

Eigenschaften der syntaktischen Struktur

- Sätze setzen sich aus Satzteilen (Konstituenten) zusammen, die wiederum aus einfachen oder ihrerseits komplexen Satzteilen bestehen können. Sätze können deshalb beliebig lang und beliebig tief geschachtelt sein.
- Die Syntax natürlicher Sprachen erlaubt variable Wortstellung: Wörter und Konstituenten mit der gleichen Funktion können an unterschiedlichen Stellen eines Satzes stehen. Unterschiedliche Sprachen erlauben sehr unterschiedliche Freiheitsgrade.
- Die grammatischen Eigenschaften unterschiedlicher Wörter und Konstituenten im Satz hängen voneinander ab – zum Teil auch in Fällen, in denen die Wörter und Konstituenten im Satz weit auseinander liegen.

Konstituenten

- Er hat die Übungen gemacht.
- Der Student hat die Übungen gemacht.
- Der interessierte Student hat die Übungen gemacht.
- Der an computerlinguistischen Fragestellungen interessierte Student hat die Übungen gemacht.
- Der an computerlinguistischen Fragestellungen interessierte Student im ersten Semester hat die Übungen gemacht.
- Der an computerlinguistischen Fragestellungen interessierte Student im ersten Semester, der im Hauptfach Informatik studiert, hat die Übungen gemacht.
- Der an computerlinguistischen Fragestellungen interessierte Student im ersten Semester, der im Hauptfach, für das er sich nach langer Überlegung entschieden hat, Informatik studiert, hat die Übungen gemacht.

Syntaktische Verarbeitung

- Syntax (Konstituentenstruktur) natürlicher Sprachen kann einfach und elegant durch kontextfreie Grammatiken dargestellt werden.

Eine kontextfreie Grammatik für deutsche Sätze

Kompaktere Schreibweise:

- Optionale Konstituenten werden in Klammern geschrieben.
- Lexikalische Symbole werden mit Komma aneinandergehängt.

$S \rightarrow NP VI$ $NP \rightarrow ART NN (SREL)$

$SREL \rightarrow RPRO VI$ $S \rightarrow NP VT NP$

$SREL \rightarrow RPRO NP VT$

$VI \rightarrow \textit{schläft, arbeitet, wartet, ...}$

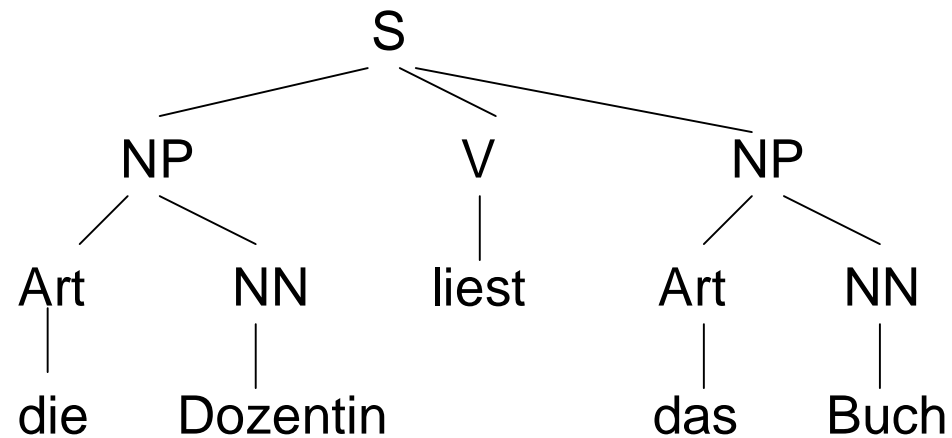
$VT \rightarrow \textit{wählt, studiert, liest, kennt, ...}$

$NN \rightarrow \textit{Student, Fach, Dozentin, Professor, Buch, ...}$

$RPro \rightarrow \textit{der, die, das}$

$ART \rightarrow \textit{der, die, das}$

Kontextfreie Grammatik: Ein Beispielsatz

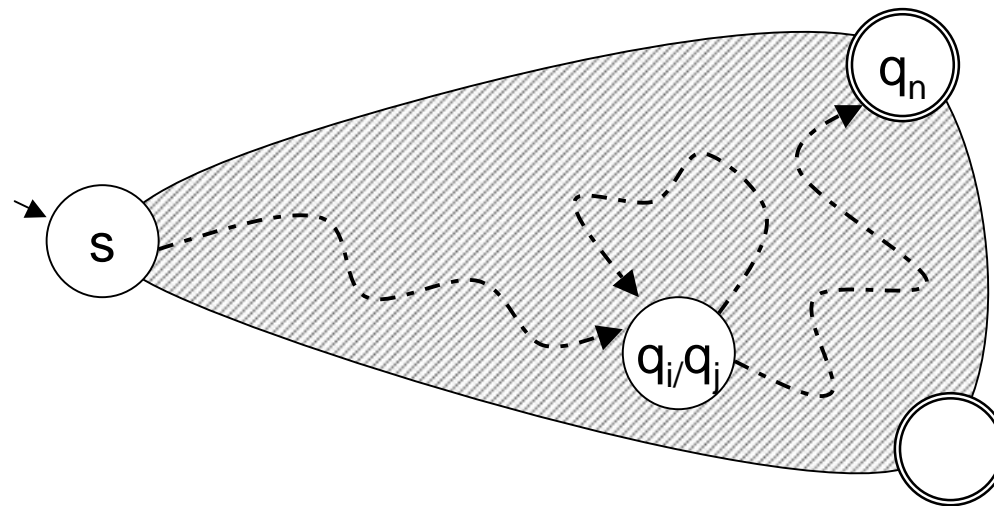


Syntaktische Verarbeitung

- Syntax (Konstituentenstruktur) natürlicher Sprachen kann einfach und elegant durch kontextfreie Grammatiken dargestellt werden.
- Syntax (Konstituentenstruktur) natürlicher Sprachen kann durch endliche Automaten nicht verarbeitet werden. - Pumping Lemma: Das „Gedächtnis“ des Automaten reicht nicht aus, um tief geschachtelte Strukturen zu analysieren.

Das "Pumping Lemma,, [2]

Sei $w = a_1 \dots a_n$ ein Wort, das vom DEA K mit k Zuständen erkannt wird, und $n = |w| \geq k$. Dann geht der Pfad vom Startzustand $q_0 = s$ zu einem Endzustand q_n , auf dem w gelesen wird, durch insgesamt $n+1$ Zustände. Das heißt, dass mindestens zwei Zustände q_i und q_j identisch sein müssen.

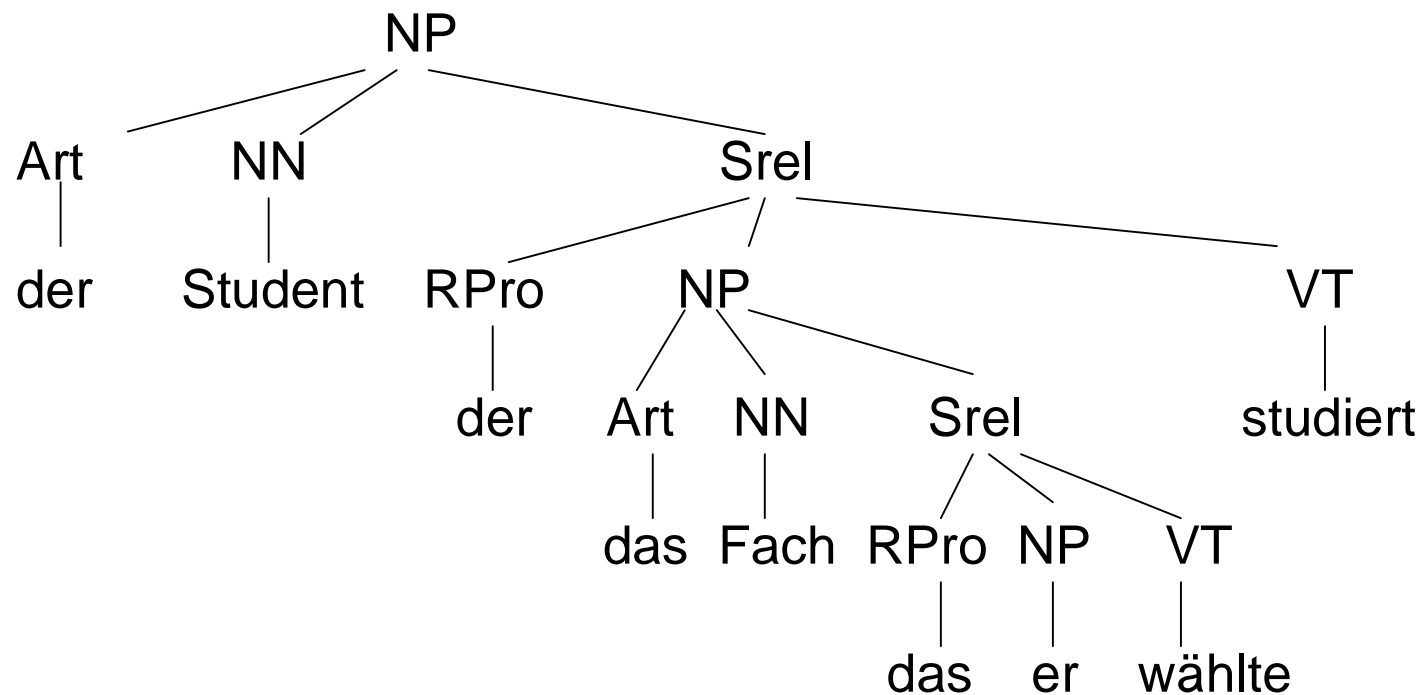


Geschachtelte Strukturen in natürlicher Sprache

*[_{NP} der an computerlinguistischen Fragestellungen
interessierte Student im ersten Semester,
[_{SRel} der [_{NP} das Fach,
[_{SRel} das [_{NP} er] nach langer Überlegung gewählt hat]],
eifrig studiert]]*

Geschachtelte Strukturen in natürlicher Sprache

[_{NP} der an computerlinguistischen Fragestellungen interessierte Student im ersten Semester, [_{SRel} der [_{NP} das Fach, [_{SRel} das [_{NP} er] nach langer Überlegung gewählt hat]], eifrig studiert]]

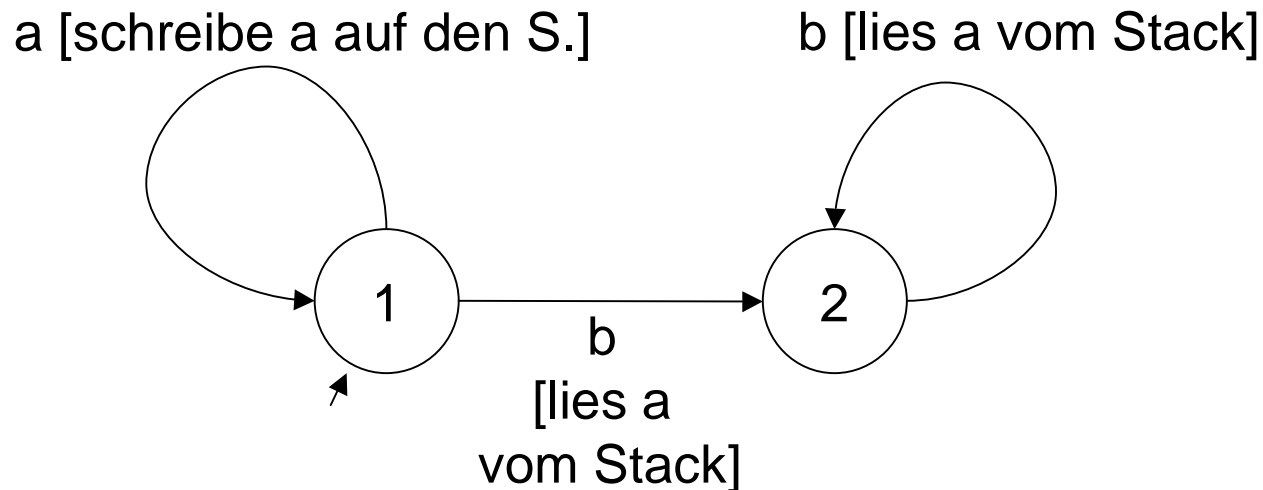


Syntaktische Verarbeitung

- Syntax (Konstituentenstruktur) natürlicher Sprachen kann einfach und elegant durch kontextfreie Grammatiken dargestellt werden.
- Natürliche Sprachen sind nicht regulär. Konstituentenstruktur kann durch endliche Automaten nicht verarbeitet werden. - Pumping-Lemma: Das „Gedächtnis“ des Automaten reicht nicht aus, um tief geschachtelte Strukturen zu analysieren.
- Der „Kellerautomat“ („push-down automaton“, PDA) besitzt zusätzlich zur endlichen Menge von Zuständen einen (unbegrenzten) linearen Speicher, der als Stapel (Stack, Keller, push-down store) verwendet wird.
- Ein Übergang beinhaltet nicht nur einen Zustandswechsel, sondern auch einen Lese- und/oder Schreibvorgang auf dem Stack.

Ein Beispiel

- Ein Kellerautomat, der $a^n b^n$ akzeptiert:



- Im Zustand 1 werden a 's gelesen und in den Stack geschrieben. Beim ersten b wechselt der Automat in den Zustand 2 und löscht für jedes gelesene b ein a vom Stack. Eingabe und Stack sind beide leer, wenn die Eingabe gleich viele a 's und b 's enthielt.

Keller-Automat: Definition

$A = \langle K, V, \Sigma, \Delta, s, F \rangle$, wobei

- $K \neq \emptyset$ endliche Menge von Zuständen
- V Alphabet
- $\Sigma \subseteq V$ nicht-leeres terminales Alphabet
- $s \in K$ Startzustand
- $F \subseteq K$ Menge von Endzuständen
- $\Delta \subseteq (K \times \Sigma^* \times V^*) \times (K \times V^*)$ Übergangsrelation

$\langle \langle q, u, v \rangle, \langle q', v' \rangle \rangle$: „Im Zustand p lies u vom Eingabestring und v vom Stack, schreibe v' auf den Stack und gehe in den Zustand q' .“

Keller-Automat: Ein Beispiel

Ein Keller-Automat, der die Sprache $a^n b^n$ akzeptiert:

$$A = \langle \{1,2\}, \{a,b\}, \{a,b\}, \Delta, 1, \{1,2\} \rangle$$

$$\Delta = \{ \langle \langle 1, a, \varepsilon \rangle, \langle 1, a \rangle \rangle, \\ \langle \langle 1, b, a \rangle, \langle 2, \varepsilon \rangle \rangle, \\ \langle \langle 2, b, a \rangle, \langle 2, \varepsilon \rangle \rangle \}$$

Keller-Automat

- Im Startzustand ist der Speicher des PDA leer.
- Ein Eingabe-Wort w wird vom PDA A akzeptiert, wenn sich A nach dem Abarbeiten von w bei leerem Speicher in einem Endzustand befindet.

Keller-Automaten und kontextfreie Grammatiken

- Kontextfreie Grammatiken (CFGs) erlauben die einfache und elegante Darstellung syntaktischer Information.
- Keller-Automaten erlauben die Verarbeitung von Konstituentenstruktur.
- Frage: Wie können CFGs für die Verarbeitung verwendet werden?

Top-Down-Parser: Ein Beispiel

$G = \langle \{a,b,S\}, \{a,b\}, \{ S \rightarrow aSb, S \rightarrow \varepsilon \}, S \rangle$
generiert $a^n b^n$.

$A = \langle \{0,1\}, \{a,b,S\}, \{a,b\}, \Delta, 0, \{1\} \rangle$ mit

$$\Delta = \{ \langle \langle 0, \varepsilon, \varepsilon \rangle, \langle 1, S \rangle \rangle, \\ \langle \langle 1, \varepsilon, S \rangle, \langle 1, aSb \rangle \rangle, \\ \langle \langle 1, \varepsilon, S \rangle, \langle 1, \varepsilon \rangle \rangle, \\ \langle \langle 1, a, a \rangle, \langle 1, \varepsilon \rangle \rangle, \\ \langle \langle 1, b, b \rangle, \langle 1, \varepsilon \rangle \rangle \}$$

erkennt $a^n b^n$.

Top-Down-Parser: Konstruktion

- Sei $G = \langle V, T, P, S \rangle$ eine kontextfreie Grammatik.
- Sei $A = \langle \{0,1\}, V, T, \Delta, 0, \{1\} \rangle$ Keller-Automat mit
$$\Delta = \{ \langle \langle 0, \varepsilon, \varepsilon \rangle, \langle 1, S \rangle \rangle \} \cup$$
$$\{ \langle \langle 1, \varepsilon, A \rangle, \langle 1, u \rangle \rangle \mid A \rightarrow u \in P \} \cup$$
$$\{ \langle \langle 1, a, a \rangle, \langle 1, \varepsilon \rangle \rangle \mid a \in T \}$$
- A akzeptiert die von G erzeugte Sprache.

Beispielgrammatik

$S \rightarrow NP\ VI$ $NP \rightarrow ART\ NN$

$S \rightarrow NP\ VT\ NP$ $NP \rightarrow PN$

$VI \rightarrow$ *schläft, arbeitet*

$VT \rightarrow$ *studiert, liest*

$NN \rightarrow$ *Student, Dozentin, Buch*

$ART \rightarrow$ *der, die, das*

$PN \rightarrow$ *Peter, Maria*

Bottom-Up-Parser: Ein Beispiel

$G = \langle \{a,b,S\}, \{a,b\}, \{ S \rightarrow aSb, S \rightarrow \varepsilon \}, S \rangle$
generiert $a^n b^n$.

$A = \langle \{0,1\}, \{a,b,S\}, \{a,b\}, \Delta, 0, \{1\} \rangle$ mit

$$\Delta = \{ \langle \langle 0, a, \varepsilon \rangle, \langle 0, a \rangle \rangle, \\ \langle \langle 0, b, \varepsilon \rangle, \langle 0, b \rangle \rangle, \\ \langle \langle 0, \varepsilon, bSa \rangle, \langle 0, S \rangle \rangle, \\ \langle \langle 0, \varepsilon, \varepsilon \rangle, \langle 0, S \rangle \rangle, \\ \langle \langle 0, \varepsilon, S \rangle, \langle 1, \varepsilon \rangle \rangle \}$$

erkennt $a^n b^n$.

Bottom-Up-Parser: Konstruktion

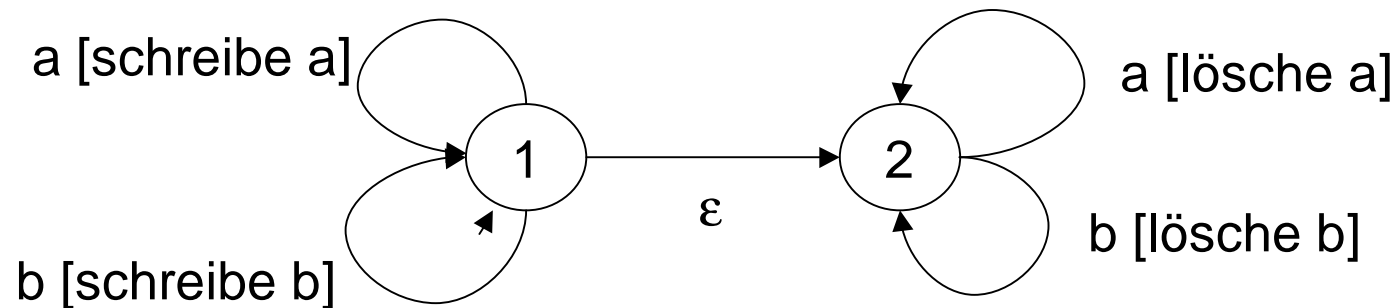
- Sei $G = \langle V, T, P, S \rangle$ eine kontextfreie Grammatik.
- Sei $A = \langle \{0,1\}, V, T, \Delta, 0, \{1\} \rangle$ Keller-Automat mit
$$\Delta = \{ \langle \langle 1, a, \varepsilon \rangle, \langle 1, a \rangle \rangle \mid a \in T \} \cup$$
$$\{ \langle \langle 1, \varepsilon, u^R \rangle, \langle 1, A \rangle \rangle \mid A \rightarrow u \in P \} \cup$$
$$\{ \langle \langle 0, \varepsilon, S \rangle, \langle 1, \varepsilon \rangle \rangle \}$$
- A akzeptiert die von G erzeugte Sprache.

Deterministische syntaktische Verarbeitung?

- Gibt es eine generelle Möglichkeit, kontextfreie Grammatiken (analog zum NEA-DEA-Überführungsalgorithmus) in ein Format zu übersetzen, das deterministisches Parsing erlaubt?
- Die Antwort ist Nein. Viele kontextfreie Sprachen sind inhärent nicht-deterministisch.

Ein einfaches Beispiel

- Die Sprache $L = \{ww^R \mid w, w^R \in \{a,b\}^*\}$ kann nicht deterministisch geparkt werden (w^R ist die Spiegelung von w).



- Problem: Woher weiß der Automat, dass er die Mitte des Wortes erreicht hat? Er muss raten, und wenn er falsch geraten hat, zurücksetzen und eine andere Option verfolgen.

Effiziente Verarbeitung

Der Detektiv verfolgte den Gangster im Sportwagen

*[_{NP} der an computerlinguistischen Fragestellungen
interessierte Student im ersten Semester,
[_{SRel} der [_{NP} das Fach,
[_{SRel} das [_{NP} er] nach langer Überlegung gewählt hat]],
eifrig studiert]]*

Chart-Parsing und Earley-Algorithmus

- Initialisierung:
 - Trage Eingabekette der Länge n in eine Chart ein, deren Knoten von 0 bis n nummeriert sind.
 - Trage neue aktive Kante $\# \rightarrow . S$ $[0,0]$ ein.

Chart-Parsing und Earley-Algorithmus

- Predictor:
 - Für aktive Kanten $A \rightarrow \beta. B \gamma [i,j]$, B nicht-lexikalische Kategorie, trage neue aktive Kanten $B \rightarrow .\delta [j,j]$ für jede Regel der Grammatik $B \rightarrow \delta$ ein.
- Scanner:
 - Für aktive Kanten $A \rightarrow \beta. A \gamma [i,j]$, A lexikalische Kategorie, $A \rightarrow a$ Regel, und a ist das $j+1$ -te Eingabewort, trage $A \rightarrow a. [j,j+1]$ ein.
- Completer:
 - Für komplette Kanten $B \rightarrow \delta. [j,k]$, aktive Kanten $A \rightarrow \beta. B \gamma [i,j]$, trage $A \rightarrow \beta B. \gamma [i,k]$ ein.