Einführung in die Computerlinguistik

Syntax

WS 2005/2006

Manfred Pinkal

Eigenschaften der syntaktischen Struktur [3]

- 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.

Syntaktische Kategorien

Konstituenten-Typen werden „syntaktische Kategorien“ genannt;

Beispiele:

- Nominalphrasen (Nominalausdrücke): *er – der Student – der interessierte Student – die Übungen – computerlinguistischen Fragestellungen*
- Präpositionalphrasen (Präpositionalausdrücke): *an computerlinguistischen Fragestellungen – im ersten Semester, – nach langer Überlegung*
- Adjektivphrasen: *interessierte – an computerlinguistischen Fragestellungen interessierte*
- Satz: Haupt- und Nebensätze unterschiedlicher Art

Kategorie und Funktion

- Die syntaktische Kategorie ergibt sich aus dem internen Aufbau einer Konstituente, insbesondere aus der Wortart ihres „lexikalischen Kopfes“: Die Konstituenten
  - *der an computerlinguistischen Fragestellungen interessierte Student*
  - *an computerlinguistischen Fragestellungen interessiert*
  - *an computerlinguistischen Fragestellungen*

  sind Nominal-, Adjektiv- und Präpositionalphrase, weil der jeweilige Kopf Substantiv („Nomen“), Adjektiv, bzw. Präposition ist.
- Die grammatische Funktion dagegen bezeichnet die Rolle, die eine Konstituente im ganzen Satz spielt. Ein Nominalausdruck z.B. kann, je nach Stellung im Satz unter anderem die Funktion von Subjekt, (direktem oder indirektem) Objekt, (Genitiv-) Attribut oder Prädikatsnomen besitzen.

Eigenschaften der syntaktischen Struktur [3]

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

Interaktion von grammatischen Merkmalen

- Kongruenz: bestimmte grammatische Merkmale verschiedener Ausdrücke müssen übereinstimmen
- Rektion: ein Ausdruck (lexikalischer Kopf) verlangt das Vorliegen bestimmter Merkmale bei abhängigen Ausdrücken
- Beispiele für Kongruenz:
  - in Nominalausdrücken (Numerus, Genus, Kasus)

    der [sg,nom,m]interessierte [sg,nom,m] Student [sg,nom,m]

    dem[sg,dat,m] interessierten[sg,dat,m] Studenten[sg,dat,m]
  - Subjekt-Verb-Kongruenz (Person, Numerus)

    Der Student[sg] arbeitet[sg] / Die Studenten[sg] arbeiten[sg]
- Beispiele für Rektion (Kasus)
  - *die Übungen [akk] abgeben / der Dozentin [dat] zuhören*
  - *mit den Kollegen [dat] / ohne die Kollegen [akk]*
  - *dem Fach [dat] zugetan / des Faches [gen] überdrüssig*

Ein komplexerer Fall: Relativpronomen

Das Relativpronomen bekommt seine grammatischen Merkmale durch zwei unterschiedliche Mechanismen zugewiesen:

- Numerus , Genus-Kongruenz mit dem Kopf der Nominalphrase
- Kasusrektion durch das Hauptverb des Relativsatzes

der Student [m,sg], den[m,sg,akk] die Computerlinguistik interessiert
der Student [m,sg], dem [m,sg,dat] die Computerlinguistik gefällt
die Studentin [f,sg], die [f,sg,akk] die Computerlinguistik interessiert
die Studentin [f,sg], der [f,sg,dat] die Computerlinguistik gefällt

Eigenschaften der syntaktischen Struktur [3]

Nominalkongruenz

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

Eigenschaften der syntaktischen Struktur [3]

Pronominalkongruenz

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

Eigenschaften der syntaktischen Struktur [3]

Subjekt-Verb-Kongruenz

- *Der [m,sg, nom]an computerlinguistischen Fragestellungen interessierte [m,sg, nom] Student [m,3, sg, nom] im ersten Semester, der [m,3, sg, nom] im Hauptfach, für das er [m,3,sg, nom] sich nach langer Überlegung entschieden hat [3,sg], Informatik studiert [3, sg], hat [3, sg]die Übungen gemacht.*

Grammatik natürlicher Sprachen und endliche Automaten

- Natürliche Sprachen sind Sprachen im Sinne der Automatentheorie:
  - Das "Alphabet" ist das (eventuell sehr große) Lexikon
  - "Worte" über dem Alphabet sind Folgen von Wörtern aus dem Lexikon
  - Die deutsche Sprache lässt sich formal charakterisieren als die Menge der Worte über dem Alphabet (also der Folgen von Wörtern im Lexikon), die grammatisch korrekte Sätze des Deutschen darstellen.
- Frage: Lassen sich natürliche Sprachen in diesem Sinn durch endliche Automaten beschreiben?
- Terminologie: Sprachen, die von einem endlichen Automaten erkannt werden, heißen auch "reguläre Sprachen". Die Frage ist also, kurz gefasst: Sind natürliche Sprachen regulär?

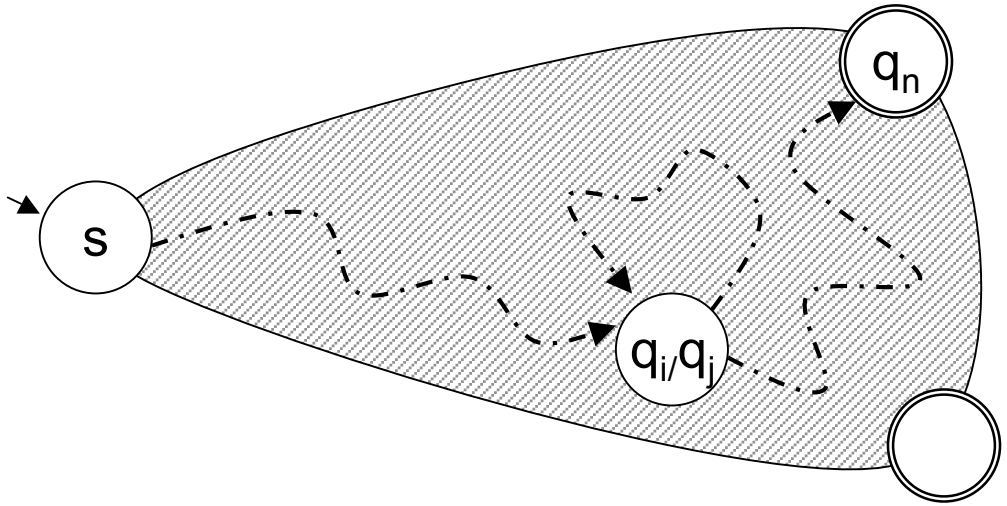
Lässt sich syntaktische Struktur mit endlichen Automaten beschreiben?

- Überlegung: DEA und NEA (und reguläre Ausdrücke) sind äquivalente Formalismen, die komplizierte Sprachstrukturen beschreiben können und, wegen der garantierten Überführbarkeit in DEAs, in linearer Zeit analysieren. Es wäre wünschenswert, wenn wir das Instrumentarium auch auf syntaktische Strukturen anwenden könnten. Frage: Können wir nicht nur die Morphologie, sondern auch die Syntax natürlicher Sprachen durch endliche Automaten analysieren?

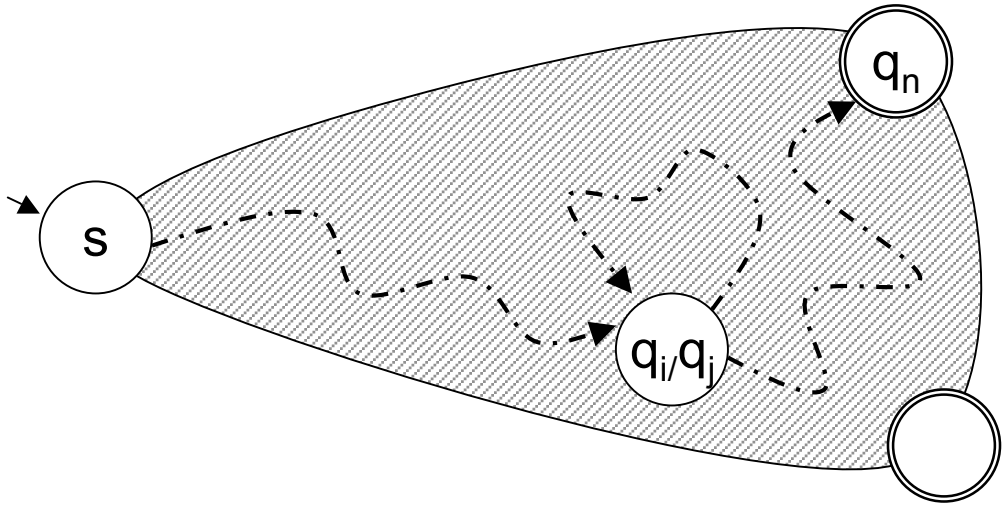
Das "Pumping Lemma„ [1]

- Das Pumping Lemma (dt. „Pump-Lemma“ oder auch „uvw-Theorem“) gibt uns, anders als das vorher gezeigte Theorem, die Möglichkeit, konstruktiv zu beweisen, dass eine Sprache nicht regulär ist.
- Der Grundgedanke ist einfach: Jede reguläre Sprache L wird von einem endlichen Automaten akzeptiert. Ein endlicher Automat hat eine bestimmte, endliche Anzahl von Zuständen. Wenn ein Wort in L mehr Symbole hat als der Automat Zustände (genau genommen reichen schon mindestens soviele Symbole aus), dann muss beim Abarbeiten des Wortes ein Zustand mindestens zweimal vorkommen. Das bedeutet aber, dass beim Abarbeiten eine Schleife durchlaufen wird. Die könnte im Prinzip aber auch mehrfach durchlaufen (oder weggelassen) werden. Das heißt, für Wörter oberhalb einer bestimmten Länge durch beliebige Wiederholung eines Teilworts „aufgepumpt“ werden können und immer noch zur Sprache gehören.

Das "Pumping Lemma"



Das "Pumping Lemma"



Sei w= a_1 ... a_n ein Wort, das vom DEA K mit k Zuständen erkannt wird, und n=|w| ≥ 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.

Das "Pumping Lemma„ [3]

Präzise Formulierung des Lemmas:

- Wenn L eine Sprache ist, die durch einen endlichen Automaten beschrieben werden kann ("reguläre Sprache"), dann gilt:

  Wenn ein Wort $x \in L$ eine bestimmte Länge k erreicht oder überschreitet ($|x| \geq k$), dann läßt sich x so in drei Teile u, v und w zerlegen (mit $|v| \geq 1$), daß mit $x=uvw$ auch jedes $x' = uv^iw$ ($i=0$ oder $i>1$) Element von L ist.

- Um zu zeigen, dass eine Sprache L nicht regulär ist, genügt es, zu zeigen, dass L ausreichend lange Worte enthält. deren Teile nicht beliebig iterierbar sind.

Eine nicht reguläre Sprache [1]

Die Sprache L = { $a^n b^n$ | n∈N} (kurz:" $a^n b^n$") wird nicht von einem endlichen Automaten akzeptiert.

Beweis:

- Angenommen, die Sprache L = { $a^n b^n$ | n∈N} (kurz:" $a^n b^n$") wird von einem endlichen Automaten akzeptiert. Nach dem Pumping Lemma gibt es dann eine Zahl k, so daß für jedes Wort x mit |x| ≥k eine Zerlegung in u, v und w möglich ist, so daß uw, uvvw, uvvvw, … ebenfalls in L sind.

- Betrachten wir das Wort $a^k b^k$. Es gilt $|a^k b^k| \geq k$, das Wort muss also einen "duplizierbaren" Teil v besitzen. Um welchen Teil könnte es sich handeln?

Eine nicht reguläre Sprache [2]

- Drei Fälle sind denkbar:
  - Fall1: v liegt vollständig in der ersten Hälfte des Wortes, besteht also nur aus a's. Dann müsste gelten, dass $uv^2w = a^{k+|v|}b^k \in L$: wegen $k+|v| \neq k$ unmöglich.
  - Fall2: v liegt vollständig in der zweiten Hälfte des Wortes, besteht also nur aus b's. Dann müsste gelten, dass $uv^2w = a^kb^{k+|v|} \in L$: wegen $k+|v| \neq k$ unmöglich.
  - Fall 3: v überspannt die Mitte des Wortes, hat also die Form a^lb^m. Dann müsste gelten, dass $uv^2w = a^kb^ma^lb^k \in L$. Geht nicht, da a's auf b's folgen.
- Es gibt also für a^kb^k keine Zerlegung in uvw mit duplizierbarem Mittelteil. Also ist $L= a^nb^n$ nicht regulär.

Was steckt hinter dem Pumping Lemma?

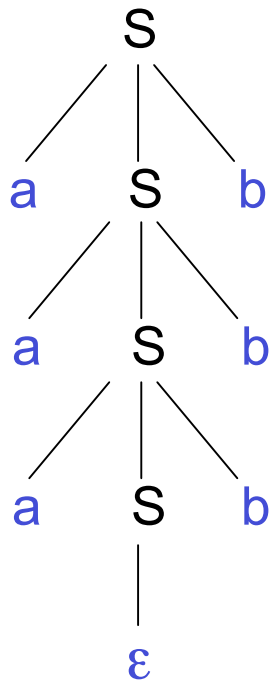
Endliche Automaten haben eine fundamentale Einschränkung: Ihr „Gedächtnis“ ist endlich, durch die Anzahl ihrer Zustände beschränkt. Ein Automat mit k Zuständen kann sich nur an einen beschränkten Kontext „erinnern“: nach spätestens k gelesenen Symbolen ist er wieder in dem Zustand, den er vorher gehabt hat. Noch anders ausgedrückt: Der Automat kann nur bis k zählen. Um zum Beispiel $a^n b^n$ zu erkennen, müsste er sich aber beliebig lange Ketten von a‘s merken können, weil er die Information anschließend beim Abarbeiten von b‘s braucht.

Kontextfreie Grammatiken: Ein Formalismus zur Beschreibung nicht-regulärer Sprachen

- $S \rightarrow aSb$, $S \rightarrow \varepsilon$ sind Produktionsregeln (Ersetzungsregeln)
- Beginne mit S (Startsymbol). Wenn in einer Kette das Symbol S vorkommt, ersetze es durch aSb (Regel 1) oder die leere Kette (Regel 2).
- Ersetze solange, bis nur noch a‘s und b‘s vorkommen („Terminalsymbole“)
- Eine gültige Ableitung der Beispielgrammatik:
  $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$
- Die Beispielgrammatik erzeugt die Sprache $a^n b^n$

Kontextfreie Grammatiken: Ein Beispiel

- Das Wort aaabbb wird durch
  $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$
  abgeleitet.
- Die Ableitung kann durch den folgenden Ableitungsbaum dargestellt werden:
- Die Blätter des Baums ergeben, von links nach rechts gelesen, das abgeleitete Wort.
- Eine weitere alternative Schreibweise:
  $[_S\ a[_S\ a[_S\ a[_S\ \varepsilon\ ]\ b]\ b]\ b]$
  Kontextfreie Grammatiken ermöglichen die Darstellung beliebig tief geschachtelter Strukturen.
- „Ableitungsbaum“ heißt auch „Parse-Tree“: Parsing ist automatische syntaktische Analyse



Kontextfreie Grammatik: Formale Definition

G = <V, Σ, P, S>, wobei

- V nicht-leere Menge von Symbolen
- $\Sigma \subseteq V$ nicht-leere Menge von Terminalsymbolen
- $P \subseteq (V- \Sigma) \times V^*$ nicht-leere Menge von Produktionsregeln
- $S \in V- \Sigma$ das Startsymbol

Die Beispielgrammatik in alternativer Notation:

- G1 = < {a,b,S}, {a,b}, {<S, aSb>, <S, ε>}, S >
- Für <A, α> ∈ P schreibt man üblicherweise $A \rightarrow \alpha$.

- Ableitungsschritt: Wenn w ein nicht-terminales Symbol A enthält und $A \rightarrow \alpha$ Produktion ist, ersetze A in w durch α.

- Die durch G erzeugte Sprache ist die Menge aller Worte über Σ^*, die sich aus S ableiten lassen.

Geschachtelte Strukturen in natürlicher Sprache

- Der Nominalausdruck *„der an computerlinguistischen Fragestellungen interessierte Student im ersten Semester, der im Hauptfach, für das er sich nach langer Überlegung entschieden hat, Informatik studiert“* enthält
- den (Relativ-)Satz „ *der im Hauptfach, für das er sich nach langer Überlegung entschieden hat, Informatik studiert*“; der enthält
- den Nominalausdruck „*Hauptfach, für das er sich nach langer Überlegung entschieden hat“;* der enthält
- den (Relativ-)Satz „ *für das er sich nach langer Überlegung entschieden hat*“; der enthält
- unter anderem den Nominalausdruck *„er“.*

Syntaktische Verarbeitung

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

Eine kontextfreie Grammatik für deutsche Sätze

S → NP VI
S → NP VT NP
NP → ART NN
VI → *schläft*
VI → *arbeitet*
VT → *studiert*
VT → *wählt*
ART → *der*

NP → ART NN SREL
SREL → RPRO VI
SREL → RPRO NP VT
NN → *Student*
NN → *Fach*
RPro → *der*
RPro → *das*
ART → *das*

- Kontextfreie Grammatiken heißen in der Linguistik auch Konstituentenstruktur-Grammatiken oder Phrasenstruktur-Grammatiken

Eine kontextfreie Grammatik für deutsche Sätze

- Schreibkonventionen: Klammernotation für optionale Konstituenten, Reihung von lexikalischen Alternativen

S → NP VI

SREL → RPRO VI

SREL → RPRO NP VT

NP → ART NN (SREL)

S → NP VT NP

VI → *schläft, arbeitet, wartet, ...*

VT → *wählt, studiert, liest, kennt, ...*

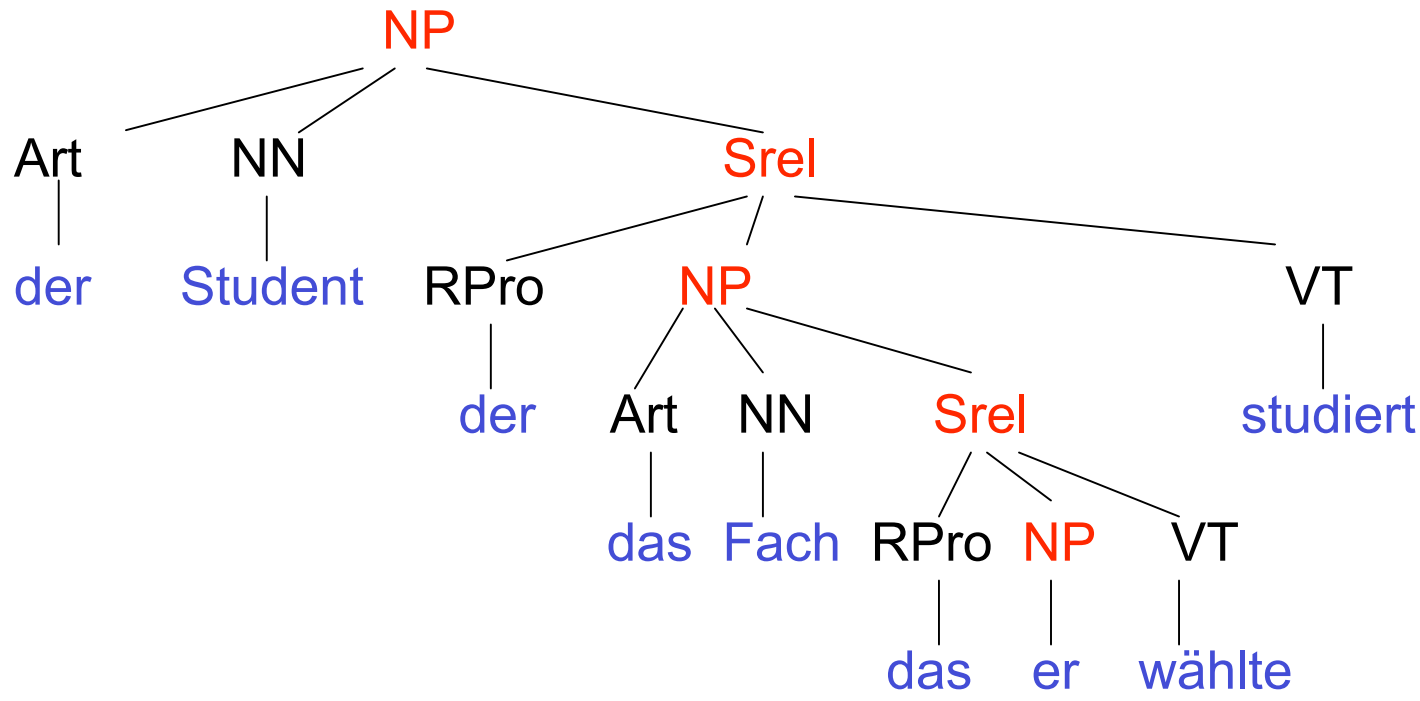
NN → *Student, Fach, Dozentin, Professor, Buch, ...*

RPro → *der, die, das*

ART → *der, die, das*

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*]]



Konstituentenstruktur

- Die Erzeugung eines Satzes mit einer kontextfreien Grammatik (und die Verarbeitung des Satzes mit einem Parser, der die Grammatik in ein Analysesystem umsetzt) liefert nicht nur die Information, ob der Satz zur Sprache gehört/"grammatisch ist". Als "Seiteneffekt" liefert sie den Ableitungsbaum/"Parse Tree", der die "Konstituentenstruktur" und damit einen wesentlichen teil der syntaktischen Struktur beschreibt.
- Eine Grammatik ist eine angemessene Beschreibung der syntaktischen Struktur einer Sprache, wenn sie
  - genau die grammatisch akzeptablen Sätze der Sprache erzeugt, und außerdem
  - für die grammatischen Sätze plausible Konstituenten-strukturen erzeugt.

Grammatische Mehrdeutigkeit

- Natürlich-sprachliche Sätze sind oft grammatisch mehrdeutig. Die grammatische Mehrdeutigkeit ist relevant. Sie führt im allgemeinen zu semantischer Mehrdeutigkeit.

Grammatische Mehrdeutigkeit, Beipiele 1

- Die folgende Grammatik erlaubt Präpositionalphrasen (PPs) als VP- (Verbphrasen-) und als NP-Modifikatoren

S → NP VP	VP → VT NP	VP → VI
NP → ART NN	NP → PN	NP → NP PP
PP → P NP	VP → VP PP	

- Der Satz: *Peter sah den Mann mit dem Teleskop* hat in dieser Grammatik zwei Analysen, die zwei Bedeutungsvarianten entsprechen:
  - [$_{S}$ *Peter* [$_{VP}$ *sah* [$_{NP}$ [$_{NP}$ *den Mann*] [$_{PP}$ *mit dem Teleskop*]]]]
  - [$_{S}$ *Peter* [$_{VP}$ [$_{VP}$ *sah* [$_{NP}$ *den Mann*]] [$_{PP}$ *mit dem Teleskop*]]]

- Der Satz *Peter benachrichtigte den Kollegen aus München mit dem Handy* hat mindestens 4 Lesarten (welche?).

Grammatische Mehrdeutigkeit

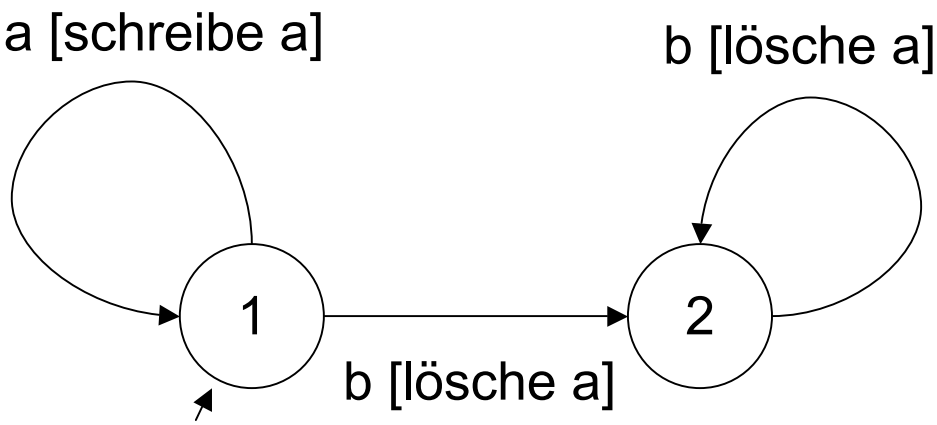
- Natürlich-sprachliche Sätze sind oft grammatisch mehrdeutig. Die grammatische Mehrdeutigkeit ist relevant. Sie führt im allgemeinen zu semantischer Mehrdeutigkeit.
- Für natürliche Sprachen benötigen wir deshalb Grammatiken, die Sätzen unterschiedliche syntaktische Strukturen zuweisen können.

Syntaktische Verarbeitung

- Kontextfreie Grammatiken erlauben die einfache und elegante Beschreibung syntaktischer Strukturen (ähnlich wie Zustandsdiagramme/deterministische endliche Automaten das für morphologische Sturkturen tun).
- Wie sehen Analysesysteme für kontextfreie Sprachen aus (syntaktische Parser)?
- Das einfachste Modell ist der "Kellerautomat": Wir nehmen einen Automaten an, der zusätzlich zu Zuständen und Übergängen über einen Speicher als Stapel (Stack, Keller) verfügt. Der Automat liest Symbole der Eingabekette und kann bei jedem Übergang Symbole in den Stack schreiben oder vom Stack nehmen. Ein Eingabe wird akzeptiert, wenn ein Endzustand erreicht, das Eingabewort abgearbeitet, und der Stack leer ist.

Ein Beispiel

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



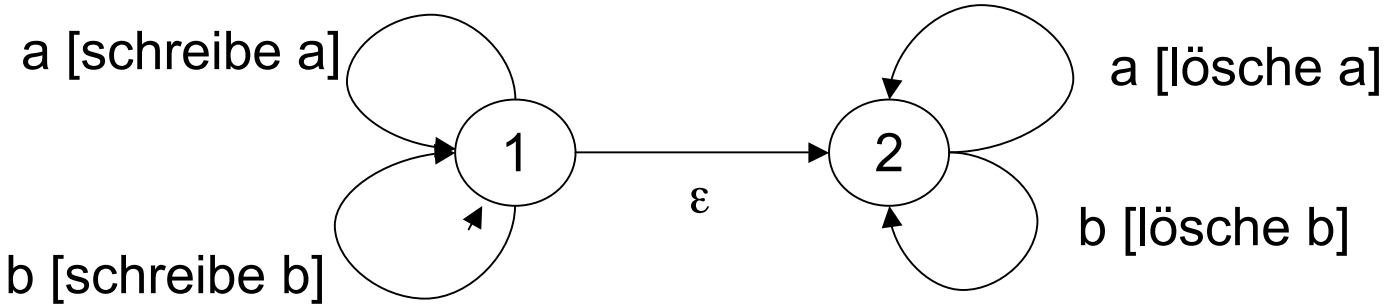
- Im Zustand 1 werden a's gelesen und in den Stack geschrieben. Beim ersten b wechselt der Automat in den Zustand 2 und löscht für jedes gelesene b ein a vom Stack. Eingabe und Stack sind beide leer, wenn die Eingabe gleichviele a's und b's enthielt. (Beide Zustände sind Endzustände.)

Deterministische syntaktische Verarbeitung?

- Der Automat aus Beispiel 1 ist deterministisch: Er führt in jedem Zustand bei jedem Eingabesymbol eine bestimmte Aktion aus, benötigt deshalb keine aufwändigen Suchverfahren und analysiert jede Eingabe in linearer Zeit, ganz wie ein DEA,
- 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 (dazu auch das Beispiel auf der nächsten Folie).

Ein einfaches Beispiel

- Die Sprache L = {ww^R | w, $w^R \in \{a,b\}^*$} ist eine kontextfreie Sprache (w^R ist die Spiegelung von w): Sie wird generiert durch die Grammatik G mit den Regeln S → aSa, S → bSb, S → ε .
- L wird durch den folgenden Automaten erkannt:



- Bis zur Mitte des Eingabewortes speichert er a's und b's im Stack, dann springt er in den zweiten Zustand, in dem er neu gelesene Symbole mit den gespeicherten Stack-Symbolen vergleicht. Der Automat kann nicht wissen, ob er die Mitte des Wortes erreicht hat. Er muss raten, und wenn er falsch geraten hat, zurücksetzen und eine andere Option verfolgen.

(Nicht-)Deterministische Grammatiken

- Programmiersprachen, Sprachen der Logik und Mathematik sind meist so konzipiert, dass sie deterministische Analyse zulassen; eine einfache Möglichkeit, Nicht-Determinismus zu vermeiden, besteht in der Verwendung von Klammern.
- Natürliche Sprachen lassen keine deterministische Analyse zu, sie erfordern nicht-lineare Suchverfahren /Parsingalgorithmen (deren Zeitbedarf im besten Fall mit der Länge der Eingabe quadratisch wächst).

Grundaufgaben der grammatischen Verarbeitung

- Effiziente Verarbeitungsverfahren für kontextfreie Grammatiken/ Konstituentenstruktur (Parsing-Techniken)
- Modellierung von Merkmalsabhängigkeiten und Wortstellungsvariation
- Entwicklung von Grammatiken für Einzelsprachen mit großer Abdeckung