Syntax natürlicher Sprachen

5: Dependenzgrammatik und Dependency Parsing

A. Wisiorek

Centrum für Informations- und Sprachverarbeitung,
Ludwig-Maximilians-Universität München

21.11.2023

1. Dependenzstruktur

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

1.1. Eigenschaften der Dependenzstruktur

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

notwendige vs. nicht-notwendige Konstituenten

- **Ergebnis Konstituentenanalyse** (Eliminierungstest):
  → *bestimmte Wörter* ***nur mit anderen eliminierbar:***

unilaterale Abhängigkeit:

eine sehr schwierige Aufgabe

**eine sehr ~~schwierige~~ Aufgabe* (* = ungrammatisch)

eine ~~sehr~~ schwierige Aufgabe

bilaterale Abhängigkeit:

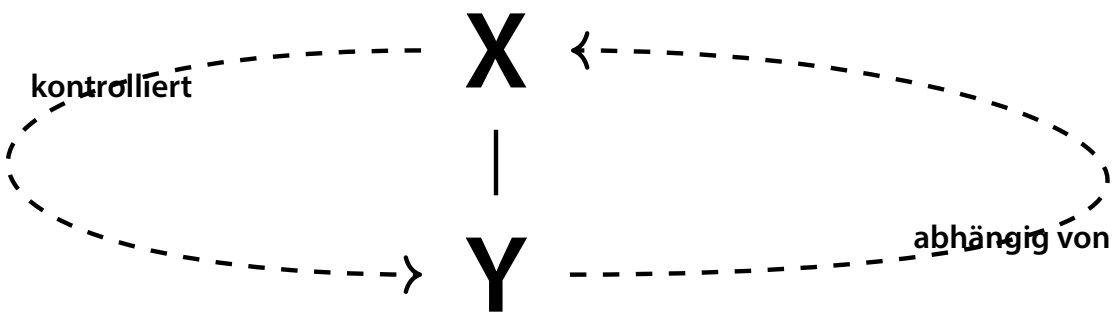
Beantworte den Brief

**~~Beantworte~~ den Brief*

**Beantworte ~~den Brief~~*

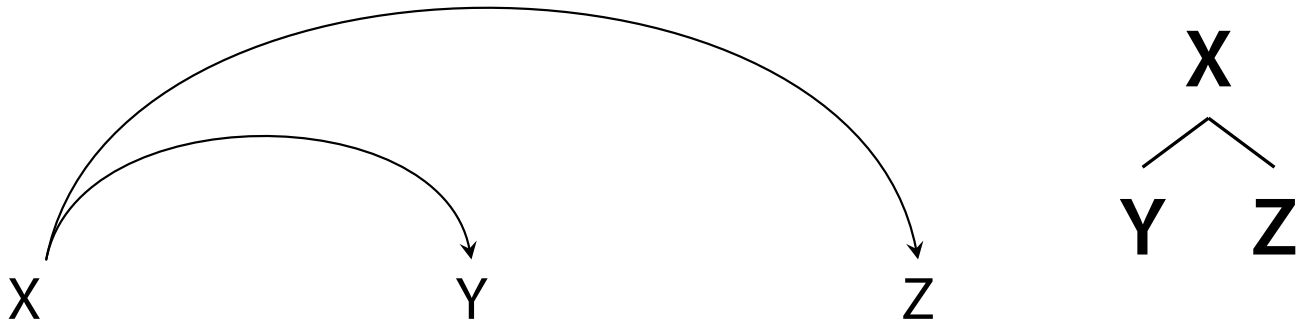
Dependenzrelation

- **Dependenzrelation** $< Y, X >$ ist eine **binäre asymmetrische Relation** zwischen einem **Kopf** X und seinem **Dependenten** Y
  - **binär:** zweistellige Relation zwischen zwei Wörtern X und Y, *wobei (das Vorkommen oder die Form von) Y von (dem Vorkommen oder der Form von) X abhängt*
  - **asymmetrisch:** *wenn Y abhängig von X ist, dann ist X nicht abhängig von Y*



Kontrollrelation

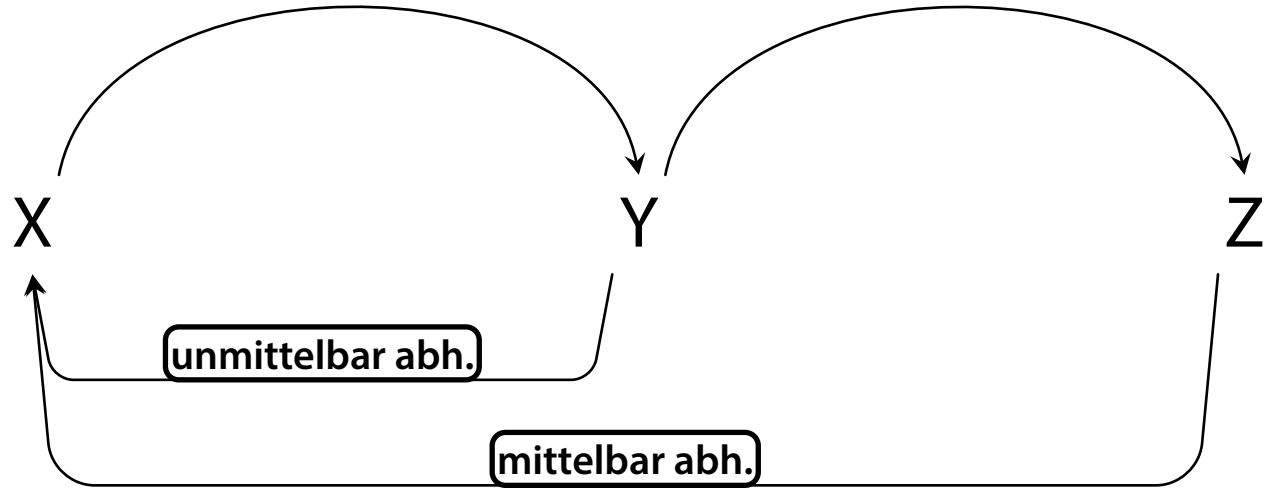
- **Kontrolle als umgekehrte Dependenzrelation:**
  $< X, Y >$: X regiert Y (X ist Kopf/Regens von Y)
- **Darstellung Kontrollrelation** mit Pfeilen (gerichteter Graph) oder implizit durch vertikale Anordnung



- **WICHTIG:** ein Wort kann nur von ***einem*** anderen Wort abhängen: **nur 1 Kopf pro Dependent!** (aber mehrere Dependenten pro Kopf möglich)

unmittelbare vs mittelbare Dependenz

- Relation der **unmittelbaren und der mittelbaren Abhängigkeit**:



- **zum Vergleich:**

Abbildung: Konstituenten- und Dependenzstrukturschema

Dependenzstruktur

- Dependenzstruktur: **Menge der durch die Relation der Dependenz/Kontrolle verbundenen lexikalischen Einheiten** (Wörter; ggf. auch Stämme und Affixe)
- direkte Untersuchung der **hierarchischen Beziehungen der Einheiten im Satz** (wie ihr Vorkommen und ihre Form voneinander abhängen)
- **Verb als Wurzelknoten** des Satzes, von dem alle anderen Knoten unmittelbar oder mittelbar abhängen
- in einer Phrase: **Kopf kontrolliert Dependenten**; Dependenten hängen von Kopf ab
- ein Wort kann nur von ***einem*** anderen Wort abhängen: **nur 1 Kopf pro Dependent!** (aber mehrere Dependenten pro Kopf möglich)

Beispiel Dependenzstruktur

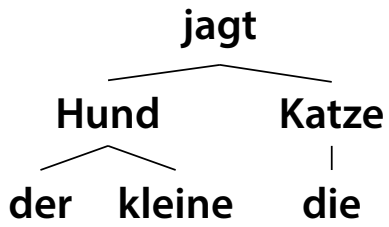


Abbildung: Einfacher Dependenzbaum (auch: Stemma)

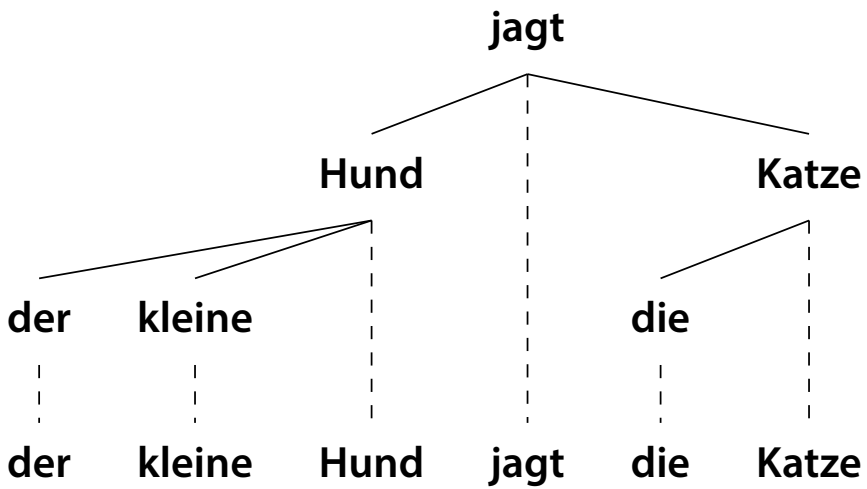


Abbildung: Dependenzbaum mit Berücksichtigung der linearen Ordnung

1.2. Vergleich Konstituenten- und Dependenzstruktur

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

Vergleich Konstituenten- und Dependenzstruktur

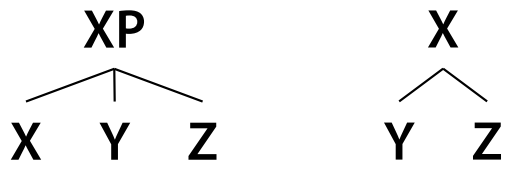


Abbildung: Konstituenten- und Dependenzstrukturschema

Übersicht Dependenzstruktur

- **Elemente der Struktur (Knoten)** $\rightarrow$ *Wörter*
- **Relationen der Struktur (Kanten)** $\rightarrow$ *Dependenzrelationen (z. B. Subjekt, Objekt)*
- **syntaktische Kategorien** $\rightarrow$ *gerichtete Kanten = Dependenzrelationen*
- **Kategorientyp** $\rightarrow$ *funktional / relational*
- **Strukturinformationen in Kanten des Syntaxbaums (funktionale Kategorien)**

Konstituentenstruktur = Phrasenstrukturgrammatik (PSG)

- Analyse des **Aufbaus der Satzstruktur durch Zergliederung** in Konstituenten
- Zusammensetzung von Wörtern zu **syntaktischen Einheiten**
- **Subjekt-Prädikat**-Grundstruktur
- **Strukturinformation in Knoten** (Kategorien des strukturellen Aufbaus)
- **phrasale Knoten**

Dependenzstruktur = relationale Wortgrammatik

- Analyse **Satzstruktur 'von innen heraus'** (vom Verb ausgehend)
- **Beziehung zwischen Wörtern**
- **Subjekt und Objekt gleichrangige Argumente des Verbs** (beide valenzgefordert)
- **Strukturinformation in Kanten** (relationale Kategorien)
  $\rightarrow$ ***Label einer Kante = syntaktische Funktion des Dependenten!***
- keine phrasalen Knoten, **flachere Struktur** als PSG

Beispiel Dependenz- vs Konstituentenstruktur

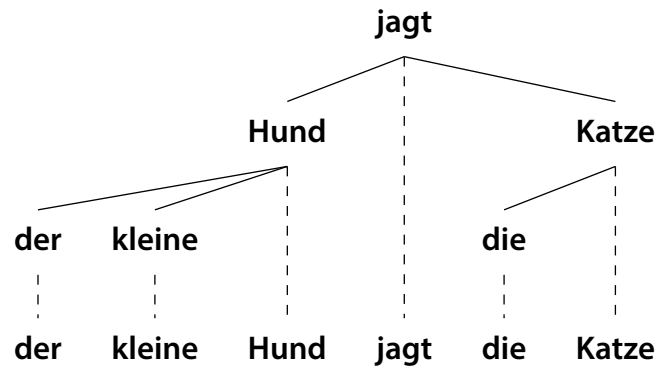


Abbildung: Dependenzbaum mit Wortartenangaben

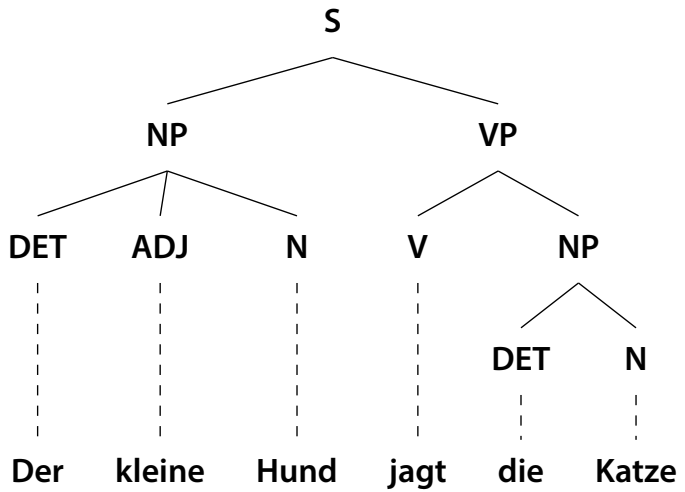
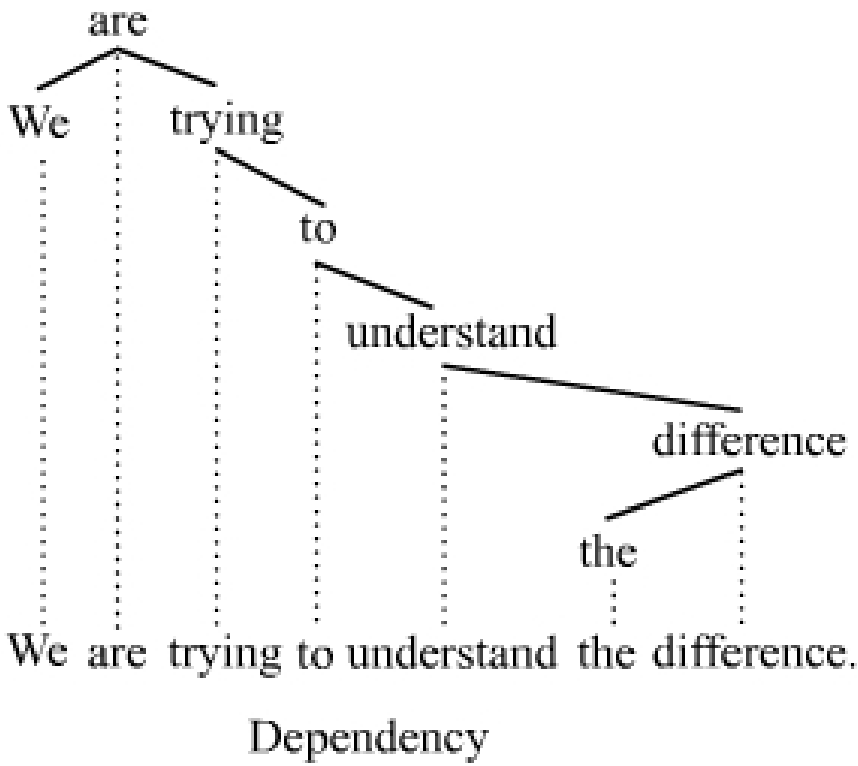
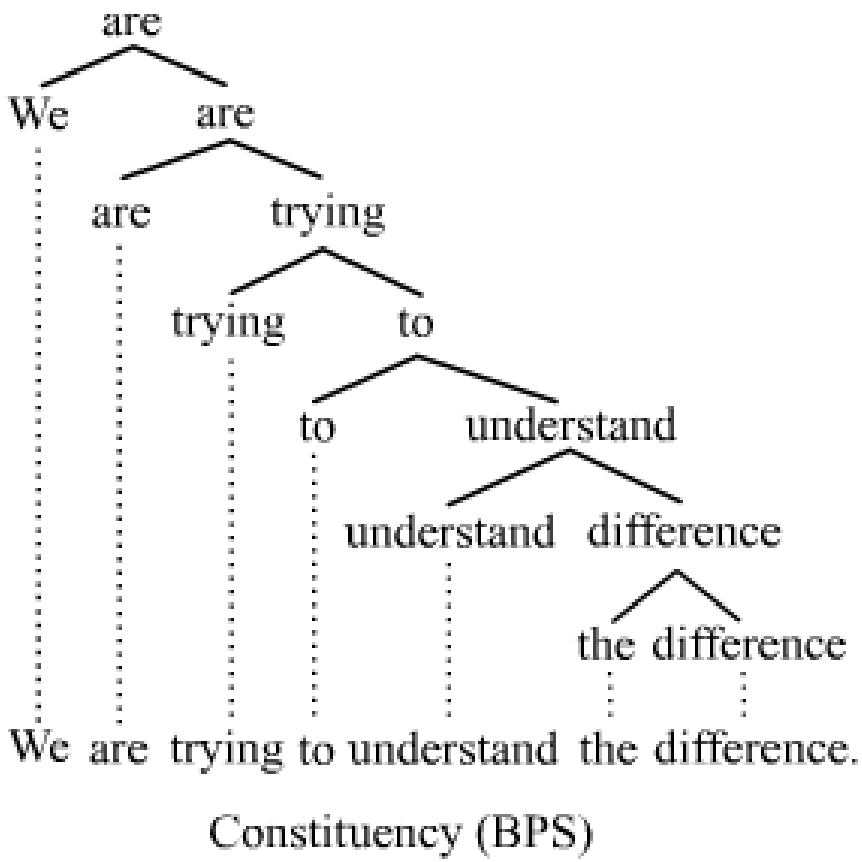


Abbildung: zum Vergleich: Konstituentenstruktur

Beispiel Dependenz- vs Konstituentenstruktur



Dependency



Constituency (BPS)

Abbildung: Geordneter Dependenzbaum - Konstituentenbaum (von Tjo3ya - eigenes Werk, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17517283)

Beziehung Dependenz / Konstituenz

- **Dependenz in Konstituentenstruktur**
  - implizite Dependenzanalyse **in Phrasenkategorien durch Kopf-Prinzip** (X-Phrase)
    $\rightarrow$ *Phrasenkopf ist Regens aller anderen Schwesterknoten*
  - **in X-Bar-Theorie**: Ergänzung und Angabe als **Komplement und Adjunkt** über **Strukturposition** definiert
- **Konstituenten in Dependenzstruktur**
  - implizite Konstituentenanalyse: **Teilbäume als Konstituenten** (aber nicht alle Konstituenten repräsentiert: VP)

Transformation Konstituenten- in Dependenzstruktur

Transformationsregeln

1. ***head-finding-rules*** *(Kopfannotation)*
   - Perkolationsregeln für Hochreichen des Kopfes:
     $\rightarrow$ *head(NP) = head(N)*
     $\rightarrow$ *head(VP) = head(V)*
     $\rightarrow$ *head(PP) = head(NP), gemäß UD-Schema*
     (alternativ: head(PP) = head(P))
     $\rightarrow$ *head(S) = head(VP)*
2. **Regel für die Bestimmung der Dependenzrelationen**:
   - die Köpfe der Ko-Konstituenten einer Einheit sind die Dependenten ihres Kopfes
     $\rightarrow$ *außer der Kopf selbst: asymmetrische Relation ist irreflexiv*
3. **Regeln für das Labeln** der Relationen:
   - Label der Dependenzrelation: syntaktische Funktionen
     $\rightarrow$ *im einfachsten Fall auch Wortart des Dependenten*

von Phrasenstrukturbaum zu Dependenzbaum

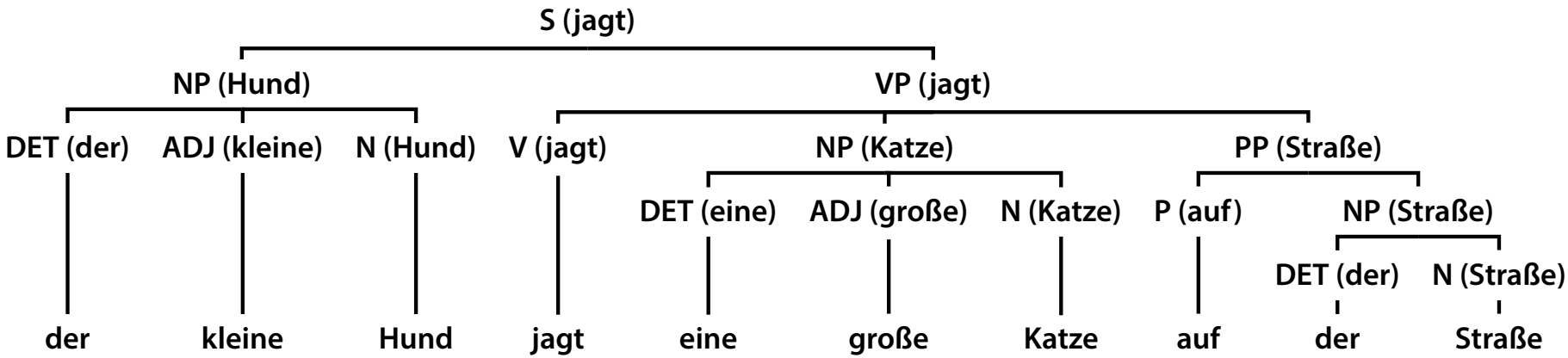


Abbildung: Phrasenstruktur mit Kopfannotation

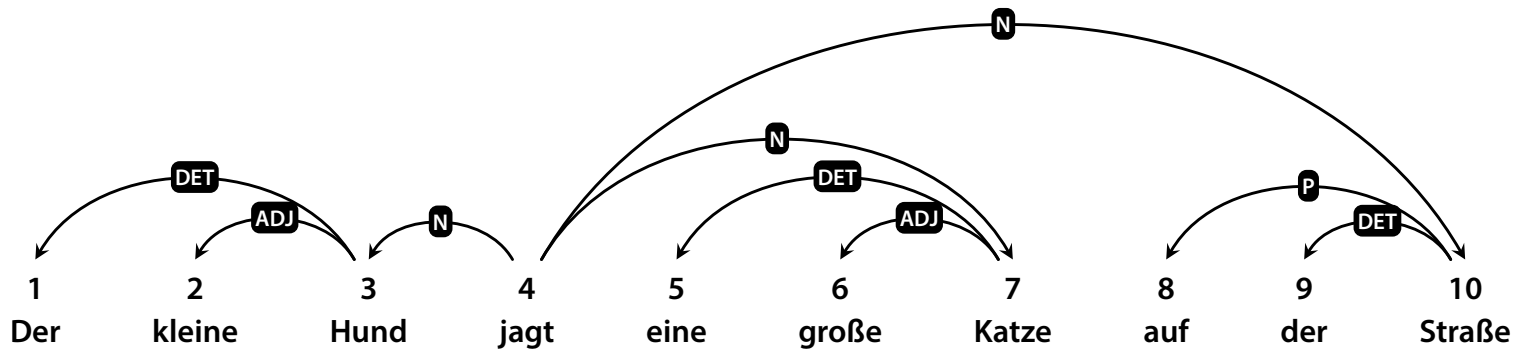


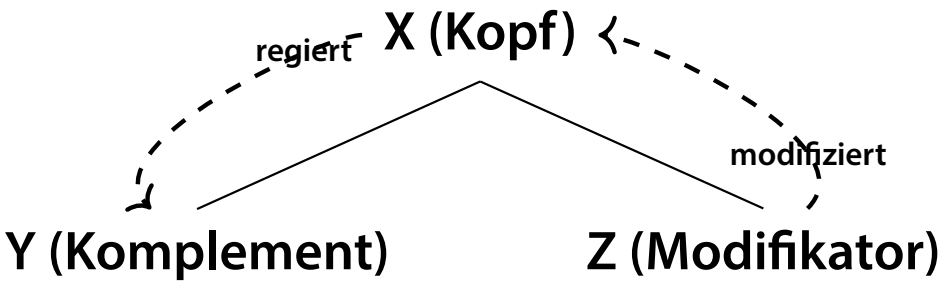
Abbildung: daraus abgeleitete Dependenzstruktur

1.3. Typen von Dependenzrelationen

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

Typen von Dependenzrelationen

- **Typ 1: Rektion**
  - **bilaterale Dependenz**: Kopf kann nicht ohne Dependent auftreten; (Kasus-)Form des Dependenten von Kopf bestimmt
    → *Dependent ist **Komplement***
- **Typ 2: Modifikation**
  - **unilaterale Dependenz**: Kopf kann ohne Dependent auftreten; (Kasus-)Form des Dependenten unabhängig von Kopf
    → *Dependent ist **Adjunkt oder Attribut***

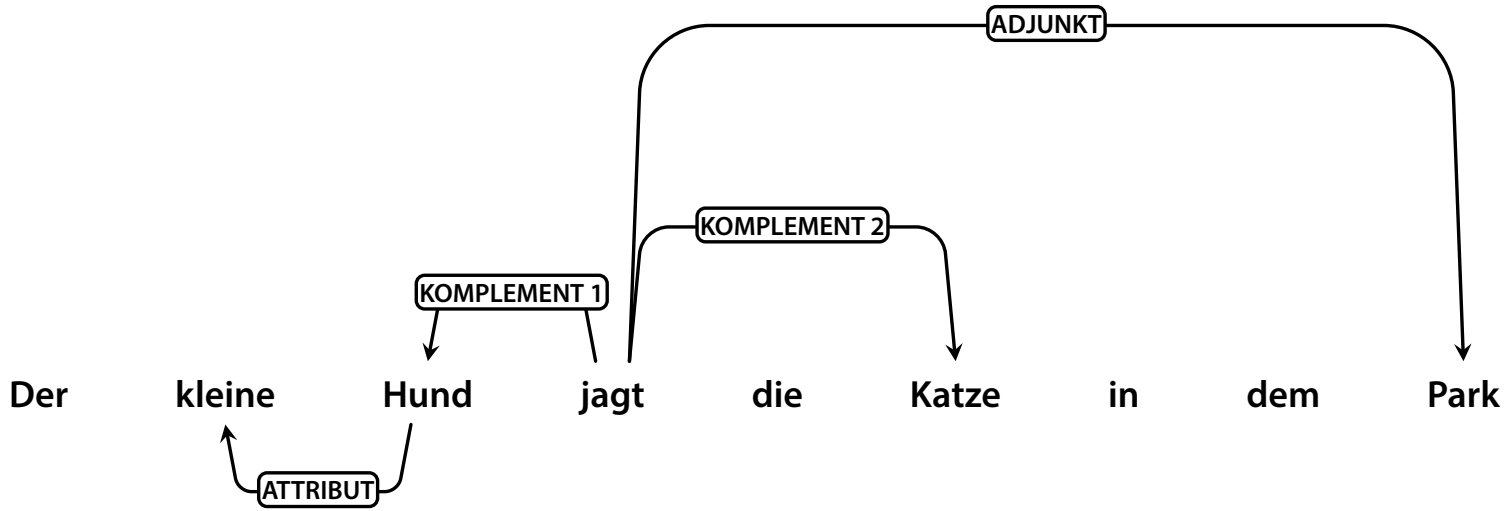


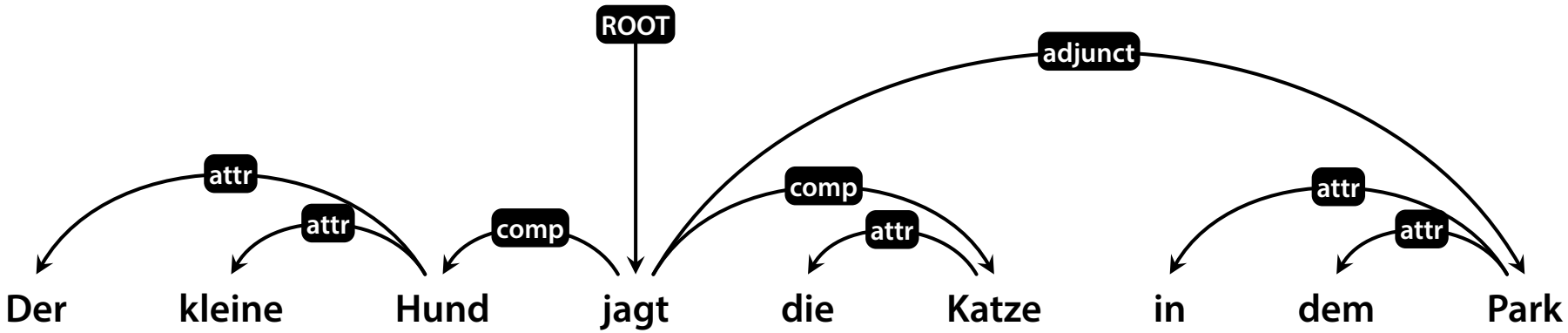
Übersicht Rektion vs Modikfikation

REKTION		MODIFIKATION
Komplement (= Ergänzung)	**verbal**	Adjunkt (= Angabe)
(nominales Komplement)	**nominal**	Attribut

- eingeschränkter Adjunkt-Begriff gegenüber X-Bar-Theorie! (nominale Modifkatoren als Attribute)
- Anmerkung: bestimmte nominale Attribute können auch als Komplemente aufgefasst werden, z.B. Genitivattribut: *der Beweis des Theorems*

Beispiel Komplement vs Adjunkt vs Attribut





Komplement (auch: Ergänzung / Aktant / Argument)

- **Vorkommen** des Dependents **vom Kopf gefordert**
  $\rightarrow$ *meist auch die **Form** gefordert: **quantitative vs qualitative Valenz**)*
- ***valenzgebundener Dependent*** (***obligatorisch***)
- *Leerstelle* (Bühler) beim Kopf (insbes. beim Verb), die mit bestimmter Konstituente in bestimmter Form zu füllen ist
- Anzahl der Leerstellen = **Valenz, Subkategorisierungsrahmen, Argumentstruktur**
- *weiter Komplementbegriff: enthält auch Subjekt*

Modifikator (Adjunkt / Angabe oder Attribut)

- **Vorkommen und Form** des Dependents **NICHT vom Kopf gefordert**
- ***nicht-valenzgebundener Dependent*** (***optional***)
- Leerstellen beim Dependent, mit der er sich an einen Kopf bestimmten Typs andocken kann (Ergebnis ist ein Syntagma gleichen Typs wie der Kopf)

- **Adjunkt als verbaler Modifikator** (auch: **Angabe** / Zirkumstant)
- **Attribut als nominaler Modifikator**
- *eingeschränkter Adjunkt-Begriff gegenüber X-Bar-Theorie!*

2. Dependenzgrammatik

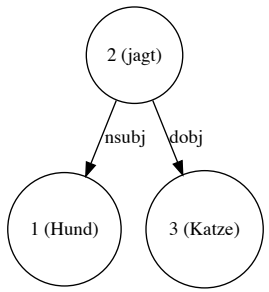
1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

2.1. Formale Eigenschaften

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

Formale Eigenschaften der Dependenzstruktur

- **gerichteter Graph** als Repräsentationsformalismus: $G =< M, R >$
  - M: Elementmenge = Knoten (hier: Wörter)
  - R: Relation zwischen Elementen von M
    $\rightarrow$ *Menge geordneter Paare = gerichtete Kanten (directed edges /arcs)*
    $\rightarrow$ *hier: Abhängigkeitsrelation*
- Dependenzstruktur hat **genau einen Wurzelknoten** (**ROOT**)
- ein Wort kann mehrere Dependenten haben
- ein Wort kann nur von von einem Wort abhängen (und nicht von sich selbst)
- Kanten können **markiert (gelabelt)** oder unmarkiert sein



2.2. Dependenzregeln

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - **Dependenzregeln**
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

Dependenzregeln

- Modellierung Dependenzstruktur mit formaler Grammatik möglich
- **Dependenzregeln** geben die Beziehung zwischen Kopf und Dependent an:
  $\rightarrow$ ***Wortgrammatik:***
  - LHS: Kopf
  - RHS: Dependent
- Beispielregeln: `jagt -> Hund`, `jagt -> Katze`

Dependenzgrammatik in NLTK

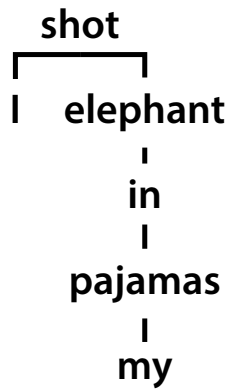
```
1  sent= 'der Mann schenkt der Frau das Buch'
2
3  grammar = nltk.DependencyGrammar.fromstring("""
4  'gibt' -> 'Mann' | 'Frau' | 'Buch'
5  'schenkt' -> 'Mann' | 'Frau' | 'Buch'
6  'Mann' -> 'der'
7  'Frau' -> 'der' | 'die'
8  'Buch' -> 'das'
9  """)
10
11 parser = nltk.ProjectiveDependencyParser(grammar)
12 for tree in parser.parse(sent.split()):
13     print(tree)
14
15 #(schenkt (Mann der) (Frau der) (Buch das))
```

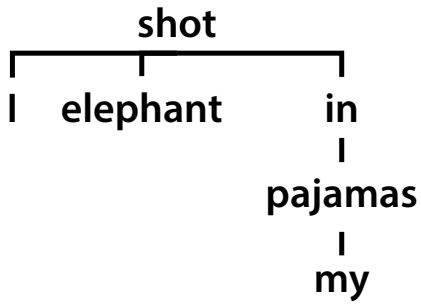
2.3. Ambiguität und PP-Modellierung

Ambiguität in Dependenzgrammatik

- auch in der Dependenzstruktur kann syntaktische Ambiguität ausgedrückt werden
- im Beispiel (vgl. NLTK book) ermöglichen die folgenden 2 Dependenzregeln die PP-Attachment-Ambiguität:

```
shot -> in
elephant -> in
```

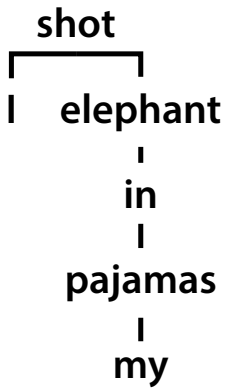


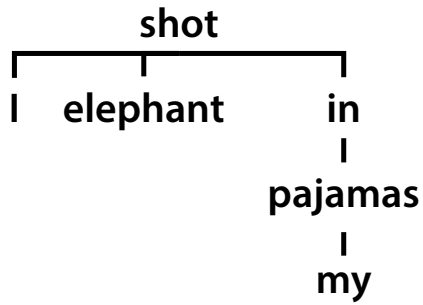


→ *HINWEIS: diese Analyse der PP-Relationen mit P als Kopf der PP ist konträr zu der in der Vorlesung bevorzugten Analyse mit dem Kopf der NP als Kopf der PP!*

PP-Attachment-Ambiguität in Dependenzgrammatik

```
1 sent= 'I shot an elephant in my pajamas'
2
3 grammar = nltk.DependencyGrammar.fromstring("""
4 'shot' -> 'I' | 'elephant' | 'in'
5 'elephant' -> 'an' | 'in'
6 'in' -> 'pajamas'
7 'pajamas' -> 'my'
8 """)
```





2.4. Projektivität

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

Projektivität und Nichtprojektivität

- **projektive Struktur:** nur **projektive** Kanten, d. h. es gibt einen Pfad vom Kopf der Relation **zu jedem Wort** zwischen Kopf und Dependent
- **nicht-projektive Struktur = Überschneidung von Kanten**
  $\rightarrow$ *c) und e) in folgendem Beispiel enthalten eine nicht-projektive Kante:*

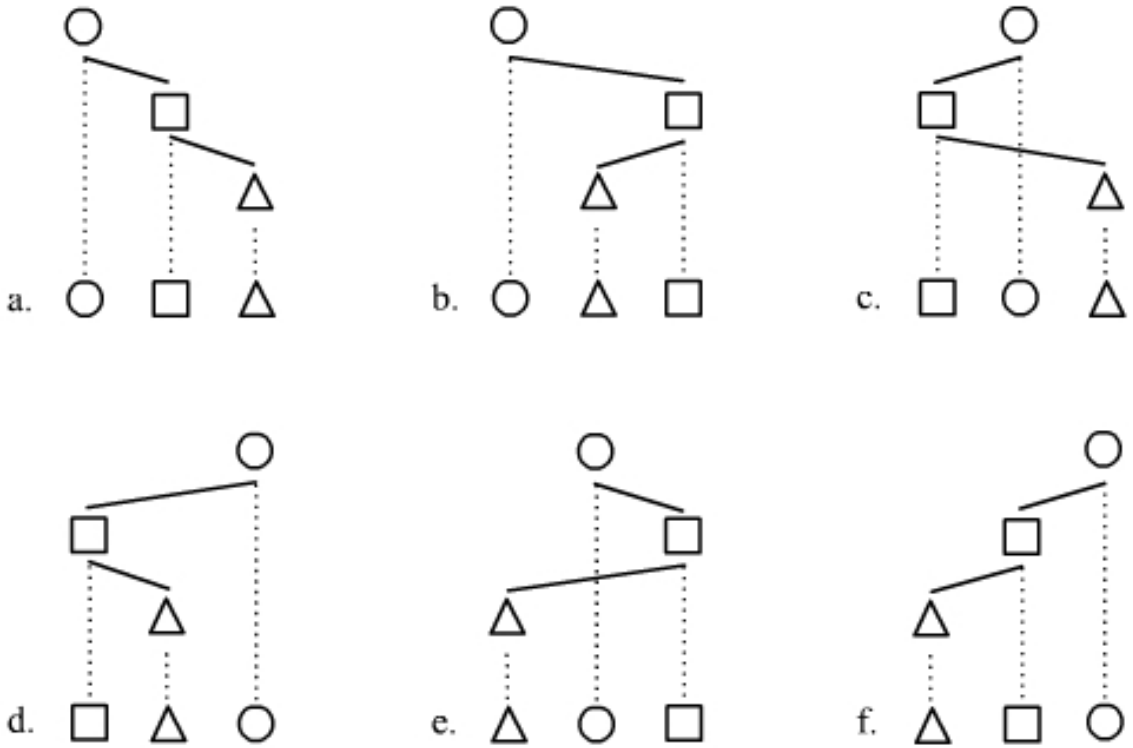
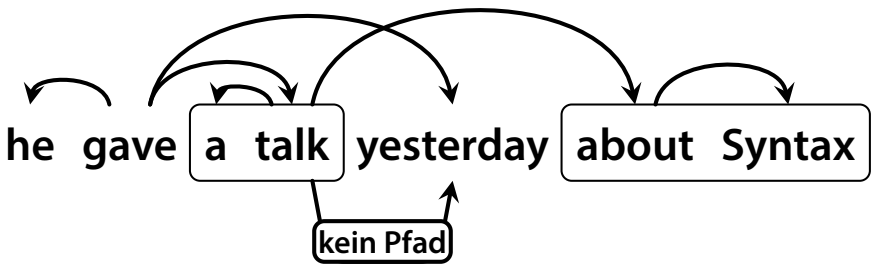


Abbildung:
`https://en.wikipedia.org/wiki/Discontinuity_(linguistics)`

Beispiel: nicht-projektive Struktur



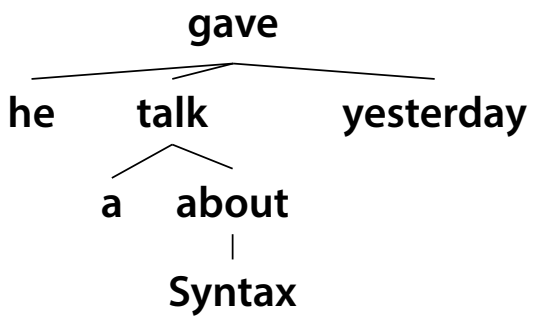


Abbildung: Dependenzanalyse diskontinuierlicher = nicht-projektiver Struktur (mit und ohne Berücksichtigung linearer Ordnung)

- **linguistisch: nicht-projektive Strukturen** entstehen durch **diskontinuierliche Elemente**
  $\rightarrow$ ***freie Wortstellung und long distance dependencies***

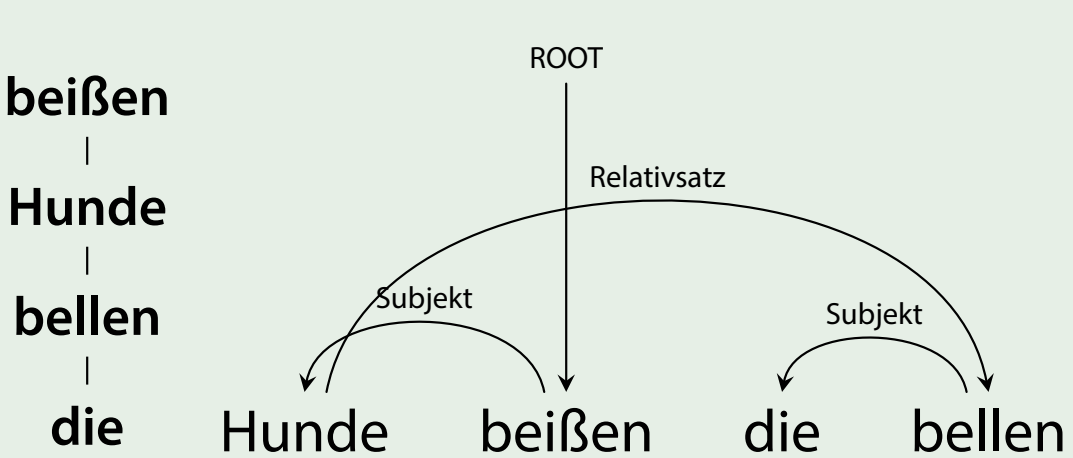
Nicht-projektive Strukturen: Dependenzgrammatik

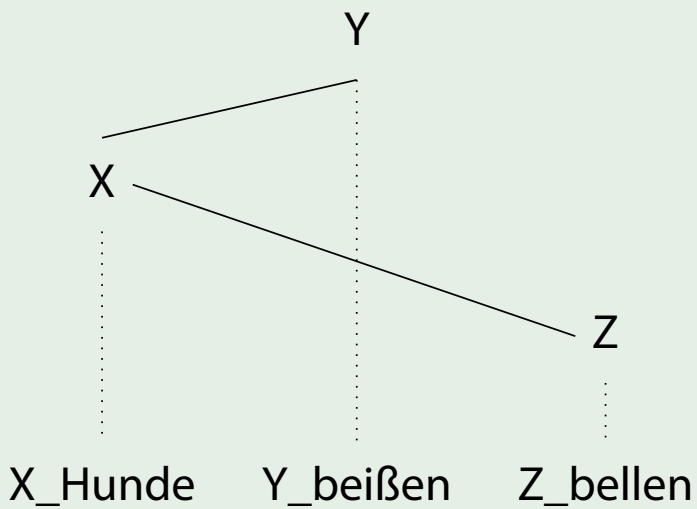
- Dependenzgrammatik: unabhängig von linearer Anordnung (Abhängigkeitsrelationen)

- aber: bestimmte Dependency Parsingalgorithmen können nicht-projektive Strukturen nicht verarbeiten

Beispiel Relativsatz

- Dependent eines Wortes folgt nach dessen Kopf, vgl c) oben

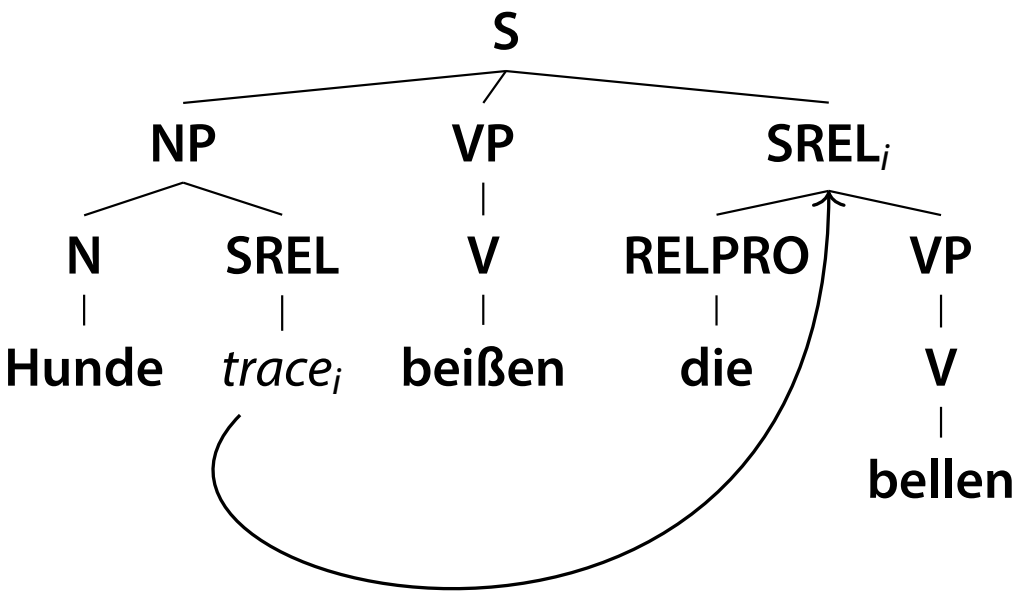




Nicht-projektive Strukturen: CFG

- diskontinuierliche Strukturen entstehen durch Herausbewegung von Phrasen/Konstituenten

- mit CFGs: nicht direkt modellierbar (überkreuzende Kanten)
- nur mit *traces* (Leerstelle + Movement/Transformationsregeln)



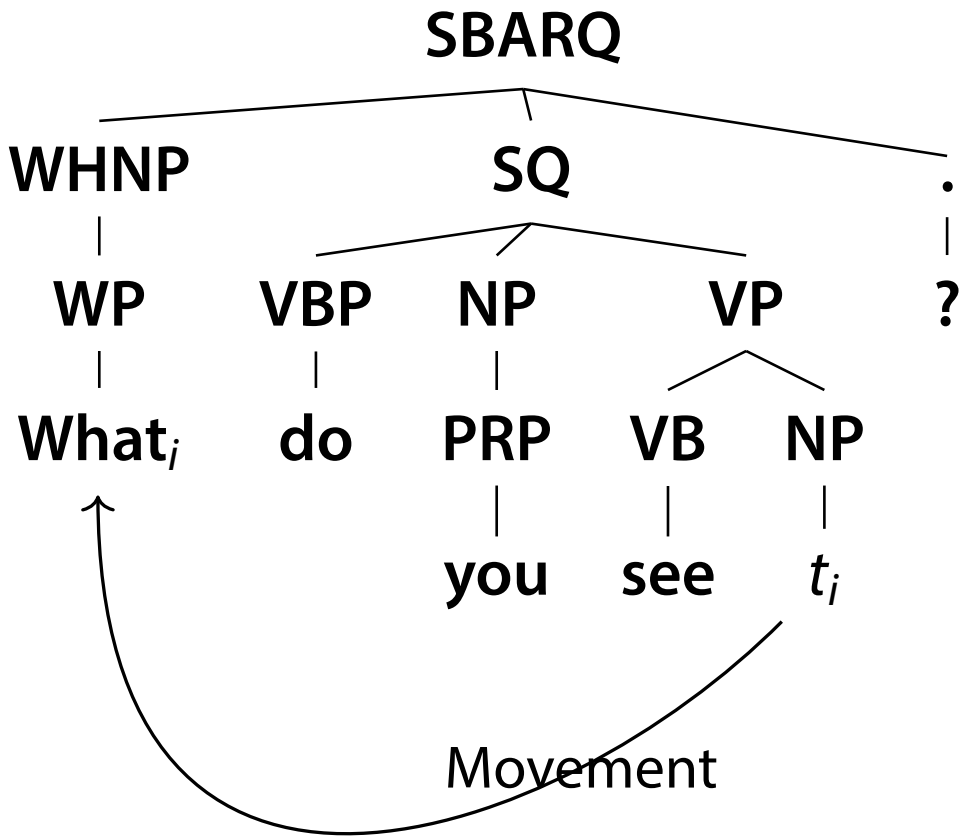
Vorteil Dependenzgrammatik

- Dependenzgrammatiken sind **besser** als Konstituentengrammatiken **geeignet**, **diskontinuierliche Strukturen abzubilden**
  → *Modellierung* ***relationaler Struktur****, nicht der linearen Anordnung*
  → *Dependenzstruktur* ***abstrahiert von der linearen Anordnung***
  → ***bei Verarbeitung*** *(Parsing) können* ***nicht-projektive Strukturen aber problematisch*** *sein*
- bei **Ableitung Dependenzgrammatik** von PSG-Treebanks durch *head-finding-rules* ergeben sich **automatisch projektive** Strukturen

Exkurs: Diskontinuierliche Konstituenten

- auch: ***long distance dependencies***
- Heraustrennung von Teilkonstituenten einer Konstituente
- Problem für Baumdarstellung:
  $\rightarrow$ ***Überkreuzung*** = *nicht-projektiv*
- Lösung: **leere Knoten** (*empty nodes*: `0`, ϵ, `t`, `NONE`)
  $\rightarrow$ *trace (**Spur**): Konzept der Transformationsgrammatik*
- **Transformationsgrammatik**:
  $\rightarrow$ *Annahme: **Tiefen- und Oberflächenstruktur***
  $\rightarrow$ *abstrakte vs. beobachtbare Form von Sätzen*
  $\rightarrow$ *z. B.: Annahme deutsche Tiefenstruktur der VP: OV (den Hund sehen)*
  $\rightarrow$ ***Transformationsregelanwendung** zur Erzeugung der Oberflächenstruktur: **läßt Spur zurück***
- im Englischen relativ begrenzt: z.B. Topikalisierung, Extraposition, **Wh-fronting**

Analyse *long distance dependencies* mit Spur (t)



3. Dependency Parsing

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

Konstituenten- vs. Dependenzgrammatik in CL

- in Computerlinguistik waren historisch **Konstituenten-basierte Formalismen dominant** (Chomsky-Tradition Generativer Grammatik)
  - siehe `Stanford PCFG Parser`
- aber: **Dependenzbasierte Syntaxmodelle** werden immer wichtiger
  - siehe u.a. `spaCy`
  - Syntaxmodelle von binären Abhängigkeitsrelationen zwischen Wörtern statt Phrasenstruktur-Grammatikregeln (PSG)
  - Dependency-Parsing-Modelle können aus **Dependency-Treebanks** induziert werden
    $\rightarrow$ *Dependency-Treebanks können handannotiert sein oder abgeleitet aus PSG-Treebank*
  - Dependenzanalysen können auch **sekundär aus Analysen mit konstituentenbasierten Parsern** erzeugt werden
    $\rightarrow$ *z. B. ursprünglich beim Stanford-Parser*
    $\rightarrow$ *inzwischen auch natives Dependency-Modell in Stanford-NLP-Tools (stanza = python-Package)*

Vorteile von Dependenzmodellen

- **Relationale Informationen direkt** vorhanden statt indirekt über Position in Strukturbaum
  $\rightarrow$ *Verwendung z. B. für* ***Informationsextraktion und semantisches Parsing***
- **Wortgrammatik** = direkte Modellierung von Relation zwischen Wörtern
  $\rightarrow$ ***keine Lexikalisierung notwendig***
- **Dependenzgrammatik als Wortgrammatik**
  $\Rightarrow$ **reduziert *sparse data*-Problem bei Parameterabschätzung**

3.3. Dependency-Treebanks

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

Dependency-Treebanks

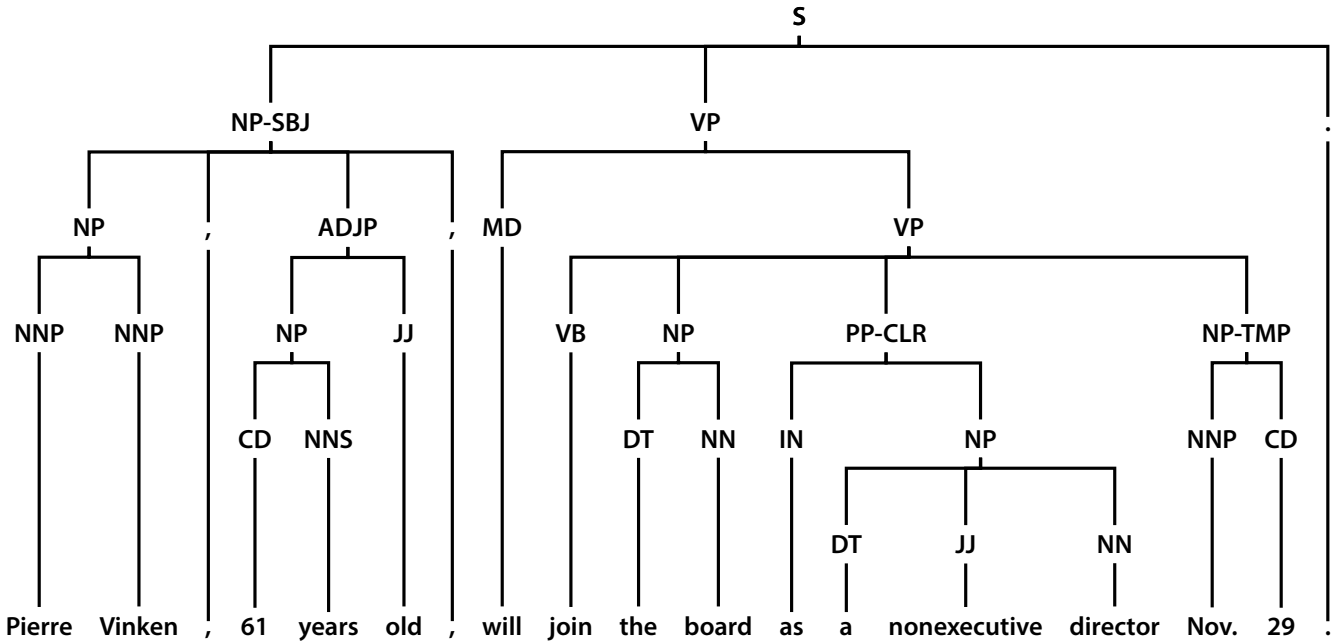
- von Experten erstellte **dependenzsyntaktisch annotierte Korpora**:
  - **relationsannotierte Tokenlisten** = Knoten + Relationen
  - verschiedene Formate: dot-Format, CoNLL-Format
- Einsatz zu **Training und Evaluation** von Dependenz-Parsing-Systemen

Gewinnung *Dependency-Treebanks* aus PSG-Treebanks:

- **Transformation** von **kopfannotierten Konstituenten-Bäumen** in einen Dependenzgraph (s. Sitzung 5):
  1. **Finden aller *head-dependent*-Relationen** über *head-finding-rules*
  2. **Labeln der Relationen** über handgeschrieben Regeln
     $\rightarrow$ *Bestimmung Relationstyp* ***über Strukturposition:***
     `NP mit Mutterknoten S ist subj`
     $\rightarrow$ *bei Penn-Treebank: Verwendung* ***funktionaler Informationen in den Nichtterminalen:*** `NP-SBJ`

Funktionale Kategorien in Penn Treebank:

- **Grammatische Relationen/funktionale Angaben** in den phrasalen Kategorien, z. B.: `NP-SBJ`
  - → *`PP-CLR`: 'closely related', z. B. für präpositionales Objekt*
  - → *`NP-PUT`: adverbiales Komplement von put*
  - → *`NP-ADV`: für Kasusadverbial*

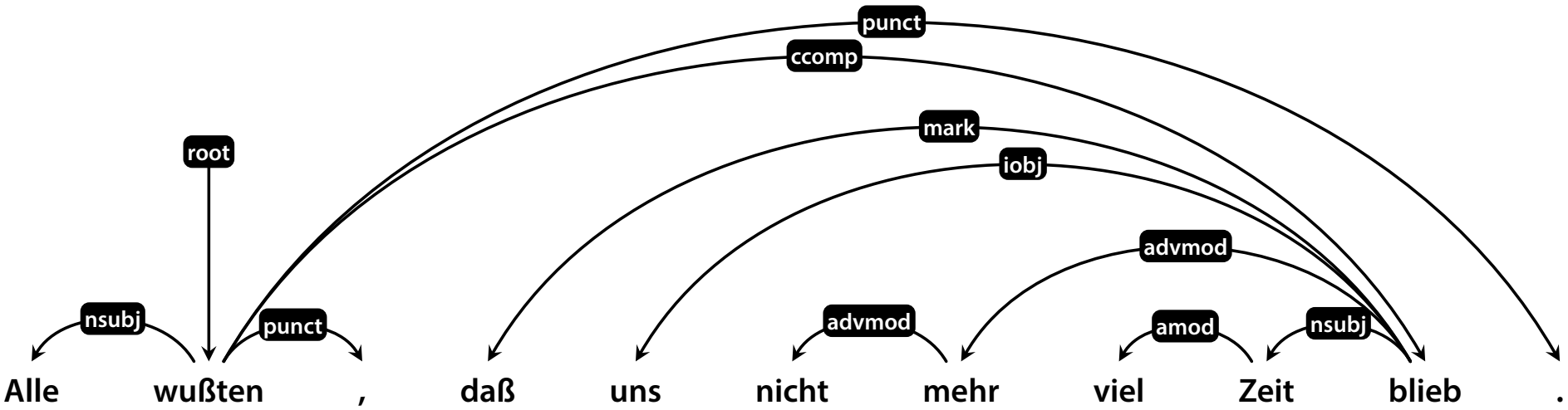


CoNLL-Dependency-Treebanks:

- CoNLL: Shared Tasks zu **Dependency Parsing**: mit annotierten Treebanks für Evaluation der Systeme
- **UD-Treebanks (>30 Sprachen)** im CoNLL-U-Format:
  $\rightarrow$ *http://universaldependencies.org/format.html*
- **TIGER Dependency Bank** (in Dependency-Format konvertiertes TIGER-Korpus, deutsch) verwendet in CoNLL und UD-Treebanks, konvertiert in Stanford bzw. Universal Dependencies

1	Alle	alle	PRON	PIS	Case=Nom..	2	nsubj	_	_
2	wußten	wissen	VERB	VVFIN	Number=Plur..	0	root	_	SpaceAfter=No
3	,	,	PUNCT	$,	_	2	punct	_	_
4	daß	daß	SCONJ	KOUS	_	10	mark	_	_
5	uns	wir	PRON	PPER	Case=Dat..	10	iobj	_	_
6	nicht	nicht	PART	PTKNEG	Polarity=Neg	7	advmod	_	_
7	mehr	mehr	ADV	ADV	_	10	advmod	_	_
8	viel	viel	ADJ	PIAT	Case=Nom..	9	amod	_	_
9	Zeit	Zeit	NOUN	NN	Case=Nom..	10	nsubj	_	_
10	blieb	bleiben	VERB	VVFIN	Number=Sing..	2	ccomp	_	SpaceAfter=No
11	.	.	PUNCT	$.	_	2	punct	_	_

Tabelle: Satz im CoNLL-Format (deutsches UD-Korpus)



3.4. Statistische Dependency-Parsing-Modelle

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

Typen von Dependency-Parsing-Modellen

1. **Übergangsbasiertes Dependenz-Parsing:**
   → ***Stack-basierter Shift-Reduce-Parser***
   → ***Auswahl des Übergangs*** *von einem Zustand (**Konfiguration** von Stack, Buffer und erkannten Relationen) zum nächsten **über Klassifikator***
   → ***Klassifikator: bildet Konfigurationen auf Übergänge ab***
   → ***trainiert anhand von Dependency-Treebank***
2. **Graphbasiertes Dependenz-Parsing:**
   → ***Auswahl von am besten bewerteten Baum*** *im Graph aller möglichen Relationen zwischen den Wörtern eines Satzes*
   → *Lernen der **Gewichte der Relationen** anhand von **Dependency-Treebank***
   → *Vorteil: **Parsing nicht-projektiver Strukturen** möglich (diskontinuierliche Strukturen)*
   → *Vorteil: **globale Bewertung der Dependenzstruktur von Sätzen** statt lokaler Entscheidungen*

Übergangsbasierte Parsing-Systeme

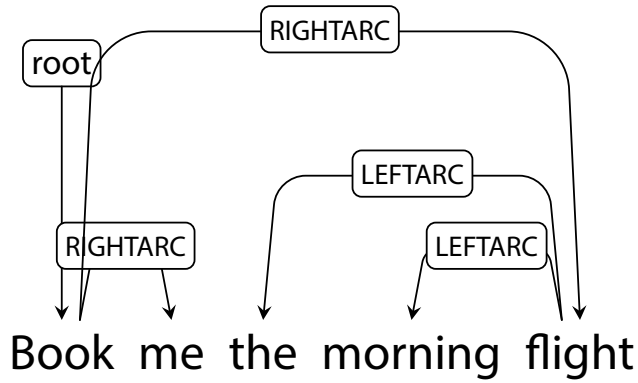
- **Malt-Parser** (Nivre et al.): *transition-based Dependency Parser*
- **Stanford-Dependency-Parser** (Manning et al.):
  - neben der **Transformation von** PCFG-geparsten **Konstituentenbäumen in Dependenzgraphen** (`englishPCFG.ser.gz`):
  - **Transition-based Dependency-Parsing-Modell:**
    $\rightarrow$ *`englishFactored.ser.gz`: verwendet PCFG-Parser und Dependenz-Parser und vergleicht Ergebnisse*
- **spaCy**: *transition-based* Dependency-Parsing; Modelle gelernt mit neuronalen Netzen (`https://spacy.io/models/#architecture`)

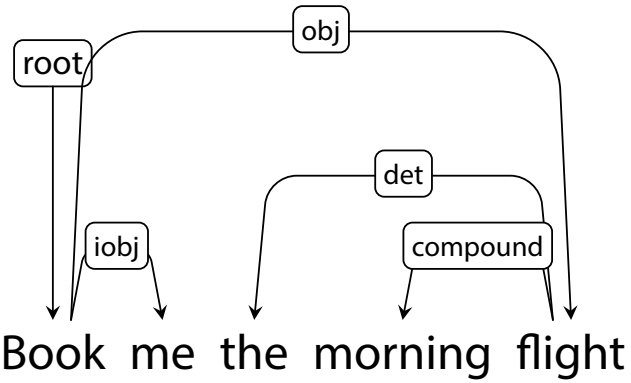
3.5. Übergangsbasiertes Shift-Reduce-Dependency-Parsing

1. Dependenzstruktur
   - Eigenschaften der Dependenzstruktur
   - Vergleich Konstituenten- und Dependenzstruktur
   - Typen von Dependenzrelationen
2. Dependenzgrammatik
   - Formale Eigenschaften
   - Dependenzregeln
   - Ambiguität und PP-Modellierung
   - Projektivität
3. Dependency Parsing
   - Dependency-Treebanks
   - Statistische Dependency-Parsing-Modelle
   - Übergangsbasiertes Shift-Reduce-Dependency-Parsing

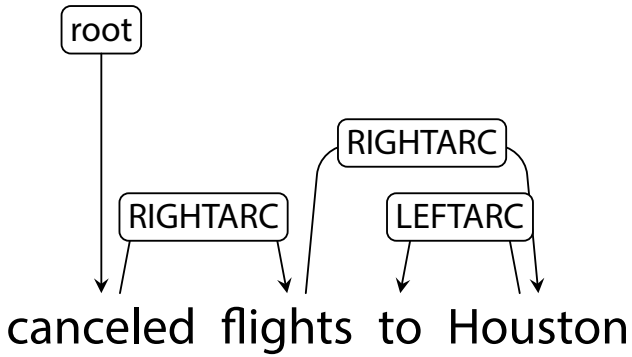
- `SHIFT`-**Operation**: Wörter in **Wortliste** (Buffer) auf Stack
  $\rightarrow$ *Stack wird mit* `root`*-Knoten* ***initialisiert***
  $\rightarrow$ *Abschluss, wenn Wortliste leer und nur noch* `root` *auf Stack*
- `REDUCE`-**Operation**:
  $\rightarrow$ *statt Ersatz durch Nonterminal (CFG):*
  $\Rightarrow$ ***Hinzufügen von Relation zwischen den beiden obersten Elementen auf dem Stack***
  $\Rightarrow$ ***Löschen des Dependents vom Stack***

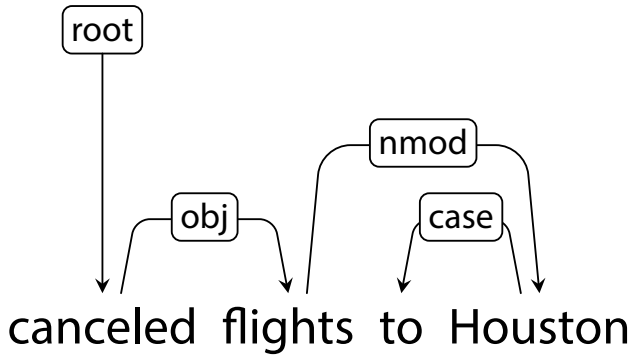
- **2 mögliche `REDUCE`-Operationen** (je nach Position Kopf):
  - **LEFTARC** (Kopf rechts): the $\leftarrow$ flights
  - **RIGHTARC** (Kopf links): book $\rightarrow$ me
- **Einschränkung bei *RIGHTARC***: nur, wenn der Dependent der möglichen Relation nicht Kopf einer der Relationen aus der Menge offener Relationen ist
  $\rightarrow$ *Einschränkung verhindert, dass **Wort zu früh vom Stack** genommen wird*
  $\rightarrow$ *dagegen `LEFTARC`: immer möglich (d.h. nur projektive Strukturen, siehe unten)*





Step	Stack	Word List (Buffer)	Transition	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning $\leftarrow$ flight)
7	[root, book, the, flight]	[]	LEFTARC	(the $\leftarrow$ flight)
8	[root, book, flight]	[]	RIGHTARC	(book $\rightarrow$ flight)
9	[root, book]	[]	RIGHTARC	(root $\rightarrow$ book)
10	[root]	[]	Done	





	Stack	Word List (Buffer)	Transition	
	[root,canceled,flights]	[to, Houston]	**SHIFT oder RIGHTARC ?**	
mögliche Übergänge:				Relation Added
SHIFT	[root,canceled,flights,to]	[Houston]		-
RIGHTARC	[root,canceled]	[to, Houston]		(canceled → flights)

- **richtiger Übergang:** `SHIFT`

  → *bei `RIGHTARC` wird flights zu früh vom Stack entfernt; Relation (flights*

  → *Houston) wäre dann nicht mehr möglich*

Evaluation von Dependenz-Parsing-Systemen

- **Überprüfung an Testmenge** (Teilmenge Dependency-Treebank)
- ***unlabeled attachment accuracy***: **korrekte Zuweisung Dependent zu Kopf**
- ***labeled attachment accuracy***: **korrekte Zuweisung** und **korrekte Relation** zwischen Dependent und Kopf