Syntax natürlicher Sprachen

Tutorium

Feature-basierte Grammatiken

Sarah Anna Uffelmann

22.12.2023

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"
"""
```

Beispiel von Constraint-Regeln in NLTK

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.

Subjektkasus und Subjekt-Verb-Agreement

Bisher hatten wir diese Regel verwendet: S -> NP VP

Wir wollen jedoch sicherstellen, dass die NP (das Subjekt) im Nominativ steht und bezüglich Person und Numerus mit dem Verb der VP kongruent ist:

$$\begin{bmatrix}\text{CAT} & S\end{bmatrix} \rightarrow \begin{bmatrix}\text{CAT} & NP \\ \text{CASE} & NOM \\ \text{AGR} & \boxed{1}\end{bmatrix} \begin{bmatrix}\text{CAT} & VP \\ \text{AGR} & \boxed{1}\end{bmatrix}$$

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.)

Subjektkasus und Subjekt-Verb-Agreement

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

-> AGR unifiziert

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

-> AGR unifiziert nicht

(Widerspruch im Numerus)

Boolsche Merkmale: Invertierte Wortstellung

Aussagesatz: "You like chocolate."

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

Die invertierte Wortstellung lässt sich vermittels boolscher Mekmale 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 invertieter 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:

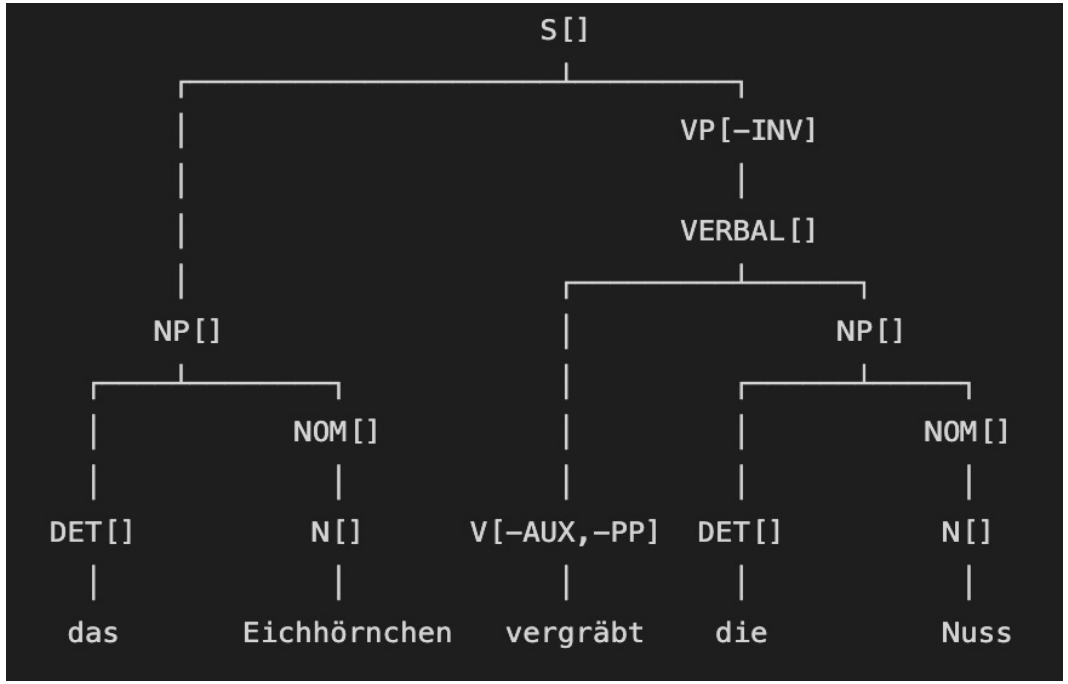
VP[-INV] -> VERBAL (für Präsens)
VP[+INV] –> V[+AUX] VERBAL (für Perfekt)

Komplementregeln:

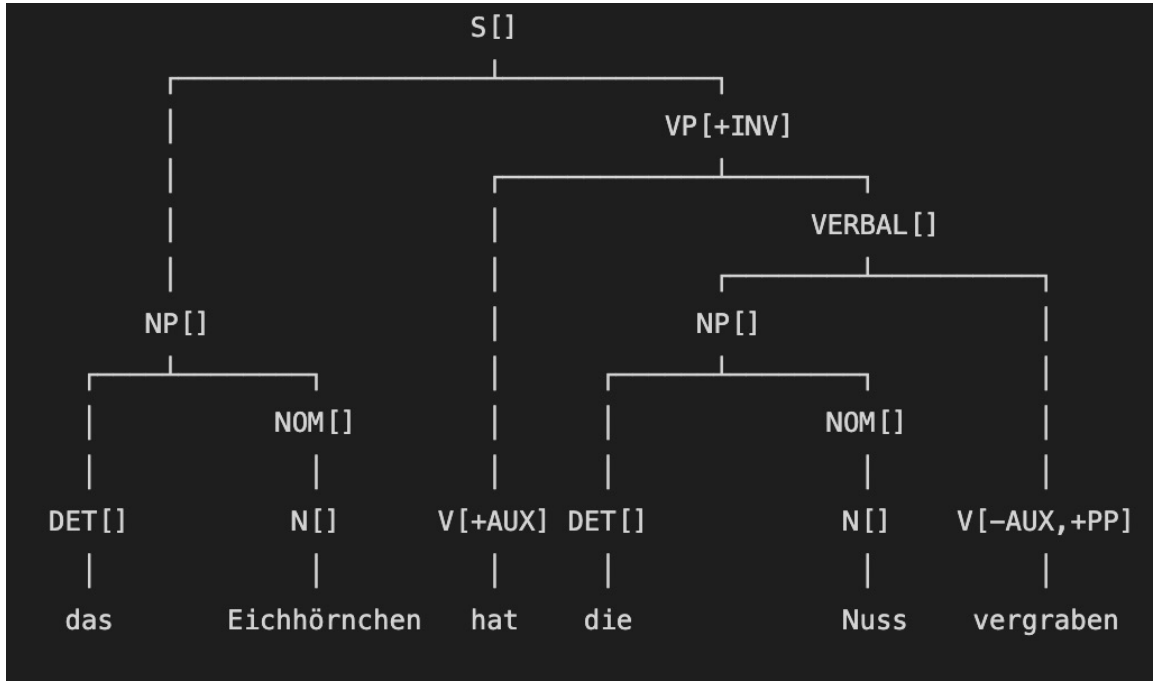
VERBAL -> V[-PP] NP (für Präsens)
VERBAL -> NP V[-AUX, +PP] (für Perfekt)

(Adjunktregeln brauchen wir hierfür nicht.)

Boolsche Merkmale: Hilfsverbkonstruktionen



Präsens (nicht invertiert)



Perfekt (invertiert)

Gap-Features mit Slash-Kategorien

Aussagesatz: “You like chocolate.“

Entscheidungsfragesatz: “Do you like chocolate?“

Ergänzungsfragesatz: “What do you like?"

Im Ergänzungsfragesatz wird die Objekt-NP („chocolate“) aus der VP herausbewegt und (als Fragepronomen) an den Satzanfang gestellt. An der ursprünglichen Stelle der NP bleibt eine Leerstelle (Gap) zurück: “What do you like __ ?“

Dies lässt sich mit Hilfe von Slash-Kategorien modellieren.

Gap-Features mit Slash-Kategorien

Aussagesatz: "You like chocolate."

S –> NP VP

Ergänzungsfragesatz: "What do you like?"

S -> NP S/NP (= gap-introduction) (S/NP = „S ohne NP")

Die gap-information wird in der Grammatik „heruntergereicht", bis wir an der Stelle ankommen, an der die NP im Aussagesatz stehen würde. An dieser Stelle definieren wir die Regel:

NP/NP –>

Die rechte Seite bleibt leer (= gap-realisation).
(An dieser Stelle steht das leere Wort, im Code schreiben wir jedoch nichts.)

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] -> NP[-QUEST] VP[+INV]
    VP[+INV] -> V[+AUX] NP[-QUEST] V[+PP]
    NP[QUEST=?x] -> PRON[QUEST=?x]

    S[+INV] -> V[+AUX] NP VP
    VP -> NP V[+PP]

#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 ->

    PRON[-QUEST]   -> "sie"
    PRON[-QUEST]   -> "ihn"
    PRON[+QUEST]   -> "wen"
    V[+AUX] -> "hat"
    V[+PP]   -> "gesehen"
```