Einführung in die Computerlinguistik

Syntax I: Kontextfreie Grammatiken

WS 2010/2011

Manfred Pinkal

Morphologie und Syntax

- Gegenstand der Morphologie ist die Struktur des Wortes: der Aufbau von Wörtern aus Morphemen, den kleinsten funktionalen oder bedeutungstragenden Einheiten der Sprache.
- Gegenstand der Syntax ist die Struktur des Satzes: der Aufbau von Sätzen aus Wörtern.
- Morphologie beschreibt die grammatischen Eigenschaften von Wörtern, die durch Wortform oder Flexionsmorpheme kodiert werden.
- Syntax beschreibt die Interaktion der grammatischen Eigenschaften unterschiedlicher Wörter im Satz.

Eigenschaften der syntaktischen Struktur [1]

- *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.*

Eigenschaften der syntaktischen Struktur [2]

Peter hat der Dozentin das Übungsblatt heute ins Büro gebracht.

Das Übungsblatt hat Peter der Dozentin heute ins Büro gebracht.

Der Dozentin hat Peter heute das Übungsblatt ins Büro gebracht.

Ins Büro hat heute Peter der Dozentin das Übungsblatt gebracht.

Heute hat Peter das Übungsblatt der Dozentin ins Büro gebracht.

?Ins Büro hat das Übungsblatt der Dozentin Peter heute gebracht.

** Ins Büro heute Peter das Übungsblatt hat gebracht der Dozentin.*

** Ins heute Büro der Peter Dozentin das hat Übungsblatt gebracht.*

Eigenschaften der syntaktischen Struktur [3]

- *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.*

Eigenschaften der syntaktischen Struktur [3]

- Wie finden Sie stattdessen die angehängten Bilder? Das sind Fotos, die im Rahmen des TALK-Projektes entstanden sind, uns gehören, und von BMW schon freigegeben waren. Außerdem vermitteln sie besser den Bezug zur Forschung.

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.

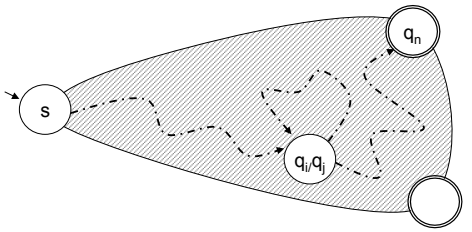
Syntax natürlicher Sprachen und endliche Automaten

- Natürliche Sprachen sind Sprachen im Sinne unserer Definitionen zur Automatentheorie:
  - Das "Alphabet" Σ 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 Sprache/ Wortmenge L 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? Gibt es einen NEA/DEA A, sodass L(A) genau die grammatisch korrekten Sätze der deutschen Sprache enthält?
- Sprachen, die von einem endlichen Automaten erkannt werden, heißen auch "reguläre Sprachen". Die Frage ist also, kurz gefasst: Sind natürliche Sprachen regulär?

Sind natürliche Sprachen regulär?

- Wir zeigen, dass Sprachen, die von einem endlichen Automaten akzeptiert werden, eine bestimmte Eigenschaft haben müssen (das sog. Pumping Lemma, dt. „Pump-Lemma“ oder auch „uvw-Theorem“).
- Wir zeigen für eine bestimmte Sprache, dass sie diese Eigenschaft nicht besitzt: Es gibt nicht-reguläre Sprachen, und sie lassen sich einfach beschreiben.
- Wir argumentieren, dass natürliche Sprachen wie das Deutsche sich ebenso verhalten.

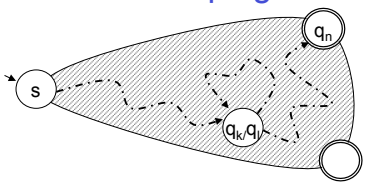
Das Pumping Lemma



Pumping Lemma: Grundgedanke

- Der Grundgedanke ist einfach:
  - Jede reguläre Sprache L wird von einem endlichen Automaten akzeptiert. Ein endlicher Automat hat eine bestimmte, endliche Anzahl von Zuständen.
  - Endliche Sprachen sind immer regulär. Unendliche Sprachen enthalten Worte beliebiger Länge.
  - 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, dass beim Abarbeiten eine Schleife durchlaufen wird.
  - Die kann aber auch mehrfach durchlaufen oder weggelassen werden.
  - Das heißt, dass Wörter oberhalb einer bestimmten Länge durch beliebige Wiederholung eines Teilworts „aufgepumpt“ (bzw. gekürzt) werden können und immer noch zur Sprache gehören.

Das Pumping Lemma



- Sei w= a1 ... an ein Wort, das vom DEA K mit k Zuständen erkannt wird, und n=|w| ≥ k. Dann geht der Pfad vom Startzustand q0=s zu einem Endzustand qn, auf dem w gelesen wird, durch insgesamt n+1 Zustände. Das heißt, dass mindestens zwei Zustände q_k und q_l identisch sein müssen.
- Wenn L regulär ist/ durch einen endlichen Automaten definiert wird, dann gilt:

  Wenn ein Wort x∈L eine bestimmte Länge k erreicht oder überschreitet (|x| ≥k), dann läßt sich x so in drei Teile u, v und w zerlegen (mit |v|≥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:" $a^n b^n$") ist nicht regulär.

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 $uv^2w = a^k b^{k+|v|} \in L$: wegen $k+|v| \neq k$ unmöglich.
  - Fall 3: v überspannt die Mitte des Wortes, hat also die Form $a^l b^m$. Dann müsste gelten, dass $uv^2w = a^k b^m a^l b^k \in L$. Geht nicht, da a's auf b's folgen.
- Es gibt also für $a^k b^k$ keine Zerlegung in uvw mit duplizierbarem Mittelteil. Also ist L= $a^n b^n$ 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", nämlich maximal die k voraufgegangenen Symbole. (Anders ausgedrückt: Er kann nur bis k zählen.)

Ein endlicher Automat kann deshalb nur solche Sprachen erkennen, bei denen die Zulässigkeit eines Symbols in einer Zeichenfolge auf der Grundlage eines Vorkontextes von begrenzter Länge entschieden werden kann. Diese Eigenschaft heißt die „Markov-Eigenschaft".

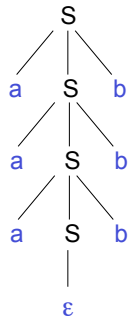
Um Zugehörigkeit zu $a^n b^n$ zu erkennen, müsste sich der Automat beliebig lange Ketten von a's merken können, weil er die Information anschließend beim Abarbeiten von b's braucht.

Kontextfreie Grammatik: Ein neuer Formalismus

- Kontextfreie Gramatiken („KFG", „CFG") beschreiben Sprachen mithilfe von Ersetzungsregeln ("rewrite rules", Produktionen) der Form A → w
  - Beispiel: S → aSb, S → ε beschreibt L= $a^n b^n$
- A → u ist zu lesen als: Ein Vorkommen von A in einer Symbolfolge/ einem Wort kann durch u ersetzt werden
  - Beispiel: aaSbb wird zu aaaSbbb oder zu aaεbb = aabb
- Eine solche Ersetzung ist ein zulässiger Ableitungsschritt. Wir schreiben: aaSbb ⇒ aaaSbbb bzw. aaSbb ⇒ aabb.
- Um ein Wort über der Sprache {a, b} abzuleiten, beginnen wir mit S (dem „Startsymbol").
- Wir wenden Ersetzungsregeln an, bis ein Wort w entsteht, das nur noch a's und b's enthält („Terminalsymbole").
  - Beispiel: S ⇒ aSb ⇒ aaSbb ⇒ aaaSbbb ⇒ aaabbb
- Wir haben damit gezeigt, dass w durch die Regeln der Grammatik aus S ableitbar ist: w ein Wort der durch die Grammatik beschriebenen (erzeugten) Sprache L.

Kontextfreie Grammatiken

- Die Ableitung
  S ⇒ aSb ⇒ aaSbb ⇒ aaaSbbb ⇒ aaabbb
  kann alternativ durch den Ableitungsbaum rechts dargestellt werden.



- Die Wurzel des Baumes ist das Startsymbol. Die Blätter des Baums ergeben, von links nach rechts gelesen und aneinandergehängt, das abgeleitete Wort.
- Alternative Schreibweise:
  $[_S$ a$[_S$ a$[_S$ a$[_S$ ε] b] b] b]

Kontextfreie Grammatik: Definitionen

G = <V, Σ, P, S>, wobei

- V nicht-leere Menge von Symbolen
- Σ ⊆ V nicht-leere Menge von Terminalsymbolen
- P ⊆ (V- Σ)×V* nicht-leere Menge von Produktionsregeln
- S ∈ V- Σ das Startsymbol

Die Beispielgrammatik für $L= a^n b^n$ in formaler Notation:

- G1 = < {a,b,S}, {a,b}, {<S, aSb>, <S, ε>}, S >

- Für <A, α> ∈ P schreibt man üblicherweise $A \rightarrow \alpha$.

Kontextfreie Grammatik: Definitionen

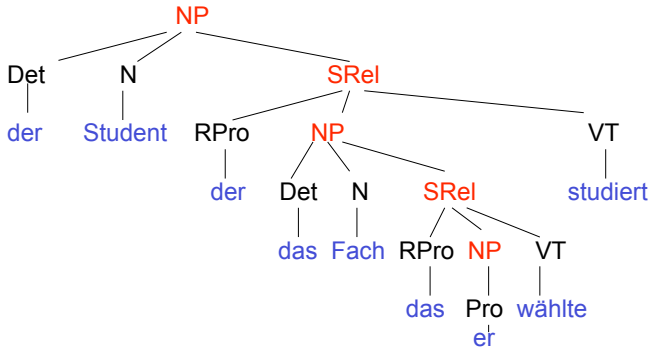
- Wenn A→α Produktion, w = uAv und w' = uαv, so ist w' aus w in einem Schritt ableitbar: w ⇒ w'
- w' ist aus w ableitbar: w ⇒* w' gdw. es eine Folge von Ableitungsschritten gibt, die mit w beginnt und mit w' endet.
- Die durch G erzeugte Sprache L(G) ist die Menge aller Worte über Σ*, die aus S ableitbar sind: L(G) = {w∈ Σ*| S ⇒* w}
- Sprachen, die durch kontextfreie Grammatiken erzeugt werden, heißen kontextfreie Sprachen.

Eigenschaften der syntaktischen Struktur [1]

- *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.*

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*]]



Eine erste kontextfreie Grammatik für deutsche Sätze

G1 = <V, Σ, P, S> mit

V = {S, SRel, NP, VI, VT, N, Det, RPro} ∪ Σ

Σ = {*schläft, arbeitet, studiert, wählte, Student, Fach, der, das, er*}

P =

S → NP VI	NP → Det N
S → NP VT NP	NP → Det N SRel
SRel → RPro NP VT	NP → Pro
SRel → RPro VI	
VI → *schläft*	N → *Student*
VI → *arbeitet*	N → *Fach*
VT → *studiert*	RPro → *der*
VT → *wählte*	RPro → *das*
Det → *der*	Det → *das*
Pro → *er*	Pro → *sie*

Eine kontextfreie Grammatik für deutsche Sätze

Notationskonventionen:

- Alternative Elemente werden durch „|"zusammengefasst (manchmal auch durch geschweifte Klammern)
- Optionale Elemente werden durch runde Klammern notiert.

Kompaktere Notation der Grammatik:

S → NP VI	S → NP VT NP
SRel → RPro VI	SRel → RPro NP VT
NP → Det N (SRel)	NP → Pro
VI → *schläft* | *arbeitet*	VT → *wählte* | *studiert*
N → *Student* | *Fach*	RPro → *der* | *das*
Det → *der* | *das*	Pro → *er* | *sie*

Kontextfreie Sprachen und reguläre Sprachen

- Kontextfreie Sprachen sind eine echte Obermenge der regulären Sprachen: Jede reguläre Sprache kann von einer CFG erzeugt werden, und es gibt kontextfreie Sprachen, die nicht regulär sind.
- Endliche Automaten verwenden Iteration: Der Automat läuft beliebig oft durch Schleifen und arbeitet dabei Wiederholungen gleicher Symbolfolgen ab.
- Kontextfreie Grammatiken verwenden Rekursion. Produktionsregeln verwenden in der Definition eines Ausdruckstyps den Ausdruckstyp selbst: Nicht-Terminale Symbole tauchen auf der linken und der rechten Seite von Regeln auf. Die Regel S → aSb besagt, dass ein Ausdruck, der mit einem a beginnt, mit einem b endet und dazwischen einen korrekten Ausdruck des Typs S enthält, ebenfalls ein korrekter Ausdruck vom Typ S ist.
- Rekursive Regeln erlauben die tiefe Schachtelung von Strukturen, und sie ermöglichen, dass eine Regel Elemente in Beziehung setzt, die in der Kette beliebig weit voneinander entfernt sind.

Kontextfreie Sprachen und natürliche Sprachen

- Kontextfreie Grammatiken sind ein Standardformalismus zur Beschreibung der Grammatik natürlicher Sprachen.
- Kontextfreie Grammatiken bilden den Standard-Formalismus zur syntaktischen Beschreibung von formalen Sprachen (Logik, Arithmetik, Programmiersprachen).
- Ein alternatives, der CGF ähnliches Format zur Beschreibung kontextfreier Sprachen ist BNF (die „Backus-Naur-Form“).

CFG: Konstituentenstruktur

- Anders als endliche Automaten beschreibt eine CFG nicht nur die zulässigen Ausdrücke einer Sprache, sondern implizit auch deren Struktur.
- Sie ordnet den Sätzen der Sprache Ableitungsbäume zu (auch „Parse-Bäume“ genannt, Parsing = automatische syntaktische Analyse).
- Durch den Ableitungsbaum/ Parse-Baum werden Teilausdrücke (Teilketten) u des analysierten Wortes einer „Kategorie“ zugeordnet: dem nicht-terminalen Symbol A, aus dem u abgeleitet wurde. Wir nennen u eine „Konstituente“ von der Kategorie A, und sagen, dass A die Elemente von u „dominiert“.

Beispiel 1

CFG für einfache arithmetische Gleichungen:

S → Term = Term
Term → (Term Op Term)
Term → – Term

Term → x | y | z
Op → + | – | * | :

Konstituenten der Kategorie „Term“ sind zum Beispiel
x, y, –z, –(x*(y+z))

Beispiel 2

Die obige CFG für ein Fragment des Deutschen.

- *er* ist eine Konstituente der Kategorie Pro
- *er* , *der Student* , *der Student, der Informatik studiert* sind Konstituenten der Kategorie NP
- *der Informatik studiert* - *der arbeitet* sind Konstituenten der Kategorie SRel