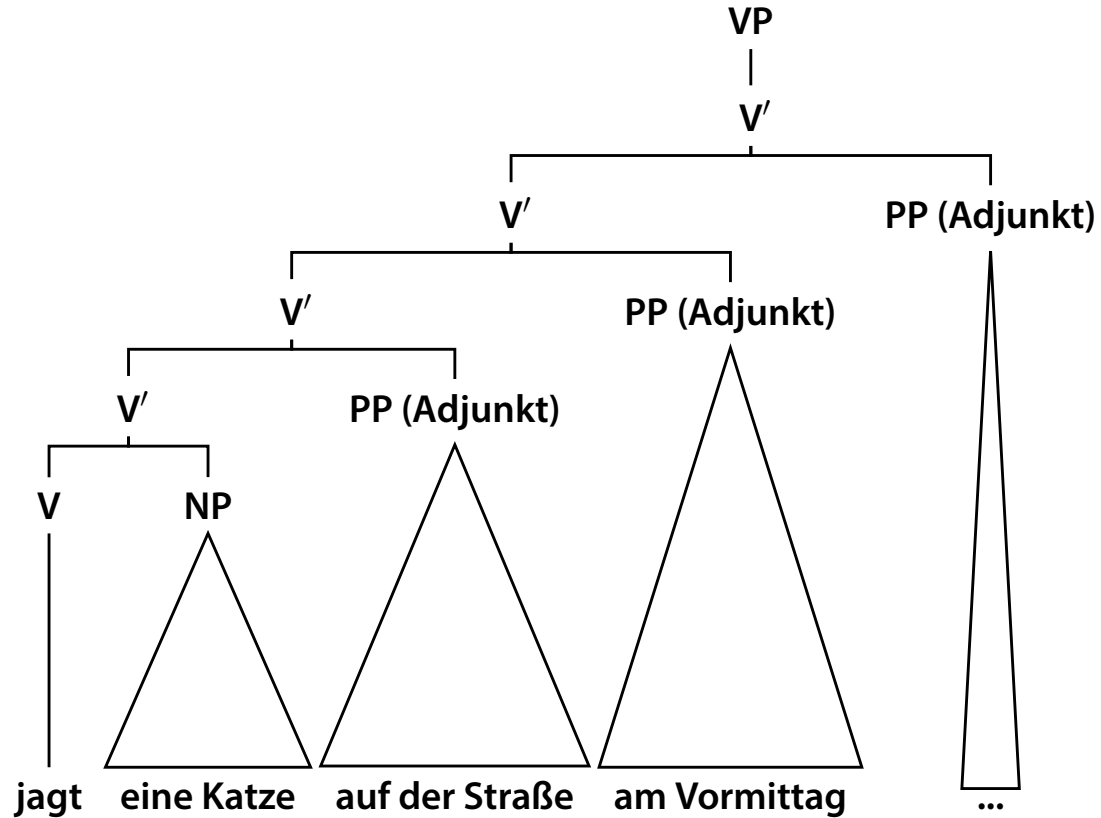


Abbildung 13: X-Bar-Analyse VP mit rekursiver Adjunktionsregel

Parallele X-Bar-Strukturanalyse für NP und VP:

Auxiliar als Spezifizierer

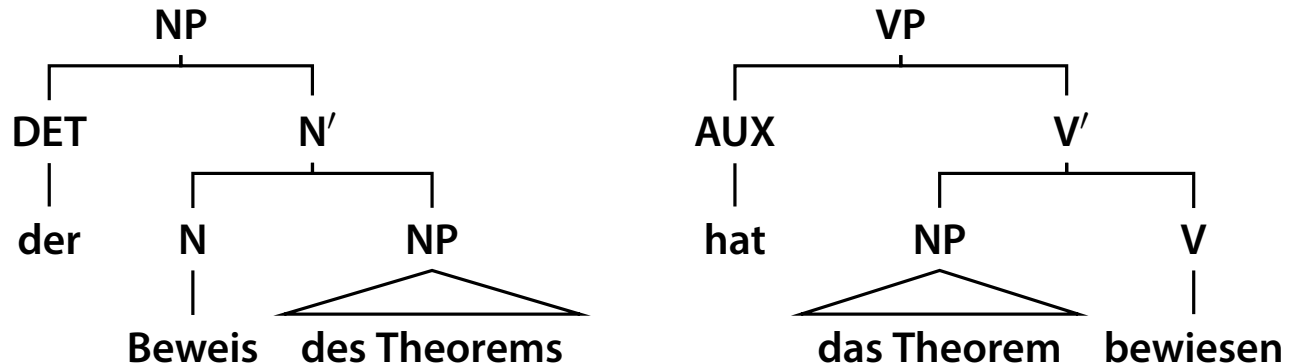


Abbildung 14: Parallele X-Bar-Strukturanalyse für NP und VP (Spezifizierer und Komplement)

Auxiliarkonstruktionen

- **Auxiliare = Hilfsverben**
 - begleiten Verb (Träger lexikalischer Bedeutung)
 - Ausdruck von **grammatischen Merkmalen** (Tempus, Modus; Diathese; Flexionsmerkmale)
- Analyse ist stark **theorieabhängig** (Strukturposition)
- u. a. als Spezifizierer (s. o.)
- ebenso: Analyse als Verbgruppe (Verb + Auxiliare)
- ebenso: Eintrag in Subkategorisierungslexikon

4.4 Adäquatheit einer CFG als Syntaxmodell

- X-Bar-Phrasenstrukturgrammatiken schränken zwar durch ihre Strukturrestriktionen die Übergenerierung ein
- Übergenerierung bleibt aber weiter bestehen

Gründe für Übergenerierung

- Nichtberücksichtigung von **Morphologie** (Kongruenz, Rektion):
**der Mann sieht des Kindes*
**der Mann sieht das Kinder*
- Nichtberücksichtigung von **Subkategorisierung** (Verbvalenzrahmen)
**der Hund starb die Katze*

Lösungen

- **komplexere atomare Kategorien**

- Problem: Regelvervielfachung

- z.B. Numerus-Kongruenz NP: SgN, PlN, SgDET, PlDET, SgNP, PlNP; SgNP → SgDET SgN, PlNP → PlDET PlN

- **Merkmale in Lexikon**

- Merkmalsstrukturen und Unifikation (Feststellung Merkmalskongruenz)

- Auswahl durch **probabilistisches Modell** (PCFG)

- Übergenerierung erlauben

- ungrammatische Sätze als unwahrscheinliche ausschließen

4.5 Treebanks und Grammatiken

- **Treebank** = Sammlung von per Hand annotierten Syntaxbäumen in bestimmtem Annotationsschema
- **Penn-Treebank**
 - relativ flache Struktur
 - arbeitet mit *traces* (Label: NONE) um *long-distance-dependencies* zu markieren
 - Schema verwendet im englischen Stanford-Parser-Modell

- **NEGRA-Korpus**

- Grundlage deutsches Stanford-Parser-Modell
- TIGER-Annotationsschema basiert auf NEGRA-Schema
- noch flacher als Penn-Treebank

- ***grammar induction***

- Treebanks als **implizite Grammatik**
- **CFG-Regeln können aus Treebank-Korpus gewonnen werden**

- **tgrep**: Programm zur Suche in Syntaxbäumen (s. u.)

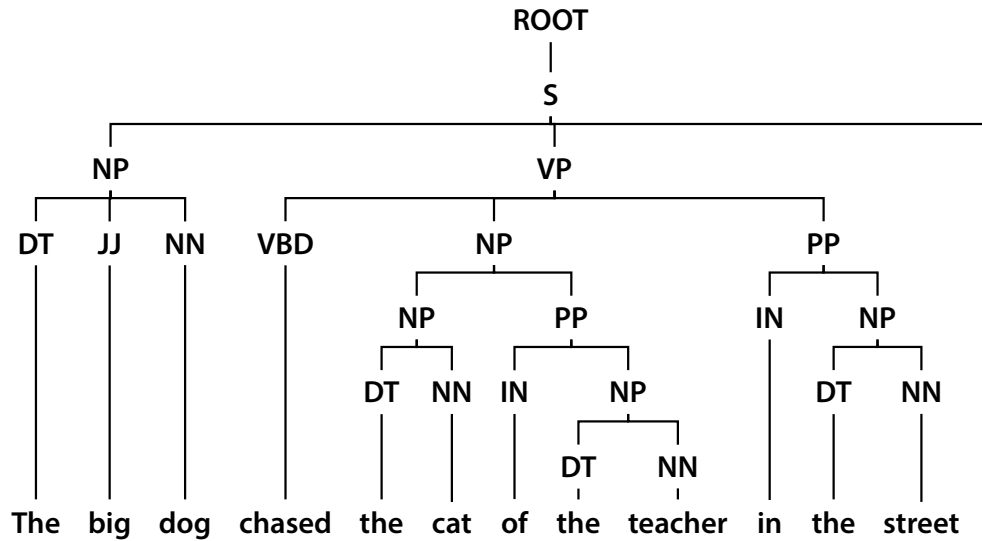
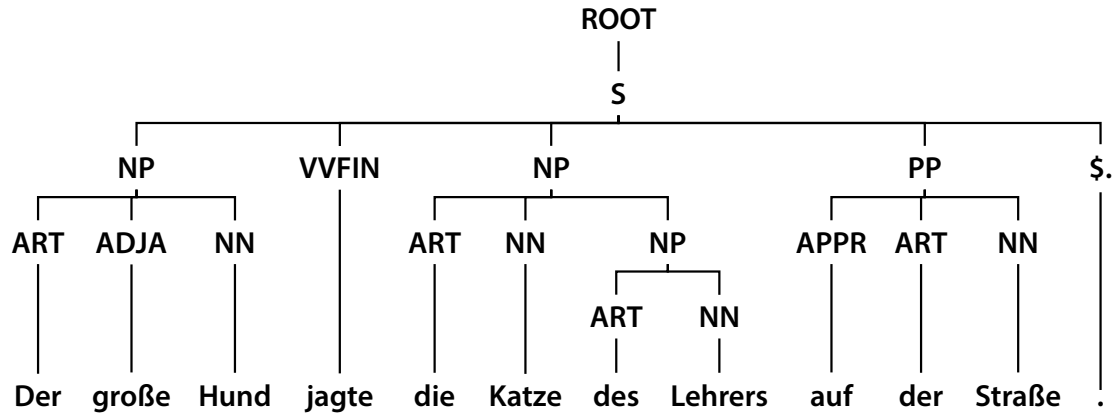


Abbildung 15: Beispiel-Parse Stanford-Parser

Abbildung 16: *Beispiel-Parse Stanford-Parser (Deutsches Modell)*

Auflistung 8: NLTK: Parsing mit Stanford-Parser; verschiedene Ausgabeformate

```
1 http://www.nltk.org/howto/tree.html
2
3 from nltk.parse.stanford import StanfordParser
4 jar = "stanford-corenlp-3.8.0.jar"
5 model = "stanford-corenlp-3.8.0-models.jar"
6 parser=StanfordParser(jar,model,model_path="edu/stanf
7
8 tree_list = list(parser.raw_parse('The big dog
    chased the cat of the teacher in the
    street.'))
9 tree = tree_list[0]
10
11
12
13
```

```
14 print(tree)
15 # (ROOT
16 #   (S
17 #     (NP (DT The) (JJ big) (NN dog))
18 #     (VP
19 #       (VBD chased)
20 #       (NP
21 #         (NP (DT the) (NN cat))
22 #         (PP (IN of) (NP (DT the) (NN
23 #           teacher))))))
24 #     (PP (IN in) (NP (DT the) (NN street))))
25 #   (. .)))
26
27
28
```



```
29 print(tree.pformat_latex_qtree())
30 # \Tree [.ROOT
31 #       [.S
32 #       [.NP [.DT The ] [.JJ big ] [.NN
33 #         dog ] ]
34 #       [.VP
35 #       [.VBD chased ]
36 #       [.NP
37 #       [.NP [.DT the ] [.NN cat ] ]
38 #       [.PP [.IN of ] [.NP [.DT the ]
39 #         [.NN teacher ] ] ] ]
40 #       [.PP [.IN in ] [.NP [.DT the ]
41 #         [.NN street ] ] ] ]
42 #       [.. . ] ] ]
```

```
42 tree.label()
43 #'ROOT'
44
45 print(tree[0,0])
46 #(NP (DT The) (JJ big) (NN dog))
47
48 print(tree[0,2])
49 #(. .)
50
51 print(tree[0,1,2])
52 #(PP (IN in) (NP (DT the) (NN street)))
53
54 print(tree[0,0,0]); print(tree[0,0,1])
55 #(DT The)
56 #(JJ big)
```

Auflistung 9: *nltk_tgrep*

```
1 #https://pypi.python.org/pypi/nltk\_tgrep/
2 sudo pip install nltk_tgrep
3
4 from nltk.tree import ParentedTree
5 import nltk_tgrep
6 tree = ParentedTree.fromstring('(S (NP (DT
   the) (JJ big) (NN dog)) (VP bit) (NP (DT a)
   (NN cat))))')
7 nltk_tgrep.tgrep_nodes(tree, 'NN')
8 #\[ParentedTree\('NN', \['dog'\]\),
   ParentedTree\('NN', \['cat'\]\)\]
9 nltk_tgrep.tgrep_positions(tree, 'NN')
10 #\[\(0, 2\), \(2, 1\)\]
11 nltk_tgrep.tgrep_nodes(tree, 'DT $ JJ')
12 #\[ParentedTree\('DT', \['the'\]\)\]
```