Lösung der Wiederholungsaufgaben

1 Syntaktische Ambiguität 2

2 Syntaktische Kategorien 5

3 Konstituentenstruktur 6

4 Parsingalgorithmen 8

5 Dependenzstruktur 10

6 Dependency Parsing 13

7 Komplexe Sätze 15

8 Grammatische Merkmale, Unifikation und Typhierarchie 19

9 Unifikationsgrammatiken 25

10 Statistisches Parsing 27

11 Datengestützte Syntaxanalyse 28

12 Partielles Parsing 30

1 Syntaktische Ambiguität

Arten syntaktischer Ambiguität .....

(a)
- Welche Art struktureller Ambiguität liegt in folgendem Satz vor?
- Erläutern Sie und geben Sie die Dependenz-Regeln an, die die Ambiguität verursachen.

(1) *The horse raced past the barn fell.*

Lösung:

- Temporale Ambiguität

```
1 ROOT → raced
2 raced → horse (nsubj)
```

vs.

```
1 ROOT → fell
2 fell → horse (nsubj)
3 horse → raced (acl:relcl)
```

(b)
- Welche Art struktureller Ambiguität liegt in folgendem Satz vor?
- Erläutern Sie und geben Sie die CFG-Regeln an, die die Ambiguität verursachen (Label: `NP, PP, VP, Det, N, V`).

(2) *I shot an elephant in my pajamas.*

Lösung:

- PP-Attachment-Ambiguität

```
1 NP → Det N | Det N PP
2 VP → V NP | VP PP
```

(c)
- Welche Art syntaktischer Ambiguität liegt bei folgender NP vor?
- Geben Sie für beide Analysen den Klammerausdruck an.

(3) *große Hunde und Katzen*

Lösung:

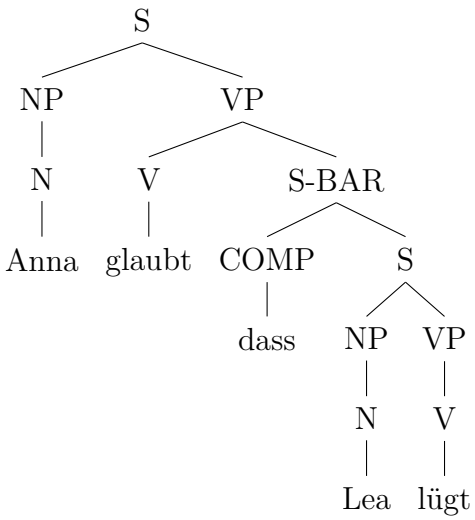
- Koordinationsambiguität
- (große (Hunde und Katzen)) **vs.** ((große Hunde) und Katzen)

Rekursionstypen .....

(a) Welche Art der Rekursion ermöglicht die verbale PP-Attachment-Regel `VP -> VP PP`?

Lösung:
direkte Rekursion (linksrekursiv): VP ist unmittelbare Konstituente von VP

(b) Erklären Sie anhand der in folgender Ableitung verwendeten Regeln, wie die rekursive Einbettung von Satzkomplementen zustande kommt?



Lösung:

- durch die Regeln VP → V S-BAR und S-BAR → COMP S können beliebig tief weitere Sätze (S) in den Hauptsatz (S) als Objektkomplemente eingebettet werden.

(c)
- Welcher Art ist die Rekursion in (b)?
- Liegt hier eine *center-embedding*-Konstruktion vor?

Lösung:

- indirekte Rekursion (eingebettetes S ist nicht unmittelbare Konstituenten von S)
- kein *center-embedding* (sondern *edge-embedding*):
  - die Objektkomplemente sind rechts eingebettet
  - dagegen *center-embedding*, z.B. bei Relativsätzen:
    Die Katze, die der Hund biss, lief weg.
    Die Katze, die der Hund, der entlaufen war, biss, lief weg.

2 Syntaktische Kategorien

Konstituententests .....

- Identifizieren Sie in folgendem Satz in fünf Schritten alle Konstituenten (oberhalb der Wortebene).
- Pro Schritt darf ein Test angewandt werden, pro Schritt dürfen mehrere Konstituenten gleichzeitig identifiziert werden.
- Geben Sie zu jedem Schritt den verwendeten Test an.

(4) *Das Auto hält an der rot leuchtenden Ampel.*

Lösung:
Schritt 1: *Das Auto hält an der Ampel.* - **Eliminierungstest**
Schritt 2: *Das Auto hält an der Straße.* - **Substitutionstest**
Schritt 3: *Das Auto hält an der rot leuchtenden Ampel und dem gelben Briefkasten.* - **Koordinationstest**
Schritt 4: *An der rot leuchtenden Ampel hält das Auto.* - **Permutationstest**
Schritt 5: *Das Auto fährt.* - **Substitutionstest**

- Geben Sie zum Schluss das daraus folgende Kastendiagramm an; verändern Sie den Satz hierbei nicht!

Kastendiagramm:

Lösung:
Das	Auto	hält	an	der	rot	leuchtenden	Ampel
Das	Auto	hält	an	der	rot leuchtenden	Ampel
Das	Auto	hält	an	der	rot leuchtenden Ampel
Das	Auto	hält	an	der rot leuchtenden Ampel
Das Auto	hält	an der rot leuchtenden Ampel
Das Auto	hält an der rot leuchtenden Ampel

3 Konstituentenstruktur

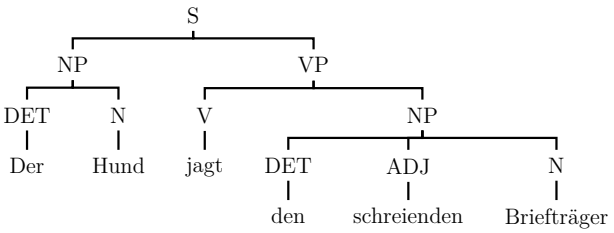
Kontextfreie Grammatiken und Syntaxbäume .....

- Erstellen sie zu dem folgenden Satz eine kontextfreie Grammatik und den dazugehörigen Syntaxbaum.
- Verwenden Sie nur folgende syntaktische Kategorien: `S, NP, VP, N, DET, ADJ, V`

(5) *Der Hund jagt den schreienden Briefträger.*

Lösung:

```
1  S   → NP VP
2  VP  → V NP
3  NP  → DET N | DET ADJ N
4  DET → "Der" | "den"
5  N   → "Hund" | "Briefträger"
6  ADJ → "schreienden"
7  V   → "jagt"
```



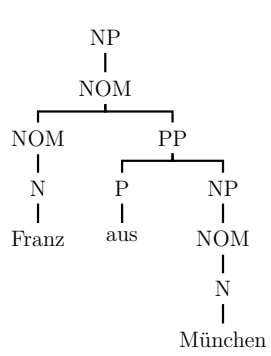
Erweiterung von kontextfreien Grammatiken .....

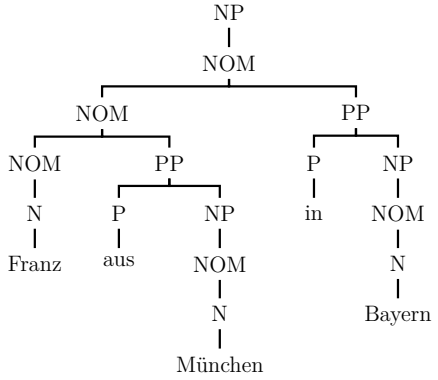
- Die Grammatik aus der vorherigen Aufgabe soll so erweitert werden, dass rekursiv PP-Adjunkte an NPs auftreten können (der Art *Franz aus München in Bayern ...*).
- Ergänzen Sie dazu im Folgenden die fehlenden rechten Seiten (PP-Regel ist gegeben; es dürfen keine neuen Kategorien eingeführt werden).

```
1  PP  → P NP
2  NP  →
3  NOM →
4  NOM →
```

Lösung:

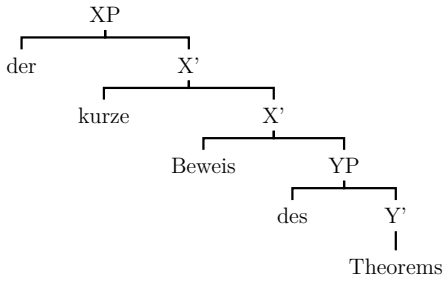
1	`PP → P NP`
2	`NP → NOM`
3	`NOM → NOM PP`
4	`NOM → N`



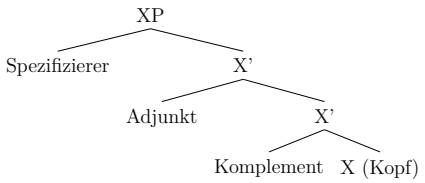


X-Bar-Konstituentenstruktur .....

Was sind in folgendem Syntaxbaum im X-Bar-Strukturschema **Spezifizierer**, **Komplemente**, **Adjunkte** und **Köpfe**?



Vgl.:



Lösung:

- **Spezifizierer:** der, des
- **Komplemente:** YP
- **Adjunkte:** kurze
- **Köpfe:** Beweis, Theorems

4 Parsingalgorithmen

Shift-Reduce-Parser

Gegeben sei folgende Grammatik:

```
1 S → NP VP
2 NP → PROPN
3 NP → DET N
4 VP → V NP
5 VP → V
6 DET → "der"
7 N → "Hund"
8 PROPN → "Max"
9 V → "kennt"
```

Mit dieser soll ein Shift-Reduce-Parser den Satz *der Hund kennt Max* analysieren. Während des Analysevorgangs wurden bereits die ersten drei Wörter eingelesen; dies ergab den folgenden Stack-Zustand:

V
NP

Der Stack-Zustand ist hier wie in der Vorlesung graphisch so dargestellt, dass neue Elemente immer oben auf den Stack kommen.

(a) Welche Parsing-Strategie verfolgt dieser Parser?

(b) Welche Operation wird als nächstes ausgeführt?

(c) Wie sieht der Stack nach Ausführen der Operation aus?

Lösung:

(a) Bottom-Up

(b)
- zunächst REDUCE (V zu VP); dies führt aber zu keinem vollständigen Parse (da NP-VP auf S zurückgeführt wird, aber ”Max” noch nicht verarbeitet)
- nach Backtracking: SHIFT (”Max”)

(c)

Max
V
NP

Recursive-Descent-Parser

Gegeben sei folgende Grammatik: wie oben

Mit dieser soll ein Recursive-Descent-Parser den Satz *der Hund kennt Max* analysieren.

(a) Welche der beiden grundlegenden Parsing-Strategien verfolgt dieser Parser?

(b) Welche zwei Operationen stehen diesem Parser zur Verfügung?

(c) Nennen Sie eine Folge von Regelanwendungen, die zu Backtracking führt.

Lösung:

(a) Top-Down

(b) PREDICT und SCAN

(c) 1-2 oder 1-2-8

Earley-Parser

Gegeben sei folgende Grammatik: wie oben.

Mit dieser soll ein Earley-Parser den Satz *der Hund kennt Max* analysieren. Während des Analysevorgangs wird folgende Zustandsmenge Q_1 erzeugt:

$$(\text{DET} \rightarrow \text{der} \cdot, 0) \tag{1}$$
$$(\text{NP} \rightarrow \text{DET} \cdot \text{N}, 0) \tag{2}$$
$$(\text{N} \rightarrow \cdot \text{Hund}, 1) \tag{3}$$

(a) Welche Operation wird als nächstes ausgeführt?

(b) Welcher Zustand wird dabei hinzugefügt?

(c) Welcher Zustandsmenge wird der Zustand hinzugefügt?

Lösung:

(a) SCAN

(b) $(\text{N} \rightarrow \text{Hund} \cdot, 1)$

(c) Q_2

5 Dependenzstruktur

Tests zur Komplement/Adjunkt-Unterscheidung .....

(a) Nennen Sie die drei Tests zur Komplement/Adjunkt-Unterscheidung. Was wird geprüft? Wann handelt es sich jeweils um ein Adjunkt?

Lösung:

- **Eliminierungstest**: wenn bei der Eliminierung der Konstituente die Grammatikalität erhalten bleibt, dann handelt es sich um ein Adjunkt
- **Adverbialsatz-Test**: wenn die Konstituente in einen Adverbialsatz ausgelagert werden kann (=grammatisch), dann Adjunkt
- ***geschehens*-Test**: wenn die Konstituente in einen Satz mit dem Verb *geschehen* ausgelagert werden kann und der Satz grammatisch bleibt, dann Adjunkt

(b) Bestimmen Sie für zwei verschiedene Konstituenten des folgenden Satzes, ob es sich jeweils um ein Komplement oder ein Adjunkt handelt. Verwenden Sie hierzu zwei der eben genannten Tests (für jede Konstituente einen).

(6) *Das blaue Krümelmonster verschlingt schmatzend alle Kekse im Park.*

Lösung:

**Das blaue Krümelmonster verschlingt schmatzend im Park.*

- **Eliminierungstest**: *alle Kekse* ist **Komplement (auch: Ergänzung)**.

Das blaue Krümelmonster verschlingt schmatzend alle Kekse, während es im Park ist.

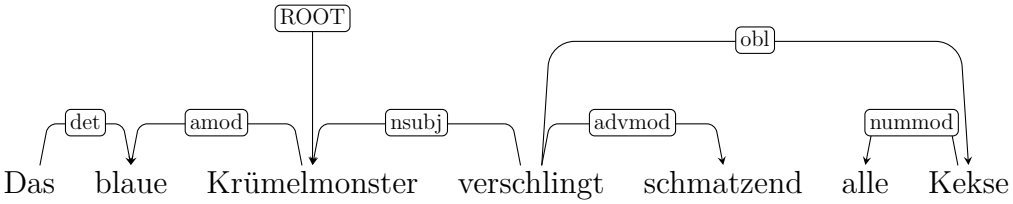
- **Adverbialsatz-Test**: *im Park* ist **Adjunkt (auch: Angabe)**.

Das blaue Krümelmonster verschlingt alle Kekse im Park und das geschieht schmatzend.

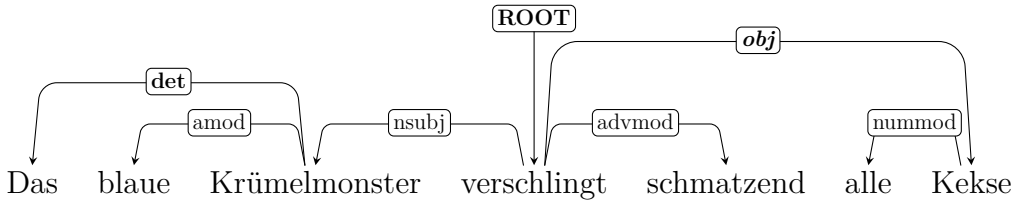
- ***geschehens*-Test**: *schmatzend* ist **Adjunkt (auch: Angabe)**.

Dependenzbäume .....

- Folgender Dependenzbaum enthält drei Fehler; diese können sich sowohl auf die Relationen als auch die Relationslabel beziehen; der Baum soll die UD-Konventionen erfüllen.
- Finden Sie die drei Fehler; markieren Sie dazu die falsche Relation bzw. schreiben Sie das korrekte Label über das falsche.
- (siehe letzte Seite für eine Übersicht der UD-Dependency-Labels)



Lösung:



1. der **ROOT-Knoten** (typischerweise das finite Verb) **kontrolliert unmittelbar oder mittelbar alle anderen Knoten** (kann also nicht Dependent sein wie *Krümelmonster* in der Angabe).

2. *Das* ist `det`-Modifikator von *Krümelmonster* (d.h. *Das* kann nicht ohne *Krümelmonster* auftreten); *Das könnte* zwar grundsätzlich auch *det*-Modifikator von *blaue* (als nominalisierte Form) sein; da aber in diesem Satz *blaue* `amod`-Modifikator von *Krümelmonster* ist, ist dies ausgeschlossen, da **kein Wort von mehr als einem Kopf abhängen kann**.

3. *Kekse* sind ***Objekt-Komplement*** von *verschlingt*; d.h. es ist valenzgefordertes Element (**Auftreten *und Form* (syntaktischer Typ) vom Kopf gefordert**, hier: Akkusativ; *verschlingt* kann also nicht ohne Akkusativ-Objekt auftreten); ***Kekse* kann also nicht nominaler Dependent des finiten Verbs mit der syntaktischen Funktion Adverbial (= `obl`)** sein (diese werden im Deutschen üblicherweise auch präpositional kodiert und sind Adjunkte, d.h. verbale Modifikatoren; ein solches Adverbial kann aber durchaus auch Komplement sein, also vom Verb gefordert, vgl. *Er stellt es neben/auf den Tisch (`obl`)*; allerdings ist hier nur das Auftreten des Adverbials gefordert, nicht dessen Form wie bei Präpositionalobjekten, vgl. *Er wartet auf sie (`obj`)*).

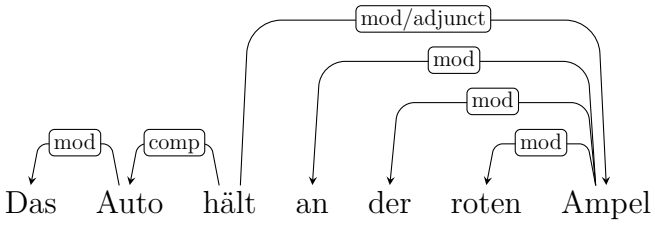
Komplement vs. Modifikator .....

- Bestimmen sie für den folgenden Satz die Abhängigkeiten. Markieren Sie, ob es sich bei der jeweiligen Abhängigkeit um ein Komplement oder einen Modifikator handelt.

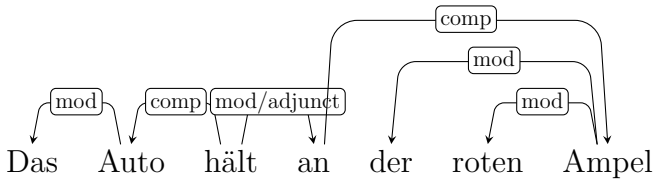
(7) *Das Auto hält an der roten Ampel*

Lösung:

UD-Schema:



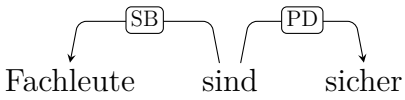
TIGER-Schema:



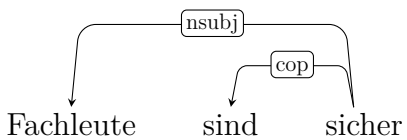
- Komplement = Ergänzung (Auftreten und Form vom Kopf gefordert)
- Verbaler Modifikator = Adjunkt / Angabe
- Nominaler Modifkator = Attribut

UD-Schema vs. TIGER-Schema .....

- Wandeln Sie folgenden Dependenzbaum im TIGER-Dependenz-Schema um in das *Universal-Dependencies*-Schema (*primacy of content word*-Maxime).
- (siehe letzte Seite für eine Übersicht der UD-Dependency-Labels)



Lösung:

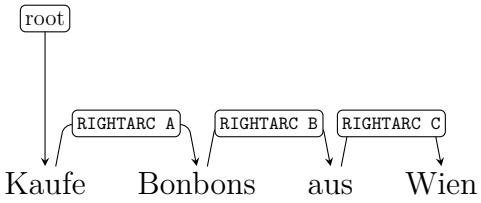


6 Dependency Parsing

Übergangsbasierter Shift-Reduce-Dependency-Parser .....

- In welcher Reihenfolge werden im Folgenden jeweils die angegebenen `REDUCE`-Übergange durchgeführt?
- Begründen Sie.
- Wie unterscheiden sich die beiden Syntaxbäume?

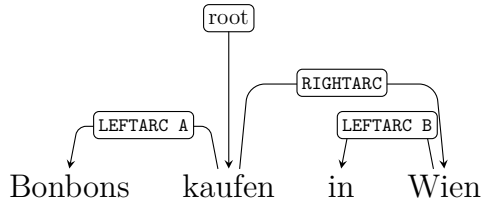
(a)



Lösung:

- `RIGHTARC C`, dann `RIGHTARC B`, dann `RIGHTARC A`
- – anderenfalls werden die Dependenten *Bonbons* (mit `RIGHTARC A`) bzw. *aus* (mit `RIGHTARC B`) zu früh vom Stack genommen
  – diese sind nämlich selbst wiederum Köpfe von Dependenten
  – würde man beispielsweise mit `RIGHTARC A` beginnen, könnte der `RIGHTARC B`-Übergang nicht mehr durchgeführt werden, da *Bonbons* als Dependent von *Kaufe* schon vom Stack gelöscht wäre.
- **`RICHTARC`-Regel:**
  – *`RICHTARC`-Übergang nur durchführen, wenn der Dependent der möglichen Relation nicht Kopf einer der Relationen aus der Menge offener Relationen ist*
  – *sonst: `SHIFT`-Übergang*

(b)



Lösung:

- `LEFTARC A`, dann `LEFTARC B`, dann `RIGHTARC`
- im Unterschied zu oben ist diese Reihenfolge (`RIGHTARC` zuletzt) nicht durch die `RICHTARC`-Regel (s.o.) bedingt, sondern kommt durch das sukzessive Hinzufügen der Wörter auf den Stack (`SHIFT`-Übergang):
  - zwischen *kaufen* und *in* gibt es hier keine Relation
  - entsprechend wird mit `SHIFT` *Wien* auf den Stack geschoben, der dann so aussieht: $[kaufen, in, Wien]$
  - nun wird mit `LEFTARC B` *in* als Dependent dieser Relation vom Stack entfernt; dieser sieht nun so aus: $[kaufen, Wien]$
  - jetzt kann mit `RICHTARC` *Wien* vom Stack gelöscht werden (da es keine offene Relation mehr gibt, in der *Wien* Kopf ist)
- Unterschied zu oben:
  - diese Dependenzanalyse folgt der *primacy of content words*-Maxime des UD-Schemas (Substantiv als Dependent statt Präposition)
  - Wortstellung (Imperativ vs Infinitiv)
  - Präpositionalphrase ist hier Adverbial, nicht Attribut

7 Komplexe Sätze

Konstituentenstruktur komplexer Sätze .....

- Geben Sie zu folgenden Regeln bzw. Regelgruppen **für komplexe Sätze des Englischen** einer Zeile jeweils deren Funktion an - welcher Typ komplexer Sätze wird jeweils erzeugt?
- Geben Sie zusätzlich auch die entsprechenden UD-Kantenlabel an.

S-BAR → COMP S

Lösung:

- S-BAR-Grundregel (eingebettete Sätze)
- UD-Label: meist `mark` für Komplementierer (z. B. für *dass*, *wenn*), z.T. auch `nsubj` usw., etwa beim Relativsatz (*der ...*)

NP → NOM, NOM → NOM S-BAR

Lösung:
Relativsätze (= Attributsätze): `acl:relcl`

VP → V S-BAR

Lösung:
Objektkomplementsatz: `ccomp`

S → S-BAR VP

Lösung:
Subjektkomplementsatz: `csubj`

S → S-BAR NP VP

Lösung:
Adverbialsatz: `advcl`

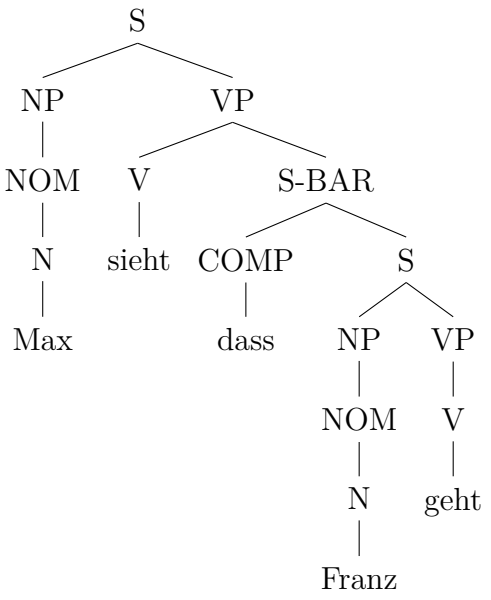
- Erstellen Sie die **Phrasenstrukturbäume** zu den folgenden beiden Sätzen; verwenden Sie dazu die dem eingebetteten Satztyp entsprechenden Regeln von oben, sowie zusätzlich folgende Regeln:
  S → NP VP, NOM → N, VP → V

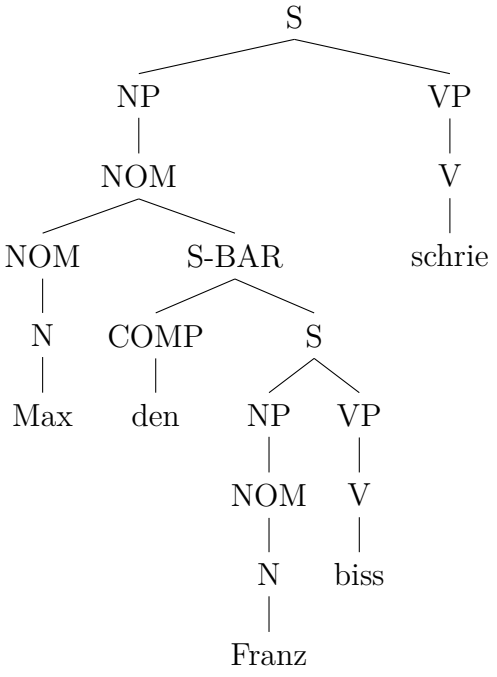
(8) *Max (N) sieht (V) dass (COMP) Franz (N) geht (V)*

(9) *Max (N), den (COMP) Franz (N) biss (V), schrie (V)*

- Verwenden Sie die in Klammern angegebenen lexikalischen Kategorien!

Lösung:

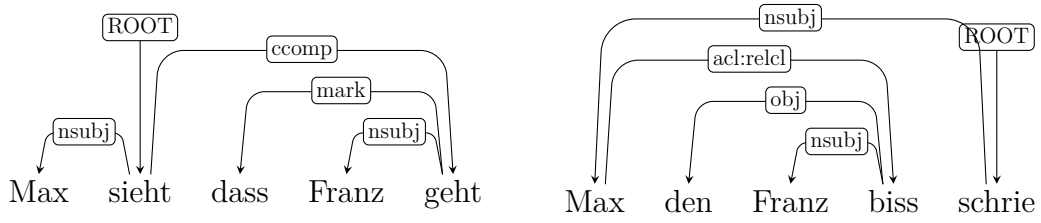




Dependenzstruktur komplexer Sätze 1 .....

- Erstellen Sie zu den Sätzen (8) und (9) auch die entsprechenden Dependenzbäume (Kopf von S = VP-Kopf; Kopf von S-BAR = Kopf von S).
- Verwenden Sie die im Anhang auf der letzten Seite angegebenen UD-Dependency-Label.

Lösung:

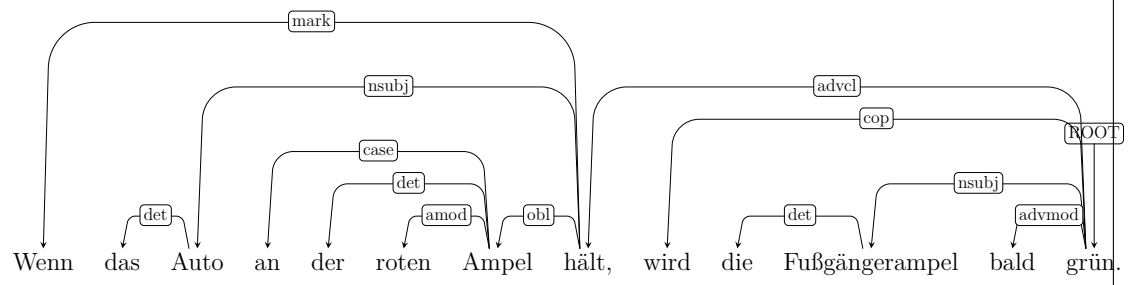


Dependenzstruktur komplexer Sätze 2 .....

- Bestimmen Sie alle Dependenzrelationen im folgenden Satz.
- Bestimmen Sie außerdem, um welche grammatische Relation es sich jeweils handelt.

(10) *Wenn das Auto an der roten Ampel hält, wird die Fußgängerampel bald grün.*

Lösung:



Infinite Konstruktionen: Subjekt- vs. Objektkontrolle .....

Liegt in folgendem Satz Subjekt- oder Objektkontrolle vor? Begründen Sie.

(11) *Sie überredeten ihn, noch zu bleiben*

Lösung:

- Objektkontrolle
- das Objekt des Matrixsatzes ist referenzidentisch mit dem (nicht explizit versprachlichten) Subjekt der Infinitivkonstruktion, also sinngemäß: *Sie überredeten ihn, dass er noch bleibt.*

8 Grammatische Merkmale, Unifikation und Typhierarchie

Kodierungstypen syntaktischer Funktionen .....

Wie werdem in den unteren transitiven Satzpaaren einer *Kunstsprache mit englischem Lexikon* die grammatischen Relationen **Subjekt** bzw. **Objekt** jeweils kodiert?

- Geben Sie jeweils eine der folgenden drei Kodierungsarten an:
  - **Kasusmorphologie** (*morphologisch* = substantielle Kodierung am Dependenten)
  - **Verbale Kongruenz** (*morphologisch* = substantielle Kodierung am Verb)
  - **Wortstellung** (strukturelle Kodierung)

(12) bird cat eat
'Die Katze frisst den Vogel.'

(13) cat bird eat
'Der Vogel frisst die Katze.'

Lösung:

- Kodierung durch Wortstellung OSV
- bird cat eat
  O S V
  cat bird eat

(14) bird-fe cat eat
'Die Katze frisst den Vogel.'

(15) bird cat-fe eat
'Der Vogel frisst die Katze.'

Lösung:

- Kodierung über Kasusmorphologie
- *-fe* = **Objektkasus**
- bird-fe cat eat
  O S V
  (vgl.: *den Vogel die Katze frisst*)

- bird cat-fe eat
  S O V
  (vgl.: *der Vogel die Katze frisst*)

(16) bird-i cat-o eat-o
'Die Katze frisst den Vogel.'

(17) bird-i cat-o eat-i
'Der Vogel frisst die Katze.'

Lösung:

- Kodierung über verbale Kongruenz
- das Verb kongruiert mit einer nominalen Kategorie (z.B. Genus)
- das Verb spiegelt die nominalen Marker *-i* und *-o*, es herrscht dann also **Subjektkongruenz** (bei Annahme Kongruenz durch gleiche Form)
- bird-i cat-o eat-o
  O S V
  (vgl. Numerus-Kongruenz im Deutschen: *die Vögel die Katze frisst*)
- bird-i cat-o eat-i
  S O V
  (vgl.: *die Vögel die Katze fressen*)

Kasusrektion, Agreement und Subkategorisierung .....

Welche der folgenden morphosyntaktischen Beschränkungen ist in den unteren Sätzen jeweils verletzt?

- **Kasusrektion**
- **Nominale Kongruenz**
- **Verbale Kongruenz** (*Agreement*)
- **Subkategorisierung** (Anzahl und Art der verbalen Argumente)

(18) *Der Auto fährt schnell.*

Lösung:
Nominale Kongruenz

(19) *Den Auto fährt schnell.*

Lösung:
Kasusrektion

(20) *Die Autos fährt schnell.*

Lösung:
Verbale Kongruenz

(21) *Das Auto fährt, dass es ankommt.*

Lösung:
Subkategorisierung

Wortstellung: Stellungsfeldermodell .....

(a) Identifizieren Sie in den folgenden Sätzen durch Unterstreichen das Mittelfeld und, falls vorhanden, das Vorfeld:

(22) *Es hat Kuchen gegeben.*

(23) *Hat es Kuchen gegeben?*

(b) Um welches *Es* handelt es sich in (22)?

- **Topik-Es** = Vorfeld-Platzhalter
- **Subjekt-Expletiv**
- **pronominaler Ersatz** für NP

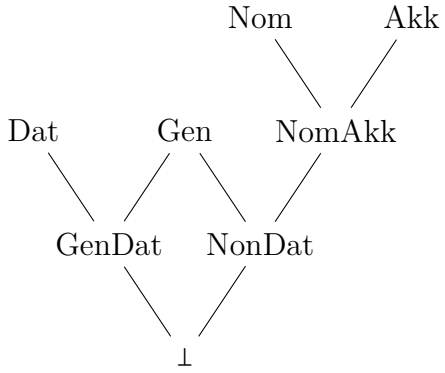
Lösung:

(a) (24) *Es hat Kuchen gegeben.*

(25) *Hat es Kuchen gegeben?*

(b)
- Subjekt-Expletiv
- Begründung: verschwindet in der Frage (V1-Wortstellung, d.h. ohne Vorfeld) nicht, sondern taucht im Mittelfeld auf
- ist also kein Topik-Es, das bei leerem Vorfeld im V2-Satz diese notwendig zu besetztende Position einnimmt (wie in *Es wird getanzt?* : *Wird getanzt?*)
- valenzsemantisch ist das Verb nullwertig, d.h. *es* ist hier kein pronominaler NP-Ersatz sondern syntaktisch gefordertes Element, dass die Subjektposition einnimmt

Unifikationsparsing und getypte Merkmalstrukturen .....



Gegeben sei folgende Typhierarchie:

Typhierarchien .....

Unifizieren Sie die folgenden Paare von Typen. Typen, die nicht unifizieren, markieren Sie als *undefiniert*.

(a) Nom ⊔ Akk = _____

(b) GenDat ⊔ NonDat = _____

(c) GenDat ⊔ Gen = _____

(d) Nom ⊔ ⊥ = _____

(e) Akk ⊔ NomAkk = _____

Lösung:

(a) undefiniert

(b) Gen

(c) Gen

(d) Nom

(e) Akk

Subsumption .....

Gegeben seien nun zusätzlich folgende Merkmalstrukturen mit $\theta(\text{FS1}) = \theta(\text{FS2}) = \theta(\text{FS3}) = \bot$.

$$\text{FS1} = \begin{bmatrix} \text{CAS} & \text{NomAkk} \\ \text{GEN} & \text{mask} \end{bmatrix} \qquad \text{FS2} = \begin{bmatrix} \text{CAS} & \text{Nom} \\ \text{GEN} & \text{mask} \\ \text{PER} & 3 \end{bmatrix} \qquad \text{FS3} = \begin{bmatrix} \text{CAS} & \text{Akk} \\ \text{PER} & 3 \end{bmatrix}$$

Entscheiden Sie jeweils mit *ja* oder *nein*:

(a) FS2 $\sqsubseteq$ FS3 ? _____

(b) FS1 $\sqsubseteq$ FS2 ? _____

(c) FS2 $\sqsubseteq$ FS1 ? _____

(d) FS3 $\sqsubseteq$ FS2 ? _____

(e) FS3 $\sqsubseteq$ FS3 ? _____

Lösung:

(a) nein

(b) ja

(c) nein

(d) nein

(e) ja

Unifikation

Unifizieren FS2 und FS3? Falls ja, unifizieren Sie; falls nein, begründen Sie.

Lösung:

Nein, da FS2@CAS = Nom, FS3@CAS = Akk und Nom $\sqcup$ Akk undefiniert ist (bzw. weil Nom und Akk nicht unifizieren).

Bedingungen

Entscheiden Sie jeweils mit *ja* oder *nein*:

(a) FS2 $\models$ CAS : $\bot$? _____

(b) FS2 $\models$ GEN : neut ? _____

(c) FS3 $\models$ PER : 3 ? _____

(d) FS1 $\models$ NomAkk $\wedge$ mask ? _____

(e) FS2 $\models$ CAS : NomAkk ? _____

Lösung:

(a) ja

(b) nein

(c) ja

(d) nein

(e) ja

9 Unifikationsgrammatiken

Modellierung von Subkategorisierung, Rektion und Agreement .....

- Betrachten Sie folgenden Ausschnitt aus einer merkmalsbasierten Grammatik für einen kleinen Ausschnitt des Englischen (s. `NLTK/book_grammars/feat0.fcfg`).
- Beantworten Sie untenstehende Fragen und geben Sie jeweils die Zeilennummern an, auf die sich ihre Antwort bezieht.

```
1 S -> NP[NUM=?n] VP[NUM=?n]
2
3 NP[NUM=?n] -> N[NUM=?n]
4 NP[NUM=?n] -> PropN[NUM=?n]
5 NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
6 NP[NUM=pl] -> N[NUM=pl]
7
8 VP[TENSE=?t, NUM=?n] -> IV[TENSE=?t, NUM=?n]
9 VP[TENSE=?t, NUM=?n] -> TV[TENSE=?t, NUM=?n] NP
```

(a) Wie wird hier Subkategorisierung modelliert?

Lösung:
durch Erweiterung von CFG-Kategoriensymbolen (Zeilen 8 und 9)

(b) Nennen Sie eine alternative Modellierung von Subkategorisierung.

Lösung:
mit SUBCAT-Merkmal als Index (`V[SUBCAT=1]` oder `V[SUBCAT=trans]`)

(c) Wie wird hier das Subjekt-Verb-Agreement modelliert?

Lösung:
Constraint auf Übereinstimmung im `NUM`-Merkmal von NP und VP in Zeile 1

(d) Geben Sie das entsprechende Constraint für das Subjekt-Verb-Agreement mittels einer Gleichung an (z.B. als Pfadgleichung der Form `CAT1@Feat = CAT2@Feat`).

Lösung:
NP@NUM = VP@NUM

Erweiterung von kontextfreien Grammatiken um Merkmale .....

- Gegeben sei folgende Grammatik:

```
 -> ADV VP
VP -> V NP PP
PP -> P NP
NP -> N
NP -> PROPN
ADV -> "gestern"
ADV -> "heute"
V -> "ging"
V -> "geht"
PROPN -> "Fritz"
N -> "Arbeit"
P -> "zur"
```

- Mit dieser können die folgenden Sätze hergeleitet werden:
  gestern ging Fritz zur Arbeit
  heute ging Fritz zur Arbeit
  heute geht Fritz zur Arbeit
- Passen sie die Grammmatik so an, dass Imperfekt und Präsens als Merkmale in der Grammatik unterschieden werden können, so dass der folgende Satz nicht mehr erkannt wird:

(26) **gestern geht Fritz zur Arbeit*

Lösung:

```
S → ADV[TIME=?x] VP[TIME=?x]
VP[TIME=?x] → V[TIME=?x] NP PP
PP → P NP
NP → N
NP → PROPN
ADV[TIME=imp] → "gestern"
ADV → "heute"
V[TIME=imp] → "ging"
V[TIME=pr] → "geht"
PROPN → "Fritz"
N → "Arbeit"
P → "zur"
```

10 Statistisches Parsing

PCFG: Gewichte und Ableitungswahrscheinlichkeit .....

Betrachten Sie folgendes PCFG-Parsing (** = unkenntlich gemacht):

```
1  grammar = nltk.PCFG.fromstring("""
2      S    -> NP VP              [1.0]
3      VP   -> TV NP              [0.4]
4      VP   -> IV                 [**]
5      VP   -> DatV NP NP         [0.3]
6      TV   -> 'saw'              [1.0]
7      IV   -> 'ate'              [1.0]
8      DatV -> 'gave'             [1.0]
9      NP   -> 'telescopes'       [0.8]
10     NP   -> 'Jack'             [0.2]
11     """)
12 viterbi_parser = nltk.ViterbiParser(grammar)
13 for tree in viterbi_parser.parse(['Jack', 'saw', 'telescopes']):
14     print(tree)
15 (S (NP Jack) (VP (TV saw) (NP telescopes))) (p=0.064)
```

(a) Geben Sie die Berechnung für die Ableitungswahrscheinlichkeit in Zeile 15 an?

Lösung:
$1.0 * 0.2 * 0.4 * 1.0 * 0.8$ oder $0.2 * 0.4 * 0.8$ oder beliebige Permutationen

(b) Welchen Wert muss das Gewicht für die Regel **VP → IV** haben?

Lösung:
0.3

- Gewichte der beiden andere VP-Regeln:
  - **VP → TV NP**: 0.4
  - **VP → DatV NP NP**: 0.3
- Gesamtwahrscheinlichkeit für VP-Regeln muss 1 ergeben:
  1 - 0.4 - 0.3 = 0.3

11 Datengestützte Syntaxanalyse

Datengestützte Methoden: Abschätzung Regelwahrscheinlichkeiten .....

(a) Für ein großes Korpus sei lediglich *part of speech* (POS) annotiert; Syntaxbäume stehen nicht zur Verfügung.
Welche Arten von Regeln einer kontextfreien Grammatik kann man mit diesen Daten automatisch generieren?

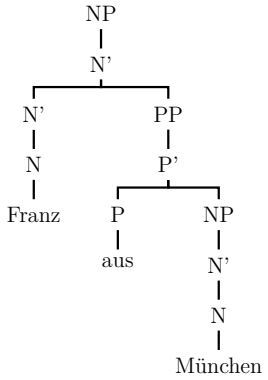
Lösung:
Lexikalische Regeln

(b) Folgende Häufigkeiten wurden aus einem Datensatz gezählt:
$count(VP \rightarrow V) = 200$, $count(VP \rightarrow V\ NP) = 100$, $count(VP \rightarrow \setminus *) = 300$.
Berechnen Sie $P(V\ NP \mid VP)$ mit der MLE-Methode.

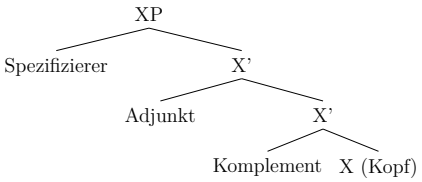
Lösung:
$\frac{100}{300} = \frac{1}{3} \approx 33.3\%$

Methoden für lexikalisierte und *history-based* PCFGs .....

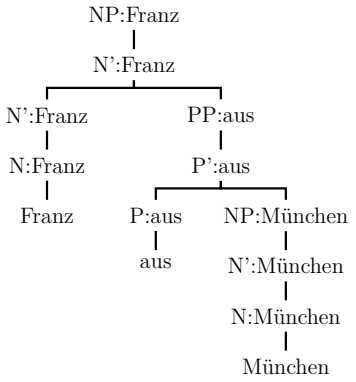
(a) Führen Sie im linken Syntaxbaum eine Kopfannotation durch; Geben Sie anschließend die lexikalisierte Regel für den Wurzelknoten an. Orientieren Sie bei der Kopfannotation an der Strukturposition des Kopfes im X-Bar-Schema (vgl. rechter Syntaxbaum).



Vgl.:

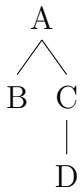


Lösung:



- **NP:Franz → N':Franz**

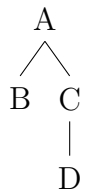
(b) Führen Sie für die CFG-Regeln **C → D** und **A → B** in unterem Syntaxbaum *parent-annotation* durch.



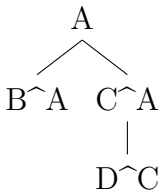
Lösung:

- **C^A → D^C**
- **A^? → B^A**

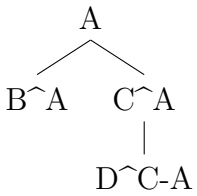
Original:



Parent Annotation:



+Grandparent Annot.:



12 Partielles Parsing

Chunking .....

Markieren Sie alle Nominalphrasen (NPs), indem Sie den folgenden deutschen Satz vollständig nach dem IOB-Tagging-Schema annotieren; verwenden Sie nur folgende Label: `B-NP, I-NP, O`.

Token	Der	junge	Mann	gab	ihr	das	Buch	.
Tag								

Lösung:

Token	Der	junge	Mann	gab	ihr	das	Buch	.
Tag	B-NP	I-NP	I-NP	O	B-NP	B-NP	I-NP	O

Kaskadierende Chunk-Parser .....

(a) Mit welcher Methode kann z.B. folgende hierarchische Struktur einer Präpositionalphrase mit flachen Chunk-Parsern erzeugt werden:

[PP auf/P [NP dem/DET Baum/N]]

Lösung:

- hintereinandergeschaltete flache Chunk-Parser
  (= **kaskadierender Chunk-Parser**)
- Output des einen als Input des folgenden Chunkers

Evaluationsmetriken .....

Berechnen Sie Accuracy, Precision und Recall für folgende korrekte Annotationen (`truth`) und folgende Hypothesen (`predict`). Geben Sie bitte jeweils Brüche an.

Sample	0	1	2	3	4	5	6	7	8	9
`truth`	PP	PP	PP	O	O	O	O	PP	O	PP
`predict`	PP	O	O	PP	O	PP	PP	O	PP	PP

Accuracy: _____

Precision (für die Klasse PP): _____

Recall (für die Klasse PP): _____

Lösung:

Accuracy: $\frac{3}{10} = 30\%$

Precision (für die Klasse PP): $\frac{2}{6} \approx 33.3\%$

Recall (für die Klasse PP): $\frac{2}{5} = 40\%$

Angabe: Hilfsmittel

Universal Dependency Relations

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