

Wiederholung Syntax Natürlicher Sprachen

Schwerpunkte wurden aus den vorherigen Tutoriumfolien zusammengefasst

Shuyan Liu 31.01.2025

Einige Beispiele kommt aus der Vorlesungsfolien, Aufgaben sowie Übungen.

Die Hauptteile der Tutoriumfolien stammen von Sarah Anna Uffelmann aus Wintersemester 2023/24 und wurden bearbeitet.

Verwendung mit Dank.

Termine

- Es gibt noch ein Tutorium am 14.02.2025
- Plan: Fragestunde

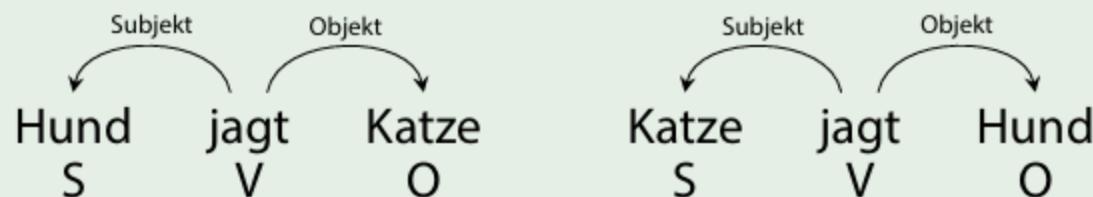
1. Einführung

Wortstellung, Kongruenz, Kasus, Konstituenten und
Konstituententests, Komplement und Adjunkt

Wortstellung

- die Reihenfolge, in der Wörter in einem Satz angeordnet sind.
 - z.B. SVO (Subject, Verb, Object); SOV (Dt. Nebensatz), usw.

SVO-Wortstellung



Kasus

- Der Kasus zeigt an, ob das Wort als Subjekt, direktes Objekt, indirektes Objekt, Besitzanzeige usw. fungiert. Im Deutschen gibt es vier Hauptkasus: Nominativ, Akkusativ, Dativ und Genitiv.

Kongruenz

- Auch als Agreement genannt.
- Grammatische Übereinstimmung zwischen verschiedenen Elementen eines Satzes.
- Beispiele: der große Hund; Ich gehe

Subjekt-Kongruenz (Numerus und Person)



→ Verb kongruiert in nominalen Kategorien (Numerus, Person) mit Subjekt-NP

- (Argument für Regel: $S \rightarrow NP\ VP$ (Subjekt > Objekt))

Konstituenten

- Konstituenten sind sprachliche Einheiten, die Teile einer größeren Einheit bilden, wie Wörter, Phrasen oder Teilsätze.
- Sätze sind in Konstituenten zerlegbar

Kastendiagramm

- Im Kastendiagramm ersetzen wir nach und nach Konstituenten durch einzelne Wörter, bis wir zu einem 2-Wort-Satz(in der Regel NP+VP) kommen.

Die	Kunst	des	Ausruhen s	ist	ein	Teil	der	Kunst	des	Arbeitens
Die	Kunst	daran		ist	ein	Teil	der	Kunst	daran	
Die	Syntax			ist	ein	Teil	der	Grammatik		
Sie				ist	ein	Teil	davo n			
Sie				ist	ein	Theoriekonstrukt				
Sie				ist	großartig					
Sie				geht						

Konstituententests

Zum Identifizieren der Konstituenten eines Satzes verwenden wir Konstituententests. Wir verändern dabei Teile Satzes unter Einhaltung der Wohlgeformtheit (Grammatikalität) des Satzes.

- Substitutionstest: Austauschbarkeit im gleichen Kontext
- Permutationstest: Wortfolge verschieben (man kann auch den Satz nach Fragesatz umstellen)
- Eliminierungstest: Wortfolge weglassen
- Koordinationstest: Ergänzung von Konstituenten **des gleichen Typs**

Vorsicht: Eine Konstituente kann manchmal beim Test durchfallen, bleibt aber dennoch als Konstituente bestehen.

Adjunkt vs. Komplement

Adjunkt:

- Nicht-notwendiges Satzglied
- Auftreten ist nicht von Kopf (meist Verb) gefordert
- Zusätzliche Information
- Immer eliminierbar

Komplement:

- Notwendiges Satzglied
- Vom Kopf gefordert
- kann kontextabhängig eliminierbar sein

Adjunkt vs. Komplement

Tests (Auf die Tests ist nicht zu 100% Verlass):

Pass: Nach dem Test ist der Satz noch grammatisch korrekt

- **Eliminierungstest:**
 - Adjunkt: Eliminierbar
- **Adverbialsatztest: als er ... war**
 - Adjunkt: Lässt sich in „...“ plazieren
- **Geschehen-test**
 - Adjunkt: Pass the test

Obligatorische, fakultative und optionale Satzglieder

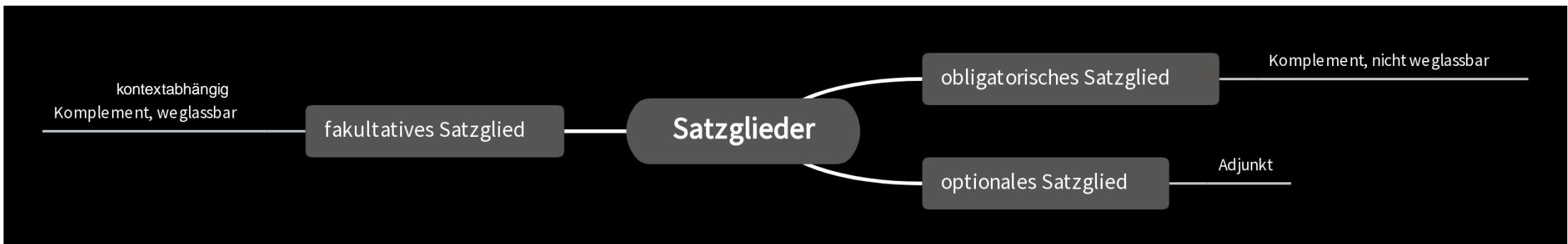
Obligatorisches Satzglied: Komplement, nicht weglassbar

Fakultatives Satzglied: Komplement, aber nach Kontext weglassbar(Achtung auf Kopf)

Beispiel: Er sieht den Hund. Der Kopf „sehen“ fordert ein Subjekt und ein Objekt. Aber der Satz „Er sieht“ macht auch Sinn.

Optionales Satzglied: Adjunkt

Obligatorische, fakultative und optionale Satzglieder



2. Phrasenstrukturgrammatik

Dt. Wortarten, Konstituente vs. Phrase, Phrasen,
Phrasenstrukturgrammatik, Kopfperkolation, Regeltypen

Wort

- Wörter sind Terminate von einem Syntaxbaum
- Wörter mit gemeinsamen Eigenschaften bilden unterschiedliche Klassen von Wortarten

Dt. Wortarten

- Lexikalische Hauptkategorien(Kopffähig):

Nomen(NP), Verben(VP), Adjektive(ADJP),
Adverbien(ADVP)

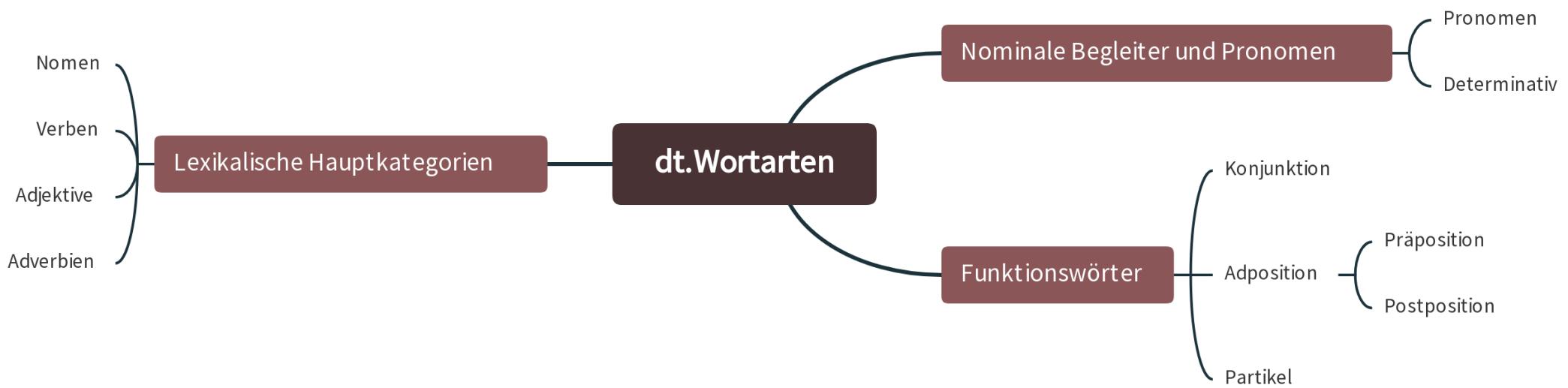
- Nominale Begleiter und Pronomen

Pronomen, Determinativ

- Funktionswörter

Adposition Prä- und Postposition(PP), Konjunktion, Partikel

Dt. Wortarten



Detail siehe “syntax_t_02“

POS Tag

- Wir taggen Wörter nach ihren Wortarten.
- Tagsets

Universal POS tags

- ADJ: adjective
- ADP: adposition
- ADV: adverb
- AUX: auxiliary
- CCONJ: coordinating conjunction
- DET: determiner
- INTJ: interjection
- NOUN: noun
- NUM: numeral
- PART: particle
- PRON: pronoun
- PROPN: proper noun
- PUNCT: punctuation
- SCONJ: subordinating conjunction
- SYM: symbol

Penn Treebank POS tags

12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural

27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present

Tiger / STTS POS tags

VVFIN	finites Vollverb
VAFIN	finites Voll- oder Kopulaverb
VMFIN	finites Modalverb
VVINF	infinites Vollverb
VAINF	infinites Hilfsverb oder Kopulaverb
VMINF	infinites Modalverb
VVIMP	Vollverb im Imperativ
VAIMP	Kopulaverb im Imperativ
VVPP	partizipiales Vollverb (Partizip II)
VAPP	partizipiales Hilfs-/Kopulaverb (Partizip II)
VMPP	partizipiales Modalverb (Partizip II)
VVIZU	Vollverb/Partikelverb im "zu"-Infinitiv

Konstituente

Alle Phrasen sind Konstituente, aber nicht alle Konstituenten sind Phrasen.

Wie z.B:

Einzelne Wörter eines Satzes.

In **Der Hund jagt die Katze** ist „Hund“ eine Konstituente, aber keine Phrase.

Vorsicht: Einzelne Wörter können auch Phrasen sein!

In **Er jagt die Katze** ist „Er“ sowohl Konstituente als auch Phrase.

Phrasenkategorien des Deutschen

- NP – Nominalphrase: “der **Elefant**“
- VP – Verbalphrase: “ins Kino **gehen**“
- PP – Präpositionalphrase: „**in** meinem Schlafanzug“
- ADJP – Adjektivphrase: „sehr **großer**“
- ADVP – Adverbialphrase: „sehr **oft**“

Grün markiert: Kopf einer Phrase

Phrasenstrukturgrammatik

Phrasen bilden Sätze

syntaktische Regeln

$S \rightarrow NP\ VP$

$NP \rightarrow Det\ N$

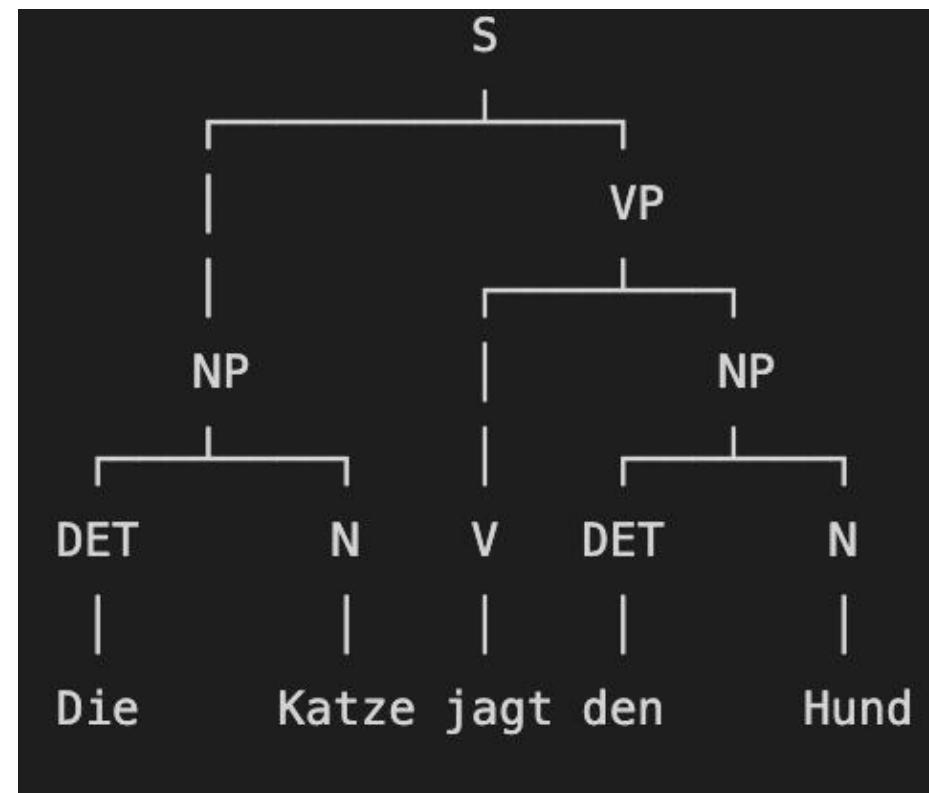
$VP \rightarrow V\ NP$

lexikalische Regeln

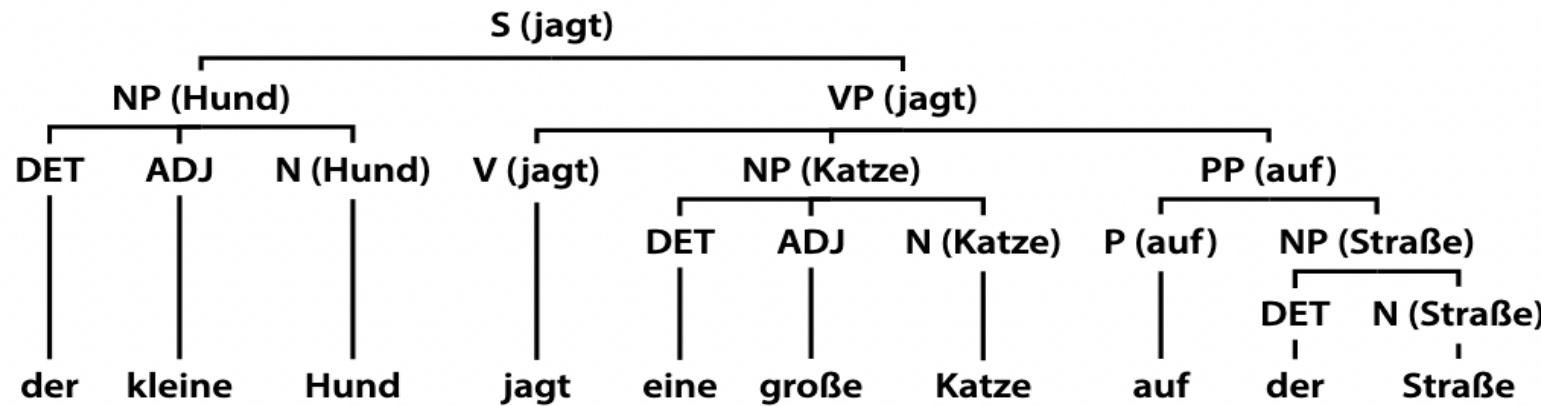
$Det \rightarrow \text{„die“} \mid \text{„den“}$

$N \rightarrow \text{„Katze“} \mid \text{„Hund“}$

$V \rightarrow \text{„jagt“}$



Kopfperkolation



vgl. Vorlesungsfolien

Kopf einer NP: N (hier „Hund“, „Straße“ und „Katze“)

Kopf einer VP: V (hier „jagt“)

Kopf einer PP: P (hier „auf“)

Kopf von S = Kopf der VP = Verb (Kopf wird nach oben weitergereicht)

Regeltypen

syntaktische Regeln

links: syntaktische Kategorie, rechts syntaktische oder lexikalische Kategorie

$S \rightarrow NP\ VP$, $NP \rightarrow Det\ N$

lexikalische Regeln

links: lexikalische Kategorie, rechts Wörter

$N \rightarrow \text{„Katze“}$

*rekursive Regeln

Nur bei syntaktischen Regeln; ein Symbol auf der rechten Seite entspricht dem Symbol auf der linken Seite

NP $\rightarrow Det\ N\ NP$

Regeltypen

Direkte Rekursion

NP -> Det N **NP** (rechtsrekursiv)

VP -> **VP** PP (linksrekursiv)

NP NP NP
„(der Hund) (der Frau) (des Nachbarn)“
Det N Det N Det N

Indirekte Rekursion

NP -> Det N PP

PP -> P **NP**

PP PP
„(die Lampe) (auf dem Tisch) (neben der Tür)“
Det N P Det N P Det N
NP NP NP

3. Ambiguität und X-Bar-Struktur

Ambiguität im Satz, X-Bar-Struktur

Ambiguität

- Strukturelle (syntaktische) Ambiguität, auch als PP-Attachment-Ambiguität genannt:
 - The man see the woman with the telescope
- Koordinationsambiguität:
 - Alte Männer und Frauen
- Temporale Ambiguität:
 - The old man the boat

X-Bar Struktur

Behandelt Übergenerierung

- z.B. NP -> DET NP | N zugelassen: *der der große Hund

Kernprinzipien:

- Kopfprinzip (obligatorischer Kopf)
- Binäre Verzweigung
- Einziger Spezifizierer
- Einführung phrasaler Zwischenebene X'
- Beschränkung auf **eine Struktur für alle Phrasen**

X-Bar Struktur

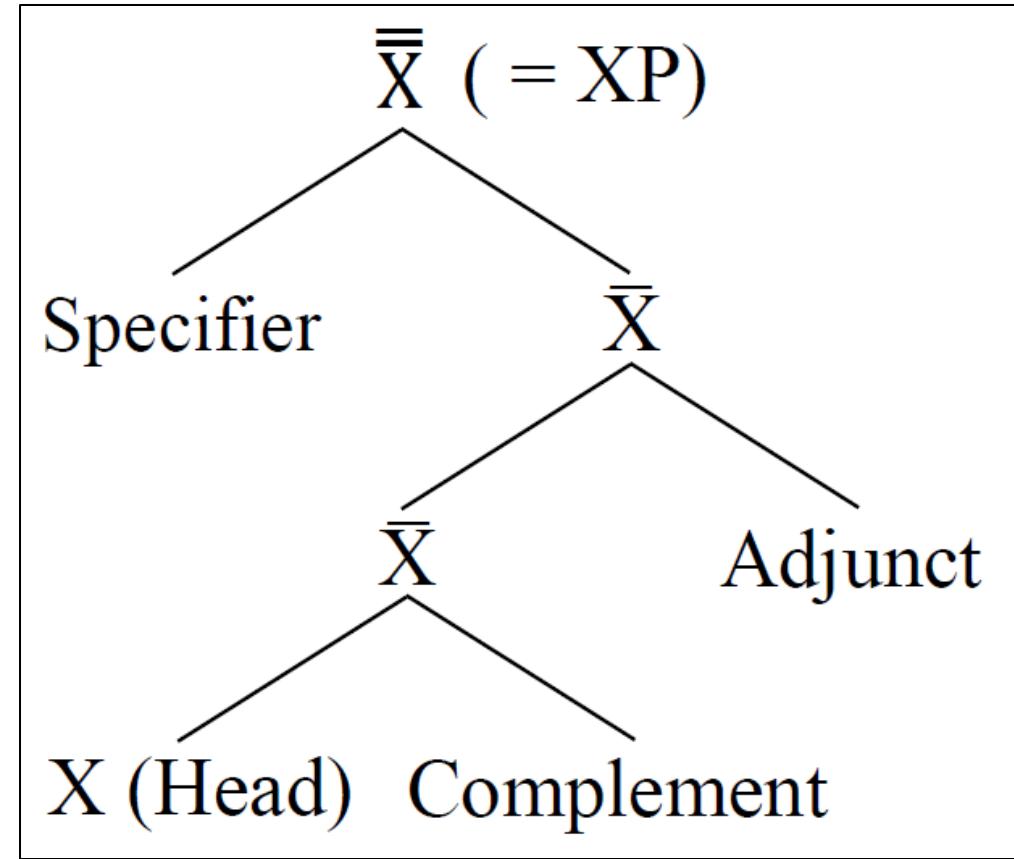
- X-Bar(\bar{X}) wird oft als X' geschrieben
- Durch die Einführung von X' können wir **beliebig viele** Adjunkten einfügen.
- X ist der Kopf von XP
- Ein Spezifizierer ist **nicht obligatorisch**,
- Ein Spezifizierer kann **nur einmal** vorkommen

Regeln:

$XP \rightarrow (\text{Spezifizierer}) X'$

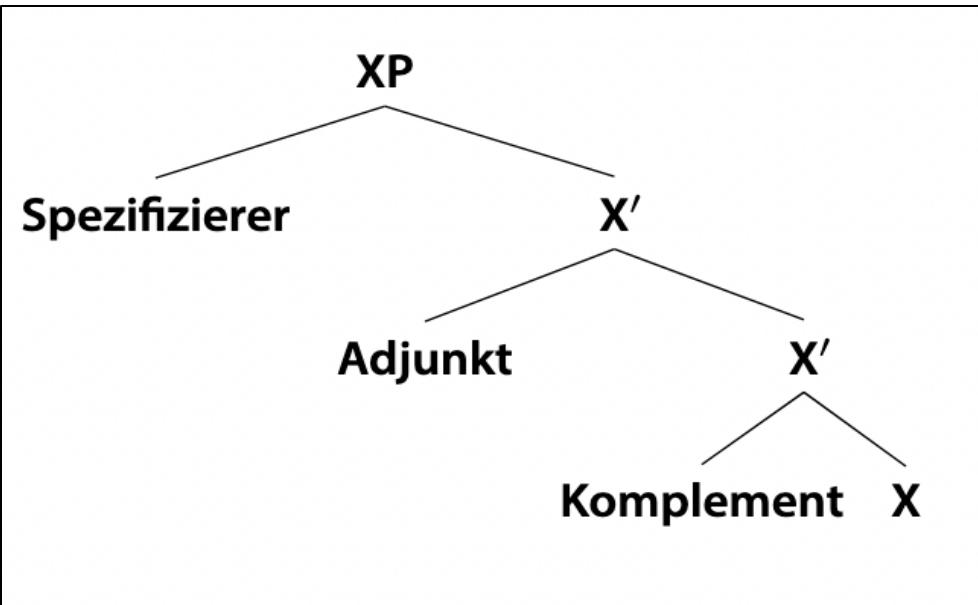
$X' \rightarrow (\text{Adjunkt}) X'$ (beliebig viele, da Rekursion)

$X' \rightarrow (\text{Komplement}) X$



By Dragoniez - Own work, CC BY-SA 4.0,
<https://commons.wikimedia.org/w/index.php?curid=111477173>

Bsp. X-Bar-Schema mit $X = N$



$NP \rightarrow (\text{Spezifizierer}) \text{ NOM}$

$\text{NOM} \rightarrow (\text{Adjunkt}) \text{ NOM}$

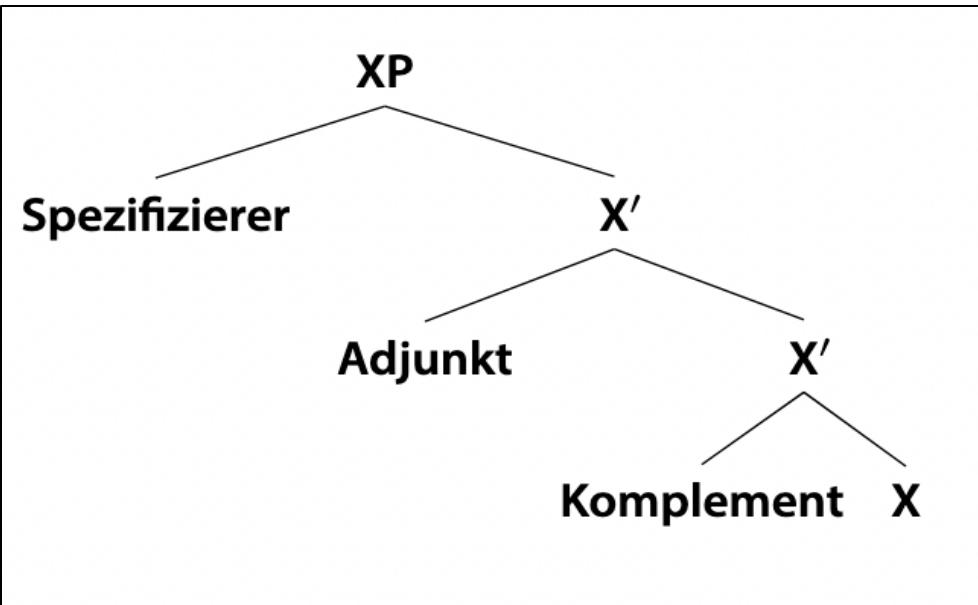
$\text{NOM} \rightarrow (\text{Komplement}) \text{ N}$

$NP \rightarrow \text{Det NOM}$

$\text{NOM} \rightarrow \text{ADJP NOM}$

$\text{NOM} \rightarrow (\text{Komplement}) \text{ N}$

Bsp. X-Bar-Schema mit X = V



VP → (Spezifizierer) VERBAL

VERBAL -> (Adjunkt) VERBAL

VERBAL -> (Komplement) V

VP → (Spezifizierer) VERBAL

VERBAL -> VERBAL PP
Reihenfolge vertauscht!

VERBAL -> V NP
(für transitive Verben mit einem
Objekt als Komplement)

Grundregeln aus der Probeklausur

X-Bar Grundregeln

- 1 | NP → NOM | DET NOM
 - 2 | NOM → ADJP NOM | NOM PP
 - 3 | NOM → N | N NP
-
- 1 | VP → VERBAL | AUX VERBAL
 - 2 | VERBAL → VERBAL PP | VERBAL ADJP
 - 3 | VERBAL → V | V NP | V NP NP

SBAR Grundregeln

- 1 | SBAR → COMP S
- 2 | S → SBAR VP
- 3 | VP → V SBAR
- 4 | S → NP VP SBAR
- 5 | NP → NP SBAR

Koordination Grundregeln

- 1 | S → S CC S
- 2 | NP → NP CC NP

Anmerkung

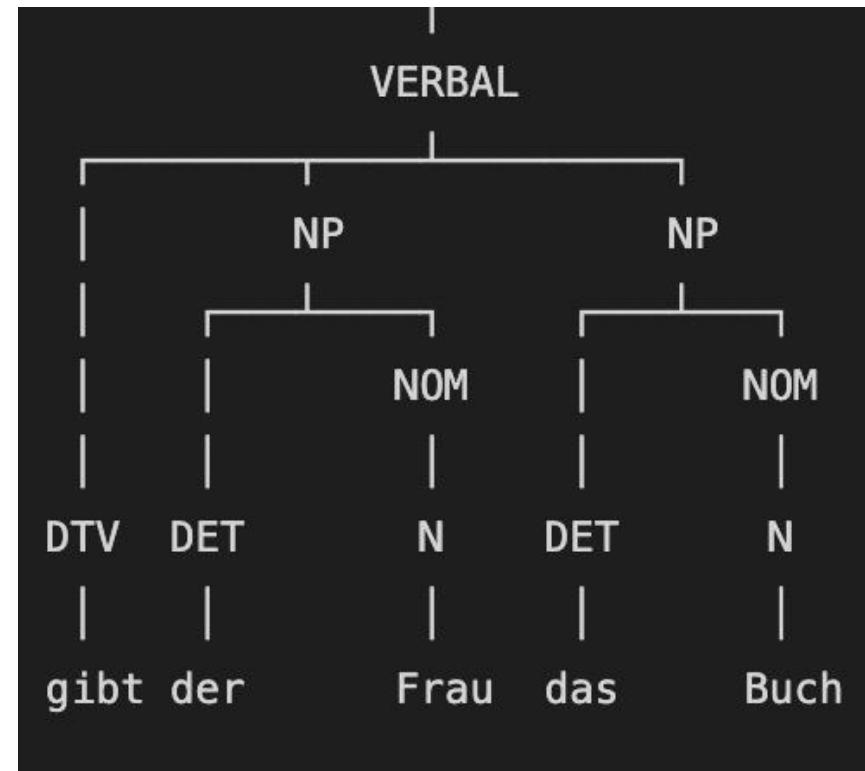
Wie sieht im X-BAR-Schema eine Verzweigung mit 2 Komplementen aus?

Das X-Bar-Schema zeichnet sich u.a. durch eine **Beschränkung auf binäre Verzweigungen** aus. Bei **ditransitiven Verben** haben wir jedoch **zwei Komplemente** und brauchen daher **drei** statt nur zwei **Tochterknoten**:

VERBAL -> V NP NP

In der richtigen X-BAR-Theorie gibt es für solche Fälle weitere Knotentypen!

Wir verwenden jedoch ein **vereinfachtes Modell** und nehmen in Kauf, dass bei zwei Komplementen die Verzweigung **nicht binär** ist.



DTV = ditransitives Verb

4. CFG-Parsing

Top-Down/Bottom-up Parsing, Recursive Descent Parser, Shift-Reduce Parser, Earley Parser

Top-Down vs. Bottom-Up Parsing

- **Top-Down:** Ausgehend vom Startsymbol wird versucht, mit Hilfe der Produktionsregeln den gegebenen Satz abzuleiten.
 - -> **Recursive Descent**
 - -> **Earley**
- **Bottom-Up:** Ausgehend von den Terminalsymbolen wird versucht, diese zu größeren syntaktischen Einheiten zu verbinden, bis man beim Startsymbol angelangt ist.
 - -> **Shift-Reduce**

Recursive Decent Parser

2 Operationen: **Predict** & **Scan** **NLTK: Expand + Match**

- Probiert jede anwendbare Regel aus
- Führt die Regel nicht zum Erfolg, nutzt der Parser **Backtracking** und probiert die nächste Regel aus, etc.

The recursive descent parser builds a parse tree during the above process. With the initial goal (find an S), the S root node is created. As the above process recursively expands its goals using the productions of the grammar, the parse tree is extended downwards (hence the name *recursive descent*)

<https://www.nltk.org/book/ch08.html#recursive-descent-parsing>

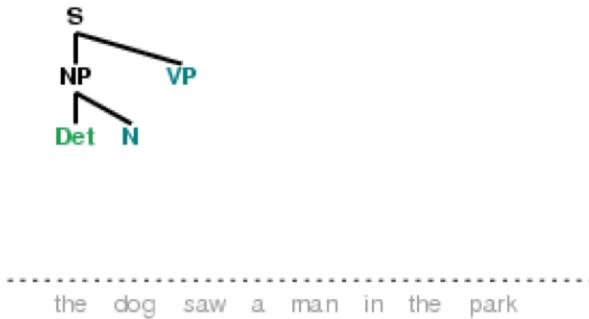
Recursive Descent Parser

<https://www.nltk.org/book/ch08.html#recursive-descent-parsing>

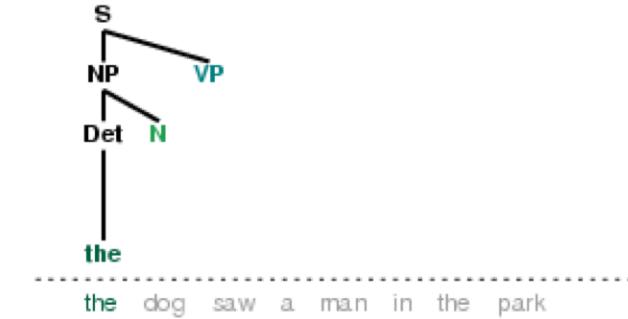
1. Initial stage



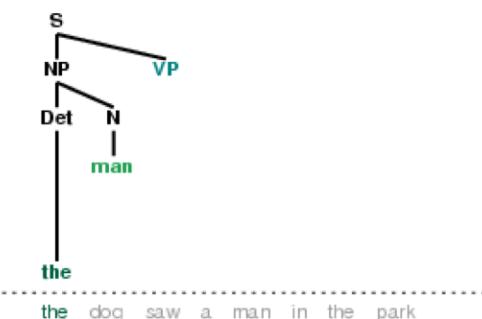
2. Second production



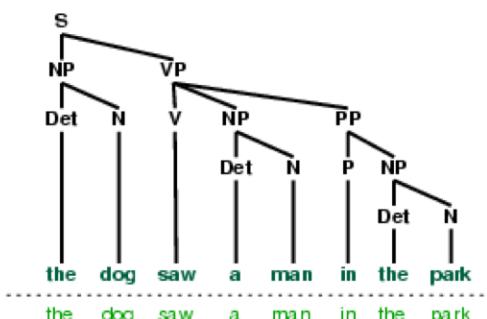
3. Matching *the*



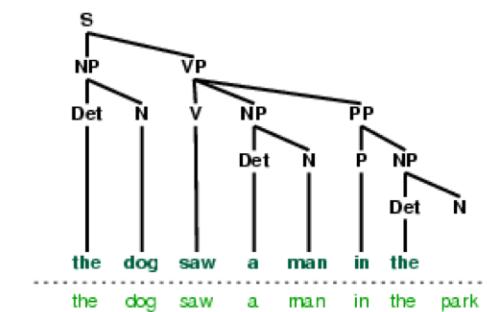
4. Cannot match *man*



5. Completed parse



6. Backtracking



Six Stages of a Recursive Descent Parser: *the parser begins with a tree consisting of the node S; at each stage it consults the grammar to find a production that can be used to enlarge the tree; when a lexical production is encountered, its word is compared against the input; after a complete parse has been found, the parser backtracks to look for more parses.*

Recursive Descent Parser

Drei wesentliche Nachteile:

- Linksrekursive Regeln führen Produktionen wie $NP \rightarrow NP\ PP$ zu einer Endlosschleife.
- Der Parser verschwendet viel Zeit damit, Wörter und Strukturen zu berücksichtigen, die nicht mit dem Eingabesatz übereinstimmen.
- Der Backtracking-Prozess kann analysierte Konstituenten verwerfen, die später erneut aufgebaut werden müssen.

Shift-Reduce Parser

2 Operationen: Shift & Reduce

Verwendet ein Stack (Stapel) und verschiebt eingelesene Wörter auf den Stapel, um sie auf passende Produktionsregeln zurückzuführen.

Shift-Reduce Parser

1. Initial state

Stack	Remaining Text
	the dog saw a man in the park

3. After reduce shift reduce

Stack	Remaining Text
Det N the dog	saw a man in the park

2. After one shift

Stack	Remaining Text
the	dog saw a man in the park

4. After recognizing the second NP

Stack	Remaining Text
NP V NP in Det N saw Det N the dog a man	the park

5. After building a complex NP

Stack	Remaining Text
NP V NP Det N saw Det N NP the dog a man in Det N the park	

6. Built a complete parse tree

Stack	Remaining Text
S NP VP Det N V NP PP the dog saw NP P NP a man in Det N the park	

<https://www.nltk.org/book/ch08.html#shift-reduce-parsing>

Shift-Reduce Parser

2 Operationen: **Shift & Reduce**

Verwendet ein Stack (Stapel) und verschiebt eingelesene Wörter auf den Stapel, um sie auf passende Produktionsregeln zurückzuführen.

Effizienter als ein Top-Down Parser, da er vom Eingabesatz ausgeht.

Probleme:

- kann Teilstrukturen erzeugen, die zu keinem Ergebnis führen
- benötigt daher Backtracking (ist aber nicht immer implementiert)
- Probleme bei PP-Attachment-Ambiguität
- Probleme mit temporaler und lexikalischer Ambiguität, wenn kein Backtracking verwendet wird. Der Eingabesatz wird dann evtl. nicht erkannt, obwohl er ableitbar ist.

Earley Parser

3 Operationen: **Scan, Predict & Complete**

- Vermeidet doppelte Berechnungen durch dynamisches Programmieren
- Zwischenergebnisse werden in einem Chart gespeichert: Chart Parser

Vorteile und Nachteile:

- **komplizierter** als Recursive Descent und Shift-Reduce
- dafür aber viel **schneller**
- benötigt **kein Backtracking** und erzeugt **keine unnötigen Teilstrukturen**

Manuelles Parsen mit Earley

	0	1	2	3	4	5	6	7		
	Er	sieht	das	Huhn	mit	dem	Fernglas			
	$S \rightarrow \bullet NP VP$ $NP \rightarrow \bullet Det N$ $NP \rightarrow \bullet NP PP$ $NP \rightarrow \bullet Pron$ $S \rightarrow NP \bullet VP$ $Det \rightarrow \bullet das$ $Det \rightarrow \bullet dem$ $Pron \rightarrow \bullet Er$	$Pron \rightarrow Er \bullet$ $NP \rightarrow Pron \bullet$ $NP \rightarrow NP \bullet PP$ $S \rightarrow NP \bullet VP$		$S \rightarrow NP VP \bullet$			$S \rightarrow NP VP \bullet$			
SCAN		$PP \rightarrow \bullet P NP$ $P \rightarrow \bullet mit$ $VP \rightarrow \bullet VP PP$ $VP \rightarrow \bullet V NP$ $V \rightarrow \bullet sieht$	$V \rightarrow sieht \bullet$ $VP \rightarrow V \bullet NP$		$VP \rightarrow V NP \bullet$ $VP \rightarrow VP \bullet PP$		$VP \rightarrow VP PP \bullet$ $VP \rightarrow V NP \bullet$ $VP \rightarrow VP \bullet PP$			
PREDICT			$NP \rightarrow \bullet Det N$ $NP \rightarrow \bullet NP PP$ $NP \rightarrow \bullet Pron$ $Det \rightarrow \bullet das$ $Det \rightarrow \bullet dem$ $Pron \rightarrow \bullet Er$	$Det \rightarrow das \bullet$ $NP \rightarrow Det \bullet N$	$NP \rightarrow Det N \bullet$ $NP \rightarrow NP \bullet PP$			$NP \rightarrow NP PP \bullet$ $NP \rightarrow NP \bullet PP$		
COMPLETE				$N \rightarrow \bullet Huhn$ $N \rightarrow \bullet Fernglas$	$N \rightarrow Huhn \bullet$					
					$PP \rightarrow \bullet P NP$ $P \rightarrow \bullet mit$	$P \rightarrow mit \bullet$ $PP \rightarrow P \bullet NP$		$PP \rightarrow P NP \bullet$		
						$NP \rightarrow \bullet Det N$ $NP \rightarrow \bullet NP PP$ $NP \rightarrow \bullet Pron$ $Det \rightarrow \bullet das$ $Det \rightarrow \bullet dem$ $Pron \rightarrow \bullet Er$	$Det \rightarrow dem \bullet$ $NP \rightarrow Det \bullet N$	$NP \rightarrow Det N \bullet$ $NP \rightarrow NP \bullet PP$		
							$N \rightarrow \bullet Huhn$ $N \rightarrow \bullet Fernglas$	$N \rightarrow Fernglas \bullet$		
							$PP \rightarrow \bullet P NP$ $P \rightarrow \bullet mit$			

5. Dependenzgrammatik & Dependency Parsing

Konstituenz vs. Dependenz, Rektion und Modifikation, Arten von
Dependenten, Primacy of Content, Shift-Reduce Dependency
Parsing

2 Ansätze zur Syntaktischen Analyse

- **Konstituentenstruktur:**

- Wie kann man ein Satz in syntaktischen Einheiten (Konstituente) zerlegen?
- Untersucht **Strukturregeln** zur Erklärung des Aufbaus eines Satzes

- **Dependenz:**

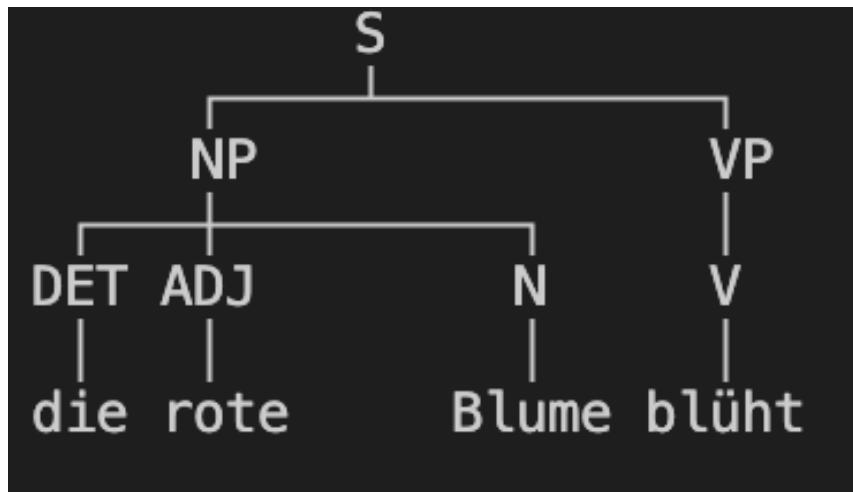
- In welcher syntaktischen Beziehung stehen **Wörter** in einem Satz?
- Untersucht Abhängigkeitsverhältnisse zwischen Wörtern eines Satzes

Dependenzrelation

- Dependent:
 - y hängt von x ab → y ist **DEPENDENT** von x
 - x regiert y → x ist **KOPF** von y
- Beispiel:
 - Satz: der Mann rennt
 - „der“ ist Dependent von „Mann“
 - „Mann“ ist Kopf von „der“

Konstituenz vs. Dependenz

Konstituenz



S → NP VP

NP → Det ADJ N

"Blume"

VP → V

"rote"

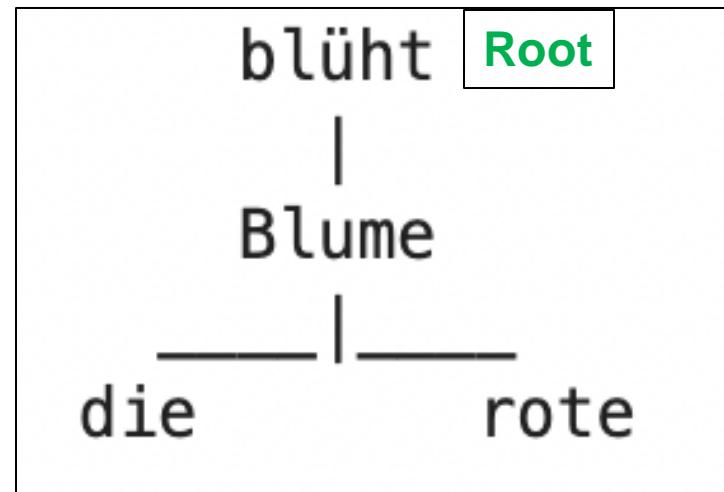
Det → " die"

N →

ADJ →

V → "blüht"

Dependenz



Darstellung
mit NLTK

Kopf → Dependent

"blüht" → „Blume“

„Blume“ → „die“ | „rote“

Rektion und Modifikation

- Rektion:
 - Kopf **kann nicht** ohne Dependent auftreten (und ein Dependent kann nie ohne Kopf auftreten)
 - Bilaterale Dependenz: Kopf und Dependent sind voneinander abhängig
 - Dependent ist **Komplement** des Kopfes
 - Modifikation
 - Kopf **kann** ohne Dependent auftreten
 - Unilaterale Dependenz
 - Dependent ist **Adjunkt** des Kopfes
- „Sie betrachtet das Bild.“

* „Sie betrachtet.“ **Nicht wohlgeformt**

* „Sie das Bild.“ **Nicht wohlgeformt**
- „Sie betrachtet das Bild am Nachmittag.“

„Sie betrachtet das Bild.“ **Wohlgeformt**

Arten von Dependenten

- Obligatorischer Dependent
 - Komplement
- Fakultativer Dependent
 - Komplement, aber nach Kontext weglassbar
 - Klassisches Beispiel: „Er sieht das Huhn“ → „Er sieht“
- Optionaler Dependent
 - Adjunkt (zusätzliche Information)

Primacy of Content Words

- In Konstituentenstruktur
 - X ist Kopf einer XP
 - Beispiel: P ist Kopf einer PP

Aber...

- Bei Dependenz
 - Nur **Inhaltswörter** sind Köpfe
 - Beispiel: „an der Uni“ ist Kopf von im (UD-Schema)

Shift-Reduce Dependency Parsing

- Ein Satz ist eine Liste (Buffer) von Wörtern
- Der Parser parst Wort für Wort
- Ein Stack für Wörter und Strukturen
- Operationen
 - **Shift**: Schiebe ein Wort aus der Liste auf das Stack
 - **Reduce**: Relation zwischen den beiden obersten Elementen auf das Stack hinzufügen und den Dependenten vom Stack löschen
 - **LEFTARC**: Kopf liegt rechts vom Dependent
 - **RIGHTARC**: Kopf liegt links vom Dependent

ein Beispiel
macht Sinn



Richtung
Pfeil

Shift-Reduce Dependency Parsing

Beginn:

- Alle Wörter sind in der Liste (Buffer)
- Stack ist mit ROOT initialisiert
- Es sind noch keine Pfeile (Dependenzrelationen) vorhanden
- Erste Operation: SHIFT

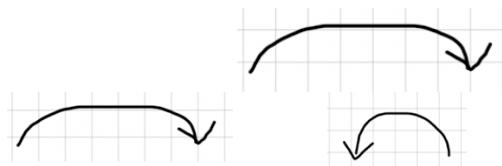
REDUCE-LEFTARC: ist immer möglich

REDUCE-RIGHTARC: nur dann möglich, **wenn der Dependent der Relation selbst nicht Kopf einer der noch offenen Relation ist.**

Diese Einschränkung verhindert, dass ein Wort zu früh vom Stack genommen wird.

Shift-Reduce Dependency Parsing

Buffer	Stack	Operation
[Kauf Tickets, nach, München]	[ROOT]	SHIFT
[Tickets, nach, München]	[ROOT, Kauf]	SHIFT
[nach, München]	[ROOT, Kauf, Tickets]	SHIFT
[]	[ROOT, Kauf, Tickets, nach, München]	LEFTARC
[]	[ROOT, Kauf, Tickets, München]	RIGHTARC
[]	[ROOT, Kauf, Tickets]	RIGHTARC
[]	[ROOT, Kauf]	RIGHTARC
[]	[ROOT]	DONE



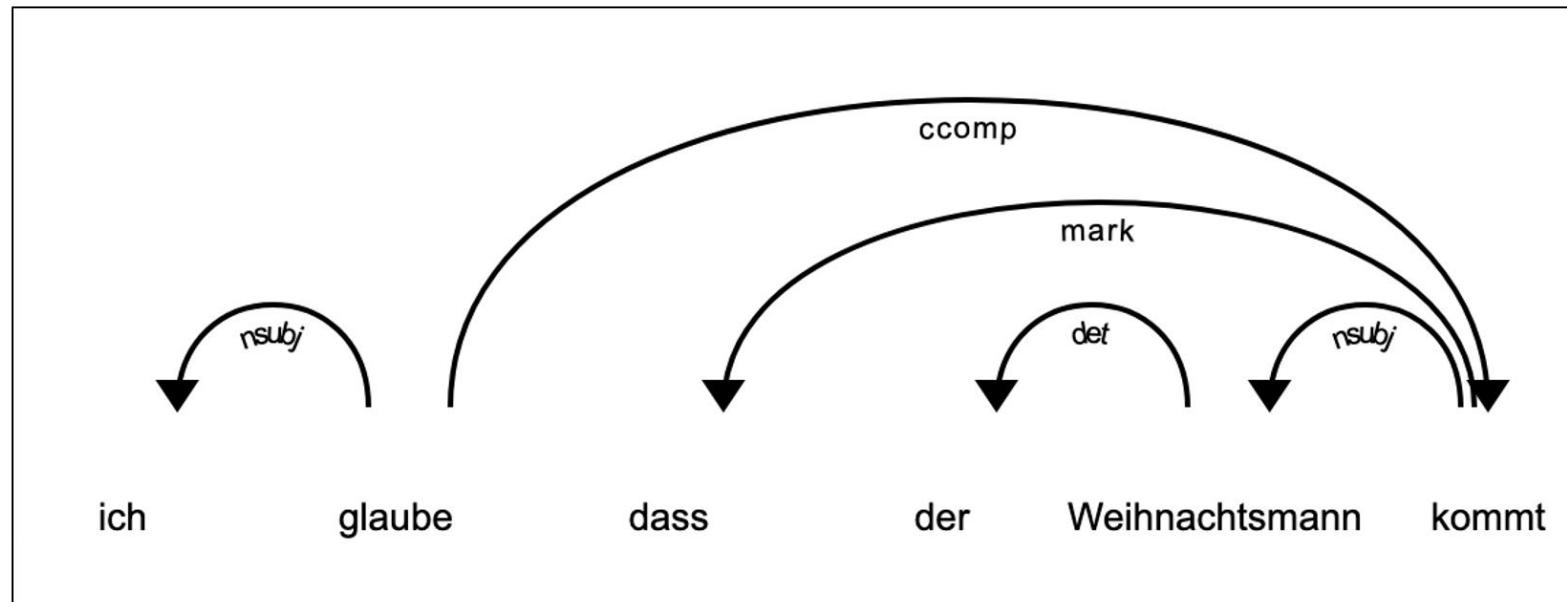
Kauf Tickets nach München.
ROOT

Shift-Reduce Dependency Parsing

Projektive Struktur:

Für jede Dependenzrelation im Satz gilt, dass es einen Pfad vom Kopf zu allen Wörtern zwischen Kopf und Dependent gibt.

Beispiel:



Z.B. können wir jedes Wort zwischen „glaube“ (Kopf) und „kommt“ (Dependent) von „glaube“ aus erreichen.

Shift-Reduce Dependency Parsing

Nicht-projektive Struktur:

Es gibt Dependenzrelation im Satz, für die die Bedingung für projektive Strukturen nicht gilt, d.h. wir können **nicht** bei allen Dependenzrelationen im Satz jedes Wort zwischen Kopf und Dependent erreichen.

Beispiel:



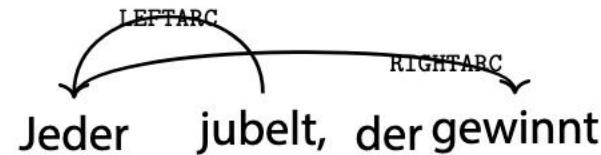
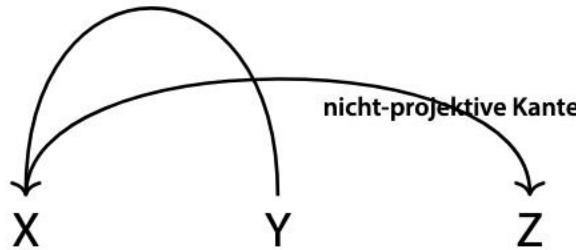
In diesem Beispiel ist „talk“ Kopf von „about“, und „yesterday“ steht zwischen Kopf und Dependent.

Es gibt keinen Pfad vom Kopf („talk“) zu „yesterday“, daher handelt es sich um eine nicht-projektive Struktur.

vgl. Vorlesungsfolien

Shift-Reduce Dependency Parsing

Beim Dependency-Parsing können nicht-projektive Strukturen problematisch sein.
Der übergangsbasierte **Shift-Reduce-Dependency-Parser kann beispielsweise mit nicht-projektiven Strukturen nicht umgehen.**



vgl. Vorlesungsfolien

Was ist das Problem bei Parsen einer solchen Struktur?

Wir würden hier X und Y auf das Stack shiften, dann REDUCE-LEFTARC anwenden und den Dependenten (X) vom Stack nehmen. Wenn wir X vom Stack nehmen, können wir aber den Rest des Satzes nicht mehr parsen!

Shift-Reduce Dependency Parsing

Jeder jubelt, der gewinnt

Stack: jeder, jubelt

Jeder jubelt, der gewinnt

Stack: jubelt

Jeder jubelt, der gewinnt

Stack: jubelt, der

Jeder jubelt, der gewinnt

Stack: jubelt, der

Jeder jubelt, der gewinnt

Macht nicht weiter!

„jeder“ ist schon weg. Kann nicht mehr geparsst werden!

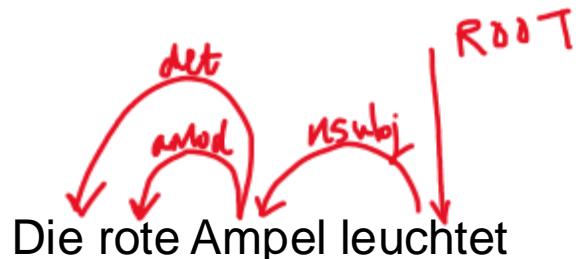
6. Dependenzgrammatik 2

Dependenztags

Dependenzanalyse

- Satz: Die rote Ampel leuchtet
 - Schritt 3: UD-Dependency Labels notieren

1 Die	3	det
2 rote	3	amod
3 Ampel	4	nsubj
4 leuchtet	0	ROOT



Anmerkung: Pfeile gehen immer vom Kopf zum Dependenten. Laut UD-Dependency sind die Tags KLEINGESCHRIEBEN

Angabe: Hilfsmittel

Universal Dependency Relations

	Nominals	Clauses	Modifier words	Function Words
Core arguments	<u>nsubj</u> <u>obj</u> <u>iobj</u>	<u>csubj</u> <u>ccomp</u> <u>xcomp</u>		
Non-core dependents	<u>obl</u> <u>vocative</u> <u>expl</u> <u>dislocated</u>	<u>advcl</u>	<u>advmod</u> * <u>discourse</u>	<u>aux</u> <u>cop</u> <u>mark</u>
Nominal dependents	<u>nmod</u> <u>appos</u> <u>nummod</u>	<u>acl</u>	<u>amod</u>	<u>det</u> <u>clf</u> <u>case</u>
Coordination	MWE	Loose	Special	Other
<u>conj</u> <u>cc</u>	<u>fixed</u> <u>flat</u> <u>compound</u>	<u>list</u> <u>parataxis</u>	<u>orphan</u> <u>goeswith</u> <u>reparandum</u>	<u>punct</u> <u>root</u> <u>dep</u>

Häufige UD-Tags im Dt. (Nicht Vollständig)

Relation (Tag)	Bezeichnung (Deutsch)	Beschreibung
nsubj	Nominales Subjekt	Das Subjekt des Satzes, das die Handlung ausführt (Er läuft).
obj	Direktes Objekt	Ziel der Handlung (z. B. Ich sehe den Hund).
iobj	Indirektes Objekt	Empfänger der Handlung (Ich gebe dem Mann das Buch).
obl	Adverbiale Bestimmung	Präpositionale Zeit, Ort, Art oder Mittel (z. B. Jeden Tag läuft er im Park).
advmod	Adverbiale Modifikation	Einfache Adverbien (oft, bald, schnell), die das Verb (oder Adjektiv) modifizieren.
amod	Adjektivmodifikation	Adjektiv, das ein Nomen beschreibt (die rote Blume).
det	Determinierer	Artikel oder andere Nomenmodifikatoren (die, ein, einige).
case	Präposition/Postposition	Funktionale Wörter, die den Kasus einer Phrase bestimmen (in, auf, mit).
mark	Subjunktor	Subjunktion, die Nebensätze einleitet (dass, weil, obwohl, da, zu).
ccomp	Komplement-Klausel	Nebensatz, der Argument des Verbs ist (Er sagt, dass er kommt).
xcomp	Infinitivsatz ohne Subjekt	Nebensatz ohne eigenes Subjekt (Er versucht zu schlafen).
acl	Relativsatz oder Infinitiv	Relativsatz oder Phrase, die ein Substantiv modifiziert (Das Buch, das ich lese).
nmod	Nominale Modifikation	Präpositionales oder genitivisches Satzglied (Der Freund des Mannes).
appos	Apposition	Zusatz, der ein Nomen näher beschreibt (Herr Müller, der Lehrer).
flat	Flache Struktur	Namen und feste Ausdrücke (Angela Merkel, München Hauptbahnhof).
conj	Koordination	Koordinierte Phrase (Haus und Garten).
cc	Koordinierende Konjunktion	Konjunktion, die Teile in einer Koordination verknüpft (und, oder, aber).
cop	Kopula	Kopulaverben (sein, werden, bleiben → Er ist müde).
expl	Expletives Element	Platzhalter (es regnet).
root	Wurzel	Der Hauptknoten des Satzes (z. B. das konjugierte Verb).

Reflexivverben und Reflexivpronomen

- Fall 1: Als direktes Objekt(obj)
 - Ich erinnere **mich** an_{obl}...
 - Ich kümmere **mich** um_{obl} ...
- Fall 2: Als indirektes Objekt(iobj)
 - Ich kaufe **mir** ein Buch
 - Ich mach **mir** keine Sorge

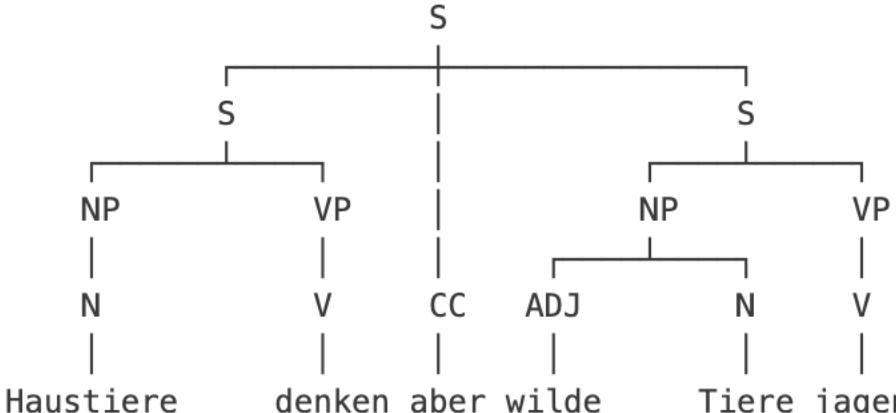
7. Konstituenz und Dependenz Komplexer Sätzen

Typen Komplexer Sätzen

- Koordination
- Subordination
 - Subjektsatz
 - Objektsatz
 - Relativsatz
 - Adverbialsatz
- Infinitiv
 - Als Objekt
 - Als Subjekt
- Sätze mit Hilfsverb

Koordination

- und/oder/aber/sowie
- https://universaldependencies.org/treebanks/de_gsd/de_gsd-pos-CCONJ.html
- Regeln für S:
 - $S \rightarrow S \text{ CC } S$ (CC=CCONJ)
 - $S \rightarrow NP VP$
- Regel für NP:
 - $NP \rightarrow NP \text{ CCONJ } NP$



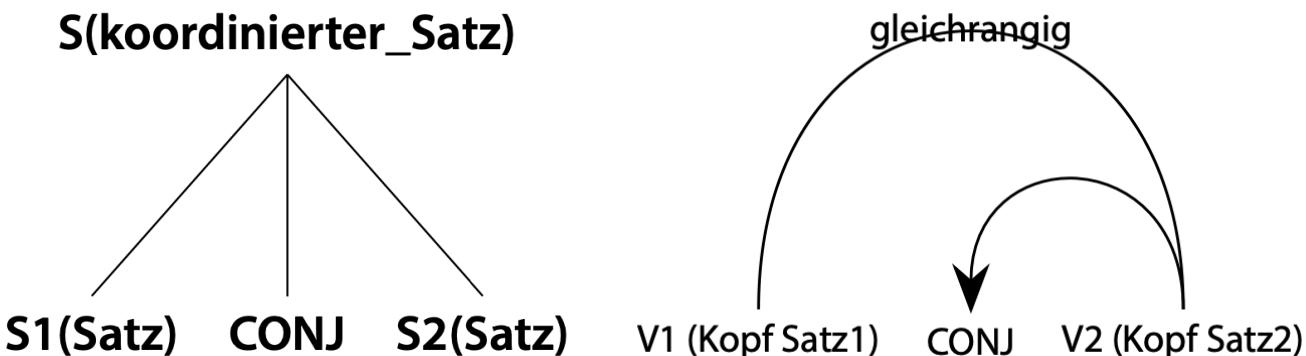
Koordination ist möglich zwischen:

Sätzen: „Er geht, aber er kommt später wieder.“

Nomen: „der Weihnachtsmann und seine Rentiere“

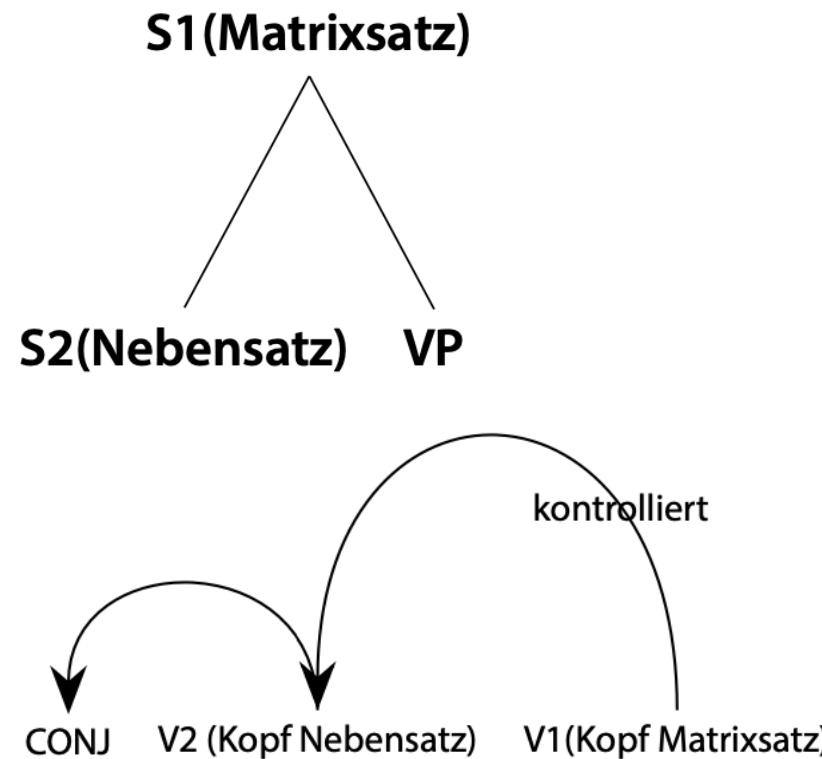
Adjektiven: „der große und schöne Weihnachtsbaum“

Verben: „Der Weihnachtsmann singt und lacht.“



Subordination

- Subjektsatz
- Objektsatz
- Relativsatz
- Adverbialsatz



Nebensätze in Satzgliedfunktion

Subjektsatz (Komplementsatz)

- **Beispiel:** *Wer anderen eine Grube gräbt, fällt selbst hinein.*
- **Funktion als Subjekt-Komplement des Matrixsatzes**

Objektsatz (Komplementsatz)

- **Beispiel:** *Er sagte, dass er keine Zeit habe.*
- **Funktion als Objekt-Komplement des Matrixsatzes**

Indirekter Objektsatz (Komplementsatz)

- **Beispiel:** *Sie musste zusehen, wie er sich betrank.*
- **Funktion als Indirektes Objekt-Komplement des Matrixsatzes**

Adverbialsatz

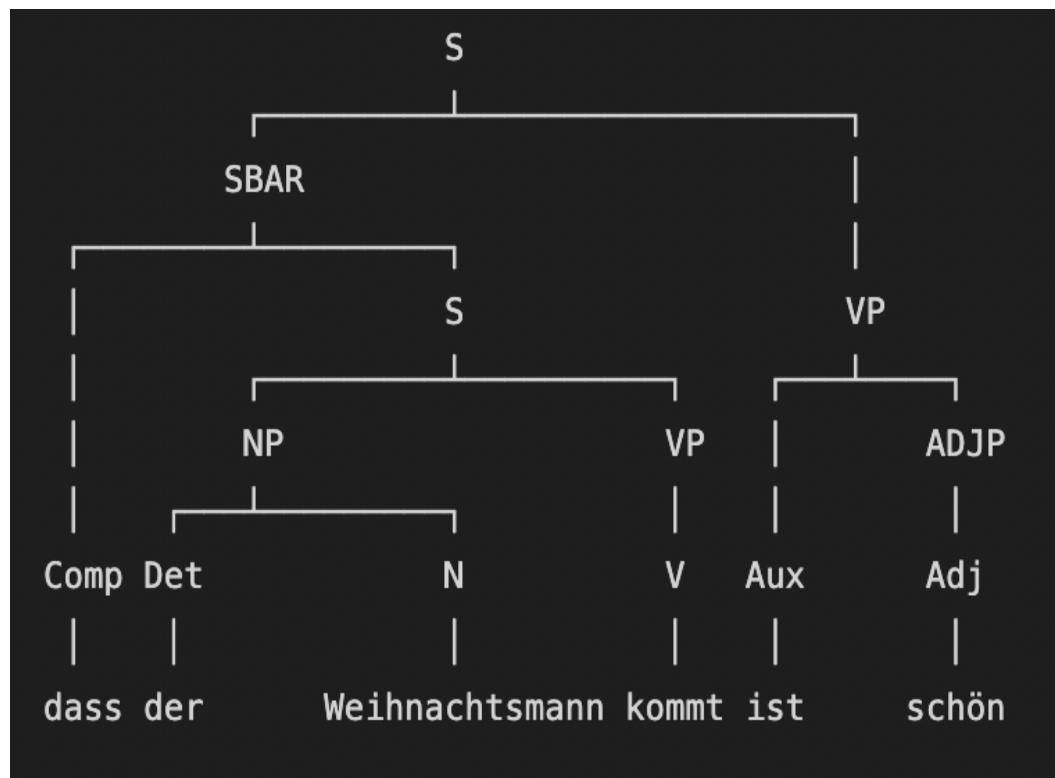
- **Beispiel:** *Er weinte, weil sie ihn nicht beachtete.*
- **Funktion als Adverbial des Matrixsatzes;** Klassifizierung nach semantischen Kriterien: **Kausal-, Temporal-Satz usw.**

1. Komplementsatz
2. Adverbialsatz
3. Attributivsatz (oft Relativsatz)

• Aus den Vorlesungsfolien

Subordination: Subjektsatz

- Nebensatz funktioniert als das Subjekt des Matrixsatzes.



Für finite eingebettete Sätze verwenden wir das Label **SBAR**

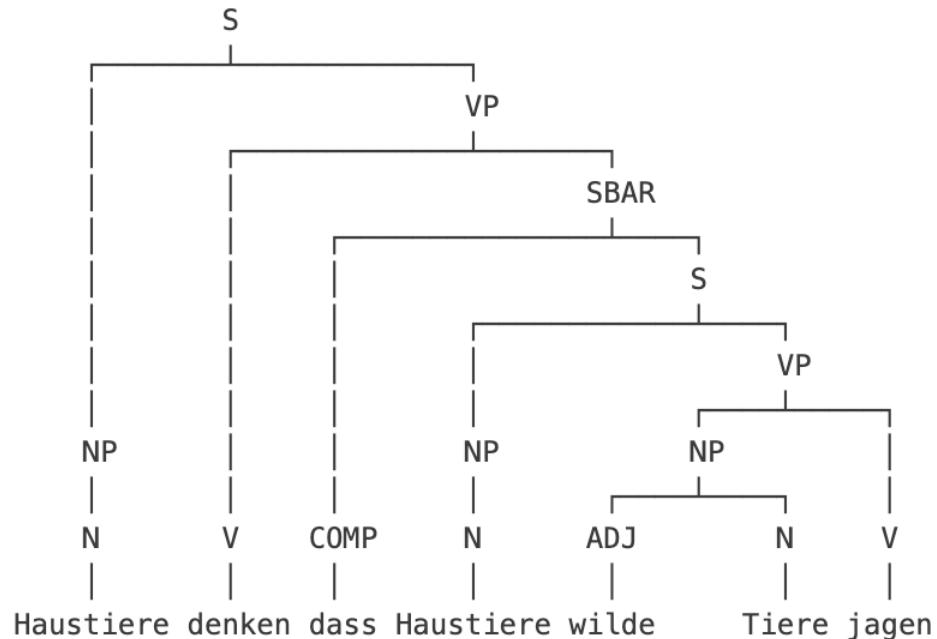
Wichtige Regeln:

$S \rightarrow SBAR\ VP$

$SBAR \rightarrow COMP\ S$

Subordination: Objektsatz

- Nebensatz funktioniert als das Objekt des Matrixsatzes.



aus Übung 7, editiert

Für finite eingebettete Sätze verwenden wir das Label **SBAR**

Wichtige Regeln:

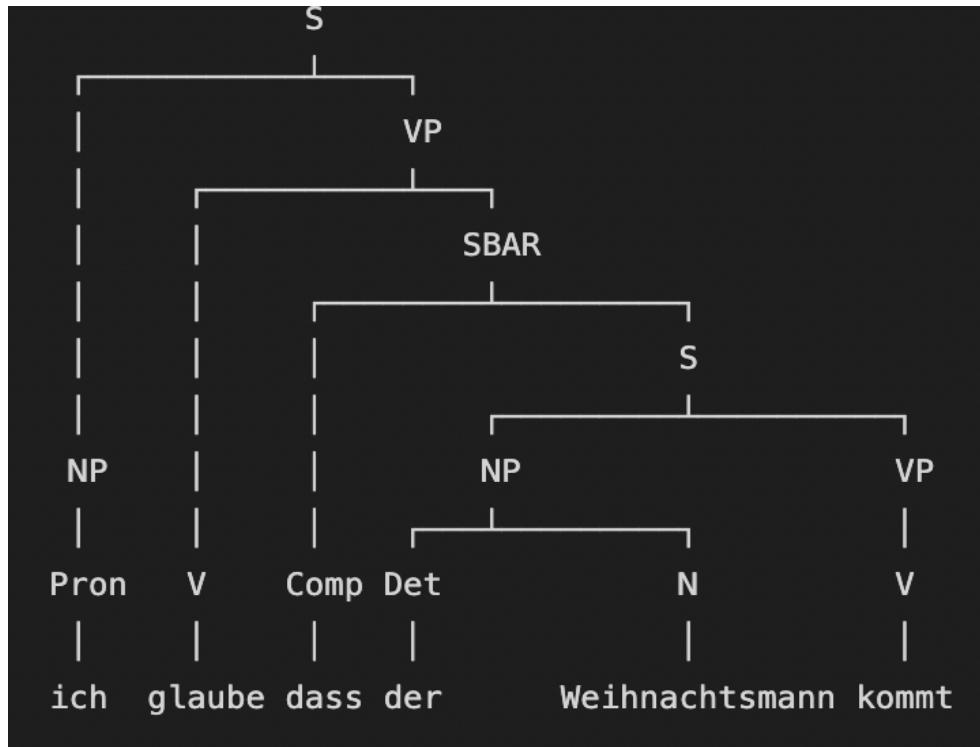
$S \rightarrow NP\ VP$

$VP \rightarrow V\ SBAR \mid NP\ V$ (invertiert für Nebensatz)

$SBAR \rightarrow COMP\ S$

Subordination: Objektsatz

- Nebensatz funktioniert als das Objekt des Matrixsatzes.



Für finite eingebettete Sätze verwenden wir das Label **SBAR**

Wichtige Regeln:

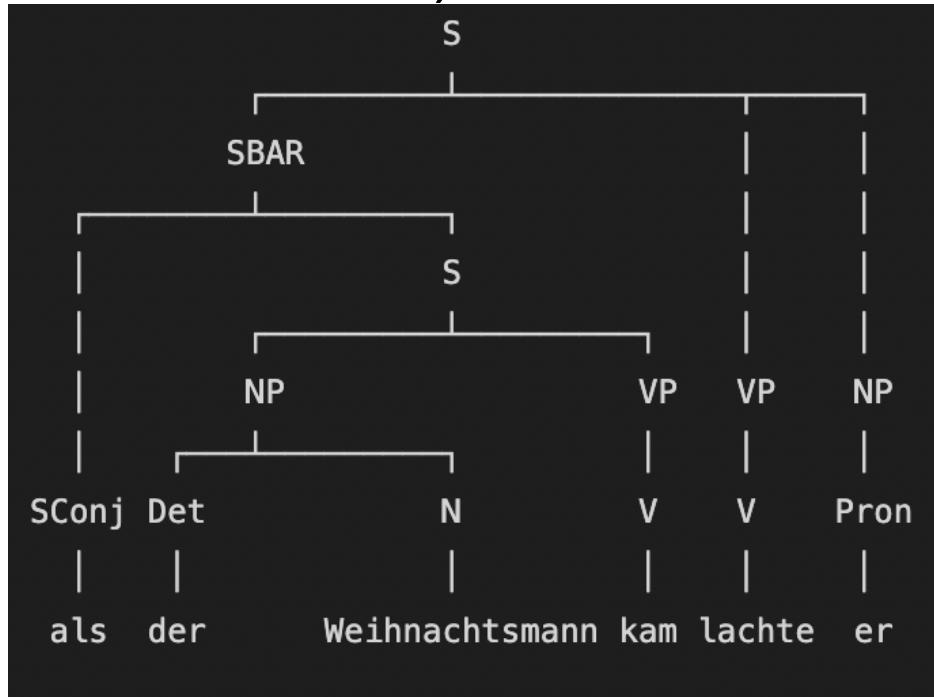
$S \rightarrow NP\ VP$

$VP \rightarrow V\ SBAR$

$SBAR \rightarrow COMP\ S$

Subordination: Adverbialsatz

- Nebensatz funktioniert als Adverbial (liefert zusätzliche Information) des Matrixsatzes.



Für finite eingebettete Sätze verwenden wir das Label **SBAR**

Wichtige Regeln:

$S \rightarrow \text{SBAR } \text{VP } \text{NP}$

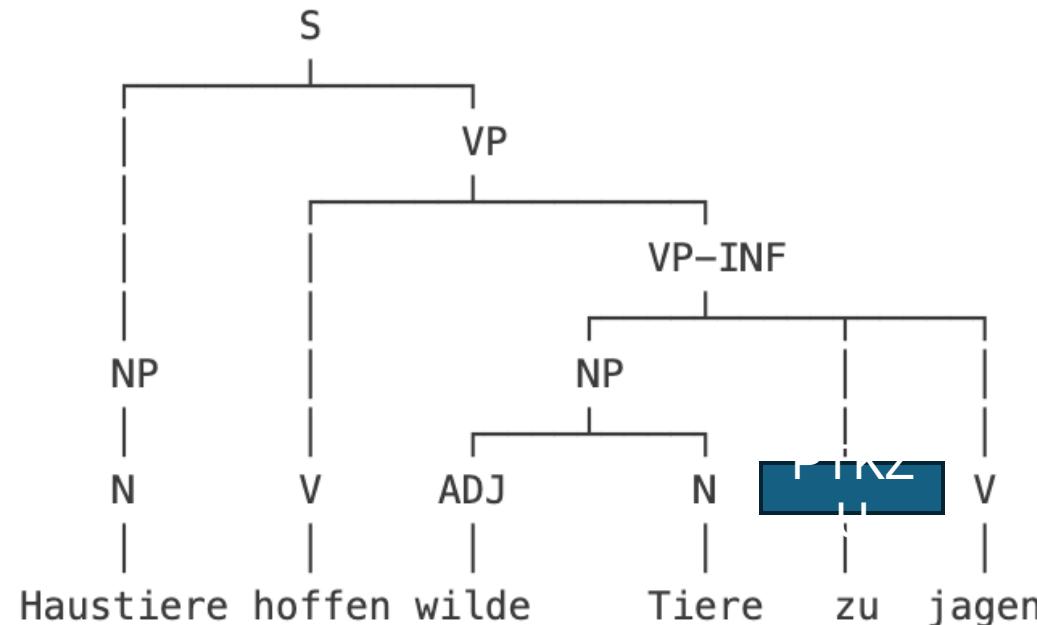
(alternativ z.B. $S \rightarrow \text{NP } \text{VP } \text{SBAR}$)

$\text{SBAR} \rightarrow \text{SConj } S$

Im Aufgabblatt 7 aber: „als“ wird als COMP betrachtet (wegen Vereinfachung)
deshalb **SBAR -> COMP S**

Infinitiv: Als Objekt des Matrixsatzes

- Eingebetteter Satz ist Teil der VP
- **INFINITIV**, kein SBAR



Wichtige Regeln:

$S \rightarrow NP\ VP$

$VP \rightarrow V\ (NP)\ VP-INF$

$VP-INF \rightarrow (NP)\ PTKZU\ V$

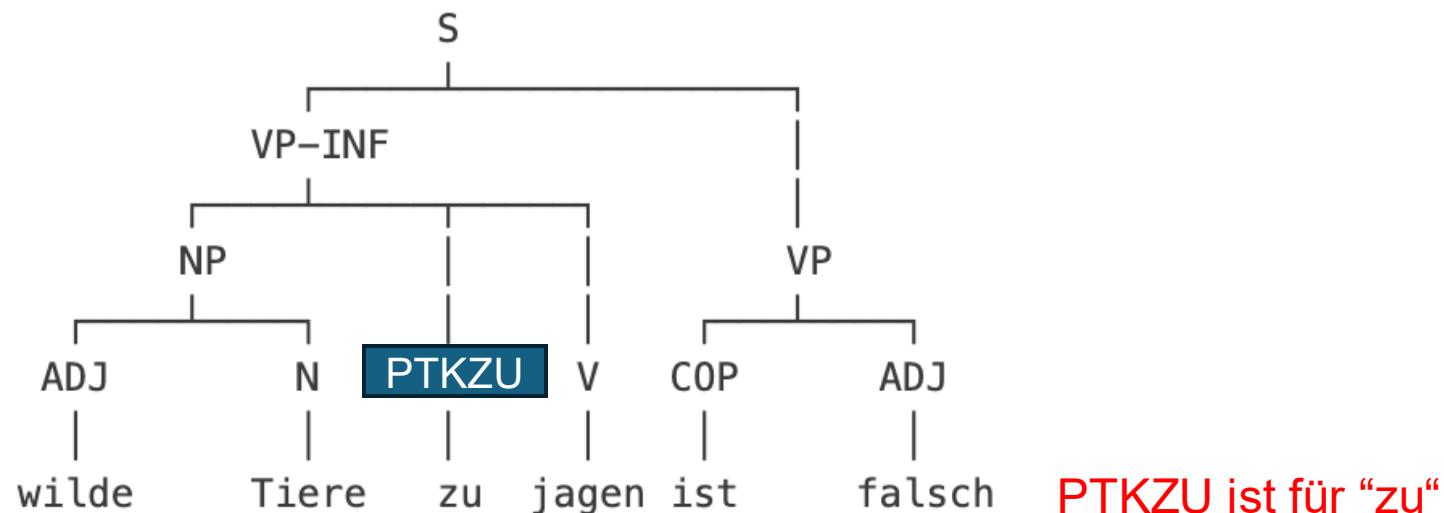
Infinitiv: Als Subjekt des Matrixsatzes

- Eingebetteter Satz ist Teil der VP
- **INFINITIV**, kein SBAR

Wichtige Regeln:

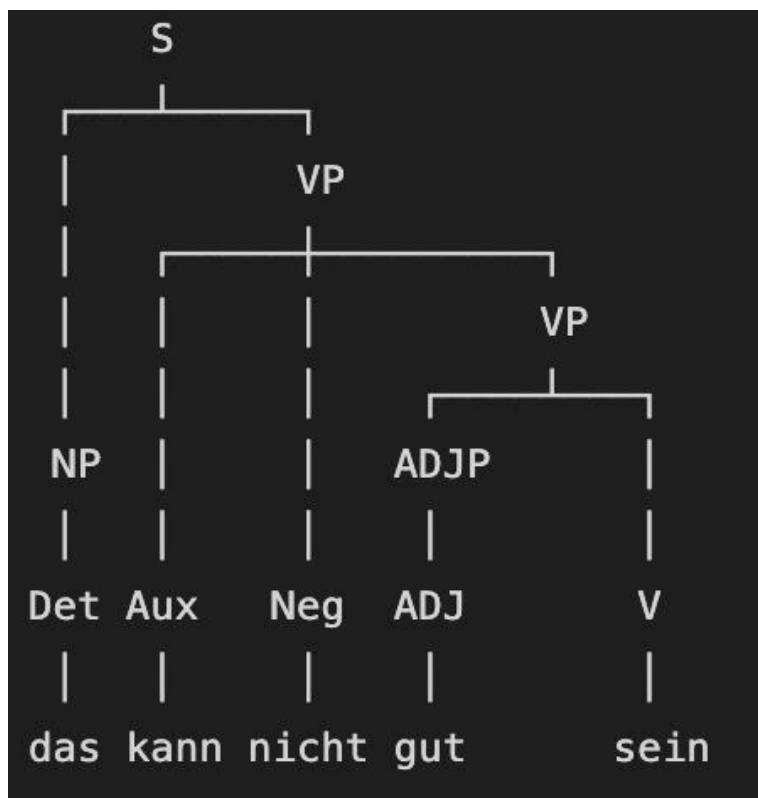
$S \rightarrow VP-INF\ VP$

$VP-INF \rightarrow (NP)\ PTKZU\ V$



Satz mit Auxiliar (Hilfsverb)

- Das Hilfsverb konjugiert.
- VP als Teil der VP



Wichtige Regeln (z.B.):

VP → AUX VP

VP → ADJ V

Häufige UD-Tags im Dt. (Nicht Vollständig)

Relation (Tag)	Bezeichnung (Deutsch)	Beschreibung
nsubj	Nominales Subjekt	Das Subjekt des Satzes, das die Handlung ausführt (Er läuft).
obj	Direktes Objekt	Ziel der Handlung (z. B. Ich sehe den Hund).
iobj	Indirektes Objekt	Empfänger der Handlung (Ich gebe dem Mann das Buch).
obl	Adverbiale Bestimmung, auch Präpositionalobjekt	Präpositionale Zeit, Ort, Art oder Mittel (z. B. Jeden Tag läuft er im Park). Präpositionalobjekt: Ich kümmere mich um dich
advmod	Adverbiale Modifikation	Einfache Adverbien (oft, bald, schnell), die das Verb (oder Adjektiv) modifizieren.
amod	Adjektivmodifikation	Adjektiv, das ein Nomen beschreibt (die rote Blume).
det	Determinierer	Artikel oder andere Nomenmodifikatoren (die, ein, einige).
case	Präposition/Postposition	Funktionale Wörter, die den Kasus einer Phrase bestimmen (in, auf, mit).
mark	Subjunktor	Subjunktion, die Nebensätze einleitet (dass, weil, obwohl, da, zu).
ccomp	Komplement-Klausel	Nebensatz, der Argument des Verbs ist (Er sagt, dass er kommt).
xcomp	Infinitivsatz ohne Subjekt	Nebensatz ohne eigenes Subjekt (Er versucht zu schlafen).
acl	Relativsatz oder Infinitiv	Relativsatz oder Phrase, die ein Substantiv modifiziert (Das Buch, das ich lese).
nmod	Nominale Modifikation	Präpositionales oder genitivisches Satzglied (Der Freund des Mannes).
appos	Apposition	Zusatz, der ein Nomen näher beschreibt (Herr Müller, der Lehrer).
flat	Flache Struktur	Namen und feste Ausdrücke (Angela Merkel, München Hauptbahnhof).
conj	Koordination	Koordinierte Phrase (They left but they came back).
cc	Koordinierende Konjunktion	Konjunktion, die Teile in einer Koordination verknüpft (und, oder, aber).
cop	Kopula	Kopulaverben (sein, werden, bleiben → Er ist müde).
expl	Expletives Element	Platzhalter (es regnet).
root	Wurzel	Der Hauptknoten des Satzes (z. B. das konjugierte Verb).

Häufige UD-Tags im Dt. (Nicht Vollständig)

Relation (Tag)	Bezeichnung	Beschreibung	Beispiel
csubj	Satzsubjekt (Nebensatz als Subjekt)	Ein Nebensatz, der als Subjekt in einem größeren Satz fungiert.	Dass er kommt, ist sicher.
acl	adjectival clause(EN)	Ein Relativsatz oder attributiver Nebensatz, der ein Nomen näher beschreibt. „acl marks finite and non-finite clauses that modify a noun.“	Beispiel aus UD-Seite: Haben Sie den Eindruck , daß das Licht heller geworden ist ?
acl:relcl	Spezifischer Typ von acl, für Relativsatz	Ein spezifischer Typ von acl, bei dem ein Relativsatz (eingeleitet durch Relativpronomen) ein Nomen genauer beschreibt.	Der Mann, der im Park sitzt, liest ein Buch.
advcl	Adverbialsatz	Ein Nebensatz, der als Adverbiale Bestimmung dient (z. B. des Grundes, der Zeit, des Zwecks usw.).	Ich bleibe zu Hause, weil es regnet.
parataxis	Parataxe	Zwei Sätze oder Satzteile werden gleichrangig und ohne Konjunktion oder mit lockerem Zusammenhang verbunden.	<pre> graph TD root[root] --> Das[Das] root --> Ambiente[Ambiente] root --> stimmt[stimmt] root --> Punc[punct] Punc --> Komma[„“] Punc --> Verkäufer[Verkäufer] Verkäufer --> die[die] Verkäufer --> sind[sind] Verkäufer --> nett[nett] </pre>
aux	Hilfsverb	Ein Verb, das ein anderes Verb grammatisch unterstützt (z. B. zur Bildung von Zeitformen oder Modus).	Ich habe eine Entscheidung getroffen.

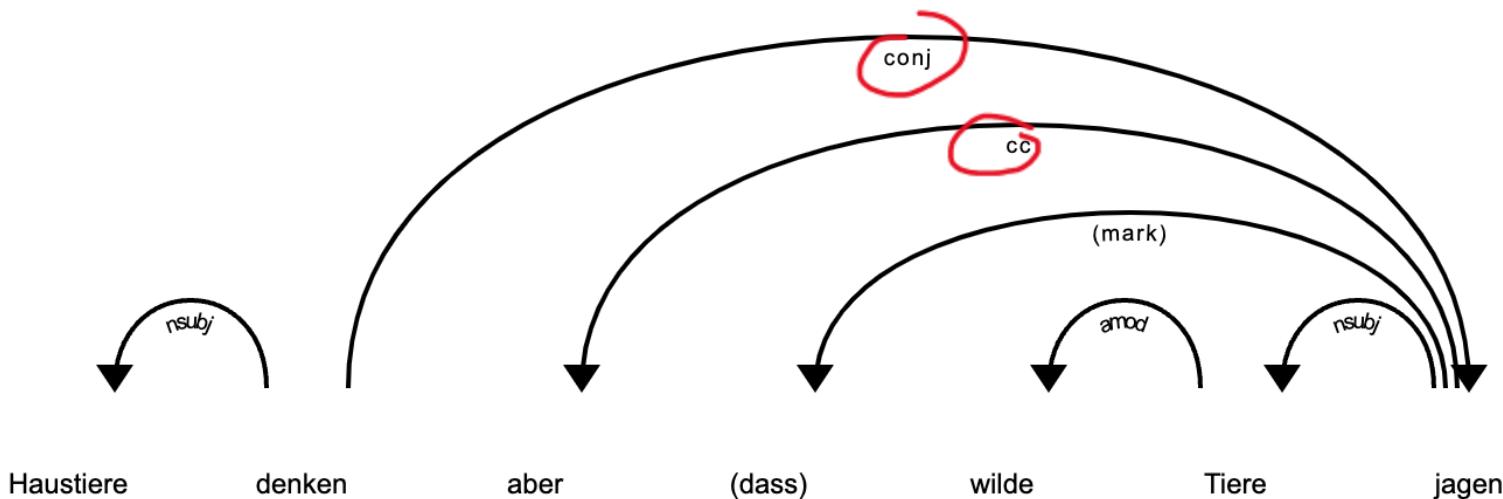
Grundregeln aus Probeklausur

Koordination Grundregeln

1 | S → S CC S
2 | NP → NP CC NP

Koordination

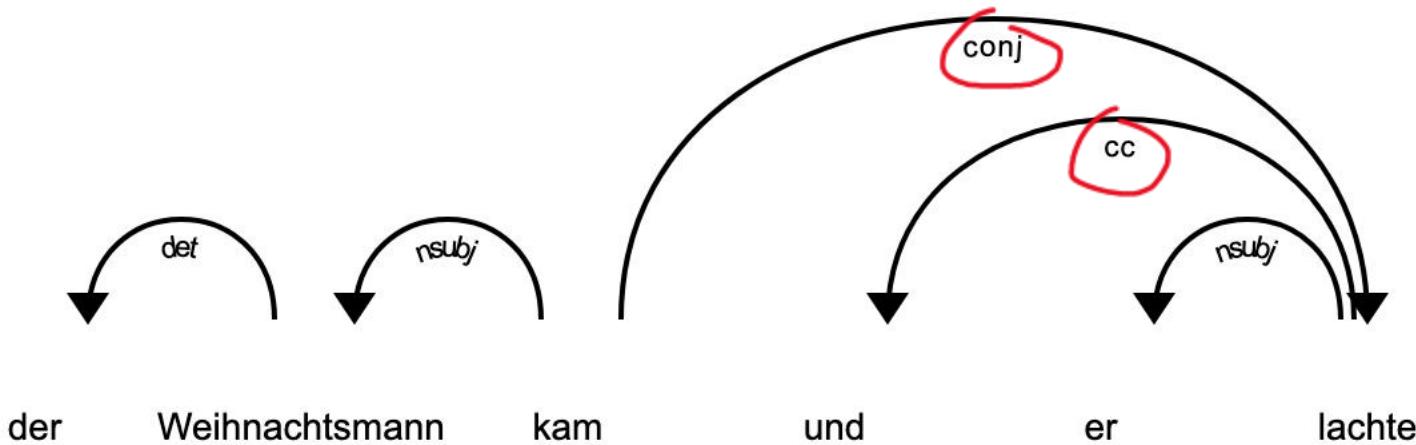
- Häufige benutzte Tags: conj, cc



conj für Koordination (*Erste Ko-Konstituent wird als Kopf der zweiten, sogar dritten, vierten usw. interpretiert*); cc markiert die Konjunktion (*ist Dependent vom verb des koordinierten Satzes*)

Koordination

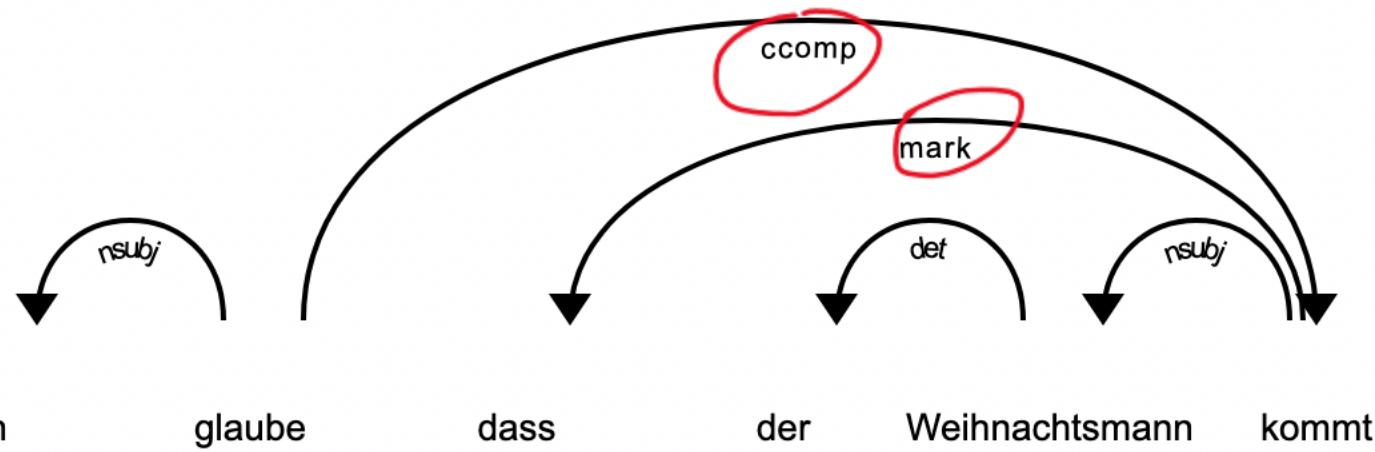
- Häufige benutzte Tags: conj, cc



conj für Koordination (*Erste Ko-Konstituent wird als Kopf der zweiten, sogar dritten, vierten usw. interpretiert*); cc markiert die Konjunktion (*ist Dependent vom verb des koordinierten Satzes*)

Subordination: Objektsatz

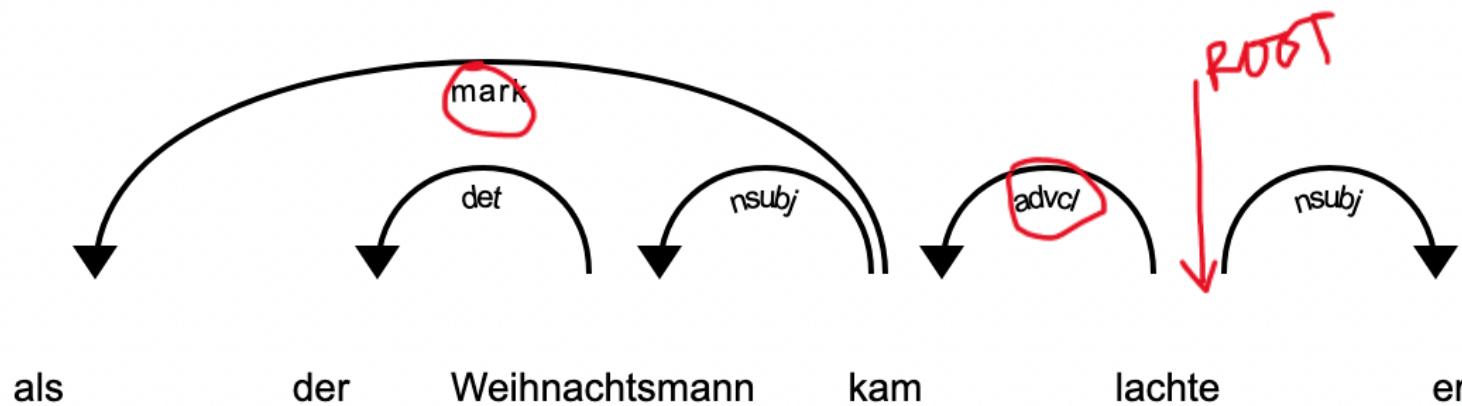
- Häufige benutzte Tags: ccomp, mark



- Das Verb des eingebetteten Satzes ist Dependent des Wurzelknotens des gesamten Satzes und erhält das Label ccomp.
- Die subordinierende Konjunktion erhält das Label mark und ist Dependent vom Verb des eingebetteten Satzes.

Subordination: Adverbialsatz

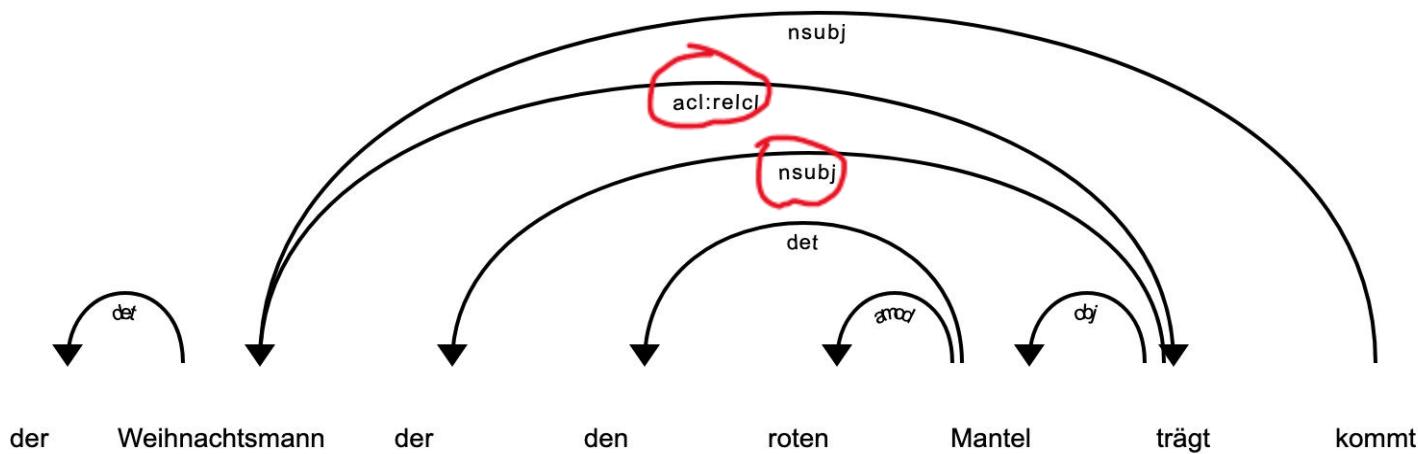
- Häufige benutzte Tags: advcl, mark



- Das Verb des Adverbialsatzes ist Dependent des Wurzelknotens des gesamten Satzes und erhält das Label **advcl**.
- Die subordinierende Konjunktion erhält das Label **mark** und ist Dependent vom Verb des eingebetteten Satzes.

Subordination: Relativsatz

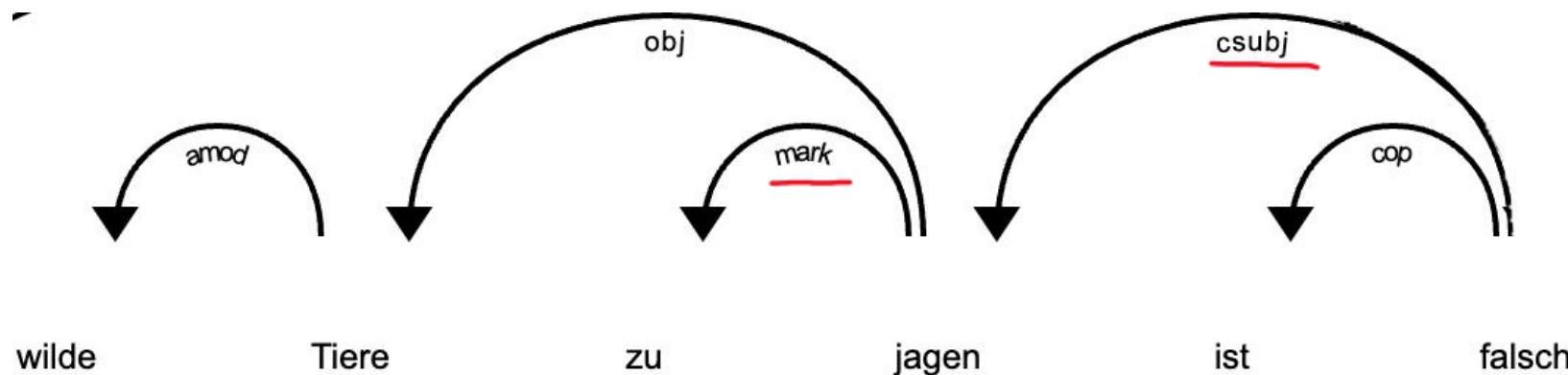
- Häufige benutzte Tags: acl:relcl, nsubj



- Das Verb des Relativsatzes ist Dependent der NP, die der Relativsatz modifiziert, und erhält das Label **acl:relcl**.
 - Das Relativpronomen ist **nsubj** des Relativsatzes.

Infinitiv: als Subjekt

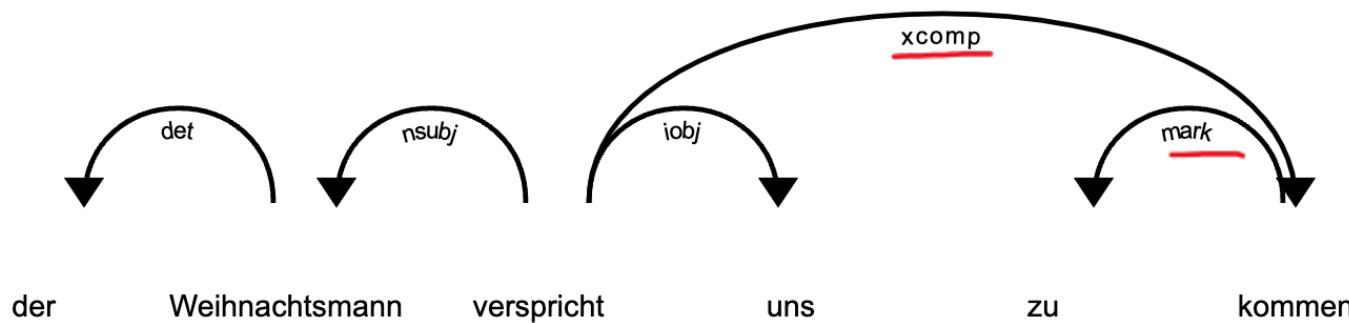
- Häufige benutzte Tags: csubj, mark



- Das infinite Verb ist Dependent des Wurzelknotens des gesamten Satzes und erhält das Label **csubj**.
- Das Wort „zu“ erhält das Label **mark** und ist abhängig vom infiniten Verb.

Infinitiv: als Objekt

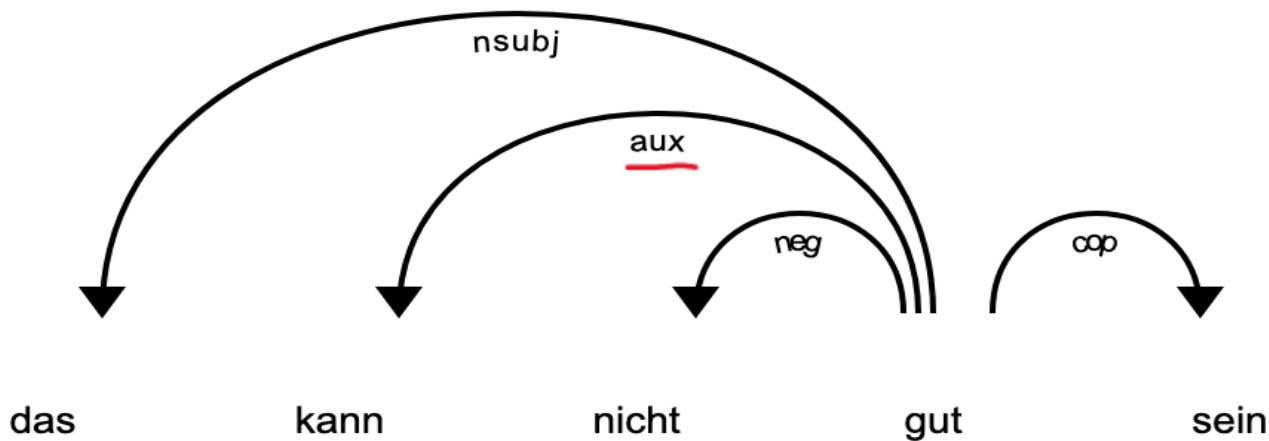
- Häufige benutzte Tags: xcomp, mark



- Das infinite Verb ist Dependent des Wurzelknotens des gesamten Satzes und erhält das Label **xcomp**.
- Das Wort „zu“ erhält das Label **mark** und ist abhängig vom infiniten Verb.

Satz mit Hilfsverb (Auxiliar)

- Häufige benutzte Tag: aux



Keine Einbettung!

“sein“ ist hier Kopula und der Wurzelknoten ist „gut“ (-> „gut sein“ als Prädikat)

Sowohl das Hilfsverb **aux** als auch die Kopula **cop** sind Dependenten von “gut“.

8. Grammatische Merkmale

Feature-basierte Grammatik, Morphosyntaktische Constraints,
Grammatische Merkmale, Unify & Subsumes, Typenhierarchie

Feature-Based Grammar

Wir lösen Übergenerierung, indem wir morphosyntaktische Merkmalsstrukturen (Features) modellieren.

-> **Feature-based Grammar = FCFG (Feature-based context free grammar)**

Wir müssen unsere **Regeln** so **einschränken**, dass ungrammatische Sätze nicht mehr abgeleitet werden können. Dabei müssen wir vor allem die folgenden drei zentralen **morphosyntaktischen Constraints** berücksichtigen:

- **Kasusrektion (Case Government)** z.B. für + Akkusativ; mit + Dativ
- **Kongruenz (Agreement)** Übereinstimmung, z.B. ich bin; er ist; *er bin; *ich ist
- **Subkategorisierung** Anzahl und Art der verbalen Argumente

Morphosyntaktische Constraints

Kasusrektion

Bestimmte Wörter (hauptsätzlich Verb, Präposition) erfordern (regieren) einen bestimmten Fall (Kasus).

Beispiele:

- „warten auf“ + Akkusativ
- „folgen“ + Dativ
- „an etwas glauben“ + Akkusativ
- „gedenken“ + Genitiv
- „mit“ + Dativ
- „für“ + Akkusativ

Morphosyntaktische Constraints

Kongruenz

Übereinstimmung von Wörtern oder Satzteilen hinsichtlich grammatischer Merkmale

Beispiele:

„**Ein** schöner grüner **Baum** steht im Wald.“ (Nom. SG. mask. --- 3. Pers. SG.)

„**Der** schöne grüne **Baum ist** eine Tanne.“ (Nom. SG. mask. --- 3. Pers. SG.)

„Im Wald stehen viele große **Tannen**.“ (Nom. PL. fem. --- 3. Pers. PL.)

„Sieh**st** Du den dunklen **Wald**?“ (Akk. SG. mask. --- 2. Pers. SG.)

-> Übereinstimmung von Subjekt und Verb in Person und Numerus

-> Übereinstimmung von Kasus, Numerus und Genus in NPs

Morphosyntaktische Constraints

Subkategorisierung

Art und Anzahl der Argumente (Valenz)

Beispiele:

- **intransitive Verben**: Subjekt im Nominativ
wachsen -> „Der Baum wächst.“
- **transitive Verben**: Subjekt im Nominativ + Objekt (meistens im Akk., aber nicht immer)
beantworten -> „Meine Freundin beantwortet den Brief.“ (Akkusativobjekt)
gehorchen -> „Der Hund gehorcht ihm.“ (Dativobjekt, in UD auch als obj markiert)
- **ditransitive Verben**: Subjekt im Nominativ + Akkusativobjekt + Dativobjekt
bringen -> „Der Postbote bringt ihr ein Paket.“

2 Typen von Flexion

- **Deklination** (Nomen, Pronomen, Adjektiven und Artikeln)
 - Grammatische Kategorien: Kasus, Numerus, Person, Genus, Definitheit
- **Konjugation** (Verben)
 - Grammatische Kategorien: Person, Numerus, Tempus, Modus, Genus Verbi

Grammatische Merkmale

- Jedes Merkmal hat seine Merkmalsausprägung (Wert)

Grammatisches Merkmal	Werte (Merkmalsausprägungen)
Numerus	Singular, Plural
Person	1, 2, 3
Kasus	Nominativ, Genitiv, Akkusativ, Dativ

Merkmalsstrukturen

Merkmalsstrukturen (= **Feature Structures**) sind Merkmal-Wert-Paare:

$$\text{Merkmalsstruktur} = \begin{bmatrix} \text{MERKMAL1} & \text{WERT1} \\ \text{MERKMAL2} & \text{WERT2} \end{bmatrix}$$

Grammatische Merkmale lassen sich als Merkmalsstruktur darstellen:

$$N \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{CASE} & \text{NOM} \end{bmatrix} \quad \text{oder} \quad \begin{bmatrix} \text{CAT} & \text{N} \\ \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{CASE} & \text{NOM} \end{bmatrix} \quad \text{CAT} = \text{Category}$$

Strukturierung Sprachlicher Kategorien

- **Unifikation (Vereinigung)**
 - Zwei Merkmalsstrukturen **unifizieren**, wenn sie vereinbar sind.
- **Subsumption (Unterordnung)**
 - Eine Merkmalsstruktur **f1 subsumiert** eine andere Merkmalsstruktur **f2** genau dann, wenn die **in f1 enthaltene Information auch in f2** enthalten ist.

Unifikation

- Zwei Merkmalsstrukturen unifizieren, wenn sie vereinbar sind.
- Das bedeutet, dass es **keine widersprüchlichen Merkmal-Wert-Paare** in den Merkmalsstrukturen geben darf.
(Voraussetzung, **sonst schlägt die Unifikation fehl**)
- Wenn diese Voraussetzung erfüllt wird, dann ist diese Operation ähnlich wie A ∪ B in Mengenoperationen.

$$\left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{CASE} & \text{NOM} \end{array} \right] \sqcup \left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{CASE} & \text{NOM} \end{array} \right] = \left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{CASE} & \text{NOM} \end{array} \right]$$

Unifikation: Festlegung Fehlender Informationen

In diesem Beispiel ist die zweite Merkmalsstruktur **underspezifiziert**, weil kein Merkmal-Wert-Paar für den Kasus angegeben wird:

$$\left[\begin{array}{ll} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{CASE} & \text{NOM} \end{array} \right] \sqcup \left[\begin{array}{ll} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{CASE} & \text{NOM} \end{array} \right] = \left[\begin{array}{ll} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{CASE} & \text{NOM} \end{array} \right]$$

Beide Merkmalsstrukturen unifizieren, da sie nicht widersprüchlich sind.

Zum Beispiel könnte die erste Merkmalsstruktur den Artikel „der“ repräsentieren, die zweite das Substantiv „Baum“. „Baum“ kann aber nicht nur Nominativ sondern auch Akkusativ und Dativ sein. Deshalb legen wir in der Merkmalsstruktur den Kasus nicht fest. In dem Beispiel unifiziert „Baum“ mit dem Artikel „der“ zu der NP „der Baum“. Dadurch wird auch der Kasus von „Baum“ (Nominativ) in diesem konkreten Fall festgelegt.

Subsumption

- Eine Struktur A subsumiert eine andere Struktur B, wenn alle Merkmale von A in B enthalten sind
- Wenn **keine spezifischen Merkmale angegeben sind**, enthält die Struktur **alle möglichen Informationen**.
 - Z.B.:

A: [CASE: Nominativ] ==
A: [NUM: -, CASE: Nominativ, GEN: -, andere Merkmale (muss man nicht schreiben)]
wenn man anderen unspezifizierten Merkmalen schreiben möchte, kann man einfach wie „GEN: -“ schreiben.
B: [NUM: Singular, CASE: Nominativ, GEN: Maskulin] → A subsumiert B
 - D.h. B ist spezifischer als A; A ist allgemeiner als B.

Unifikation und Subsumption in NLTK

Methode unify():

f1.unify(f2)

-> gibt bei erfolgreicher Unifikation die unifizierte Merkmalsstruktur zurück, ansonsten NONE

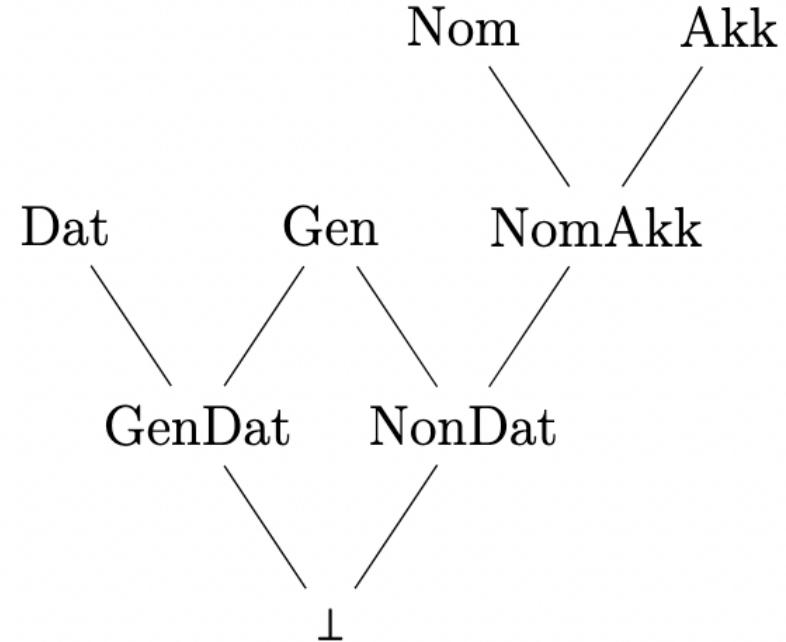
Methode subsumes():

f1.subsumes(f2)

-> gibt bei erfolgreicher Subsumption TRUE zurück, ansonsten FALSE

Typenhierarchie

- Die Typenhierarchie ist als ein Baumdiagramm dargestellt, das verschiedene Typen und deren Beziehungen zueinander beschreibt.
- **Allgemeine Typen** befinden sich weiter unten in der Hierarchie, während **spezifischere Typen** weiter oben angeordnet sind.
- Allgemeine Typen subsumiert ihre Tochterknoten. Wie z. B. NonDat subsumiert NomAkk und Nom.



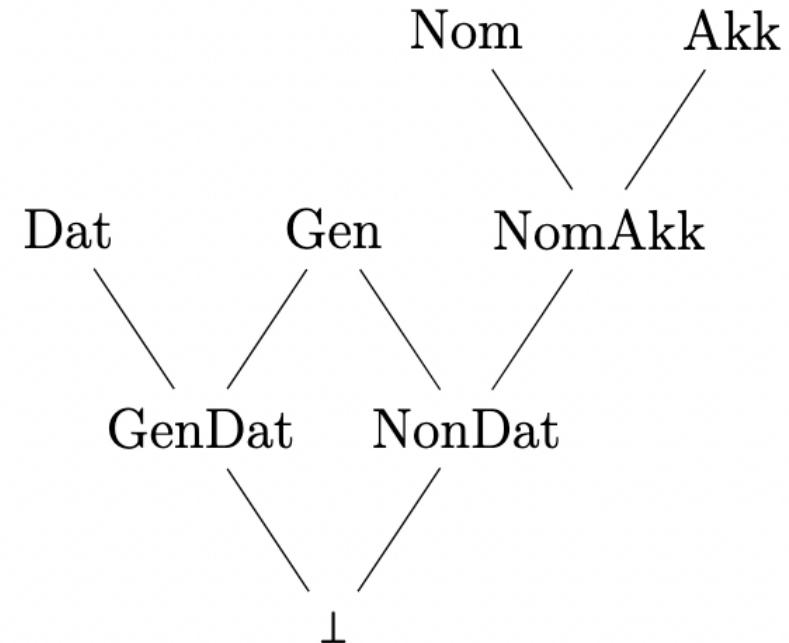
Typenhierarchie

- \perp (**Bottom-Typ**)

Am unteren Ende der Hierarchie befindet sich der Typ \perp , der als der **allgemeinste Typ** gilt.

- Eigenschaften von \perp :

- Er ist die Basis für alle anderen Typen und enthält keine spezifischen Merkmale.
- Jeder andere Typ der Hierarchie ist eine Spezialisierung von \perp .
- \perp **subsumiert alle anderen Typen**, da er die allgemeinste Kategorie darstellt.



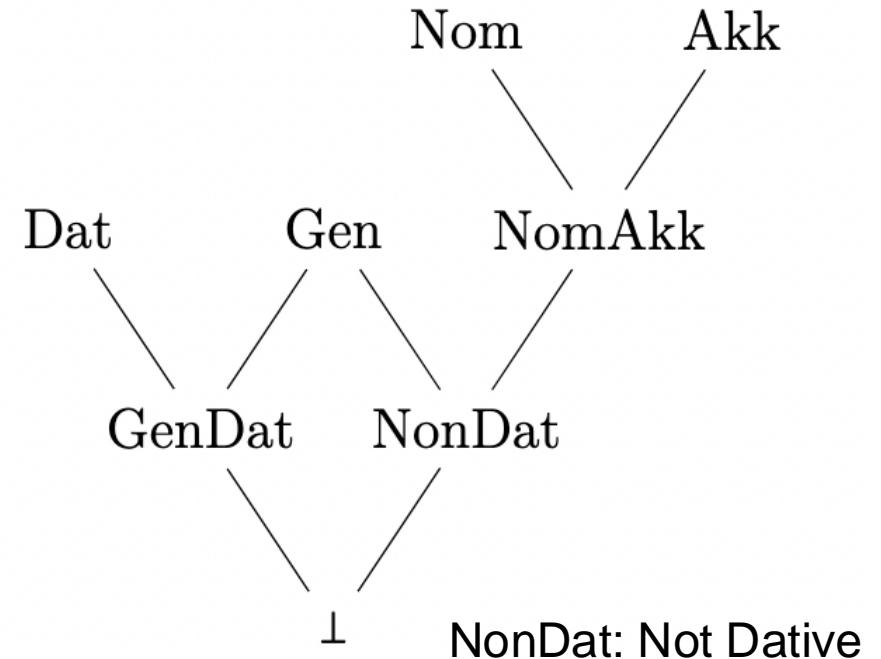
Typenhierarchie

- **Spezifische Typen:**

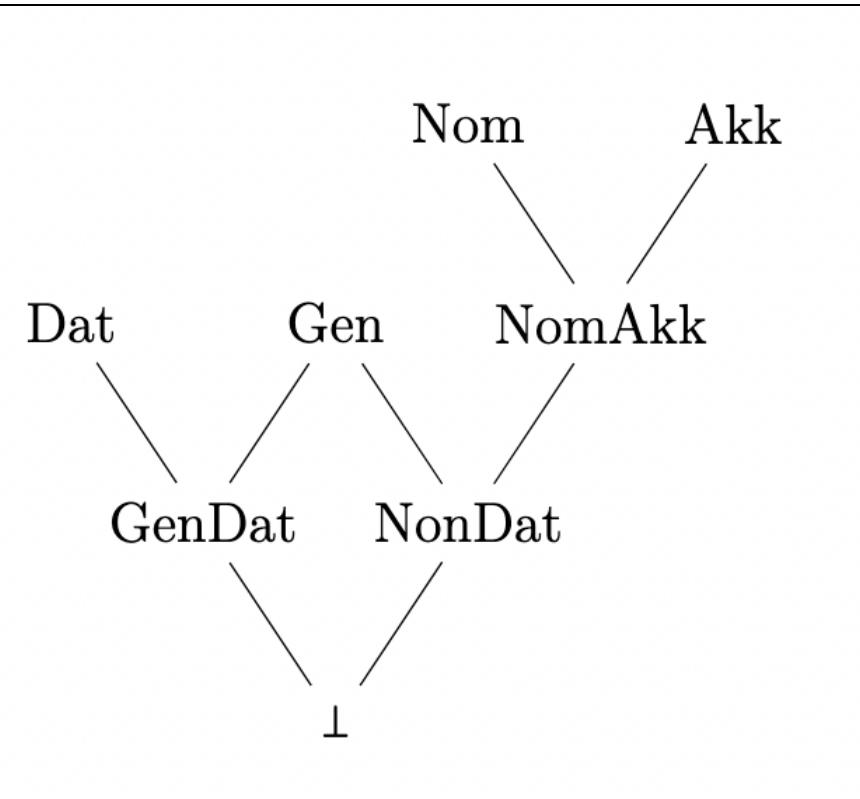
Höher in der Hierarchie sind spezifischere Typen zu finden, die jeweils besondere Merkmale besitzen. In diesem Fall zeigt die Typenhierarchie verschiedene **Kasus-Typen** wie **Nom** (Nominativ), **Akk** (Akkusativ), **Dat** (Dativ) und **Gen** (Genitiv).

- **Beispiel:**

- **Nom** ist ein Typ, der sich nur auf den Nominativ bezieht (spezifisch).
- **NomAkk** ist ein Typ, der sowohl den Nominativ als auch den Akkusativ umfasst (allgemeiner).



Beispiel einer Typenhierarchie:



In NLTK definieren wir die Typenhierarchie als Dictionary:

```
case_hierarchy = {  
    "nondat": ["gen", "nomakk"],  
    "gendat": ["gen", "dat"],  
    "nomakk": ["nom", "akk"],  
    "nom": [],  
    "gen": [],  
    "dat": [],  
    "akk": []  
}
```

9. Feature-basierte Grammatik

Constraint-Regeln

```
gramstring = r"""
% start S
S   -> NP VP

VP  -> V[OBJCASE=?y] NP[CASE=?y]
VP -> V

NP[CASE=?x]  -> DET[CASE=?x] N
DET[CASE=nom] -> "der"
DET[CASE=akk] -> "den"
DET[CASE=dat] -> "dem"
N   -> "Hund" | "Briefträger"
V   -> "schläft"
V[OBJCASE=akk] -> "jagt"
V[OBJCASE=dat] -> "gehört"
....
```

OBJCASE von V stimmt mit CASE von NP überein,
damit V NP eine gültige VP formt

Beispiel von Constraint-Regeln in NLTK

Vererbung: CASE der NP wird vom CASE des DET vererbt

Subkategorisierung

Koreferenz

Zwei Merkmale sind **koreferent**, wenn sie denselben Wert (bzw. bei komplexen Merkmalen dieselbe Merkmalsstruktur) teilen.

```
gramstring = r"""
% start S
S   -> NP VP

VP  -> V[OBJCASE=?y] NP[CASE=?y]
VP -> V

NP [CASE=?x]  -> DET [CASE=?x] N

DET [CASE=nom] -> "der"
DET [CASE=akk] -> "den"
DET [CASE=dat] -> "dem"
N   -> "Hund" | "Briefträger"
V   -> "schläft"
V[OBJCASE=akk] -> "jagt"
V[OBJCASE=dat] -> "gehört"

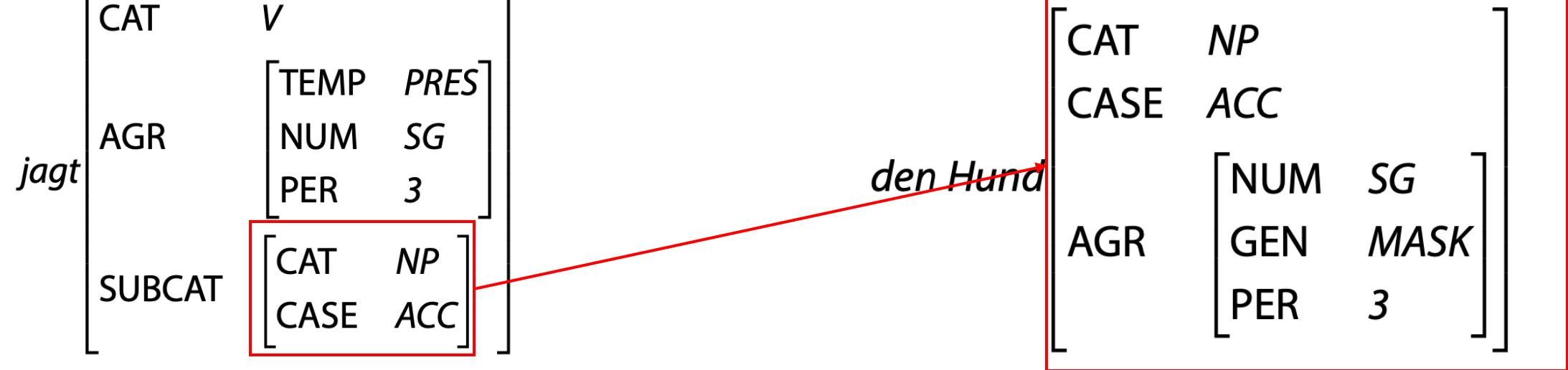
....
```

Z.B. fordern wir in dieser Grammatik bei der Regel

VP -> V[OBJCASE=?y] NP[CASE=?y]

dass, die Merkmale OBJCASE und CASE koreferent sind.

Constraintregeln: Unifikation



intransitives Verb:

SUBCAT: NONE (ohne Komplement)

Unifikation zwischen SUBCAT [CAT: NP, CASE: ACC] und [CAT: NP, CASE: ACC, AGR = [...]] **erfolgt**

Richtige Kombination = Erfolgreiche Unifikation

Constraintregeln: Unifikation

	CAT	V		
<i>bellt</i>	AGR			
	SUBCAT	<table border="1"><tr><td>NONE</td></tr></table>	NONE	
NONE				

	CAT	NP				
	CASE	ACC				
<i>den Hund</i>	AGR					
	SUBCAT	<table border="1"><tr><td>NUM SG</td></tr><tr><td>GEN MASK</td></tr><tr><td>PER 3</td></tr></table>	NUM SG	GEN MASK	PER 3	
NUM SG						
GEN MASK						
PER 3						

intransitives Verb:

SUBCAT: NONE (ohne Komplement)

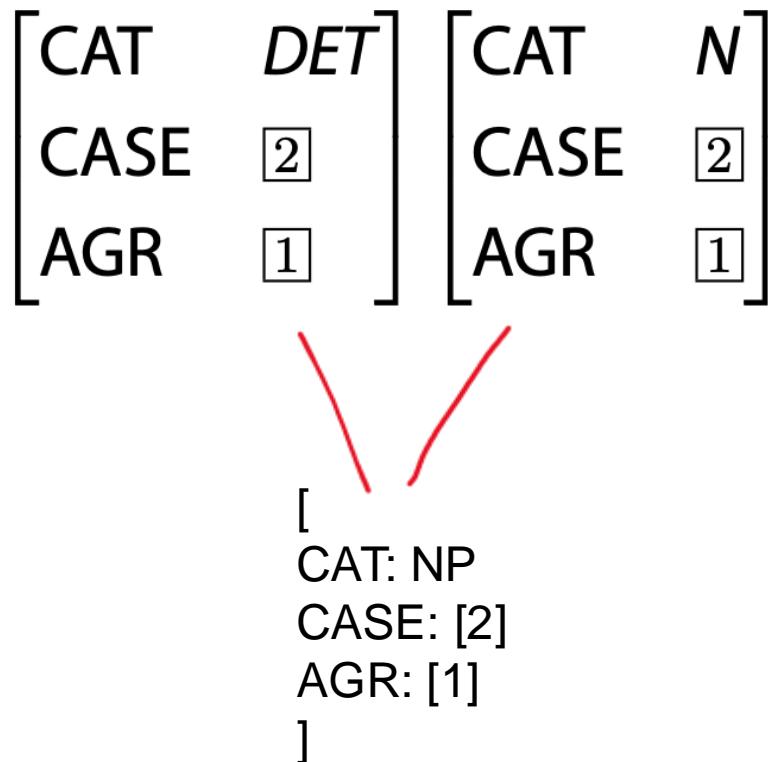
Unifikation zwischen NONE und [CAT: NP, CASE: ACC, AGR = [...]] **schlägt fehl**

Achtung: **NONE != []**

[] unifiziert mit allem, da es kein Widerspruch gibt

Kongruenz

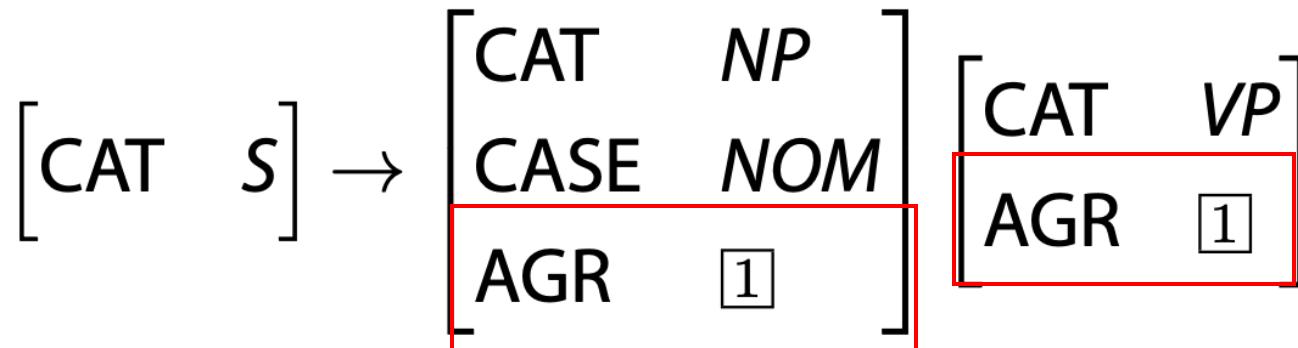
Wenn CASE und AGR in DET und N unifizieren, dann formt DET N eine NP



Artikel-Nomen-Kongruenz: Kasus [Genus, Numerus, ...]

Kongruenz

vermeidet Überproduktion!



Dieser Regel zufolge kann S nur dann zu NP VP expandieren, wenn die NP Nominativ steht (**CASE NOM**).

Zusätzlich müssen NP und VP hinsichtlich ihrer Merkmalsausprägungen von **AGR koreferent** sein.

(Die 1 ist hier eine Variable, die für eine bestimmte Merkmalsausprägung steht.)

Subjekt-Verb-Kongruenz

$$\begin{bmatrix} \text{CAT} & NP \\ \text{CASE} & \text{NOM} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix}, \quad \begin{bmatrix} \text{CAT} & VP \\ \text{AGR} & \begin{bmatrix} \text{TEMP} & \text{PRES} \\ \text{NUM} & \text{SG} \\ \text{PER} & 3 \end{bmatrix} \\ \text{SUBCAT} & \text{NONE} \end{bmatrix}$$

-> AGR unifiziert

This table shows two feature matrices for the NP and VP categories. The NP matrix includes features for Case (NOM), Number (SG), Gender (MASK), and Person (3). The VP matrix includes features for Tense (PRES), Number (SG), and Person (3). The AGR node in both cases is highlighted with a red box.

$$\begin{bmatrix} \text{CAT} & NP \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{GEN} & \text{FEM} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix}, \quad \begin{bmatrix} \text{CAT} & VP \\ \text{AGR} & \begin{bmatrix} \text{TEMP} & \text{PRES} \\ \text{NUM} & \text{PL} \\ \text{PER} & 3 \end{bmatrix} \\ \text{SUBCAT} & \text{NONE} \end{bmatrix}$$

-> AGR unifiziert nicht
(Widerspruch im Numerus)

This table shows two feature matrices for the NP and VP categories. The NP matrix includes features for Number (SG) and Gender (FEM). The VP matrix includes features for Number (PL) and Person (3). The AGR node in both cases is highlighted with a red box.

Boolsche Merkmale: Invertierte Wortstellung

Aussagesatz: "You like chocolate."

Fragesatz: "Do you like chocolate?" -> Hilfsverb an erster Position

Die invertierte Wortstellung lässt sich vermittels boolscher Merkmale modellieren:

Inversionsmerkmal: [+/-Inv] **[+Inv] = [Inv=True], [-Inv] = [Inv=False]**

Auxiliarmerkmal: [+/-Aux]

Entsprechend können wir die CFG-Regeln für invertierte Sätze so formulieren:

S[+Inv] -> V[+Aux] NP VP[-Aux]

VP[-Aux] -> V[-Aux] NP

-> Bei einem Satz mit invertierter Wortstellung muss an erster Position ein Hilfsverb stehen, gefolgt von einer NP und einer VP, wobei das Verb der VP kein Hilfsverb sein darf.

Boolsche Merkmale: Hilfsverbkonstruktionen

Das Auxiliarmerkmal können wir auch verwenden, um Hilfsverbkonstruktionen im Deutschen zu modellieren:

Präsens: „Das Eichhörnchen **vergräbt** die Nuss.“

Perfekt: „Das Eichhörnchen **hat** die Nuss **vergraben**.“

Um Übergenerierung zu vermeiden, brauchen wir zusätzlich ein boolsches Merkmal für das Partizip Perfekt: [+/-PP]

Boolsche Merkmale: Hilfsverbkonstruktionen

S → NP VP

VP[-INV] → VERBAL

VERBAL → V[-PP] NP

VP[+INV] → V[+AUX] VERBAL

VERBAL → NP V[-AUX, +PP]

NP → DET NOM

NOM → N

DET → "das" | "die"

N → "Eichhörnchen" | "Nuss"

V[-AUX, -PP] → "vergräbt"

V[+AUX] → "hat"

V[-AUX, +PP] → "vergraben"

Wir verwenden dafür das XBAR-Schema.

Spezifiererregeln:

VP[-INV] → VERBAL (für Präsens)

VP[+INV] → V[+AUX] VERBAL (für Perfekt)

Komplementregeln:

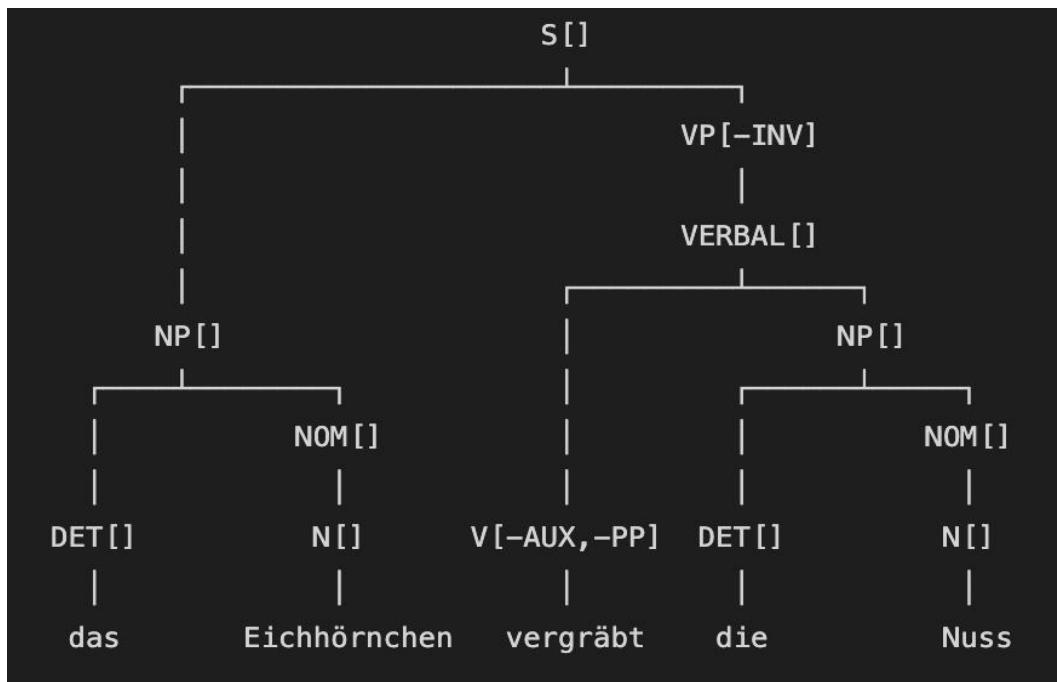
VERBAL → V[-PP] NP (für
Präsens)

VERBAL → NP V[-AUX, +PP] (für Perfekt)

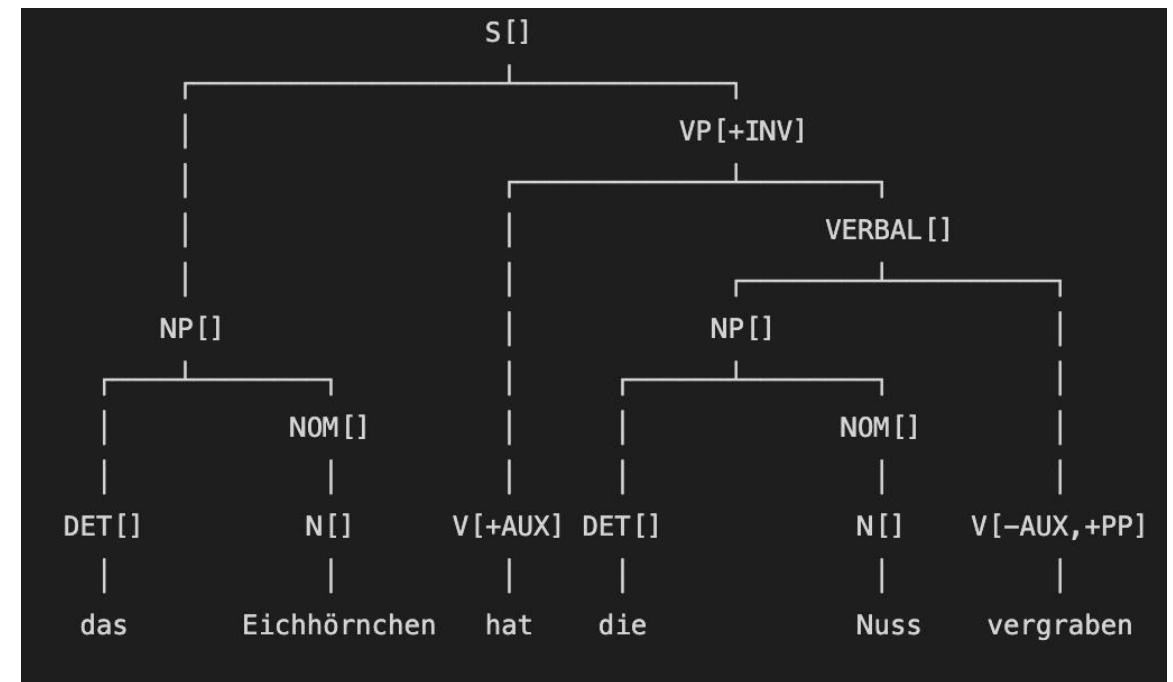
(Adjunktregeln brauchen wir hierfür nicht.)

Partizip Perfekt: [+/-PP]

Boolsche Merkmale: Hilfsverbkonstruktionen



**Präsens
(nicht invertiert)**



**Perfekt
(invertiert)**

Gap-Feature

- Gap-Feature beschreibt die Existenz von Lücken oder *missing positions* innerhalb eines Satzes. Diese Lücken entstehen meist aufgrund von Bewegung oder Extraktion (z. B. bei WH-Bewegungen oder relativen Konstruktionen).
- WH-Bewegung
 - Was hast du ___ gesehen?
 - Das Fragewort „was“ wurde verschoben, und das Gap (___) verweist auf die Position, an der ursprünglich ein Objekt stand.
- Extraktion
 - Das Buch, das ich ___ gekauft habe.
 - Das Relativpronomen "das" füllt die Lücke aus, die durch die Bewegung des Objekts verursacht wurde (___ steht für die Position von "das").

Gap-Feature

- Das Gap-Feature wird oft durch Merkmale wie [+/-WH] ausgedrückt:
 - +WH zeigt an, dass ein Satz oder Knoten ein WH-Element enthält, das eine Lücke (Gap) verursacht hat, die aufgelöst oder mit einer passenden Ergänzung verbunden werden muss.
 - WH: - bedeutet, dass keine Lücke enthalten ist.

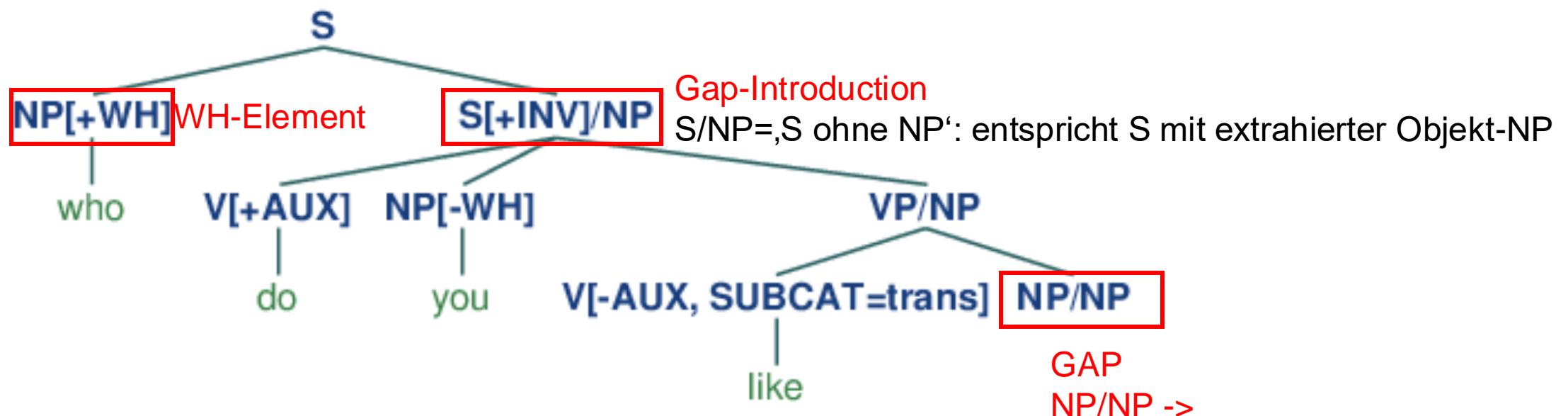


Abbildung: (http://www.nltk.org/book/tree_images/ch09-tree-16.png)

Gap-Features mit Slash-Kategorien

Wir wollen eine Grammatik schreiben, aus der wir die Sätze „Sie hat ihn gesehen“ und „Wen hat sie gesehen?“ ableiten können.

Dazu führen wir zwei boolsche Merkmale ein:

[+/-QUEST] für das Fragepronomen

[+/-MOVEMENT] für den Ergänzungsfragesatz

#Movement Objekt (Gap-Introduction):

S [+MOVEMENT] → NP [+QUEST] S [+INV]/NP

#Gap-Informationen herunterreichen:

S [+INV]/?x → V [+AUX] NP VP/?x

VP/?x → NP/?x V [+PP]

#Gap-Realisierung:

NP/NP →

Wir legen fest,

- dass im Ergänzungsfragesatz die NP ein Fragepronomen sein muss
- dass nach dem Fragepronomen ein Satz mit invertierter Wortfolge ohne NP stehen muss

(Statt der Variable ?x könnten wir auch /NP weiter herunterreichen.)

Gap-Features mit Slash-Kategorien

$S[-INV] \rightarrow NP[-QUEST] VP[+INV]$
 $VP[+INV] \rightarrow V[+AUX] NP[-QUEST] V[+PP]$
 $NP[QUEST=?x] \rightarrow PRON[QUEST=?x]$

$S[+INV] \rightarrow V[+AUX] NP VP$
 $VP \rightarrow NP V[+PP]$

#Movement Objekt (Gap-Introduction):

$S[+MOVEMENT] \rightarrow NP[+QUEST] S[+INV]/NP$

#Gap-Informationen herunterreichen:

$S[+INV]/?x \rightarrow V[+AUX] NP VP/?x$

$VP/?x \rightarrow NP/?x V[+PP]$

#Gap-Realisierung:

$NP/NP \rightarrow$

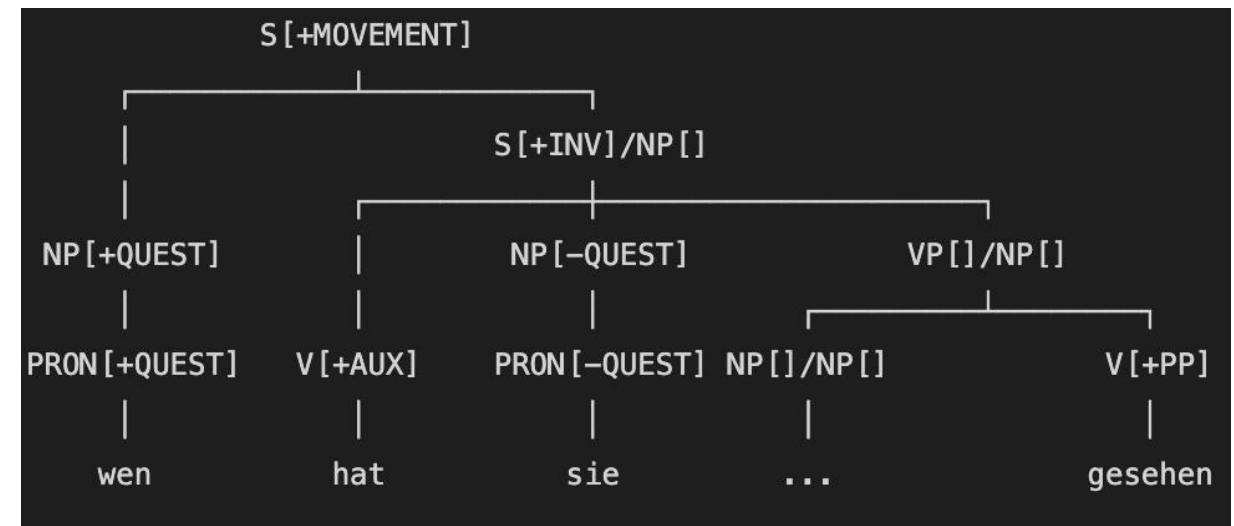
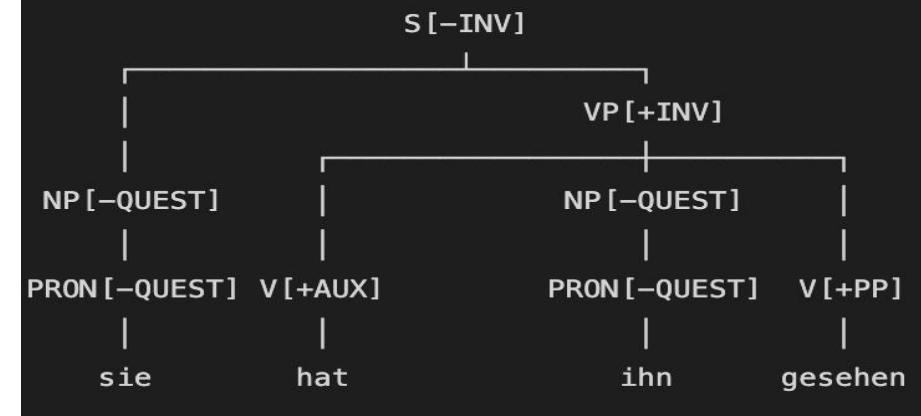
$PRON[-QUEST] \rightarrow "sie"$

$PRON[-QUEST] \rightarrow "ihn"$

$PRON[+QUEST] \rightarrow "wen"$

$V[+AUX] \rightarrow "hat"$

$V[+PP] \rightarrow "gesehen"$



(Statt der Variable ?x könnten wir auch /NP weiter herunterreichen.)

10. Probabalistic CFG

PCFG-Wahrscheinlichkeiten, 3 Parser

Was ist Probabilistic Context Free Grammar?

- Context-Free-Grammar + Probability
- Ziel:
 - korrekte, plausible oder wahrscheinlichere Struktur-Interpretationen vorzuschlagen.
 - Syntax-Disambiguierung
- Liefert die mögliche Parsebäume (basiert auf Korpusdaten) zurück
- Regeln:
 - Die Wahrscheinlichkeiten aller Regeln mit **derselben linken Seite** summieren zu 1 (d.h. sie ergeben eine Wahrscheinlichkeitsverteilung)
 - Die Wahrscheinlichkeit eines Parsebaumes ist das Produkt seiner Regelwahrscheinlichkeiten

Wahrscheinlichkeit-Regel 1

1. $S \rightarrow NP\ VP$

2. $NP \rightarrow DET\ N$

3. $VP \rightarrow V\ NP$

WK: 70% für $VP \rightarrow V\ NP$

4. $VP \rightarrow V$

Wahrscheinlichkeit für $VP \rightarrow V$?

5. $DET \rightarrow "der" | "die" | "das" | "ein" | "eine"$ Antwort: 30%

6. $N \rightarrow "Hund" | "Katze" | "Maus" | "Apfel"$

denn Wahrscheinlichkeiten aller Regeln für die Expansion eines bestimmten Nonterminals addieren sich zu 1

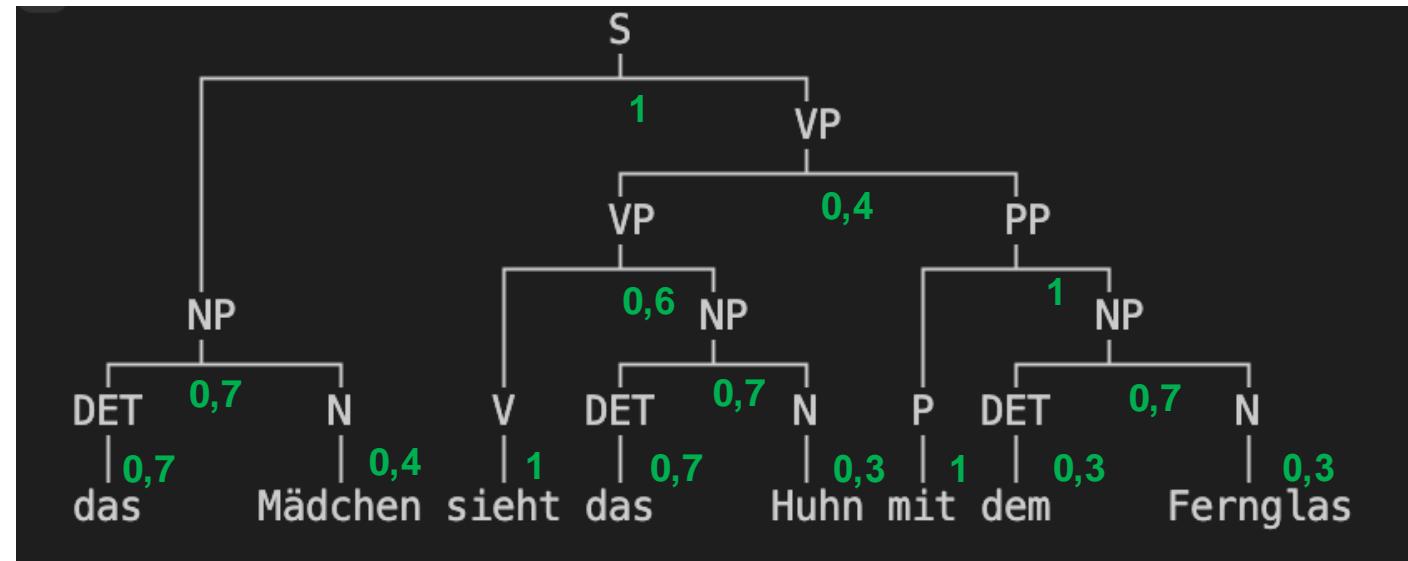
7. $V \rightarrow "frisst" | "schläft" | "läuft" | "spielt"$

Wahrscheinlichkeit-Regeln 2

„das Mädchen sieht das Huhn mit dem Fernglas.“

S	-> NP VP	(1)
NP	-> Det N	(0,7)
NP	-> NP PP	(0,3)
VP	-> V NP	(0,6)
VP	-> VP PP	(0,4)
PP	-> P NP	(1)
Det	-> das	(0,7)
Det	-> dem	(0,3)
N	-> Mädchen	(0,4)
N	-> Huhn(0,3)	
N	-> Fernglas	(0,3)
V	-> sieht	(1)
P	-> mit	(1)

Lesart 1:



Die Wahrscheinlichkeit eines Parsebaumes ist das Produkt seiner Regelwahrscheinlichkeiten:

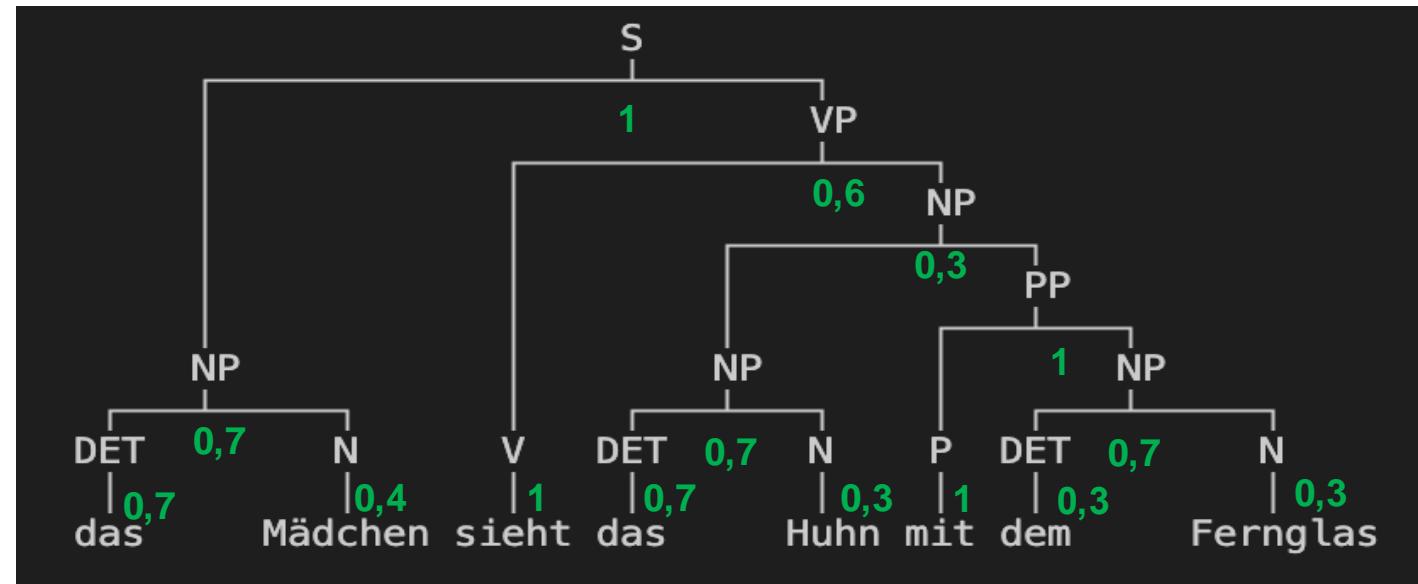
$$P(T1) = 1 * 0,7 * 0,7 * 0,4 * 0,4 * 0,6 * 1 * 0,7 * 0,7 * 0,3 * 1 * 1 * 0,7 * 0,3 * 0,3 \\ = 0,0004356374$$

Wahrscheinlichkeit-Regeln 2

„das Mädchen sieht das Huhn mit dem Fernglas.“

S	-> NP VP	(1)
NP	-> Det N	(0,7)
NP	-> NP PP	(0,3)
VP	-> V NP	(0,6)
VP	-> VP PP	(0,4)
PP	-> P NP	(1)
Det	-> das	(0,7)
Det	-> dem	(0,3)
N	-> Mädchen	(0,4)
N	-> Huhn(0,3)	
N	-> Fernglas	(0,3)
V	-> sieht	(1)
P	-> mit	(1)

Lesart 2:



Die Wahrscheinlichkeit eines Parsebaumes ist das Produkt seiner Regelwahrscheinlichkeiten:

$$\begin{aligned}P(T2) &= 1 * 0,7 * 0,7 * 0,4 * 0,6 * 1 * 0,3 * 0,7 * 0,7 * 0,3 * 1 * 1 * 0,7 * 0,3 * 0,3 \\&= 0,0003267281\end{aligned}$$

PCFG-Regeln

$P(T1) = 0,0004356374$

$P(T2) = 0,0003267281$

-> **T1 ist der wahrscheinlichere Parsebaum**

Die **Wahrscheinlichkeit des Satzes** ist die Summe der Wahrscheinlichkeiten aller seiner möglichen Ableitungen (Parsebäume).

In unserem Beispiel: $P(S) = P(T1) + P(T2) = 0,0007623655$

Woher erhalten wir die Regelwahrscheinlichkeiten?

-> Sie werden aus Regelhäufigkeiten geschätzt.

Regelhäufigkeiten erhalten wir

- (1) durch Zählen der Regeln in einer Treebank
- (2) aus einem automatisch geparstenen und disambiguierten Korpus

PCFG Parsing

Viterbi Parser

-> Gibt den **wahrscheinlichsten** Parsebaum (und **nur** diesen!) zurück

Probabilistische Chart-Parser

- probabilistische Varianten von Chart-Parsing-Algorithmen (Earley, CYK)
- NLTK: InsideChartParser, LongestChartParser
- haben Zustandsmengen (**edge queue**), die nach unterschiedlichen Kriterien sortiert werden
- können:
 - **lowest cost first** (bei InsideChartParser)
Sortierung nach Wahrscheinlichkeit, findet also immer die wahrscheinlichste Ableitung
 - **beam search** (bei Inside Chart Parser beam_size definieren)
wie lowest cost first, aber es werden nur die n (beam size) wahrscheinlichsten Ableitungen behalten
 - **best first search** (bei LongestChartParser)
Sortierung nach Länge der Ableitung (hat zur Folge, dass evtl. nicht die wahrscheinlichste Ableitung an erster Stelle steht)

11. Grammatikinduktion und Annotationen

Grammatikinduktion

- Aus annotierte Kopura (Sg. Korpus) können Grammatikregeln extrahiert werden.
- Wahrscheinlichkeiten der Grammatikregeln basiert sich auf ihren relativen Häufigkeiten.
 - Für Generierung korrekter Parsebäume wichtig.
 - Formel: $P(\alpha \rightarrow \beta | \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$
 $P(\underline{\alpha} \rightarrow \beta | \underline{\alpha})$: $\underline{\alpha} \rightarrow \beta$ gegeben $\underline{\alpha}$. d.h. $\underline{\alpha}$ (linke Seite einer Regel) ist schon festgelegt
 $\frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)}$: Anzahl von $\underline{\alpha}$ als linke Seite und eine beliebe γ als rechte Seite
- Beispiel siehe nächste Folie

Berechnung von Regelwahrscheinlichkeiten

Regelwahrscheinlichkeiten werden aus Regelhäufigkeiten berechnet.

Beispiel: In einer Treebank haben wir folgende Regelhäufigkeiten gezählt:

<u>Regel</u>	<u>Häufigkeit</u>
NP -> Det N	200
NP -> Pron	175
NP -> NP PP	125

Wie berechnen wir die Regelwahrscheinlichkeiten für die Regel NP -> Det N?

Formel: $P(\alpha \rightarrow \beta | \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$

$$P(\text{Det N} | \text{NP}) = 200 / (200 + 175 + 125) = 200 / 500 = 0,4$$

$$P(\text{Pron} | \text{NP}) = 175 / 500 = 0,35$$

$$P(\text{NP PP} | \text{NP}) = 125 / 500 = 0,25$$

Chomsky-Normalform (CNF)

Eine kontextfreie Grammatik (CFG) befindet sich in **Chomsky-Normalform (CNF)**, wenn alle Produktionsregeln die folgende Form haben:

1. $A \rightarrow BC$

1. Ein Nichtterminal A wird in **zwei Nichtterminale** B und C deriviert.
2. B und C sind ebenfalls **Nichtterminale**.

2. $A \rightarrow a$

1. Ein Nichtterminal A produziert genau **ein Terminal** a.

3. $S \rightarrow \epsilon$ (optional, nur für leere Sprache)

1. Startsymbol S produziert den leeren String ϵ , falls ϵ in der Sprache enthalten ist.

Konversion nach CNF

Wie bringen wir diese Regel in Chomsky-Normalform?

$A \rightarrow b C D e$

1. Regeln für die Terminale einführen

$B \rightarrow b$

$E \rightarrow e$

$A \rightarrow B C D E$

2. Regeln verkürzen durch das Einführen von Zwischenebenen

$A \rightarrow B X_1$

$X_1 \rightarrow C X_2$

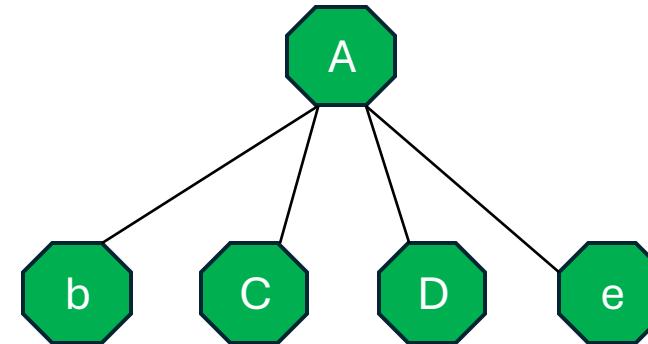
$X_2 \rightarrow D E$

(Diese drei Regeln in Kombination leisten dasselbe wie die Regel $A \rightarrow B C D E$)

Konversion nach CNF

Wie bringen wir diese Regel in Chomsky-Normalform?

$A \rightarrow b C D e$



flache Struktur

Lösung:

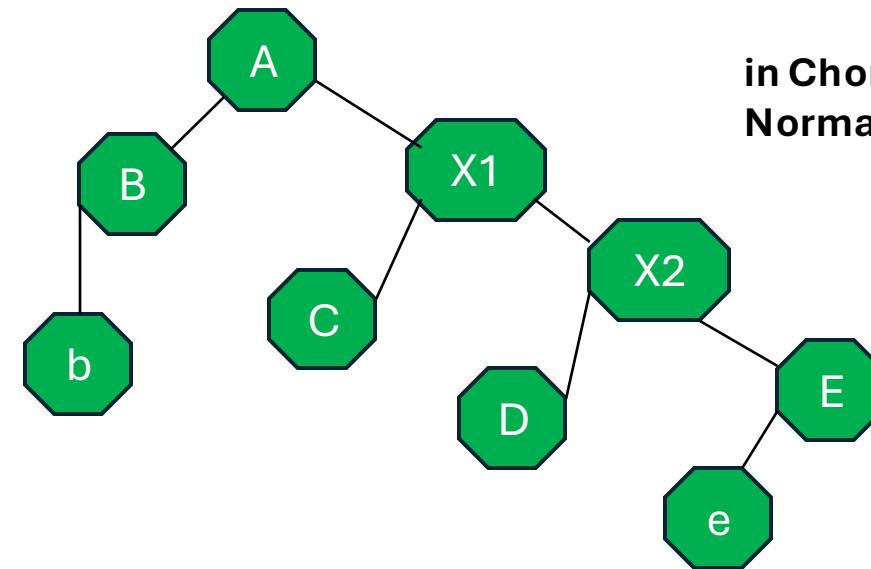
$B \rightarrow b$

$E \rightarrow e$

$A \rightarrow B X_1$

$X_1 \rightarrow C X_2$

$X_2 \rightarrow D E$



in Chomsky-
Normalform

Unabhängigkeitsannahmen von PCFGs

1. Annahme: Unabhängigkeit von lexikalischem Material

- **Erklärung:**
 - In einfachen Modellen wird angenommen, dass die **Wahrscheinlichkeiten von Teilbäumen unabhängig von den Terminalen** (den Wörtern) im Satz sind.
 - Beispiel: In einer CFG betrachten die Produktionsregeln oft nur die syntaktischen Strukturen (z. B. $NP \rightarrow Det+N$), unabhängig davon, **welche Wörter tatsächlich als Det oder N auftreten** (z. B. "der Hund" vs. "die Katze").
- **Problem:**
 - Diese Annahme ist unrealistisch, weil das lexikalische Material (Wörter) oft Einfluss auf die Syntax hat (z. B. regiert ein bestimmtes Verb einen bestimmten Kasus).
- **Modell:**
 - Lexikalierte PCFGs ⇒ Auflösung lexikalischer Ambiguität

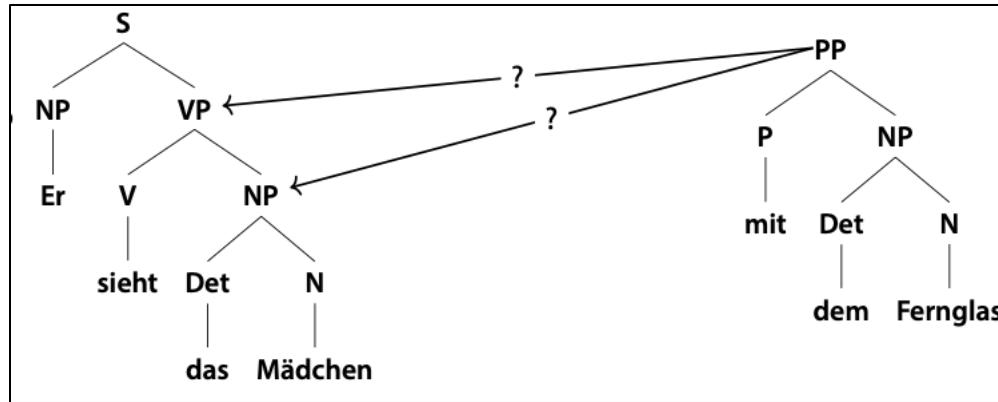
Unabhängigkeitsannahmen von PCFGs

2. Annahme: Unabhängigkeit vom Kontext

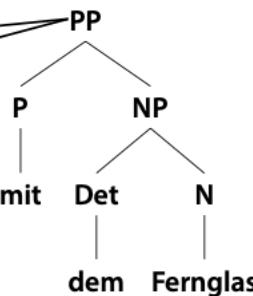
- **Erklärung:**
 - In probabilistischen Modellen könnte angenommen werden, dass die **Wahrscheinlichkeiten von Teilbäumen unabhängig vom Kontext in der Struktur** sind (z. B. welches Elternknoten den Teilbaum dominiert).
 - Beispiel: Die Wahrscheinlichkeit von $VP \rightarrow V + NP$ wird unabhängig betrachtet von ihrer Umgebung, etwa davon, ob der Satz ein Fragesatz oder ein Hauptsatz ist.
- **Problem:**
 - Dies ignoriert linguistische Zusammenhänge – z. B. beeinflusst der Hauptsatz möglicherweise die Struktur des VP.
- **Modell:**
 - History-based PCFGs \Rightarrow Auflösung kontextabhängiger struktureller Ambiguität

Lexikalische PCFGs

Eine nicht-lexikalisierte PCFG gibt bei ambigen Sätzen immer dieselbe (die wahrscheinlichere) Struktur zurück, unabhängig von den verwendeten Wörtern.



Bsp. PP-Attachment Ambiguität



Welche Struktur jedoch tatsächlich wahrscheinlichere (bzw. sinnvollere) ist, hängt vom Vokabular ab:

Er **sieht das Mädchen** mit dem Fernglas.

- Beide Lesarten sinnvoll

Er **sieht das Huhn** mit dem Fernglas.

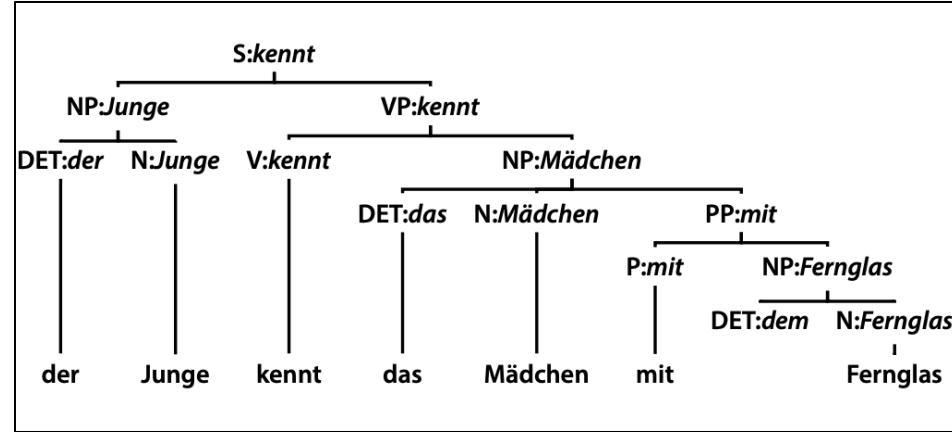
- VP-Attachment bevorzugt

Er **kennt das Mädchen mit dem Fernglas**.

- NP-Attachment bevorzugt

Lexikalische PCFGs

Lösung: Kopfannotation



Phrasenköpfe werden hochgereicht (**Kopf-Perkolation**) und jedes Nicht-Terminal wird mit dem Phrasenkopf annotiert.

Einige Probleme:

- Regelvervielfachung: statt der allgemeinen Regel $VP \rightarrow V\ NP$ haben wir jetzt die Regeln:
 $VP(\text{sieht}) \rightarrow V(\text{sieht})\ NP(\text{Mädchen})$
 $VP(\text{kennt}) \rightarrow V(\text{kennt})\ NP(\text{Mädchen})$. usw. für das gesamte Vokabular
- umfangreiche Trainingsdaten notwendig
- Probleme bei ungesehenen Wörtern (sparse-data problem)

Lexikalische PCFGs

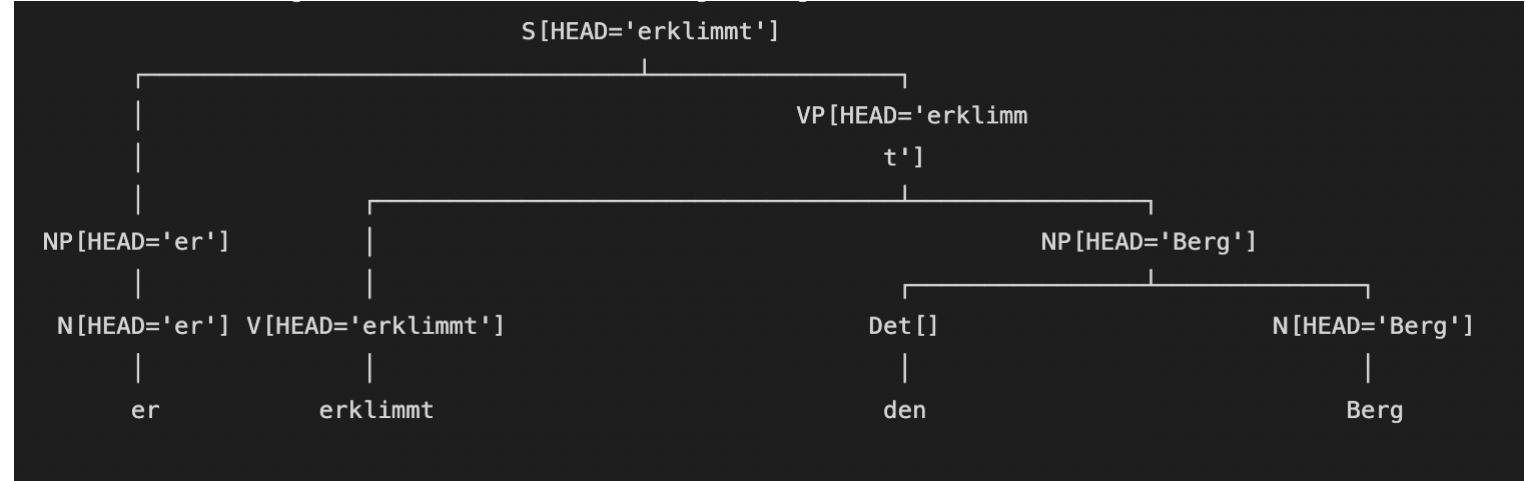
Beispiel einer **Kopfannotation** mit Hilfe eines **HEAD-Features** in einer FCFG

```
sentence = "er erklimmt den Berg"

gramstring = r"""
% start S
    S[HEAD=?v]    -> NP[] VP[HEAD=?v]
    VP[HEAD=?v]   -> V[HEAD=?v] NP[]
    NP[HEAD=?n]   -> Det[] N[HEAD=?n]
    NP[HEAD=?n]   -> N[HEAD=?n]

    Det[] -> "den"
    N[HEAD="er"]   -> "er"
    N[HEAD="Berg"] -> "Berg"
    V[HEAD="erklimmt"] -> "erklimmt"
"""

print(gramstring)
```



History-based PCFGs

Eine herkömmliche PCFG berücksichtigt nicht die Position einer Regelanwendung im Parsebaum, d.h. der **Kontext** wird in die Berechnung einer Regelwahrscheinlichkeit nicht mit einbezogen.

Oft sind die Regelwahrscheinlichkeiten jedoch abhängig von den zuvor angewandten Regeln.

Bsp.:

Die **Wahrscheinlichkeit der Regel NP -> Pron**

ist **höher bei Subjekt-NPs** (wenn die zuvor angewandte Regel S → NP VP war) **als bei Objekt-NPs** (wenn die zuvor angewandte Regel VP -> V NP war).

History-based PCFGs

erwünschte Regelgewichtung Subjekt (S-dominiert):

NP → PRON 0.91

NP → DET N 0.09

erwünschte Regelgewichtung Objekt (VP-dominiert):

NP → PRON 0.34

NP → DET N 0.66

normale PCFG (keine Differenzierung, Daten aus Korpus):

NP → PRON 0.25

NP → DET N 0.28

Lösung: Splitting NP-Kategoriensymbol (*parent annotation*):

NP^S → PRON 0.91

NP^S → DET N 0.09

NP^{VP} → PRON 0.34

NP^{VP} → DET N 0.66

Lösung: Parent Annotation

- nicht-terminale Knoten werden mit der Kategorie des Elternknotens (= history) annotiert
- als Trennzeichen verwenden wir das Zeichen ^ (z.B. NP^S)
- Nicht-Terminale werden dadurch in mehrere Kategorien aufgespalten

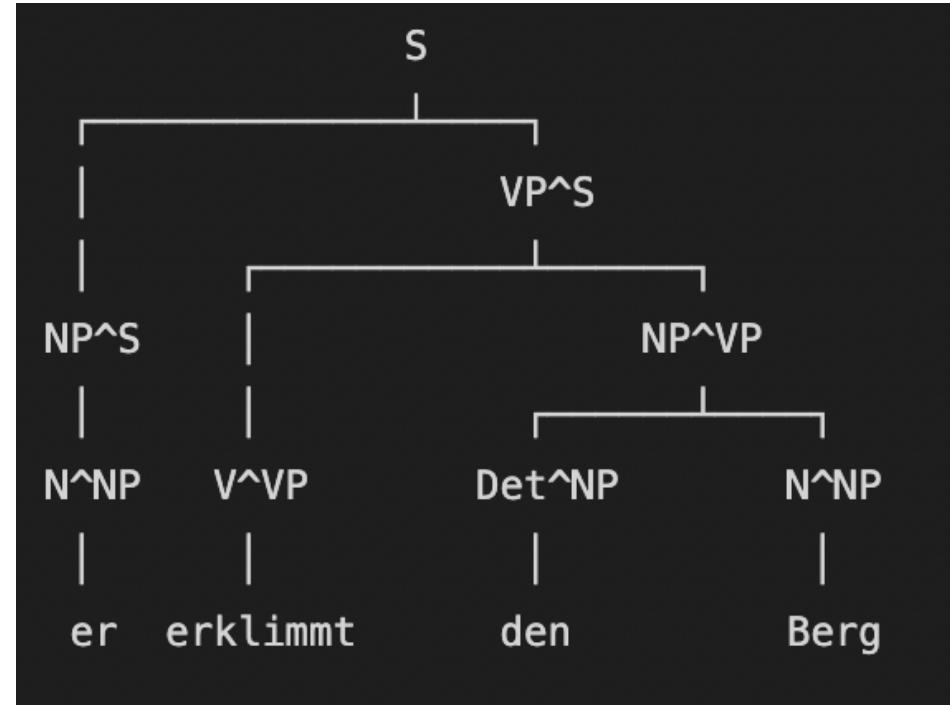
Probleme:

- Regelvervielfachung
- Probleme bei unbekannter Vorgängerkategorie

History-based PCFGs

Beispiel einer Parent Annotation

```
sentence = "er erklimmt den Berg"  
  
grammar = nltk.CFG.fromstring("""  
    S      -> NP^S VP^S  
    VP^S   -> V^VP NP^VP  
    NP^VP  -> Det^NP N^NP  
    NP^S   -> N^NP  
  
    Det^NP -> "den"  
    N^NP   -> "er"  
    N^NP   -> "Berg"  
    V^VP   -> "erklimmt"  
""")
```

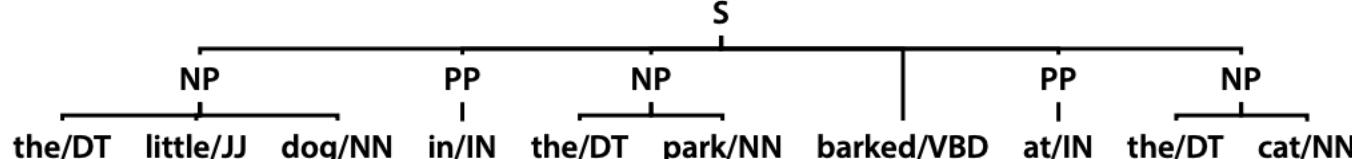


12. Partielles Parsen

Flache Struktur, IOB Marking, Evaluation

Partielles Parsen

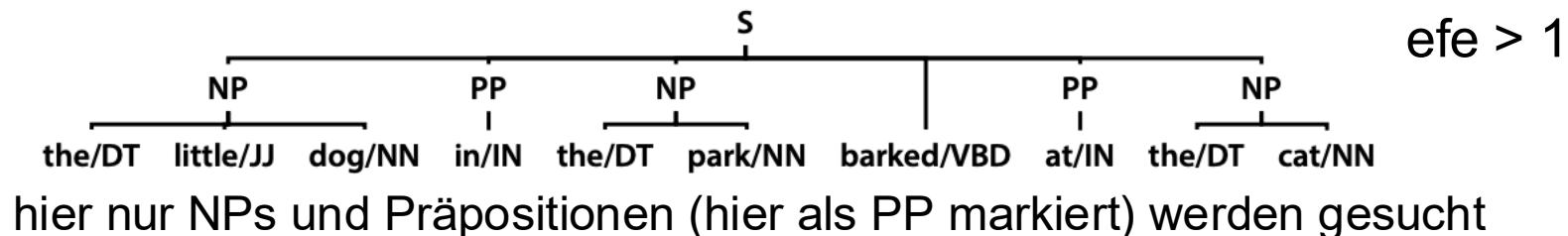
- Für viele Aufgaben (z.B. Information Extraktion und Information Retrieval) analysiert man nicht den ganzen Satz.
- Statt einen Satz komplett zu analysieren, wird dabei nur nach bestimmten Konstituenten gesucht --- NPs, VPs, PPs, (ADJPs)
- Unvollständige Analyse:
 - nur die Wörter berücksichtigt, die Element der relevanten syntaktischen Einheiten sind
 - andere Wörter werden nicht syntaktisch annotiert
 - verschachtelte Strukturen werden nicht berücksichtigt



hier nur NPs und Präpositionen (hier als PP markiert) werden gesucht

Chunking

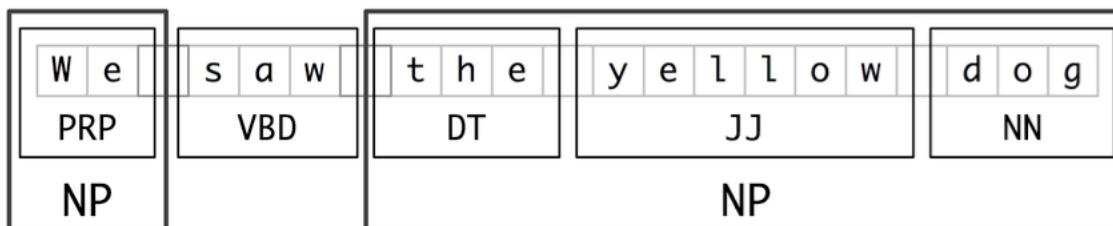
- Beim Chunking erhalten wir **flache, nicht-hierarchische** Analysen
 - Chunk-Bäume haben die Tiefe 1 (keine verschachtelten Strukturen)
- Die gewünschten Konstituenten werden als Chunks zurückgeliefert
- Beispiel:
 - Nur flache Konstituenten werden erkannt:



Chunk Parser

Was macht er?

- Er parst den ganzen Satz, findet die Grenzen von Konstituenten und liefert die erwünschten Konstituenten zurück mit ihren Labels (wie NP, PP).
- Also 2 Aufgaben:
 - Segmentierung + Tagging
- Wie segmentiert der Parser den Satz?
 - Durch IOB Labels
 - Detail in nächster Folie



IOB Format

- I: inside O: Outside B: Beginning

W e	s a w	t h e	y e l l o w	d o g
PRP	VBD	DT	JJ	NN
B-NP	O	B-NP	I-NP	I-NP

O: nicht ein Teil einer gewünschten Konstituente

B/I : Teile einer gewünschten Konstituente

Evaluation

- True Positive: positive item correctly classified as positive
- False Positive: negative item incorrectly classified as positive
- True Negative: negative item correctly classified as negative
- False Negative: positive item incorrectly classified as negative

Metrik	Formel	Worüber?
Accuracy	$(TP + TN) / \text{Gesamt}$	Gesamtgenauigkeit aller Vorhersagen
Precision	$TP / (TP + FP)$	Für alle Positive Ergebnisse, wie viel sind korrekt?
Recall	$TP / (TP + FN)$	Für alle echte Positive, wie viele wurde erkannt?
F1-Score	$2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$	Balance zwischen Precision und Recall