Einführung in die Computerlinguistik

Stefan Müller

Deutsche Grammatik
Institut für Deutsche und Niederländische Philologie
Fachbereich Philosophie und Geisteswissenschaften
FU Berlin

Stefan.Mueller@fu-berlin.de

24. November 2013

Organisatorisches

- alle Teilnehmer bitte bei E-Learning-Platform eintragen (Achtung! EMail-Weiterleitung aktivieren)
- Telefon und Sprechzeiten siehe: http://hpsg.fu-berlin.de/~stefan/
- Information zur Vorlesung: http://hpsg.fu-berlin.de/~stefan/Lehre/CL/
- Materialien z. T. von Prof. Uszkoreit: http://www.coli.uni-sb.de/~hansu/

Ziele der Vorlesung

- Überblick über das Fachgebiet
- Nützlichkeit zeigen, Interesse wecken
- Einführung ins Erstellen formaler Beschreibungen von Sprache und deren Verbarbeitung

Jobs

- Angebote auf Linguist-Liste bzw. Colibri angucken:
  - http://linguistlist.org/
  - http://colibri.let.uu.nl/

Was wird gebraucht? Beispiel aus einer Stellenanzeige

- Soll-Kriterien:
  - abgeschlossenes Studium mit Schwerpunkt Germanistik und/oder Linguistik
  - sehr gute Englischkenntnisse
- wünschenswert:
  - Programmierkenntnisse (Perl, Java, C++)
  - Erfahrung mit Datenbanken (SQL, Oracle)
  - weitere Fremdsprachenkenntnisse
  - praktische Erfahrung im Bereich Sprachverarbeitung
  - Erfahrungen in den Bereichen maschinelle Übersetzung, Korpuslinguistik, statistische Sprachverarbeitung, kontrastive Linguistik, Übersetzung, Projekt-Management

Teil I

Überblick über das Fachgebiet

Inhalt Teil I

- Aufgaben und Einordnung des Faches
- Motivationen für die Modellierung menschlicher Sprache
- Computerlinguistik als eine moderne Sprachwissenschaft
- Repräsentationen und Verarbeitungskomponenten

Anwendungen

- Mensch-Maschine-Schnittstelle:

  taz: Wieviele Fernseher (haben sie zu Hause)?
  Harald Schmidt: Zwei.
  taz: Videorecorder?
  Harald Schmidt: Ja, aber den kann ich nicht bedienen. Der staubt nur vor sich hin. Ich habe schon ewig nichts mehr aufgezeichnet, seit Jahren nicht mehr, wirklich. (taz, 30.06.2003, S. 18)
- In 60 % der deutschen Haushalte blinkt die Zeitanzeige am Videorecorder.
- Bedinung der Geräte ist zu komplex.
  Sprache ist unser natürliches Kommunikationsmittel.

(Potentielle) Anwendungen: (zukünftig) wirtschaftlich nutzbare (I)

- Mensch-Maschine-Schnittstelle
  - Spracherkennung
    - Diktiersysteme
    - Call-Center-Anwendungen (Kinokarten), Auskunftssysteme (Bahnfahrplan)
    - E-Commerce, Geschäfte über Mobil-Telefone
  - Sprachsynthese
    - je nach Komplexität der Interaktion
- maschinelle Übersetzung geschriebener und gesprochener Sprache
  - Unterstützung von Humanübersetzern
  - domänenspezifische Übersetzungen

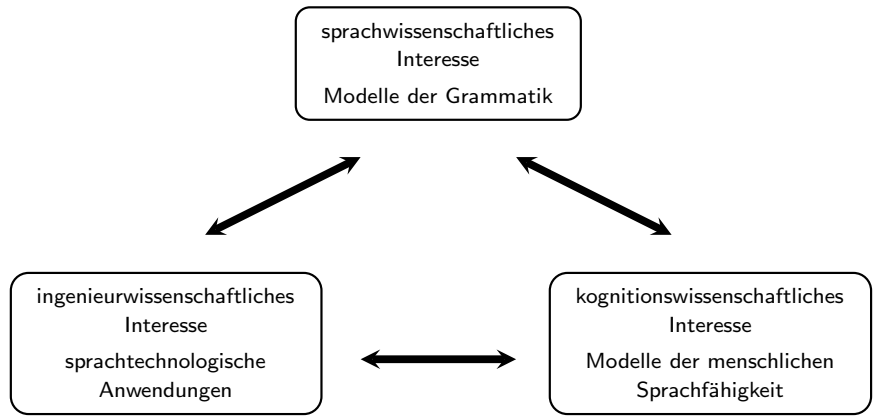
(Potentielle) Anwendungen: (zukünftig) wirtschaftlich nutzbare (II)

- Hilfsmittel für Behinderte
  - Vorlesegeräte für Blinde
  - Fernsehen für Hörgeschädigte
  - Wortvervollständigung für mechanisch Behinderte
- Information Extraction / Information Retrieval
  (Wissen im WWW, nur wo?)
- Rechtschreibprüfprogramme (Spell Checker),
  Grammatikprüfprogramme (Grammar Checker)
  (siehe dazu aktuellen Artikel: http://seattlepi.nwsource.com/business/217802_grammar28.asp)

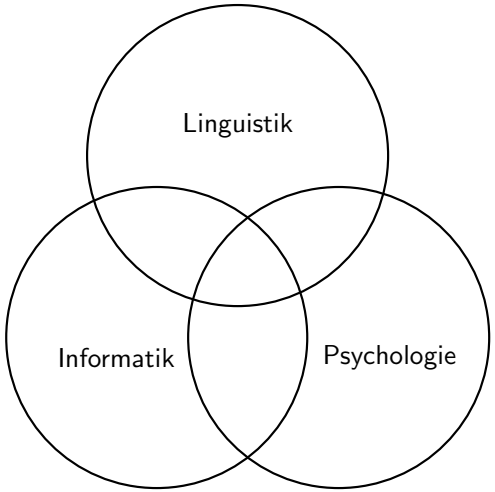
Anwendungen: akademische nutzbare

- Theorieverifikation (theoretisch, empirisch)
- Entwickeln und Testen linguistischer Theorien
- Unterstützung bei der Erstellung von (anotierten) Korpora
  Korpora sind wichtig für
  - Lexikographie (Verwendung und Valenz bestimmter Wörter)
  - Widerlegung von falschen Behauptungen
  - quantitative Aussagen
- Bereitstellung von Korpuswerkzeugen
  Beispiel: http://corpora.ids-mannheim.de/~cosmas/

Motivationen



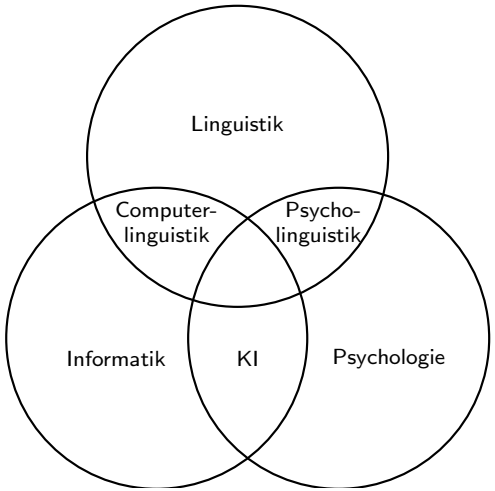
Nachbarwissenschaften



Status des Faches

- Computerlinguistik ist eine Disziplin zwischen
  - Linguistik
  - Informatik
  - Psychologie
- Modelle der menschlichen Sprache werden entworfen, formalisiert, auf dem Computer implementiert und empirisch untersucht
- Informatik →
  Überschneidung mit dem Fachgebiet maschinelle Sprachverarbeitung (*natural language processing* – NLP),
  Teilgebiet der künstlichen Intelligenz (*artificial intelligence* – AI)
- Psychologie →
  Überschneidungen mit der kognitiven Sprachpsychologie

Schnittmengen mit den Nachbarwissenschaften



Die Disziplin (I)

Computerlinguistik im weiteren Sinne
: ist ein zwischen Linguistik und Informatik liegendes interdisziplinäres Forschungsgebiet, das sich mit der maschinellen Verarbeitung natürlicher Sprachen beschäftigt.

Computerlinguistik im engeren Sinne
: ist ein Teilgebiet der modernen Linguistik, das berechenbare Modelle menschlicher Sprache entwirft, implementiert und untersucht.

Die Disziplin (II)

Theoretische Computerlinguistik
: entwirft, implementiert und untersucht die Modelle mit dem Ziel, zum Verständnis, zur Verifikation und zur Verbesserung der zugrundeliegenden linguistischen und psychologischen Theorien beizutragen.

Angewandte Computerlinguistik
: entwirft, implementiert und untersucht die Modelle mit dem Ziel, Softwareanwendungen zu ermöglichen, die über eine (eingeschränkte) Beherrschung menschlicher Sprache verfügen.

Verwandte Begriffe

Maschinelle Sprachverarbeitung
: Analyse und Generierung von natürlicher Sprache mit dem Computer
Englisch: Natural Language Processing (NLP).

Sprachtechnologie(n)
: Oberbegriff für die Technologien sprachbeherrschender Systeme.
Ingenieurwissenschaftliches Forschungsgebiet, in dem die Sprachtechnologien entwickelt werden.

Linguistische Datenverarbeitung (LDV)
: Traditionell ein Teilgebiet der elektronischen Datenverarbeitung, das sich sowohl mit der Anwendung von Methoden der Datenverarbeitung für die linguistische Forschung als auch mit maschineller Sprachverarbeitung beschäftigt. Die LDV versteht sich heute als ein Gebiet, das die Computerlinguistik einschließt.

Sprachdatenverarbeitung
: Verarbeitung von sprachlichen Daten mit dem Computer. Schließt ein: mono- und multilinguale Textverarbeitung, elektronische Wörterbücher, Konkordanzen, Terminologiebanken, maschinelle und maschinengestützte Übersetzung.

Linguistik

Die Linguistik ist eine „moderne“, synchron orientierte, auf die interne Struktur der Sprache bezogene Wissenschaft, die sprachliche Regularitäten auf allen Beschreibungsebenen untersucht und ihre Ergebnisse in explizierter (formalisierter) Beschreibungssprache und in integrierten Modellen darlegt.

(Bußmann (1990): *Lexikon der Sprachwissenschaft*)

Teilgebiete der Linguistik

- nach Beschreibungsebenen
  - Phonetik
  - Phonologie
  - Morphologie
  - Syntax
  - Semantik
  - Pragmatik/Text/Diskurs
- andere Teilgebiete
  - Historische Linguistik
  - Sozio- und Ethnolinguistik
  - Dialektologie
  - Psycholinguistik
  - Neurolinguistik
  - Mathematische Linguistik
  - Statistische Linguistik/Korpuslinguistik

Aspekte der Sprache

- **Sprachliches Wissen**
  Was sind die Inhalte und Strukturen dieses unbewußten Wissens?
- **Sprachverarbeitung**
  Wie produzieren und verstehen wir sprachliche Äußerungen?
- **Spracherwerb**
  Wie lernt das Kind seine Muttersprache?
- **Sprachwandel**
  Wie entstehen Sprachen, Dialekte, Soziolekte?

Kompetenz und Performanz

Sprachliche Kompetenz
: die endliche strukturierte Wissensbasis, die es den Sprechern einer Sprache ermöglicht, die wohlgeformten Äußerungen der Sprache zu generieren und zu interpretieren.

Sprachliche Performanz
: die Generierung oder Interpretation realer Äußerungen, bzw. die Gesamtheit der Prozesse, die beteiligt sind, wenn der Mensch auf der Basis der sprachlichen Kompetenz reale Äußerungen generiert und interpretiert.

Kompetenzmodell

- Ein Kompetenzmodell sollte beinhalten:
  - Regeln,
  - Prinzipien,
  - Beschränkungen auf jeder Beschreibungsebene,

  die in ihrem Zusammenwirken genau die wohlgeformten Sätze der Sprache charakterisieren.
- Es bietet für jede Sprache eine formalisierte endliche Definition einer unendlichen Menge von Paaren.
- (Dazu gehören: Lexikon, morphologische Regeln, syntaktische Regeln, semantische Regeln.)

Performanzmodell

Ein Performanzmodell sollte erklären:

- warum viele ungrammatische Sätze erzeugt werden
  z. B. Sprechfehler, Grammatikfehler
- warum viele ungrammatische Sätze verstanden werden
  z. B. in der Kommunikation mit Kindern oder Ausländern
- warum viele grammatische Sätze nicht erzeugt werden
  z. B. durch Präferenzen in der Generierung
- warum viele grammatische Sätze nicht verstanden werden
  z. B. Holzwegsätze
- wie die Verarbeitung zeitlich strukturiert ist
  z. B. Effizienz, Abfolge der Verarbeitungsschritte
- welchen Aufwand die Verarbeitungsschritte erfordern
  z. B. Abhängigkeiten von anderen kognitiven Belastungen

Beispiele für Performanzphänomene

- Grammatikfehler:

  (1) Das Verfassen der Kinderbücher und der Reiseberichte haben dem Autor viel Ruhm eingebracht.

- Holzwegsätze:

  (2) The canoe floated down the river sank.

  (3) a. Er bezichtigte den Vater des Schreibens unkundiger Kinder.
  b. Peter beschuldigte sie der Geheimniskrämerei ähnlichen Verhaltens.

Charakter der Forschung (I)

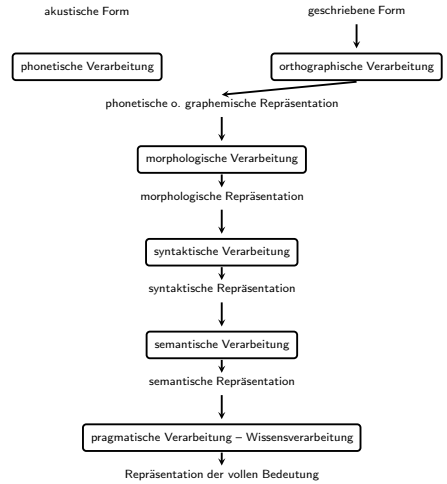
- Sprachwissenschaften gehören zu Geisteswissenschaften
- traditionelle theoretische Linguistik
  - nicht empirisch im Sinne der Naturwissenschaften
  - nicht formal
  - Sprachdaten meist exemplarisch, aus Texten oder wenig kontrollierter Feldforschung
  - Beschreibungen nicht formalisiert → Generalisierungen und Theorien nicht falsifizierbar
  - Argumentation auf Plausibilitätserwägungen gegründet
- Formalisierung Voraussetzung für Entwicklung der Computerlinguistik ohne Formalisierung keine Algorithmisierung
- Simulation der menschlichen Sprachverarbeitung → experimentell
- Zunahme empirischer Forschung

Charakter der Forschung (II)

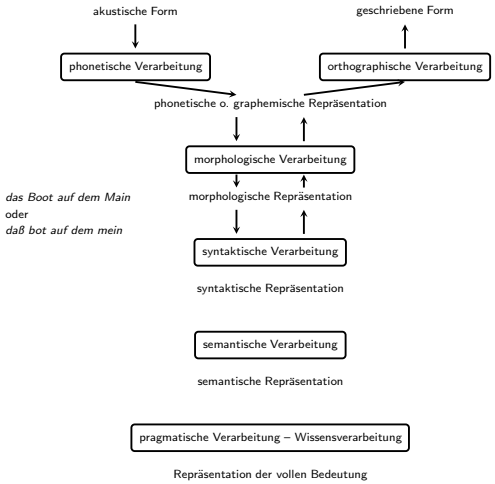
Computerlinguistik ist also:

- theoretisch
- deskriptiv
- formal
- empirisch
- experimentell
- simulativ

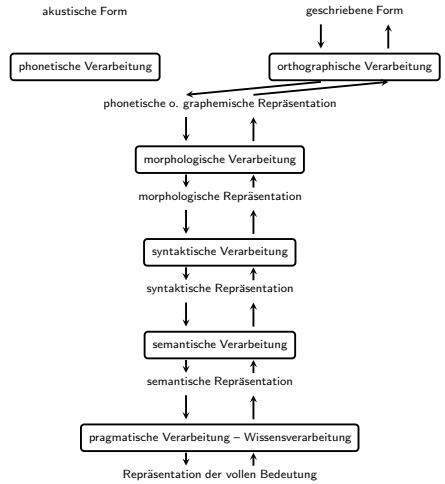
Textverstehen



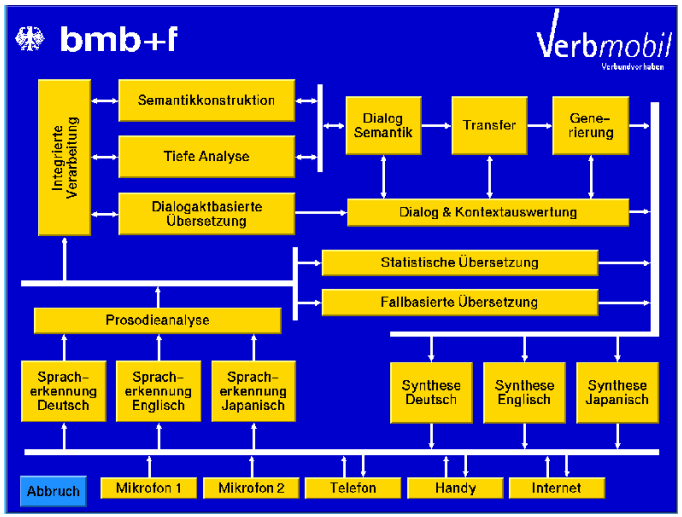
Diktat



Maschinelle Übersetzung



Verb*mobil*-Übersicht



Ambiguität (I)

- **phonetische Ambiguität (Homophone)**
  Miene (Gesichtsausdruck) – *Mine* (Bergwerk, Sprengkörper, Stift)
- **orthographische Ambiguität (Homographen)**
  übersétzen – *´übersetzen*
- **lexikalische Ambiguität (Homonyme)**
  Band (*Bänder*) – *Band* (*Bände*) – *Band* (*Bands*)
- **morphologische Ambiguität**
  Stau-becken – *Staub-ecken*
  Hauptpostsekretär
  Mädchenhandelsschule (Fourquet, 1970)

Ambiguität (II)

- **syntaktische Ambiguität**

  (4) a. Peter fuhr seinen Freund sturzbetrunken nach Hause.
  b. Visiting relatives can be boring.
  c. Unbekannte haben Mittwochabend bei einer Wahlkampfveranstaltung mit FDP-Chef Guido Westerwelle Farbbeutel geworfen. (taz, 21.5.04, S. 7)
  d. Ich traf den Sohn des Nachbarn mit dem Gewehr.

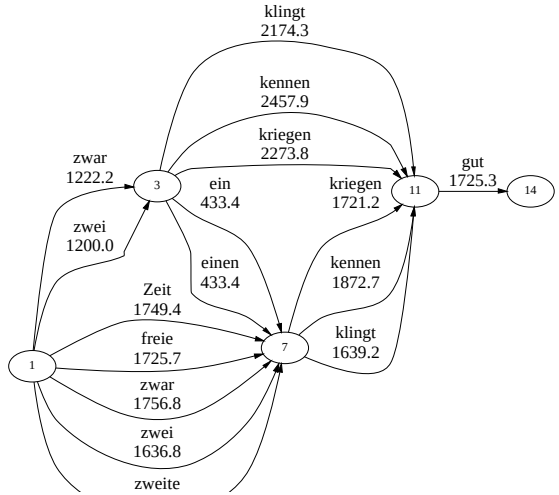
- **kompositionell-semantische Ambiguität**

  (5) a. Die zwei Mitarbeiter müssen vier Sprachen beherrschen.
  b. Im Rahmen der Schulfilmwoche zeigen 80 Kinos 50 Filme. (taz, berlin, 17.06.2003, S. 24)

- **pragmatische Ambiguität**

  (6) Könnten Sie die Aufgabe lösen?

Worthypothesengraph: *Zwei klingt gut.*



Kombinatorische Explosion (I)

Wieviele Lesarten (lexikalische, syntaktische und anaphorische Ambiguitäten) hat das folgende Beispiel (Beispiel nach H. Uszkoreit)?

(7) Früher stellten die Frauen der Insel am Wochenende Kopftücher mit Blumenmotiven her, die ihre Männer an den folgenden Montagen auf dem Markt im Zentrum der Hauptinsel verkauften.

Insgesamt sind es 258.048!

Ohne Anaphernambiguitäten sind es 64.512!

Kombinatorische Explosion (II)

Wieviele Lesarten (lexikalische, syntaktische und anaphorische Ambiguitäten) hat das folgende Beispiel (Beispiel nach H. Uszkoreit)?

(8) Früher stellten die Frauen der Inseln am Wochenende Kopftücher mit Blumenmotiven her, die ihre Männer an den folgenden Montagen auf dem Markt im Zentrum der Hauptinsel verkauften.

- *früher* kann sowohl eigenständiges Adverb als auch Komparativ von *früh* sein. (2)
- Die Verbform *stellten* ist ambig zwischen Präteritum und Konjunktiv. (2)
- *die Frauen* kann sowohl Subjekt als auch Objekt des Satzes sein. (2)
- *am Wochenende* kann *die Insel*, *die Frauen* oder das Verb modifizieren. (3)
- *mit Blumenmotiven* kann sich auf *die Kopftücher* beziehen, ein Instrument der Herstellung sein oder ein Adjunkt im Sinne von *gemeinsam mit Blumenmotiven*. (3)
- *her* hat auch eine direktionale Bedeutung. (2)
- Der Relativsatz könnte jede der vier Nominalphrasen modifizieren. (4)

Kombinatorische Explosion (III)

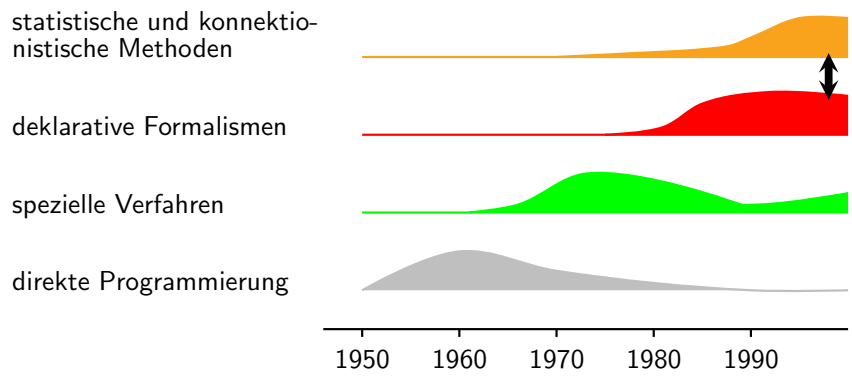
Wieviele Lesarten (lexikalische, syntaktische und anaphorische Ambiguitäten) hat das folgende Beispiel (Beispiel nach H. Uszkoreit)?

(9) Früher stellten die Frauen der Inseln am Wochenende Kopftücher mit Blumenmotiven her, die ihre Männer an den folgenden Montage auf dem Markt im Zentrum der Hauptinsel verkauften.

- Sowohl *die* als auch *ihre Männer* kann Subjekt des Relativsatzes sein. (2)
- Das Possesivpronomen *ihre* kann auf jede der vier Nominalphrasen referieren. (4)
- *Montagen* hat eine zweite Lesart als Nominalisierung von *montieren*. (2)
- *der Hauptinsel* kann im Genitiv zu der vorangegangen Nominalphrase gehören oder im Dativ die Käuferin bezeichnen. (2)
- Die drei Präpositionalphrasen des Relativsatzes können sich in insgesamt sieben Kombinationen mit jeweils vorhergehenden NPs oder mit dem Verb verbinden. (7)
- *verkauften* ist wieder ambig zwischen Präteritum und Konjunktiv. (2)

Also insgesamt: 2 * 2 * 2 * 3 * 3 * 2 * 4 * 2 * 4 * 2 * 2 * 7 * 2 = 258.048

Verschiedene Herangehensweisen



Direkte Programmierung

Direkte Programmierung in einer traditionellen Programmiersprache
Keine Trennung von Kompetenz und Performanz,
also auch keine Trennung von Grammatik und Verarbeitung
Beispiele:
SYSTRAN, SHRDLU, frühe SFB 100 Systeme

Kompetenzmodellierung: als Modelle theoretisch uninteressant, nicht überprüfbar, Kodierung linguistisch uninteressant, schwer erweiterbar

Performanzmodellierung: als Modelle theoretisch uninteressant, weil mit der Kompetenz vermischt,
keine Ansätze zur Integration psycholinguistischer Erkenntnisse

Anwendungspotential: einige wenige Systeme sind zur Anwendungsreife gelangt (z. B. SYSTRAN),
fast nicht mehr erweiterbar, für neue Entwicklungen nicht geeignet

Spezielle Verfahren

Spezielle Verfahren und Beschreibungssprachen
Trennung von Kompetenz und Performanz,
vielfach noch immer Vermischung von Wissen und Verarbeitung
Beispiele:
Augmented Transition Networks (ATN), Augmented Phrase Structure Grammar (APSG), EUROTRA Framework

Kompetenzmodellierung: verschieden von den Modellen der Linguistik, als linguistische Modelle theoretisch wenig interessant, vielfach Vermischung mit prozeduralen Elementen

Performanzmodellierung: wenige aber sehr ernsthafte Versuche, einige Gesichtspunkte der Performanzmodellierung zu berücksichtigen, Einflüsse der Psycholinguistik, Hindernis ist das Fehlen plausibler Kompetenzmodelle

Anwendungspotential: fast alle der heute marktreifen Systeme gehören zu dieser Klasse (z. B. METAL, Q&A)

Deklarative Formalismen

Deklarative Grammatikformalismen
Linguistische Grammatikmodelle und Einzelanalysen lassen sich darin kodieren.

Dadurch Aufhebung der Trennung von theoretischer Linguistik und Computerlinguistik.

Beispiele:

fast alle Unifikationsgrammatikmodelle, neuere semantische Formalismen

Kompetenzmodellierung: deklarative linguistisch fundierte Modelle; unabhängig von Verarbeitungsrichtung, Verarbeitungsreihenfolge und Verarbeitungsalgorithmen; logisch fundierte Semantik, transparente Modularisierung und Hierarchisierung des Wissens

Performanzmodellierung: deduktive Verarbeitung; in den fortgeschrittensten Systemen erfolgt die Verarbeitung durch Typdeduktion;

Anwendungspotential: noch keine marktreifen Systeme, bisher noch mangelnde Effizienz

Vorteile deklarativer Spezifikationen

- deklarative Spezifikation → bidirektional verwendbar
  Karl liest ein Buch. ↔ lesen(karl,buch)
- Analyse von Sprache:
  Schrift oder gesprochene Sprache ist Eingabe, Bedeutung Ausgabe
- Generierung von Sprache:
  Bedeutung ist Eingabe, Schrift oder gesprochene Sprache Ausgabe
- Arbeit an Grammatiken ergibt Zugewinn an Wissen über die Sprache
- große Grammatiken für verschiedene Domänen einsetzbar
- kein totaler Einbruch bei Domänenwechsel

Linguistische Grammatikmodelle

- Generalized Phrase Structure Grammar (GPSG)
  (Gazdar u. a., 1985),
- Lexical Functional Grammar (LFG)
  (Bresnan, 1982, 2001; Berman und Frank, 1996; Berman, 2003),
- Categorial Grammar (CG)
  (Wood, 1993; Steedman, 2000),
- Head-Driven Phrase Structure Grammar (HPSG)
  (Pollard und Sag, 1994; Müller, 2013)

Computerlinguistische Grammatikformalismen

- Functional Unification Grammar (FUG)
- PATR
- Stuttgart-Tübingen Unifikation Formalism (STUF)
- Typed Feature Structures System (TFS)
- Comprehensive Unification Formalism (CUF)
- Type Description Language (TDL)

Statistische u. konnektionistische Verfahren

Statistische u. konnektionistische Verfahren
in der akustischen Spracherkennung (Hidden Markov Model), und in der maschinellen Übersetzung
Beispiele:
Hidden Markov Model (HMM), Parsing mit neuronalen Netzen

Kompetenzmodellierung: für die Theoriebildung uninteressant, die Kompetenz ist nicht transparent modelliert, keine Verbindung zu den Theorien der Linguistik, unzureichende Darstellung der Rekursivität

Performanzmodellierung: Lernverfahren, massive Parallelität könnte Schlüssel zum Effizienzproblem sein, Potential für die Modellierung linguistischer Präferenzen und anderer unscharfer Konzepte z. B. in der lexikalischen Semantik, Potential für holistische Ansätze

Anwendungspotential: großes Potential in der akustischen Spracherkennung und in der akustischen Sprachsynthese,
in den letzten Jahre beeindruckende Entwicklungen auch im Bereich der statistikbasierten MÜ

Beispiel: N-Gram

- Wahrscheinlichkeit, daß Wort B dem Anfangsstück A folgt:

  (10) Der Mann kennt die Frau.

- P(Mann | der), P(kennt | der Mann), P(die | der Mann kennt), ...
- zu viele Daten nötig, um diese ganzen Wahrscheinlichkeiten zu berechnen
- Annäherung: P(Mann | der), P(kennt | Mann), P(die | kennt), ...
- zwei Wörter angeschaut → Bezeichnung: Bigram
- Trigram: schaut drei Wörter an, also zwei Wörter zurück
- Vorhersagen über nächstes Wort möglich

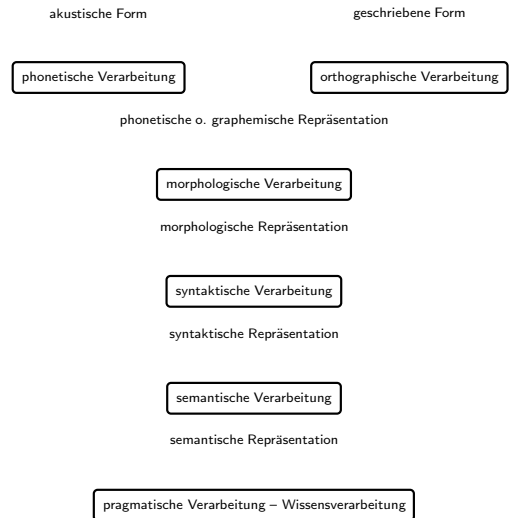
- je spezifischer die Domäne, desto besser die Ergebnisse
- allerdings → Probleme beim Domänenwechsel

- aktueller Stand: Kombination tiefer und flacher Verarbeitung (z. B. Verb*mobil*)
- Kombination von Robustheit und Akkuratheit

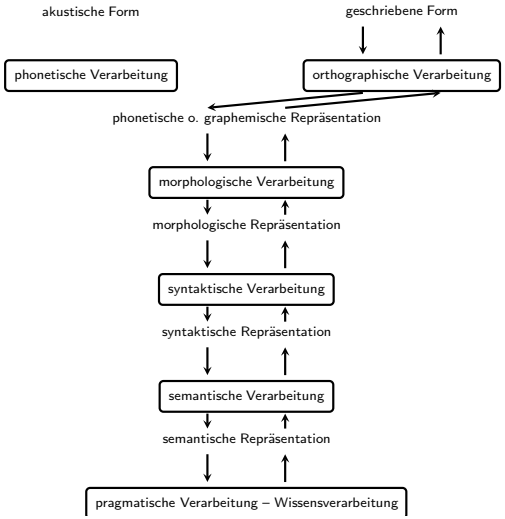
Teil II

Ansätze und Methoden

Ebenen der Verarbeitung



Maschinelle Übersetzung



Verarbeitung gesprochener Sprache

- **Eingabe**: Die Äußerung liegt als dreidimensionales analoges Signal vor. (Zeit, Frequenz, Intensität)
- **Verarbeitung**:
  Vorverarbeitung: Filterung, Transformationen (FFT)
  Aus dem zweidimensionalen Signal wird ein dreidimensionales Schallspektrum
  Verarbeitung:
  Erkennung von Segmenten
  (Phone, Diphone, Halbsilben, Silben, Morpheme, Wörter)
  Methoden:
  regelbasierte Ansätze, stochastische Ansätze (HMM),
  konnektionistische Ansätze
- **Ausgabe**:
  Repräsentationen der erkannten Segmente,
  oft versehen mit Wahrscheinlichkeiten

Morphophonologische Verarbeitung

- **Eingabe:** Segmente (Buchstaben/Graphe, Phone/Allophone)
- **Verarbeitung:**
  Jedes Wort wird in seine Morpheme zerlegt.
  Dabei müssen phonologische Prozesse rückgängig gemacht werden.
  Zugriff auf die zugrundeliegenden Morphem-Einträge im Lexikon.
  Aufbau der internen Wortstruktur.
- **Ausgabe:**
  Repräsentationen der Wörter, die mit den für die syntaktische und semantische Verarbeitung relevanten Merkmalen versehen sind.

Morphologie

- Morphem (klassisch): kleinstes bedeutungstragendes Element der Sprache, das als phonologisch-semantisches Basiselement nicht mehr in kleinere Elemente zerlegt werden kann.
- Morphem (richtig): kleinste, in ihren verschiedenen Vorkommen als formal einheitlich identifizierbare Folge von Segmenten, der (wenigstens) eine als einheitlich identifizierbare außerphonologische Eigenschaft zugeordnet ist. (Wurzel 1984:38)
- freies Morphem: *Rad*, *Schnur*, *klug*
- gebundenes Morphem: *-heit*, *-bar*, *ent-*, Plural-Morphem
- Plural-Morphem ist eine Abstraktion: ∅, *-e*, *-en*, *-er*, *-n*, *-s*, ...
- Allomorphe: gleiche Bedeutung und komplementäre Verteilung
- Morpheme bzw. Allomorphe stehen im Lexikon der Morphologiekomponente

Flexionsmorphologie

- Markierung von Tempus, Person, Kasus, Numerus, ...
  geht – ging
  der Mann – des Mannes
- flektierte Formen werden bei manchen Anwendungen einfach in einer Liste aufgeführt

Lexikon mit flektierten Formen

Aal	bade	bete	brauen
Aale	baden	beten	braust
aale	badest	betest	braut
aalen	badet	betet	brause
Aals	...	bette	brausen
aalst	baue	betten	braust
aalt	bauen	bettest	braust
aaltest	baust	bettet	...
aaltet	baut	...	Zylinder
...	...	braue	Zylinders

Derivationsmorphologie

- bedeutungsverändernde Bildung von Wörtern aus einem Stamm-Morphem und einem Derivationsmorphem
  klar – unklar
  Sache – sächlich oder *sachlich*
- produktive, d. h. regelmäßige und vorhersagbare Prozesse:
  schlagen – schlagbar – Schlagbarkeit
- abgeleitete Elemente haben teilweise Eigenschaften der ursprünglichen Form
- Listung im Lexikon ist redundant, schlecht wartbar und höchstwahrscheinlich unvollständig

Lexikon mit flektierten und abgeleiteten Formen

Aal	badet	betten	...
Aale	...	bettest	schlagbar
aalen	baue	bettet	Schlagbarkeit
Aals	bauen	...	
aalst	baust	braue	...
aalt	baut	brauen	
aaltest	...	braust	unschlagbar
aaltet	bete	braut	Unschlagbarkeit
...	beten	brause	
bade	betest	brausen	...
baden	betet	braust	Zylinder

Komposition

- Zusammensetzung von mehreren Stamm-Morphemen
  Bauer und *Hof* – *Bauernhof*
  Sonne und *baden* – *sonnenbaden*
- sehr häufig in Texten und viele Bildungsmuster
- Determinativkomposita: *Haustür*, *Katzenpfote*, *Intensivkurs*
- Rektionskomposita: *Hausverkauf*, *Preisangabe*
- oft viele Interpretationen möglich: *Blumenfan*
- meist zweigliedrig, aber auch mehrgliedrig: *Oberflächenvorbereitungsmaßnahme*, *Unfallverhütungsvorschriften*, *Grundstücksverkehrsgenehmigungszuständigkeitsübertragungsverordnung*[1]
- Aufzählung im Lexikon nur für Anwendungen mit beschränktem Wortschatz sinnvoll
- Problem: Ermittlung der richtigen Segmentierung
  Transport+band vs. # *Tran+sport+band*
- Ansatz: Zerlegung in möglichst wenig Komponenten, Frequenzlisten

Lexikon mit Komposita

Aal	bade	bette	schlagbar
Aale	baden	betten	Schlagbarkeit
aalen	badest	bettest	
Aalfang	badet	bettet	. . .
Aalfänger	. . .	. . .	unschlagbar
Aalräucherei	baue	braue	Unschlagbarkeit
Aalräuchereien	bauen	brauen	Unschlagbarkeitsbehauptung
Aalräucherer	baust	braust	
Aals	baut	braut	. . .
aalst	. . .	brause	
aalt	bete	brausen	Zylinder
aaltest	beten	braust	Zylinders
aaltet	betest	braust	Zylinderschloß
. . .	betet	. . .	Zylinderstifte

Konkatenation und nichtkonkatenative Phänomene

- Konkatenation (von Morphen):
  z. B. *geh* + *st* ↔ *gehst*
  ab + *ge* + *frag* + *t* + *e* ↔ *abgefragte*

- Nichtkonkatenative Phänomene:
  - Veränderung des Stammvokals:
    z. B. Umlaut: Pluralbildung (*Mutter* – *Mütter*)
    z. B. Ablaut: Tempusmarkierung (*geb* – *gab*)

Morpho(phonologie)

- Stamm + Endung
  z. B. *geh* + *st* – *gehst*
- Epenthese
  z. B. *bad* + *st* – *badest*
- Elision
  z. B. *ras* + *st* – *rast*
- Merkmalsveränderungen an Phonemen
  z. B. Auslautverhärtung
  (stimmhafte Konsonanten werden am Silbenende stimmlos)
  /bad/ – [bat]

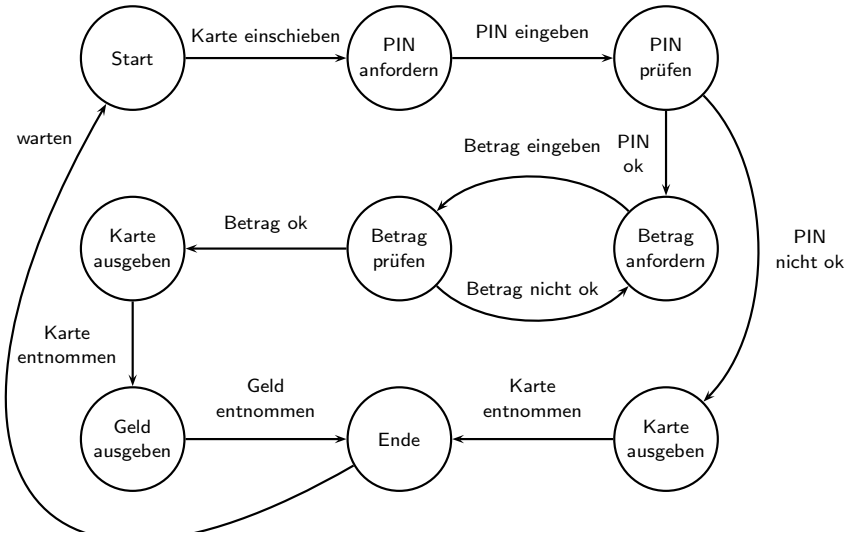
Automaten (I)

- Automaten in der weiteren Bedeutung des Wortes sind ein zentrales, aber nicht formal definiertes Konzept in der Informationsverarbeitung.
- Wenn wir durch eine Sequenz von Handlungen bestimmte Effekte auslösen, dann ist das nicht unbedingt Informationsverarbeitung.
  Normalerweise wird mit jeder Handlung eine Wirkung erzeugt.
  Beispiel: Ich öffne eine Tür mit zwei Schlössern,
  indem ich jedes Schloß mit einem Schlüssel aufschließe.
- Wenn wir ein System haben, das so gebaut ist, daß es in Abhängigkeit von meinen Handlungen „Entscheidungen“ über seine Handlungen trifft, dann liegt Informationsverarbeitung vor. In diesem Fall bewirken meine Handlungen die Handlungen der Maschine nicht direkt kausal.
- Automaten sind Systeme, die in Abhängigkeit von meinen Handlungen, bestimmte Aktionen ausführen bzw. auslösen.
- Damit man von Automaten spricht,
  muß es mindestens einen Entscheidungspunkt geben.

Automaten (II)

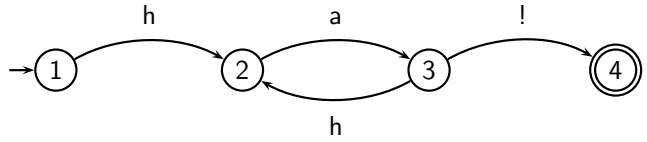
- Automaten werden verwendet, um Symbol- oder Handlungsabfolgen zu überprüfen, zu analysieren oder abzuarbeiten.
  Beispiel: Verarbeitung von Benutzereingaben
  - Überprüfung, ob Eingaben korrekt
  - in richtiger Reihenfolge bzw. in richtigem Kontext
- Automaten spielen auch eine Rolle bei der Überprüfung bzw. Übersetzung von Programmier- oder Spezifikationssprachen in die interen Sprachen des Computers.
- In der Sprachverarbeitung werden Automaten an vielen Stellen eingesetzt, so z. B. bei der morphologischen Analyse von Wörtern.

Der „endliche Geldautomat“ (nach Reinhard Völler)



Beispiel: Ein Lachautomat

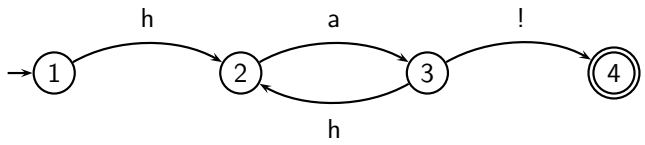
Graphische Repräsentation:
Zustandsübergangsdiagram (*transition diagram*)



- Der Startzustand ist durch einen eingehenden Pfeil gekennzeichnet.
- Endzustände
  doppelt eingekreist

Eingabe des Automaten

Der Lachautomat erhält eine Zeichenkette als Eingabe:

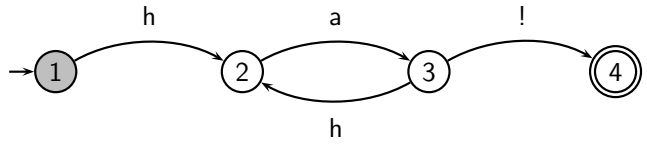


haha!

Was macht der Automat damit?

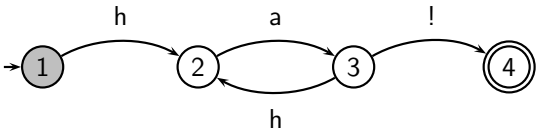
Beginn der Verarbeitung

- Der Automat ist im Startzustand.
- Er schaut auf das erste Zeichen der Eingabe.



h aha!

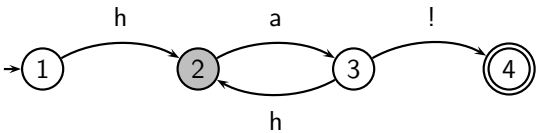
Ein einzelner Verarbeitungsschritt



h aha!

Bei einem Übergang

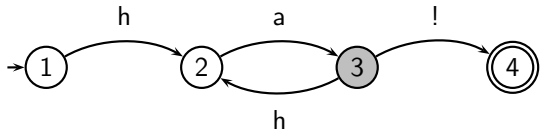
- springt der Automat in einen neuen Zustand und
- schaut auf das nächste Zeichen der Eingabekette.



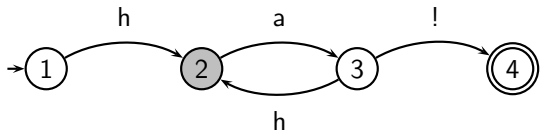
h a ha!

Abarbeiten der Eingabe

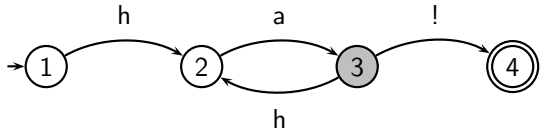
Der Automat konsumiert so Zeichen um Zeichen . . .



ha h a!



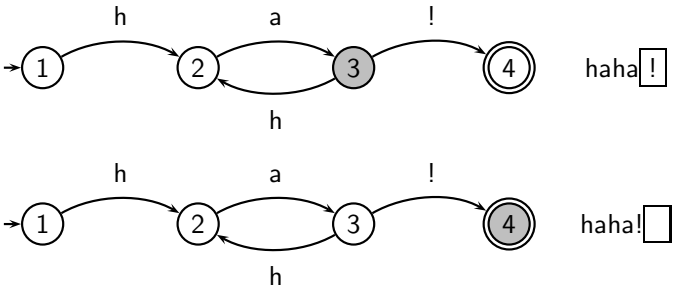
hah a!



haha !

Abarbeiten der Eingabe

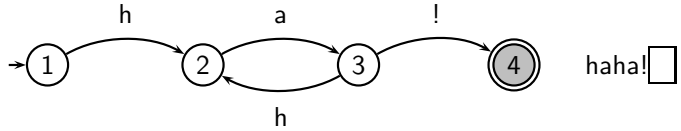
Der Automat konsumiert so Zeichen um Zeichen ...
bis auch das letzte Zeichen der Eingabe konsumiert wurde.



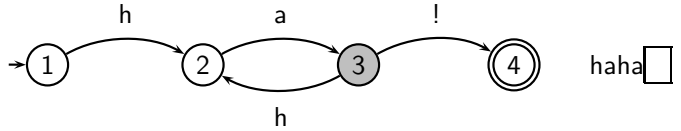
Ende der Verarbeitung (I)

Wenn die Eingabe vollständig konsumiert ist, gibt es zwei Möglichkeiten:

- Der aktuelle Zustand ist ein Endzustand (4):
  Der Automat hat die Eingabe akzeptiert.



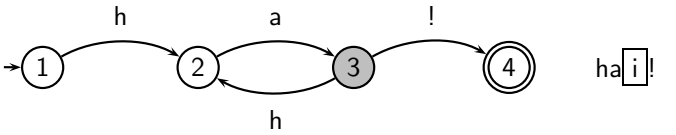
- Der aktuelle Zustand ist kein Endzustand (1,2,3):
  Der Automat hat die Eingabe nicht akzeptiert.



- Ein Automat kann mehrere Endzustände besitzen.

Ende der Verarbeitung (II)

Kommt der Automat nicht weiter, weil kein Übergang zum aktuellen Eingabezeichen paßt, wird die Eingabe ebenfalls nicht akzeptiert.



Automaten (III)

Bestandteile eines Automaten:

- endliche Menge von Zuständen
- Startzustand, Endzustand (auch mehrere)
- Eingaben: Wörter einer Sprache
- Ausgaben: Wörter einer Sprache
- Regeln für die Auswirkungen einer Eingabe
  Diese Regeln müssen vollständig sein,
  d. h. für jede Eingabe muß eine Reaktion des Automaten spezifiziert sein.
  Notfalls wird ein Fehlerzustand definiert.

Endliche Automaten

- Ein endlicher Automat ist ein mathematisches Modell eines Systems mit Ein- und Ausgaben.
- Ein solches System befindet sich immer in einer aus einer endlichen Anzahl möglicher interner Konfigurationen.
- Man sagt auch: das System befindet sich in einem Zustand.
- Auch ein Computer ist ein endlicher Automat. Allerdings ist hier das Modell wegen der großen Anzahl möglicher Zustände nicht besonders hilfreich.

Der endliche Automat

Endlicher Automat (*finite state automaton* = FSA):

- Z: endliche Menge von n Zuständen $z_0, z_1, \ldots, z_n$
- Σ: ein endliches Eingabealphabet von Symbolen
- z_0: ein Startzustand
- $E \subseteq Z$: die Menge der Endzustände
- $\delta(z, e)$: die Übergangsfunktion bzw. Übergangsmatrix zwischen Zuständen
  Für einen Zustand $z \in Z$ und ein Eingabesymbol e in Σ liefert $\delta(z, e)$ einen Zustand $z' \in Z$.

- Wenn Übergangsfunktion eindeutig ist:
  deterministischer endlicher Automat (DFSA).

DFSA als Akzeptor

Wird die Eingabekette $w = e_1, e_2, \ldots, e_n$ durch einen Automaten A akzeptiert?

Definition zweier Variablen: z für den gegenwärtigen Zustand und i als Zähler

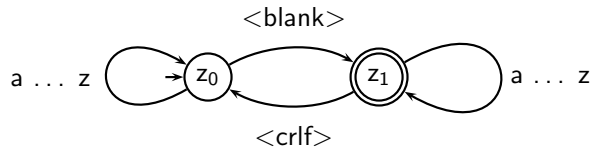
Wir setzen $z := z_0, i := 1$ und wiederholen für jedes $i \leq n$

$z := \delta(z, e_i)$

$i := i + 1$

Wenn $i = n$ und $z \in F$, dann ist die Eingabekette durch A akzeptiert, ansonsten gilt sie als zurückgewiesen.

Beispiel: Liste von Vor- und Nachnamen



Max Meier
.
.
.
Steffen Pietsch

Übergangstabelle:

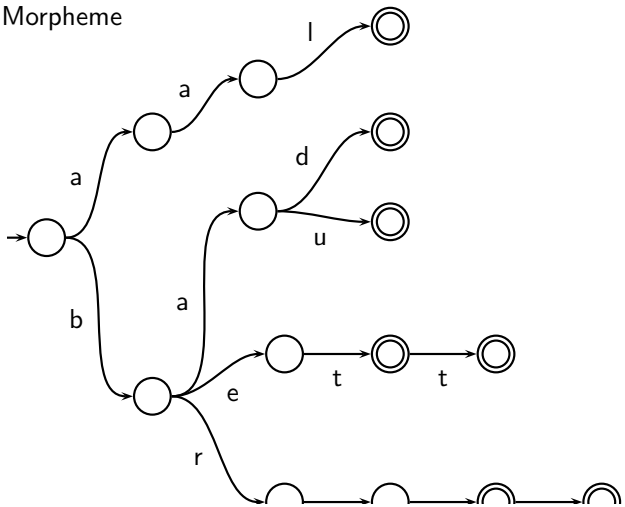
	z_0	z_1
a	z_0	z_1
.	z_0	z_1
.	z_0	z_1
.	z_0	z_1
z	z_0	z_1
<blank>	z_1	
<crlf>		z_0

Freie Universität Berlin

Buchstabenbaum: Stämme

Verbstamm-Morpheme

aal
bad
bau
bet
bett
brau
braus

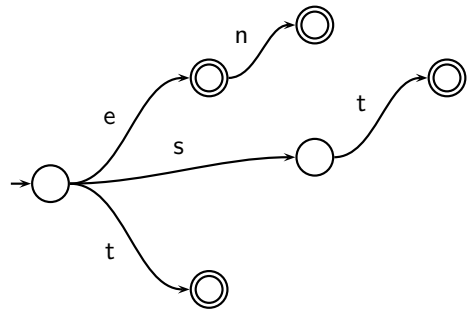


Freie Universität Berlin

Buchstabenbaum: Endungen

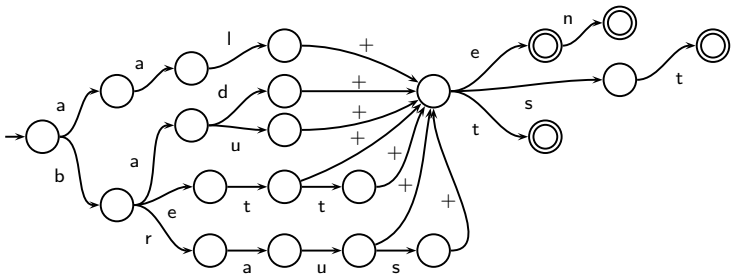
Verbsuffix-Morpheme

-e
-en
-st
-t



Freie Universität Berlin

Buchstabenbaum: Verkettung von Stamm und Endung



- Verknüpfung zweier Automaten A_1 und A_2:
  - der Anfangszustand des neuen Automaten ist der Anfangszustand von A_1
  - die Endzustände des ersten Automaten A_1 sind im neuen Automaten keine Endzustände mehr
  - von jedem Endzustand in A_1 geht eine Kante mit '+' zum Anfangszustand von A_2
  - '+' dient dazu, die Morphemgrenze für morphophonologische Veränderungen wiederzufinden
- *bad+st* ist falsch: *e* fehlt, muß an Morphemgrenze eingefügt werden, dazu Automat mit Ausgabe

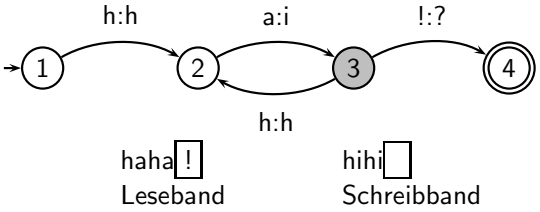
Freie Universität Berlin

Automat mit Ausgabe

Mealey-Automat (*finite state transducer* = FST)

Endliche Automaten, die zusätzlich zum Lesen auch noch Schreiben können, werden oft als Transduktoren (transducer) bezeichnet.
Transduktoren können die beim Verarbeiten durchlaufenen Schritte nach Außen kommunizieren, ohne den Formalismus aufzubrechen!

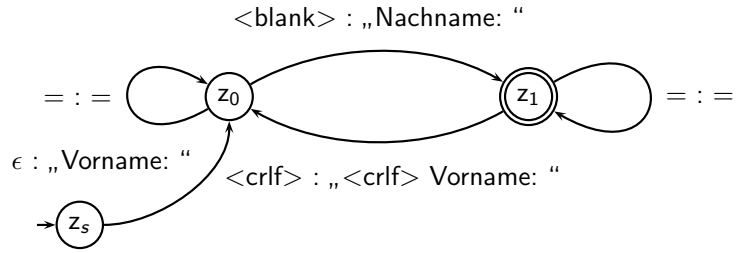
Lach-Transduktor



Konventionen im Folgenden:

- =:= steht für alle anderen Symbole, die nicht explizit im Automaten erwähnt sind (*other*)
- einzelne Buchstaben stehen für identische Ein-/Ausgabe (x ≡ x:x)
- Das Epsilonsymbol ϵ steht für das leere Symbol.
  Statt ϵ schreibt man auch 0.

Vor- und Nachnamen



Vorname: Max Nachname: Meier
.
.
.
Vorname: Steffen Nachname: Pietsch

Übergangstabelle:

	z_s	z_0	z_1
ϵ : „Vorname: “	z_0		
= : =		z_0	z_1
<blank> : „ Nachname: “		z_1	
<crlf> : „<crlf> Vorname: “			z_0

Ersetzungsregeln

- *e*-Epenthese an der Morphemgrenze: +:e ↔ {d, t} _ {s, t}

  (11) a. geh + st
  b. red + st → redest
  c. bad + st → badest

  Das Morphemgrenzenzeichen wird zu *e*,
  wenn es sich nach einem *d* oder *t* und vor einem *s* oder *t* befindet.
- *e*-Ellision nach *e*: e:0 ↔ _ + e

  (12) a. schön + er
  b. schlau + er
  c. leise + er → leiser

  Ein *e* wird gelöscht,
  wenn es sich vor einer Morphemgrenze und einem weiteren *e* befindet.
- alternative Schreibweise: a → b / c _ d

Zwei-Ebenen-Morphologie

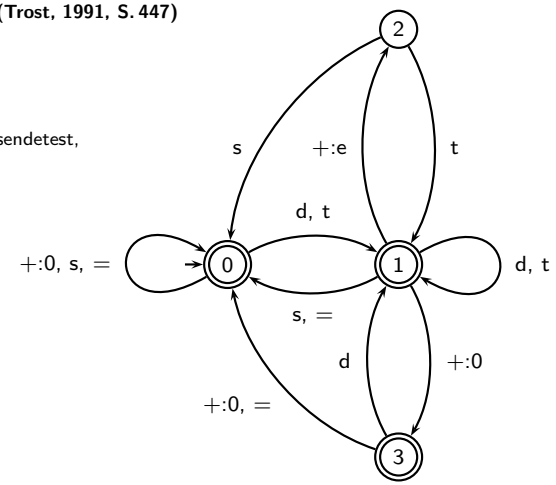
e-Epenthese an der Morphemgrenze (Trost, 1991, S. 447) (vereinfacht)

Regel: +:e ↔ {d, t} _ {s, t}
sag+t+e → sagte,
send+t+e → sendete, send+t+st → sendetest,

Übergangstabelle:

	0	1	2	3
+:e	–	2	–	–
+:0	0	3	–	0
d:d	1	1	–	1
s:s	0	0	0	–
t:t	1	1	1	–
=:=	0	0	–	0

Achtung: =:= enthält hier auch e:e,
obwohl 'e' im Automaten vorkommt



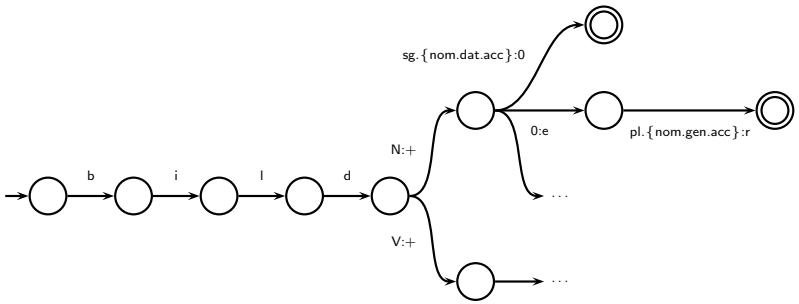
Freie Universität Berlin

Lexikonzugriff

- Aufgabe 1: Wir haben ein Wort und wollen den Stamm und die Flexionsmerkmale. (Analyse)
- Aufgabe 2: Wir haben einen Stamm und Flexionsmerkmale und wollen die Form des Wortes. (Generierung)

  B i l d N:pl.{nom.gen.acc}
  B i l d + e r

  b i l d V:{1.3}.pl
  b i l d + e n



Freie Universität Berlin

Vom Lexikonzugriff zur Oberflächenform

- Lexikonzugriff liefert Stämme, Morphemgrenzen und Endungen
- Morphemgrenzen werden für die Oberflächenform entfernt
- dabei werden entsprechende Veränderungen an den Morphemgrenzen durchgeführt

Lexikon:	b	a	d	V:2.sg		
Zwischenstufe:	b	a	d	+	s	t
Oberfläche:	b	a	d	e	s	t

- Zwischen den jeweiligen Repräsentationen sind FST.
- Automaten sind in beide Richtungen verwendbar:
  Oberfläche → Lexikon und Lexikon → Oberfläche.

Freie Universität Berlin

Übung

Entwickeln Sie einen Automaten, der folgende Zeichenfolgen akzeptiert:

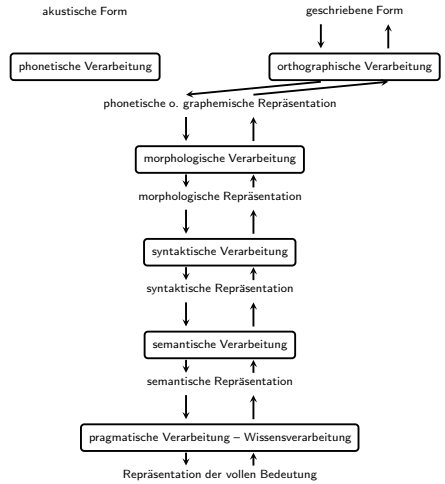
bla!	bla!!	bla!!!	...
blabla!	blabla!!	blabla!!!	...
blablabla!	blablabla!!	blablabla!!!	...
...			

Entwickeln Sie einen Transduktor, der die obigen Zeichenfolgen in die unten aufgeführten überführt:

blub?	blub??	blub???	...
blubblub?	blubblub??	blubblub???	...
blubblubblub?	blubblubblub??	blubblubblub???	...
...			

Freie Universität Berlin

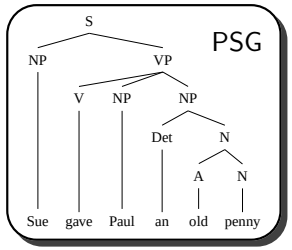
Maschinelle Übersetzung

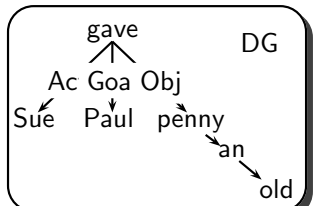


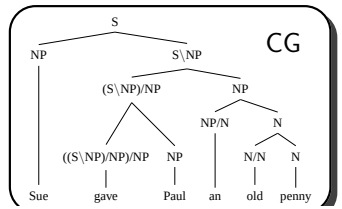
Syntaktische Verarbeitung

- **Eingabe:** Kette von Wortrepräsentationen
- **Verarbeitung:** Parsing d. h. Analyse der Eingabekette mit dem Ziel, eine Repräsentation der Satzstruktur zu erhalten, die sich als Eingabe für die semantische Verarbeitung eignet.
  Parsingverfahren: top-down, bottom-up
  links-rechts, rechts-links, Inselparsing
  Stackparsing, Chartparsing
  atomare oder komplexe Kategorien
- **Ausgabe:** Repräsentation des Satzes, die die semantisch relevante Struktur und die semantisch relevanten syntaktischen Merkmale enthält.

Drei traditionelle Modelle







Gliederung

- Grundlagen zu Phrasenstrukturgrammatiken
- Eine Grammatik für das Deutsche
- Parsing

Phrasenstrukturgrammatiken

- Sprache als Menge von Zeichenketten
- Zeichketten eine Struktur zuordnen → Grammatiken
- Zusammenhänge erkennen

- Fragen:
  - Wie komplex müssen die Grammatiken sein?
  - Wie komplex müssen die Algorithmen zur Verarbeitung sein?
- zur Beantwortung:
  - Einteilung von Grammatiken in Klassen
  - Untersuchung natürlicher Sprachen auf Klassenzugehörigkeit
  - Details werden in der Vorlesung *Computationelle Syntax* behandelt.

Grundlagen der Mengentheorie

- Menge = Ansammlung von Objekten
- Objekte werden Elemente genannt
- Mengen schreibt man als Aufzählung ihrer Elemente in geschweiften Klammern auf
  Beispiel: Menge der deutschen Wochentage:
  $M_{Woche} =$
  $\{montag, dienstag, mittwoch, donnerstag, freitag, sonnabend, sonntag\}$
- $\in$ ist Abkürzung für *ist Element von*
  $montag \in M_{Woche}$
  $dienstag \in M_{Woche}$
  $mittwoch \in M_{Woche}$
  ...
- Frage *Wie oft ist x in S enthalten?* ist nicht sinnvoll
- Objekt kann nur Element einer Menge sein oder nicht, kommt nicht mehrfach vor

Eigenschaften, Darstellung und Definition von Mengen

- Mengen sind ungeordnet.
- Zwei Mengen sind gleich, wenn sie dieselben Elemente enthalten.
  $\{1, 2, 3\}$
  $\{2, 3, 1\}$
  $\{3, 2, 1\}$
- Computerdarstellung oft als Listen, Listen sind geordnet:
  effiziente Berechnung von Teilmengen bzw. Enthaltenseinsbeziehungen
- für Definition muß man nicht unbedingt alle Elemente aufzählen:
  Definition einer Menge über Eigenschaften der Elemente
  $M_{Woche} = \{x | x \text{ ist die deutsche Bezeichnung für einen Wochentag}\}$
  = *Die Menge aller* x, *wobei* x *die deutsche Bezeichnung eines Wochentags ist.*

Operationen über Mengen: *Durchschnitt* und *Vereinigung*

- Durchschnitt zweier Mengen S_1 und S_2 ist die Menge der Objekte, die zu beiden Mengen gehören
  $S_1 \cap S_2 = \{x | x \in S_1 \textit{ und } x \in S_2\}$
  Beispiel:
  $\{1, 2, 3, 5\} \cap \{2, 4, 6, 8\} = \{2\}$
- Die Vereinigung zweier Mengen ist die Menge der Objekte, die in beiden Mengen vorkommen:
  $S_1 \cup S_2 = \{x | x \in S_1 \textit{ oder } x \in S_2\}$
  Beispiel:
  $\{1, 2, 3, 5\} \cup \{2, 4, 8\} = \{1, 2, 3, 4, 5, 8\}$
- Menge S_1 ist eine Teilmenge einer anderen Menge S_2, $S_1 \subseteq S_2$, wenn jedes Element von S_1 auch Element von S_2 ist.
- Jede Menge ist auch Teilmenge von sich selbst.
- Ansonsten spricht man von einer echten Teilmenge ($\subset$)

Grammatik mit Ersetzungsregeln

Beispielgrammatik für Telefonnummern, die mindestens dreistellig sind:

tel_nr $\rightarrow$ ziffer ziffer ziffer ziffern
ziffern $\rightarrow$
ziffern $\rightarrow$ ziffer ziffern
ziffer $\rightarrow$ 0
...
ziffer $\rightarrow$ 9

Sprachen (I)

- *Vokabular*: Menge von Symbolen, die zu Phrasen verknüpft werden können
  $V_{Deutsch} = \{Aachen, Aal, \ldots, Zytoplasma\}$
- $S*$ = Menge, die alle möglichen endlichen Sequenzen von S enthält:
  $V_{Deutsch}*$ =
  {
  Aal Aal Aal Aal Aal Zytoplasma Aal,
  Aachen Zytoplasma fliegt Aal Aal ihn,
  Er hilft dem Mann
  Das klingt gut
  ...
  }
  Der leere String ϵ ist ebenfalls Element dieser Menge.
- Nicht jedes Element der Menge $V_{Deutsch}*$ ist eine vernünftige Phrase.

Sprachen (II)

- Sprache über dem Vokabular S als Teilmenge von $S*$ mathematisch definiert
- Deutsch ist Teilmenge von $V_{Deutsch}*$:
  {
  Er hilft dem Mann
  Das klingt gut
  ...
  }
- Nacheinanderschreibung von α und β (Elemente des Vokabulars oder Sequenzen) bedeutet die Verkettung der betreffenden Phrasen
  $\alpha\ \beta$

  n-fache Wiederholung von α:
  α^n
- Schriftsprache: Leerzeichen um das Ende eines Wortes und den Anfang des nächsten zu markieren

Formale Systeme zur Definition von Sprachen

bestehen aus zwei Bestandteilen:

1. Eine Charakterisierung des Wissens über eine Sprache, das von Sprache zu Sprache unterschiedlich ist.
2. Eine Spezifikation, wie man mit einem bestimmtem Teil dieses Wissens das Enthaltensein eines bestimmten Satzes in einer Sprache feststellen kann. Mit Spezifikation kann man eine Prozedur (endliche Sequenz explizit formulierter Anweisungen) schreiben, die mechanisch ausführbar ist. Prozedur sollte ein Algorithmus sein, der auf allen Eingaben terminiert (entscheidbar).

Grammatiken

Definition Eine formale Grammatik G = [S,T,N,R] besteht aus vier Komponenten:

- T einer endlichen Menge von Terminalsymbolen,
- N einer endlichen Menge von Nichtterminalsymbolen,
- R einer endlichen Menge von Regeln der Form $\alpha \rightarrow \beta$, wobei α und $\beta \in (N \cup T)*$ sind und
- S einem besonderen Element von N – dem Startsymbol.

Definition: Satzform

- S ist eine Satzform, wenn S Startsymbol ist.
- Wenn x eine Satzform der Form $\alpha\beta\gamma$ ist und es eine Regel der Form $\beta \rightarrow \delta$ gibt, dann ist $\alpha\delta\gamma$ eine Satzform.

Definition Eine Satzform, die nur Terminalsymbole enthält wird Satz genannt. Die Menge aller Sätze, die eine Grammatik G beschreibt, ist die Sprache L(G).

Beispiel

Grammatik für Telefonnummern, die mindestens dreistellig sind:

N = {tel_nr, ziffer, ziffern}
T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
S = tel_nr
R = {
tel_nr → ziffer ziffer ziffer ziffern

ziffern →
ziffern → ziffer ziffern

ziffer → 0
...
ziffer → 9
}

Anwendung der Ersetzungsregeln

„1 2 3“ ist ein Satz der oben definierten Sprache,
da die folgenden Sequenzen Satzformen sind:

tel_nr
ziffer ziffer ziffer ziffern
1 ziffer ziffer ziffern
1 2 ziffer ziffern
1 2 3 ziffern
1 2 3

erste Sequenz nach erstem Teil der Definition für Satzform
andere Sequenzen sind jeweils aus den vorigen Sequenzen abgeleitet (zweiter Teil der Definition)

Übung

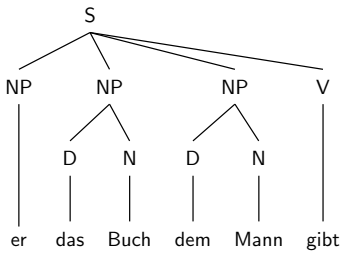
Schreiben Sie eine Grammatik, die aus Vornamen und Familiennamen bestehende Namen analysieren kann.

Eine Grammatik des Deutschen

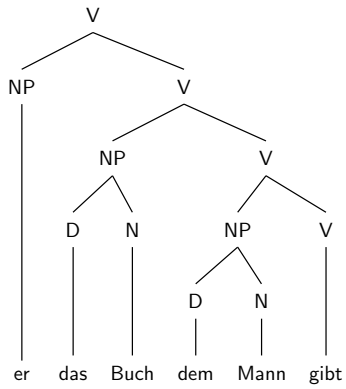
- passende Menge von Bezeichnungen für Kategorien der Wörter bzw. Wortgruppen
- Hauptkategorien sind:

Verb (Tätigkeitswort):	bezeichnen in der Zeit verlaufende Phänomene: Tätigekiten, Vorgänge, Zustände morphologisch: Konjugation, Person, Numerus, Modus, Tempus Valenzbeziehungen → Zentrum des Satzes Kongruenz mit Subjekt
Nomen (Ding-, Gegenstandswort):	morphologisch: Genus, Numerus, Kasus
Präposition (Fallfügteil, Verhältniswort):	nicht flektiert, nicht allein
Adjektiv (Beiwort, Eigenschaftswort):	morphologisch: Deklination u. Komparation Kongruenz mit Nomen

Verschiedene vorstellbare Strukturen



NP → D, N
S → NP, NP, NP, V



NP → D, N
V → NP, V

Beispielableitung bei Annahme flacher Strukturen

NP → D, N
S → NP, NP, NP, V

NP → er
D → das
D → dem

N → Buch
N → Mann
V → gab

er	das	Buch	dem	Mann	gab	
NP	das	Buch	dem	Mann	gab	NP → er
NP	D	Buch	dem	Mann	gab	D → das
NP	D	N	dem	Mann	gab	N → Buch
NP		NP	dem	Mann	gab	NP → D, N
NP		NP	D	Mann	gab	D → dem
NP		NP	D	N	gab	N → Mann
NP		NP		NP	gab	NP → D, N
NP		NP		NP	V	V → gab
					S	S → NP, NP, NP, V

Von der Grammatik beschriebene Sätze

- die Grammatik ist zu ungenau:
  NP → D, N
  S → NP, NP, NP, V

(13) a. er das Buch dem Mann gibt.
b. * ich das Buch dem Mann gibt.
(Subjekt-Verb-Kongruenz *ich*, *gibt*)
c. * er das Buch den Mann gibt.
(Kasusanforderungen des Verbs *gibt* verlangt Dativ)
d. * er den Buch dem Mann gibt.
(Determinator-Nomen-Kongruenz *den*, *Buch*)

Subjekt-Verb-Kongruenz (I)

- Übereinstimmung in Person (1, 2, 3) und Numerus (sg, pl)

(14) a. Ich schlafe. (1, sg)
b. Du schläfst. (2, sg)
c. Er schläft. (3, sg)
d. Wir schlafen. (1, pl)
e. Ihr schlaft. (2, pl)
f. Sie schlafen. (3,pl)

- Wie drückt man das in Regeln aus?

Subjekt-Verb-Kongruenz (II)

- Verfeinerung der verwedenten Symbole
  aus S → NP, NP, NP, V wird

  S → NP_1_sg, NP, NP, V_1_sg
  S → NP_2_sg, NP, NP, V_2_sg
  S → NP_3_sg, NP, NP, V_3_sg
  S → NP_1_pl, NP, NP, V_1_pl
  S → NP_2_pl, NP, NP, V_2_pl
  S → NP_3_pl, NP, NP, V_3_pl
- sechs Symbole für Nominalphrasen, sechs für Verben
- sechs Regeln statt einer

Kasuszuweisung durch das Verb

- Kasus muß repräsentiert sein:
  S → NP_1_sg_nom, NP_dat, NP_acc, V_1_sg_ditransitiv
  S → NP_2_sg_nom, NP_dat, NP_acc, V_2_sg_ditransitiv
  S → NP_3_sg_nom, NP_dat, NP_acc, V_3_sg_ditransitiv
  S → NP_1_pl_nom, NP_dat, NP_acc, V_1_pl_ditransitiv
  S → NP_2_pl_nom, NP_dat, NP_acc, V_2_pl_ditransitiv
  S → NP_3_pl_nom, NP_dat, NP_acc, V_3_pl_ditransitiv
- insgesamt 3 * 2 * 4 = 24 neue Kategorien für NP
- 3 * 2 * x Kategorien für V (x = Anzahl der Valenzmuster)

Determinator-Nomen-Kongruenz

- Übereinstimmung in Genus (fem, mas, neu), Numerus (sg, pl) und Kasus (nom, gen, dat, acc)

  (15) a. der Mann, die Frau, das Buch (Genus)
  b. das Buch, die Bücher (Numerus)
  c. des Buches, dem Buch (Kasus)
- aus NP → D, N wird

NP_3_sg_nom → D_fem_sg_nom, N_fem_sg_nom	NP_gen → D_fem_sg_gen, N_fem_sg_gen
NP_3_sg_nom → D_mas_sg_nom, N_mas_sg_nom	NP_gen → D_mas_sg_gen, N_mas_sg_gen
NP_3_sg_nom → D_neu_sg_nom, N_neu_sg_nom	NP_gen → D_neu_sg_gen, N_neu_sg_gen
NP_3_pl_nom → D_fem_pl_nom, N_fem_pl_nom	NP_gen → D_fem_pl_gen, N_fem_pl_gen
NP_3_pl_nom → D_mas_pl_nom, N_mas_pl_nom	NP_gen → D_mas_pl_gen, N_mas_pl_gen
NP_3_pl_nom → D_neu_pl_nom, N_neu_pl_nom	NP_gen → D_neu_pl_gen, N_neu_pl_gen
... Dativ	... Akkusativ

- 24 Symbole für Determinatoren, 24 Symbole für Nomen
- 24 Regeln statt einer

Probleme dieses Ansatzes

- Gernalisierungen werden nicht erfaßt
- weder in Regeln noch in Kategoriesymbolen
  - Wo kann eine NP oder NP_nom stehen? Nicht wo kann eine NP_3_sg_nom stehen?
  - Gemeinsamkeiten der Regeln sind nicht offensichtlich.
- Lösung: Merkmale mit Werten und Identität von Werten
  Kategoriesymbol: NP Merkmal: Per, Num, Kas, ...
  Wir erhalten z. B. die Regeln:
  NP(3,sg,nom) → D(fem,sg,nom), N(fem,sg,nom)
  NP(3,sg,nom) → D(mas,sg,nom), N(mas,sg,nom)

Merkmale und Regelschemata (I)

- Regeln mit speziellen Werten zu Regelschemata verallgemeinern:

  NP(3,Num,Kas) → D(Gen,Num,Kas), N(Gen,Num,Kas)
- Gen-, Num- und Kas-Werte sind egal,
  Hauptsache sie stimmen überein (identische Werte)
- der Wert des Personenmerkmals (erste Stelle in NP(3,Num,Kas))
  ist durch die Regel festgelegt: 3

Merkmale und Regelschemata (II)

- Regeln mit speziellen Werten zu Regelschemata verallgemeinern:

  NP(3,Num,Kas) → D(Gen,Num,Kas), N(Gen,Num,Kas)
  S → NP(Per1,Num1,nom),
  NP(Per2,Num2,dat),
  NP(Per3,Num3,akk),
  V(Per1,Num1)

- Per1 und Num1 sind beim Verb und Subjekt gleich.
- Bei anderen NPen sind die Werte egal. (Schreibweise für irrelevante Werte: '_')
- Die Kasus der NPen sind in der zweiten Regel festgelegt.

Bündelung von Merkmalen

- Kann es Regeln geben, in denen nur der Per-Wert oder nur der Num-Wert identisch sein muß?

  S → NP(Per1,Num1,nom),
  NP(Per2,Num2,dat),
  NP(Per3,Num3,akk),
  V(Per1,Num1)

- Gruppierung von Information → stärkere Generalisierung, stärkere Aussage

  S → NP(Agr1,nom),
  NP(Agr2,dat),
  NP(Agr3,akk),
  V(Agr1)

  wobei Agr ein Merkmal mit komplexen Wert ist: z. B. agr(1,sg)

Köpfe

Kopf bestimmt die wichtigsten Eigenschaften einer Projektion

(16) a. Karl schläft.
b. Karl liebt Maria.
c. in diesem Haus
d. ein Mann

ein Satz ist die Maximalprojektion eines finiten Verbs

Hauptkategorien sind:

Kategorie	proijzierte Merkmale	Beispiel
Verb	Kategorie, Finitheitsmerkmale	hat, schläft, lieben
Nomen	Kategorie, Kasus, Numerus, Genus	Mann, Blume, Kind
Präposition	Kategorie, Form der Präposition	an, auf, in, neben
Adjektiv	Kategorie, Kasus, Numerus, Genus	klug, schön, faul

Argumente vs. Adjunkte

- Beispiele für Adjunkte:
  Adjektive eine schöne Frau
  Relativsätze der Mann, den Maria liebt
  der Mann, der Maria liebt
  Adverbien Karl schnarcht laut.

- Adjunkte leisten keinen Betrag zur Kernbedeutung des Kopfes.
- Adjunkte sind mit größeren Klassen von Wörtern o. Phrasen kombinierbar.
- Adjunkte sind optional.
- Adjunkte sind iterierbar.

  (17) a. * Der Mann der Mann schläft.
  b. eine schöne kluge Frau

- Argumente werden unterteilt in Subjekt und Komplemente.

Abstraktion über Regeln: $\overline{\text{X}}$-Schemata

$\overline{\text{X}}$-Theorie (Jackendoff, 1977):

$\overline{\text{X}}$-Regel	mit Kategorien	Beispiel
$\overline{\overline{\text{X}}} \rightarrow \overline{\text{Spezifikator}}\ \overline{\text{X}}$	$\overline{\overline{\text{N}}} \rightarrow \overline{\text{DET}}\ \overline{\text{N}}$	das [Bild von Maria]
$\overline{\text{X}} \rightarrow \overline{\text{X}}\ \overline{\text{Adjunkt}}$	$\overline{\text{N}} \rightarrow \overline{\text{N}}\ \overline{\text{REL_SATZ}}$	[Bild von Maria] [das alle kennen]
$\overline{\text{X}} \rightarrow \overline{\text{Adjunkt}}\ \overline{\text{X}}$	$\overline{\text{N}} \rightarrow \overline{\text{ADJ}}\ \overline{\text{N}}$	schöne [Bild von Maria]
$\overline{\text{X}} \rightarrow \text{X}\ \overline{\text{Komplement}}*$	$\overline{\text{N}} \rightarrow \text{N}\ \overline{\text{P}}$	Bild [von Maria]

X steht für beliebige Kategorie, '*' für beliebig viele Wiederholungen

$\overline{\text{X}}$-Theorie wird in vielen verschiedenen Frameworks angenommen:
Chomsky, 1981; Bresnan, 1982; Gazdar, Klein, Pullum und Sag, 1985

$\overline{\text{X}}$-Schemata

- alternative Schreibweise mit Bar-Stufe als Merkmal:

  X(2) → Spezifikator(2) X(1)
  X(1) → X(1) Adjunkt(2)
  X(1) → Adjunkt(2) X(1)
  X(1) → X(0) Komplement(2) *

- Konvention:
  - Bar-Level des X auf der rechten Seite ist gleich oder kleiner als das der linken
  - bei Regeln mit Merkmalen (Num, Per, o. ä.):
    Werte beim X links und rechts gleich
- X ist der Kopf, die Merkmale sind Kopfmerkmale
- Adjunktion verändert das Level nicht → Iteration möglich

Nominalphrasen (I)

- Subjekt oder Objekt (*der große rote Quader*)
- referieren normalerweise auf bestimmte Objekte oder Konzepte
- aber auch Expletiva = semantisch leer

  (18) a. Es geht um den Montag.
  b. Er bringt es bis zum Professor.

- zwei Wortklassen, die selbst Nominalphrasen sind:
  - Pronomina: normalerweise auf vorerwähnte Dinge referieren
    auch deiktisch oder im Diskurs vorgreifend (Kataphora z. B. in der Literatur).
  - Eigennamen (*proper nouns*) bezeichnen bestimmte Dinge, wohingegen normale Nomina Klassen von Dingen bezeichnen. Beispiel *Karl*, *Bremen*

  n(2,Per,Num,Kasus,Genus) → pronoun(Per,Num,Kasus,Genus)
  n(2,Per,sing,Kasus,Genus) → proper_noun(Per,sing,Kasus,Genus)

Nominalphrasen (II)

- Determinatoren sind Spezifikatoren von Nominalphrasen
  Artikel *der*, Quantoren *einige*, Possessiva *ihre*, Demonstrtiva *diese*

  n(2,Per,Num,Kasus,Genus) → det(Num,Kasus,Genus) n(1,Per,Num,Kasus,Genus)

  (19) einige Frauen
- im Plural kann Artikel entfallen:

  n(2,Per,plur,Kasus,Genus) → n(1,Per,plur,Kasus,Genus)

  (20) Er liebt **Bäume**.

Nominalphrasen (III)

- Modifikatoren vor dem Nomen (z. B. Adjektive)

  n(1,Per,Num,Kasus,Genus) → adj(2,Num,Kasus,Genus) n(1,Per,Num,Kasus,Genus)

  (21) kluge Frau
- oder danach (z. B. Präpositionalphrasen und Relativsätze)

  n(1,Per,Num,Kasus,Genus) → n(1,Per,Num,Kasus,Genus) post_mod

  (22) a. Frau aus Bremen
  b. Frau, die wir kennen
- viele Nomina nehmen keine Komplemente (*Stuhl*, *Haus*)

  n(1,Per,Num,Kasus,Genus) → n(0,Per,Num,Kasus,Genus)

Verbphrasen

- Verbphrase = (infinites) Verb + dazugehörige Objekte

  (23) a. dem Mann ein Buch zu geben
  b. zu schlafen

  v(2,Vform) → v(1,Vform)
  v(1,Vform) → verb_mods v(1,Vform)
  v(1,Vform) → v(0,Vform)
- Modifikation durch
  - Adverbien (*schnell*, *absichtlich*)
  - Präpositionalphrasen (*im Bett*)
- Stellungsmöglichkeiten von Adverbialen durch Regeln nicht korrekt erfaßt.
- *Vform* ist ein Kopfmerkmal → Selektion durch übergeordnetes Prädikat

Präpositionalphrasen

- Verb spezifiziert, welche Präposition in Präpositionalphrase vorkommen muß

  (24) a. Max denkt an die Prüfung.
  b. * Max denkt vor die Prüfung.

  → Merkmal *Pform*: pform = {an, auf, in, neben, ...}
- Kasus muß zur Präposition passen, Verb verlangt Präposition mit NP in bestimmtem Kasus

  (25) a. Karl wartet auf den Installateur.
  b. Karl wartet auf dem Installateur.

  p(2,Pform,Kasus) → p(1,Pform,Kasus)
  p(1,Pform,Kasus) → p(0,Pform,Kasus) n(2,_,_,Kasus,_)
- Person und Numerus der NP sind egal
  '_' = nicht spezifizierter Wert ≅ alle für das entsprechende Merkmal möglichen Werte

Adjektivphrasen

- oft bestehen APen nur aus dem Adjektiv

  (26) Sie ist **klug**.

- aber auch mit NP oder PP

  (27) a. seiner Frau treu
  b. auf das Ergebnis stolz

- Gradangaben:

  (28) Sie ist **sehr** schön.

a(2,Kasus,Num,Genus) → deg a(1,Kasus,Num,Genus)
a(1,Kasus,Num,Genus) → adv a(1,Kasus,Num,Genus)
a(1,Kasus,Num,Genus) → a(0,Kasus,Num,Genus)

Sätze (I)

- Satzregel für Grammatiken des Englischen in $\overline{\text{X}}$-Schreibweise

  s → n(2,Per,Num,nom,_) v(2,fin,Per,Num)

  (Kategorie 'S' fällt aus $\overline{\text{X}}$-Schema raus, deshalb werden in neueren Arbeiten zusätzliche Kategorien IP & CP verwendet.)
- Subjekt-Verb-Kongruenz:
  Person und Numerus kommen vom Verb den Kopfpfad entlang zur VP
- für das Deutsche ist s → np,vp fragwürdig:
  - Stellung des Subjekts:

    (29) a. weil das Buch keiner liest.
    b. Deshalb gab gestern der Frau keiner das Buch.

  - subjektlose Konstruktionen:

    (30) a. Dem Student graut vor der Prüfung.
    b. Dort wird noch gearbeitet.

Sätze (II)

- Alternativ:
  kombinieren Verb gleich mit allen abhängigen Elementen,
  z. B. transitives Verb:
  s(initial) → v(0,fin,Per,Num) n(2,Per,Num,nom,_) n(2,_,_,akk,_)
  s(final) → n(2,Per,Num,nom,_) n(2,_,_,akk,_) v(0,fin,Per,Num)

- Unterscheidung der zwei Stellungen: Verbintial- und Verbfinalstellung

Komplemente und Adjunkte

- Komplemente können obligatorisch sein, Adjunkte nie
- Komplemente dürfen nur einmal auftreten
- Adjunkte modifizieren normalerweise Kategorien einer bestimmten Klasse
- Komplemente sind wort- bzw. wortklassenspezifisch

  (31) a. Karl kennt den Mann.
  b. * Karl kennt an den Mann.
  c. Karl denkt an den Mann.

Spezifikation möglicher Komplemente im Lexikon

- allgemeine Idee: Lexikoneintrag verlangt bestimmte Komplemente
- Art der möglichen Komplemente eines Wortes wird im Lexikon spezifiziert (*Subcat*-Merkmal)
- Auswahl einer bestimmten Grammatikregel über das *Subcat*-Merkmal
  subcat = {intrans, trans, emotion, meinungsaustausch,...}
- Unterschiede in Grammatiken in Bezug auf Bar-Level

Subkategorisierung und Phrasenstrukturregeln (I)

- Komplemente sind immer Maximalprojektionen
  (phrasale Kategorien mit Bar-Level 2)

```
n(1,Per,Num,Kasus,Genus) → n(0,mitPP_ueberPP,Per,Num,Kasus,Genus) p(2,mit,dat) p(2,über,akk)   % Streit
n(1,Per,Num,Kasus,Genus) → n(0,zuInf,Per,Num,Kasus,Genus) v(2,inf,_)                            % Plan

a(1,Per,Num,Kasus,Genus) → p(2,auf,akk) a(0,aufPP,Per,Num,Kasus,Genus)                          % stolz
a(1,Per,Num,Kasus,Genus) → n(2,_,_,dat,_) a(0,datNP,Per,Num,Kasus,Genus)                        % treu

v(1,Vform,Per,Num)       → n(2,_,_,dat,_) n(2,_,_,akk_) v(0,ditrans,Vform,Per,Num)              % geben
v(1,Vform,Per,Num)       → n(2,_,_,akk,_) v(2,inf,_) v(0,obj_control,Vform,Per,Num)             % zwingen
```

- Kopf schränkt Menge der mgl. Komplemente ein
- bildet nur mit einer bestimmten Menge von Komplementen eine wohlgeformte und sinnvolle Phrase

Subkategorisierung und Phrasenstrukturregeln (II)

- Kopf selegiert nach syntaktischen und auch nach semantischen Kriterien:
  - *töten* braucht lebendes Objekt, Subjekt kann belebt oder unbelebt sein
  - *morden* verlangt ein Subjekt, das für seine Handlung verantwortlich sein kann
- Wörter können mit verschiedenen Subkategorisierungsmustern auftreten:

  (32) a. Er kocht die Suppe.
  b. Die Suppe kocht.

- Alternativen einfach ins Lexikon schreiben bzw.
  über Lexikonregeln eine oder beide Formen ableiten

Unzulänglichkeiten dieser Grammatik

kontextfreie Grammatik → viele Dinge, die nicht korrekt modelliert wurden:

- (Morphologie)
- Konstituentenstellung
- Subkategorisierungsregelmäßigkeiten
- Fernabhängigkeiten
- Koordination

Morphologie

- vollständig flektierte Wörter im Lexikon
- Einträge für *schläft*, *schläfst*, ... stehen in keiner Beziehung zueinander
- keine allgemeinen Prinzipien für Flexion →
  großes, redundantes, schlecht wartbares Lexikon
- keine derivationelle Morphologie: *Ableitbarkeit* = *ab* + *leit* + *bar* + *keit*

Konstituentenstellung

- Kombination von Symbolen nur in der in Regeln vorgegebenen Reihenfolge
- für das Englische kein Problem, da die Konstituentenfolge relativ festgelegt ist
- gibt jedoch Sprachen mit sehr viel freierer Wortstellung
- Deutsch: diskontinuierliche Komplemente
  Teile von Maximalprojektionen können abwechselnd mit Elementen auftreten, die von anderen Köpfen abhängen

  (33) weil es ihm jemand zu lesen versprochen hat. (Haider, 1990)

Subkategorisierungsregelmäßigkeiten nicht erfaßt (I)

- Komplemente durch *Subcat*-Merkmal bestimmt
- passende Regel der Form $v(1) \rightarrow v(subcat)\ldots$ wird gewählt
- mehrfache Lexikoneinträge für Verb mit verschiedenen *Subcat*-Werten, wenn Möglichkeiten nur für bestimmte Verben auftreten.

  (34) a. Er glaubt an ihn.
  b. Er glaubt das.
  c. Er glaubt zu träumen.

Subkategorisierungsregelmäßigkeiten nicht erfaßt (II)

- Änderungen des *Subcat*-Merkmals, die für fast alle Verben gleich sind
  z. B. Passiv:

  (35) a. Sie schlägt ihn.
  b. Er wird geschlagen.

- Wenn man für jede Regel mit v(1) auf der linken Seite eine Passivregel schreiben muß, ist der Grammatikformalismus nicht ausdrucksstark genug.

  v(1) → v(trans,fin) n(2) ...
  v(1) → v(intrans,pas) ...

Fernabhängigkeiten (*Nonlocal Dependencies*)

- Vorfeldbesetzung (Besetzung der Position vor dem finiten Verb):

  (36) [Über dieses Thema]$_i$ [hat er ihn gebeten] [[einen Vortrag $__i$] zu halten].

- Relativierung:

  (37) das Thema, [über das]$_i$ er Peter gebeten hat, [[einen Vortrag $__i$] zu halten],

- Nachfeldbesetzung (Verschiebung nach rechts hinter die Verben):

  (38) Der Mann [hat [der Frau] den Apfel gegeben], [die er am schönsten fand].

- Beschreibung mit Folge lokaler Abhängigkeiten
  Weitergabe von Informationen im Syntaxbaum von Knoten zu Knoten siehe Vorlesung: *Computationelle Syntax* und Müller, 1994.
  Auch HPSG-Vorlesung.

Koordination

- symmetrische Koordination:
  gleiche Konstituenten können durch Koordination verbunden werden

  (39) a. [der Mann und die Frau]
  b. dein [Freund und Arbeitskollege]
  c. Er [kennt und liebt] diese Schallplatte.
  d. Er ist [dumm und arrogant].

- → viele Regeln fehlen
- Regeln für jede Form von Konstituenten auf jedem Bar-Level nötig
- Vergrößerung der Grammatik → Generalisierung nicht ausgedrückt, Unübersichtlichkeit, keine Wartbarkeit

- Koordination von Satzteilen, die keine Konstituenten sind

Kriterien für einen guten Parser

Korrektheit Alle gefundenen Analysen sind mögliche Analysen.
manchmal unkorrekte Analysen erlaubt und hinter aussortiert

Vollständigkeit Jede mögliche Analyse wird gefunden.
Wenn Grammatik unendlich viele Analysen für einen String zuläßt →
Es gibt keine Analyse, die der Parser nicht irgendwann einmal findet.
Parser terminiert nicht, sondern gibt nach und nach die Analysen aus
Manchmal ist Vollständigkeit nicht notwendig (zeitkritische Anwendungen).
Alle zu einem bestimmten Zeitpunkt vorhandenen Ergebnisse werden berücksichtigt,
angenommene beste Analyse wird genommen

Effizienz Arbeit, die nur einmal getan werden muß, nicht wiederholen

Parsestrategien: zielgesteuert vs. datengesteuert

- zielgesteuert (*goal-driven*) vs. datengesteuert (*data-driven*)
- zielgesteuert:
  - beginne bei S, dem Startsymbol der Grammatik
  - suche Regel mit S auf der linken Regelseite
  - ersetze S durch die Symbole auf der rechten Regelseite
  - setze den Prozeß mit den Symbolen der rechten Regelseite fort
  - wenn die zu parsende Wortkette durch die Grammatik lizenziert wird, gelangt man eventuell zur Wortkette
- datengesteuert:
  - beginne bei den Wörtern
  - suche Regeln der Form PT → w
  - vergleiche Ketten von Präterminalen mit rechten Regelseiten
  - ersetze rechte Regelseite durch Symbol auf der linken Regelseite
  - wenn die zu parsende Wortkette durch die Grammatik lizenziert wird, gelangt man eventuell zu S

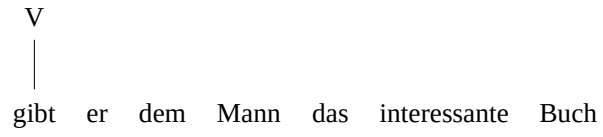
Parsestrategien: *Top-Down* vs. *Bottom-Up*

- Ableitungen werden in Syntaxbäumen repräsentiert →
  zielgesteuert ≡ von oben nach unten (*top-down*)
  datengesteuert ≡ von unten nach oben (*bottom-up*)

Bottom-Up Parsing (I)

S → V NP NP NP
NP → D N
N → A N
N → N PP

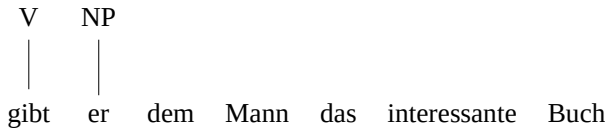
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Bottom-Up Parsing (II)

S → V NP NP NP
NP → D N
N → A N
N → N PP

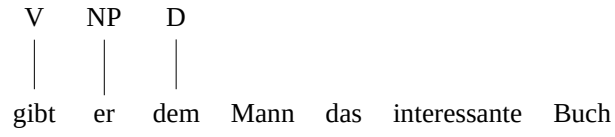
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Bottom-Up Parsing (III)

S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Bottom-Up Parsing (IV)

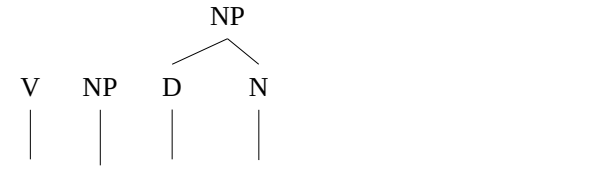
S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat

Bottom-Up Parsing (V)

S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Bottom-Up Parsing (VII)

S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat

Bottom-Up Parsing (VIII)

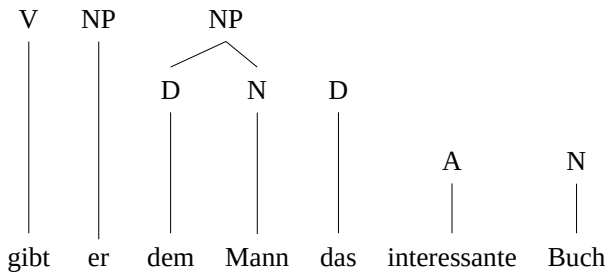
S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat

Bottom-Up Parsing (IX)

S → V NP NP NP
NP → D N
N → A N
N → N PP

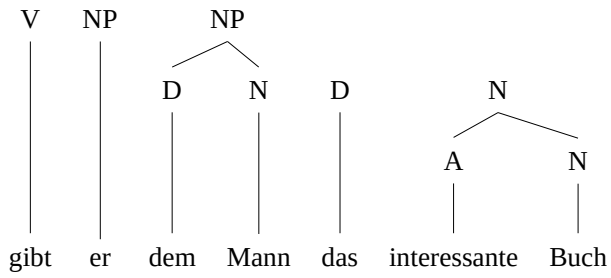
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Bottom-Up Parsing (X)

S → V NP NP NP
NP → D N
N → A N
N → N PP

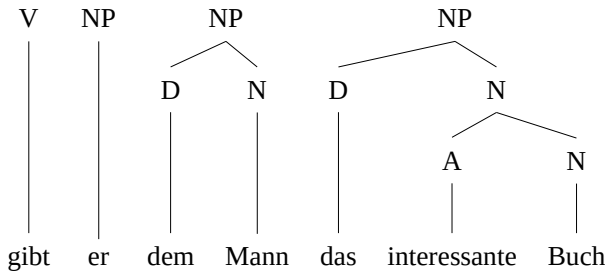
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Bottom-Up Parsing (XI)

S → V NP NP NP
NP → D N
N → A N
N → N PP

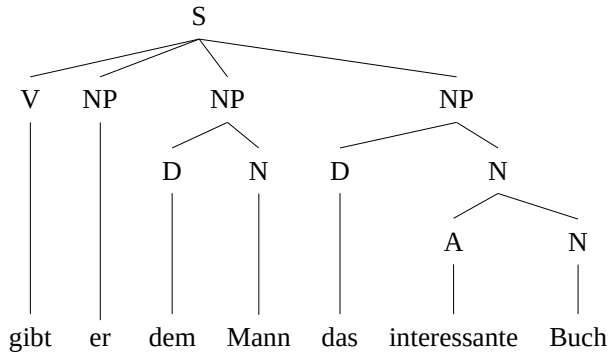
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Bottom-Up Parsing (XII)

S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Der Kellerspeicher (*Stack*)

- Keller:
  - Wir steigen in den Keller und legen etwas ab.
  - Das nächste, was abgelegt wird, wird oben aufgelegt.
  - Es entsteht ein Stapel.
  - Wir können nur von oben wieder wegnehmen.
  - Um an das Unterste zu gelangen, müssen wir alles aus dem Keller holen.
- formales Beispiel Liste:
  - Wir dürfen nur am Listenanfang Elemente hinzufügen bzw. wegnehmen
  - leere Liste $\langle\rangle$
  - einelementige Liste $\langle\ a\ \rangle$
  - zweielementige Liste $\langle\ b,\ a\ \rangle$
  - dreielementige Liste $\langle\ c,\ b,\ a\ \rangle$
  - an *a* kommen wir nicht direkt dran
    müssen erst *c* und *b* wieder abräumen
  - zweielementige Liste $\langle\ b,\ a\ \rangle$
  - einelementige Liste $\langle\ a\ \rangle$

Algorithmus: Bottom-Up-Parsen

Ein Zustand eines Shift-Reduce-Erkenners ist durch folgendes gekennzeichnet:

1. eine Sequenz von Nichtterminalen, die zum Anfang des Strings gefunden wurden
2. eine Sequenz von Wörtern, die noch nicht berücksichtigt wurden.

Sequenz der Symbole von erkannten Nichtterminalen = Stack

Erkenner schiebt die Kategorie eines neuen Wortes auf den Stack (*shift*), oder er reduziert die Symbole auf dem Stack, indem er eine Symbolsequenz auf dem Stack findet, die einer rechten Regelseite entspricht und diese Sequenz durch die linke Seite der Regel ersetzt (*reduce*).

Startsymbol *S* auf dem Stack und gleichzeitig die Wortsequenz leer $\rightarrow$ Eingabestring erfolgreich erkannt

Beispiel: Bottom-Up-Parsen

Stack	Wortsequenz
	Karl den Mann kennt
np	den Mann kennt
np d	Mann kennt
np d n	kennt
np np	kennt
np np v	
s	

zwei Entscheidungen:

- bei jedem Schritt entweder schieben oder reduzieren
- Entscheidung: welche Regel für das Wort und welche Regel beim Reduzieren

Probleme beim Bottom-Up-Parsen

- ϵ-Produktion
  Wenn Regel $C \rightarrow \epsilon$ existiert, kann man an jeder Stelle des Strings beliebig viele *C*s auf den Stack schieben.
- große lexikalische Ambiguität
  Kann das erste Wort im String zu 10 verschiedenen Kategorien gehören, so muß der Bottom-Up-Erkenner alle berücksichtigen und sucht auch nach größeren Phrasen, die die entsprechende Kategorie enthalten, auch wenn nur eine wirklich zu Beginn eines *S* möglich wäre.
- Kombination von Top-Down- und Bottom-Up-Parser kann diese Probleme verkleinern. (Left-Corner-Parser)

Top-Down-Parsing (I)

S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat

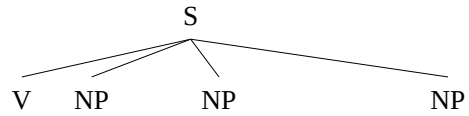
S

gibt er dem Mann das interessante Buch

Top-Down Parsing (II)

S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat

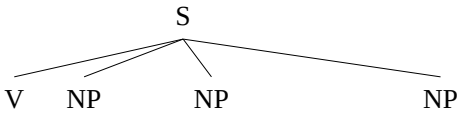


gibt er dem Mann das interessante Buch

Top-Down Parsing (III)

S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat

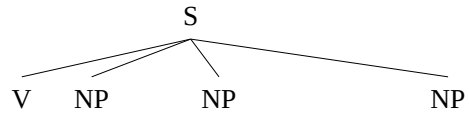


gibt er dem Mann das interessante Buch

Top-Down Parsing (IV)

S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat

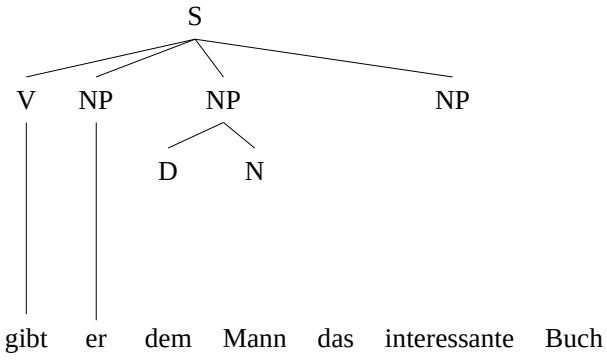


gibt er dem Mann das interessante Buch

Top-Down Parsing (V)

S → V NP NP NP
NP → D N
N → A N
N → N PP

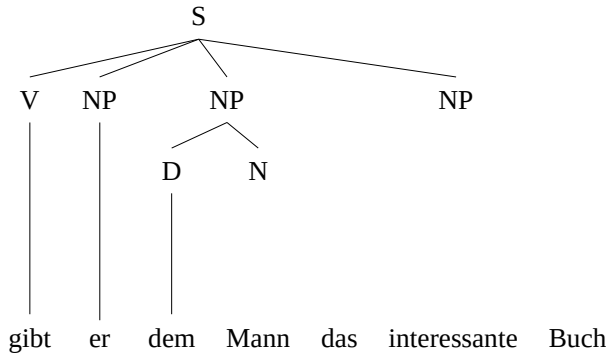
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Top-Down Parsing (VI)

S → V NP NP NP
NP → D N
N → A N
N → N PP

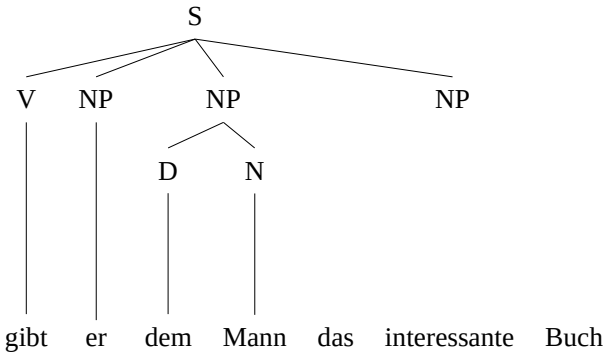
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Top-Down Parsing (VII)

S → V NP NP NP
NP → D N
N → A N
N → N PP

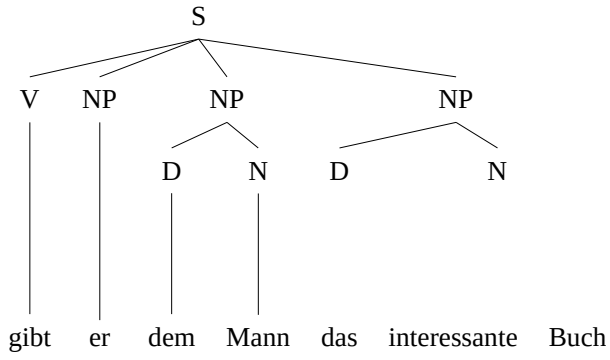
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Top-Down Parsing (VIII)

S → V NP NP NP
NP → D N
N → A N
N → N PP

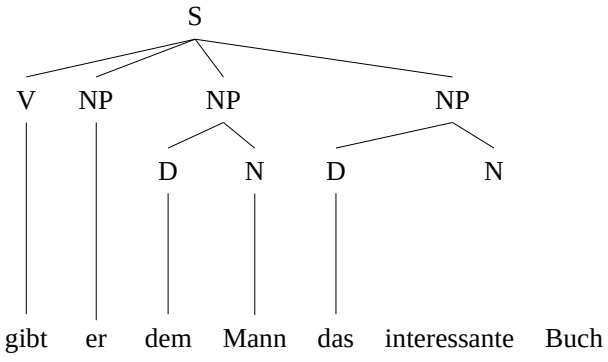
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Top-Down Parsing (IX)

S → V NP NP NP
NP → D N
N → A N
N → N PP

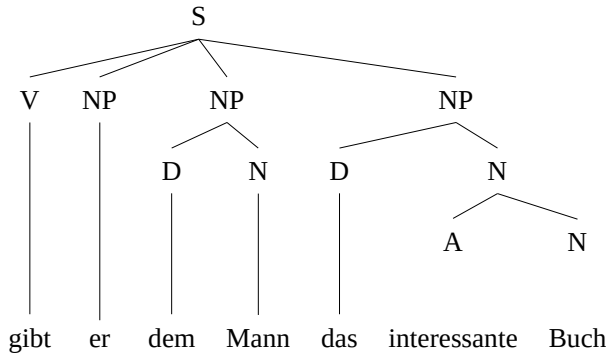
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Top-Down Parsing (X)

S → V NP NP NP
NP → D N
N → A N
N → N PP

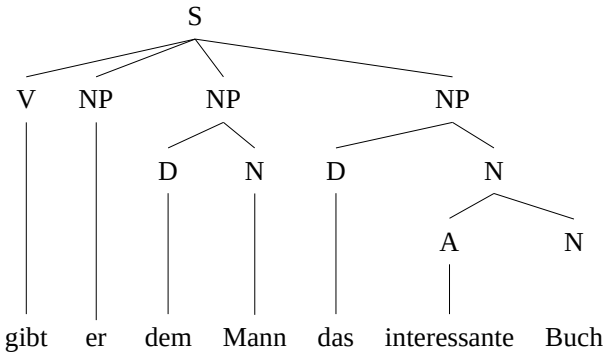
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Top-Down Parsing (XI)

S → V NP NP NP
NP → D N
N → A N
N → N PP

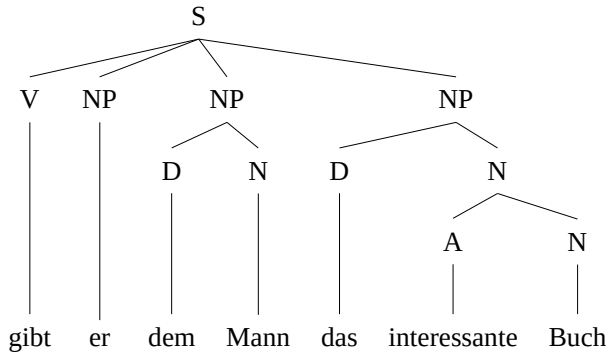
D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Top-Down Parsing (XII)

S → V NP NP NP
NP → D N
N → A N
N → N PP

D → das
D → dem
A → interessante
A → kluges
NP → er
N → Mann
N → Buch
V → gibt
V → schläft
V → hat



Algorithmus: Top-Down-Parsen

Zustand eines Top-Down-von-links-nach-rechts-Erkenners wird durch zwei Dinge beschrieben:

1. eine Sequenz von Zielen – bestimmte Nichtterminalsymbole, nach denen gesucht wird, und
2. die Wortsequenz, in der die Phrasen gefunden werden müssen.

Erkenner nimmt das erste Ziel aus der Liste,
sucht eine Grammatikregel, die Ziel-Kategorie auf der linken Seite hat,
und ersetzt das Ziel durch die Sequenz der Symbole auf der rechten Seite.

Wenn das erste Ziel-Symbol dem Wort in der Wortsequenz gleicht,
kann es aus beiden Listen entfernt werden.

Wenn beide Listen leer sind, wurde die Wortsequenz erkannt.

Beispiel: Top-Down-Parsen

Ziele	Wörter
s	Karl den Mann kennt
np np v	Karl den Mann kennt
Karl np v	Karl den Mann kennt
np v	den Mann kennt
det n v	den Mann kennt
den n v	den Mann kennt
n v	Mann kennt
Mann v	Mann kennt
v	kennt
kennt	kennt

- Ersetzung von *v* durch alle Lexikoneinträge, die Verben sind → ineffizient
  in der Praxis arbeitet der Parser nur bis zu den Präterminalen
- Änderung der Verarbeitungsrichtung durch Anhängen bzw. Entfernen der Ziele und Wörter ans Ende der Sequenzen
- Erkenner wird zum Parser, wenn man sich die Ableitung merkt

Probleme beim Top-Down-Parsen (I)

- von-links-nach-rechts-Top-Down-Parser:
  n1 → n1 post_mod
- Problem: keine Grenze für Rekursion

Ziel	Wörter
...	...
n1 ...	...
n1 adjunct ...	...
n1 adjunct adjunct ...	...
n1 adjunct adjunct adjunct ...	...
...	...

- Regel muß irgendwann angewendet werden,
  da alle Analysen gefunden werden müssen.
- Bezeichnung: linksrekursiv

Probleme beim Top-Down-Parsen (II)

- Linksrekursion kann durch Regelkombination entstehen:

  (40) des Teufels General

  n(2,Cas) → det n(1,Cas)
  det → n(2,gen)
- Wenn es 600 Regeln für die Kategorie *S* gibt,
  müssen diese berücksichtigt werden,
  auch wenn 599 mit einer NP anfangen und der Satz mit einem Verb.

Behandlung des Nichtdeterminismus

- wenn es mehrere Ersetzungsmöglichkeiten gibt:
  - genau eine wählen und zurückgehen, falls die Weiterverfolgung dieser Analyse sich als sinnlos erweist (*Backtracking*) ≡ Tiefe-zuerst-Suche (*depth-first-search*) oder
  - mehrere Alternativen gleichzeitig verfolgen ≡ Breite-zuerst-Suche (*breadth-first-search*)
- Um theoretisches Minimum für Parsezeiten zu erhalten, muß man komplexe Datenstruktur verwenden:
  Tabelle für erfolgreiche Teilanalysen

Tiefe-zuerst-Suche

- aus einer Menge alternativer Folgezustände einen auswählen
  alle möglichen Nachfolgezustände dieses Zustands berücksichtigen, bevor man zurückkommt und eine andere Alternative wählt.
- Bezeichnung: *Backtracking*
- effizient implementierbar:
  Stack vorangegangener Entscheidungspunkte (*choice points*)
  falsche Entscheidung → Rückkehr zum nächsten (obersten) Entscheidungspunkt
  Fortsetzung der Arbeit nach erneuter Entscheidung
- Größe des Stacks proportional zur Tiefe des Suchraums

Breite-zuerst-Suche

- Simulation des gleichzeitigen Ausprobierens aller Möglichkeiten
- heutige Maschinen zumeist sequentiell →
  etwas Zeit für die Abarbeitung der ersten Alternative,
  etwas Zeit für die zweite, usw.
- gesamter Suchbaum mit allen aktuellen Abarbeitungszuständen muß repräsentiert werden
- meistens wesentlich aufwendiger als die Abspeicherung der Entscheidungspunkte für die Tiefe-zuerst-Suche
- bei unendlichem Suchraum ist Tiefe-zuerst-Suche nicht vollständig
  (z. B. für Top-Down-Parser und eine Grammatik mit Linksrekursion)
- Breite-zuerst-Suche findet immer alle möglichen Antworten, auch wenn die Suche wegen des unendlichen Suchraums eventuell nicht terminiert.

Grammatikübersetzung

- statt Parsestrategie zu ändern kann man Grammatik transformieren
- Epsilons eliminieren bzw. Linksrekursion eleminieren
  (Bar-Hillel, Perles und Shamir, 1961)
- Ergebnis ist nur schwach äquivalent
  (ð, Strukturen, die man nach Transformation bekommt, sehen anders aus) →
  Parsebäume mit transformieren

Unnötige Arbeit

- Tiefe-zuerst-Suche: Abarbeitung von Teilzielen
- schlägt Teilziel fehl, werden alle Zwischenergebnisse verworfen

  vp → np(dat), v(nom_dat)
  vp → np(dat), np(akk), v(ditrans)

  (41) a. Er hat *dem Mann, den alle kennen und verehren, obwohl er nicht wirklich singen kann,* ein Buch geschenkt.
  b. Er hat *dem Mann, den alle kennen und verehren, obwohl er nicht wirklich singen kann,* geholfen.

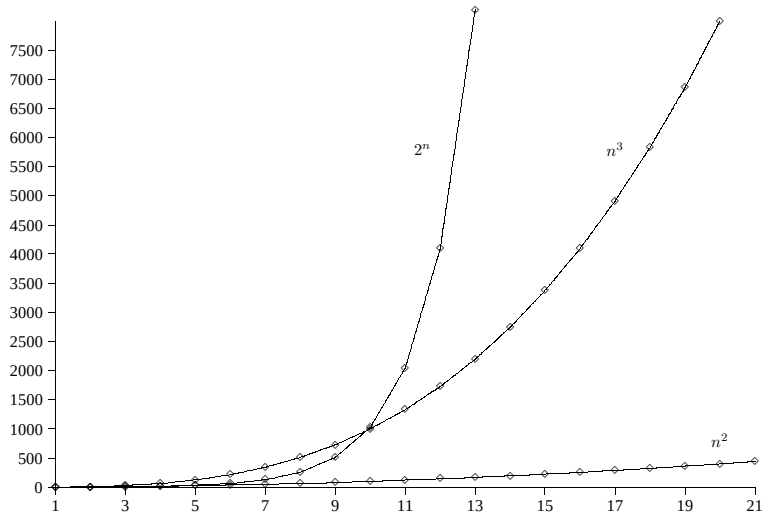
- Arbeit für Analyse der komplexen Dativ-NP in (41a) mit erster Regel umsonst → Erkenner und Parser sind ineffizient

Well Formed Substring Table

Verschiebung der Komplexität von der Kontroll- in die Datenstruktur:

- Platz-Komplexität des Backtrackings eines Parsers ist linear
- Datenstruktur mit Größe $< n^2$ bei einer Stringlänge n → Zeit-Komplexität polynomial (n^x) im Gegensatz zu exponentiell (x^n)
- Parser speichert erfolgreich geparste Konstituenten
- gespeicherte Resultate werden für weitere Verarbeitung benutzt
- abstrakte Datenstruktur = *Well-Formed Substring Table* oder *Chart*

Verschiedene Komplexitäten



CFG Parsing: Der Cocke-Kasami-Younger-Algorithmus

- Grammatik muß in Chomsky Normal Form (CNF) sein, nur
  - RHS mit einem Terminalsymbol: $A \rightarrow a$
  - RHS mit zwei Nichtterminalsymbolen: $A \rightarrow BC$
  - Keine ϵ-Regeln ($A \rightarrow \epsilon$)
- Repräsentation der Eingabekette (*string*) mit Positionen und Wortindices:

  $\cdot_0$ w_1 $\cdot_1$ w_2 $\cdot_2$ w_3 $\cdot_3$ w_4 $\cdot_4$ w_5 $\cdot_5$ w_6 $\cdot_6$

  z. B. : $\cdot_0$ the $\cdot_1$ young $\cdot_2$ boy $\cdot_3$ saw $\cdot_4$ the $\cdot_5$ dragon $\cdot_6$

Die Tabelle (*Chart*)

- Die well-formed substring table (ab jetzt (passive) chart) für einen String der Länge n ist eine Matrix der Größe $n \times n$.
- Das Feld (i, j) der Chart enthält die Menge aller Kategorien von Konstituenten, die an Position i beginnen und an Position j enden, ð
  chart(i,j) $= \{A \mid A \Rightarrow^* w_{i+1} \ldots w_j\}$
- Matrix ist dreieckig, da keine Konstituente vor ihrem Anfangspunkt endet

In der Chart repräsentierte Abdeckung

Ein Eingabesatz mit 6 Wörtern:

$\cdot_0\ w_1\ \cdot_1\ w_2\ \cdot_2\ w_3\ \cdot_3\ w_4\ \cdot_4\ w_5\ \cdot_5\ w_6\ \cdot_6$

in der Chart repräsentierte Abdeckung:

VON: \ BIS:	1	2	3	4	5	6
0	0–1	0–2	0–3	0–4	0–5	0–6
1		1–2	1–3	1–4	1–5	1–6
2			2–3	2–4	2–5	2–6
3				3–4	3–5	3–6
4					4–5	4–6
5						5–6

Beispiel für in der Chart repräsentierte Abdeckung

Beispielsatz

$\cdot_0$ the $\cdot_1$ young $\cdot_2$ boy $\cdot_3$ saw $\cdot_4$ the $\cdot_5$ dragon $\cdot_6$

in der Chart repräsentierte Abdeckung:

	1	2	3	4	5	6
0	the	the young	the young boy	the young boy saw	the young boy saw the	the young boy saw the dragon
1		young	young boy	young boy saw	young boy saw the	young boy saw the dragon
2			boy	boy saw	boy saw the	boy saw the dragon
3				saw	saw the	saw the dragon
4					the	the dragon
5						dragon

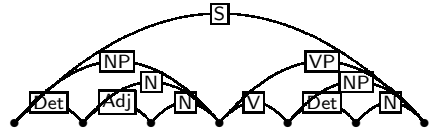
Ein Beispiel für eine ausgefüllte Tabelle

Eingabesatz:

$\cdot_0$ the $\cdot_1$ young $\cdot_2$ boy $\cdot_3$ saw $\cdot_4$ the $\cdot_5$ dragon $\cdot_6$

Chart:

	1	2	3	4	5	6
0	{Det}	{}	{NP}	{}	{}	{S}
1		{Adj}	{N}	{}	{}	{}
2			{N}	{}	{}	{}
3				{Vt, N}	{}	{VP}
4					{Det}	{NP}
5						{N}



Grammatik:

S → NP VP
VP → Vt NP
NP → Det N
N → Adj N
Vt → saw
Det → the
Det → a
N → dragon
N → boy
N → saw
Adj → young

Ausfüllen der Chart

- Es ist wichtig, die Chart systematisch auszufüllen.
- bauen alle Konstituenten, die an einem bestimmten Punkt enden, bevor wir irgendeine Konstituente bauen, die an einem späteren Punkt endet.

	1	2	3	4	5	6
0	*1*	**3**	**6**	**10**	**15**	**21**
1		*2*	**5**	**9**	**14**	**20**
2			*4*	**8**	**13**	**19**
3				*7*	**12**	**18**
4					*11*	**17**
5						*16*

for $j := 1$ to length(*string*)
  lexical_chart_fill$(j-1, j)$
  for $i := j-2$ down to 0
    syntactic_chart_fill(i, j)

lexical_chart_fill(j-1,j)

- Idee:
  Nachschlagen im Lexikon (*Lexical lookup*). Das Feld $(j-1, j)$ in der Chart wird mit dem Präterminalsymbol gefüllt, das das Wort j dominiert.
- Umsetzung:

$chart(j-1, j) := \{X \mid X \rightarrow word_j \in R\}$

syntactic_chart_fill(i,j)

- Idee:
  Führe alle Reduktionsschritte durch, indem syntaktische Regeln so benutzt werden, daß das reduzierte Symbol den String von i nach j überspannt.
- Umsetzung:

$$chart(i,j) = \left\{ A \;\middle|\; \begin{array}{l} A \rightarrow BC \in R, \\ i < k < j, \\ B \in chart(i,k), \\ C \in chart(k,j) \end{array} \right\}$$

- Explizite Schleife über alle möglichen Werte von k und alle Grammatikregeln:

$chart(i,j) := \{\}$.
for $k := i+1$ to $j-1$
  for every $A \rightarrow BC \in R$
    if $B \in chart(i,k)$ and $C \in chart(k,j)$ then
      $chart(i,j) := chart(i,j) \cup \{A\}$.

The Complete CYK Algorithm

Eingabe: Startsymbol S und Eingabe *string*

$n :=$ length(*string*)

for $j := 1$ to n
  $chart(j-1, j) := \{X \mid X \rightarrow word_j \in R\}$
  for $i := j-2$ down to 0
    $chart(i,j) := \{\}$
    for $k := i+1$ to $j-1$
      for every $A \rightarrow BC \in R$
        if $B \in chart(i,k)$ and $C \in chart(k,j)$ then
          $chart(i,j) := chart(i,j) \cup \{A\}$

Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

Lexikoneintrag: *the* (j = 1, field chart(0,1))

	1	2	3	4	5
0	d				
1					
2					
3					
4					

Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

Lexikoneintrag: *cat* (j = 2, field chart(1,2))

	1	2	3	4	5
0	d				
1		n			
2					
3					
4					

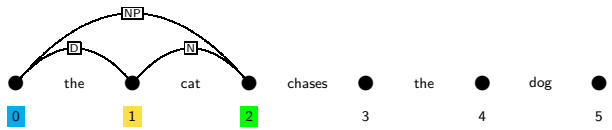
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 2
i = 0
k = 1

	1	2	3	4	5
0	d	np			
1		n			
2					
3					
4					



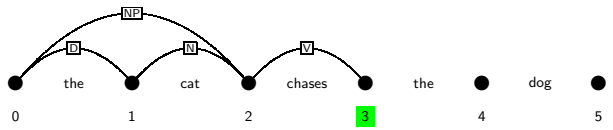
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

Lexikoneintrag: *chases* (j = 3, field chart(2,3))

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3					
4					



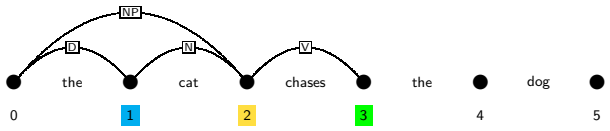
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 3
i = 1
k = 2

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3					
4					



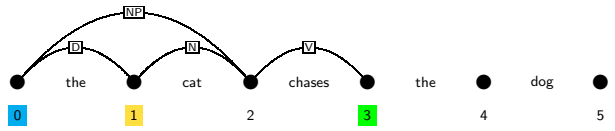
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 3
i = 0
k = 1

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3					
4					



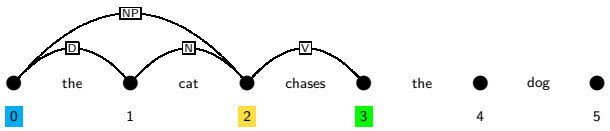
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 3
i = 0
k = 2

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3					
4					



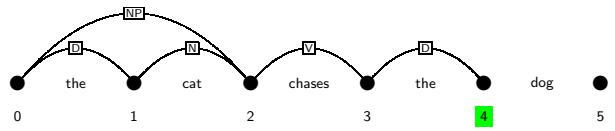
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

Lexikoneintrag: *the* (j = 4, field chart(3,4))

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					



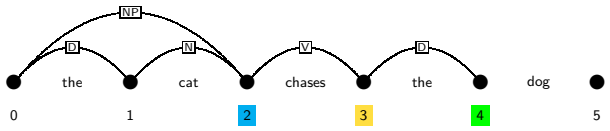
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 4
i = 2
k = 3

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					



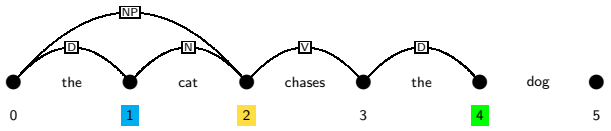
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 4
i = 1
k = 2

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					



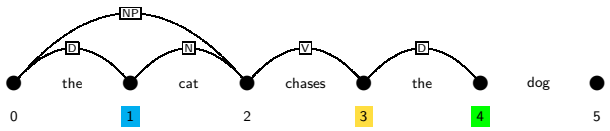
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 4
i = 1
k = 3

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					



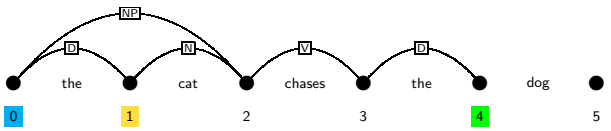
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 4
i = 0
k = 1

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					



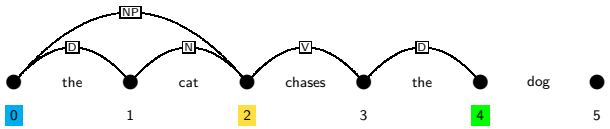
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 4
i = 0
k = 2

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					



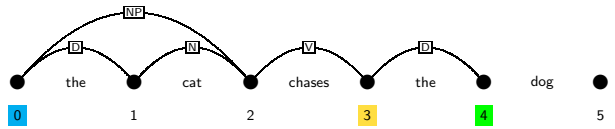
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 4
i = 0
k = 3

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					



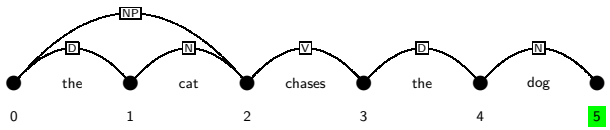
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

Lexikoneintrag: *dog* (j = 5, field chart(4,5))

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	
4					n



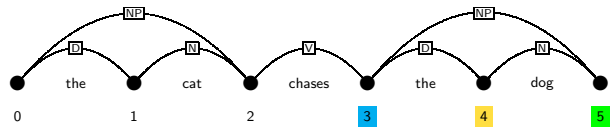
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 3
k = 4

	1	2	3	4	5
0	d	np			
1		n			
2			v		
3				d	np
4					n



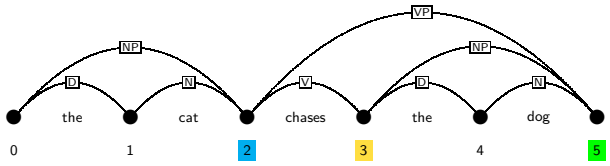
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 2
k = 3

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n



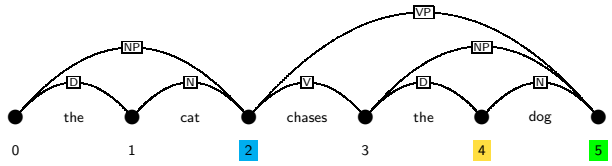
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 2
k = 4

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n



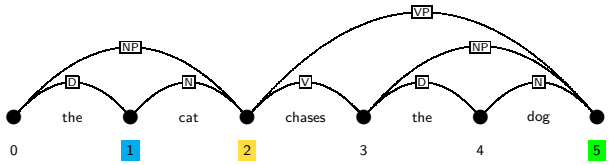
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 1
k = 2

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n



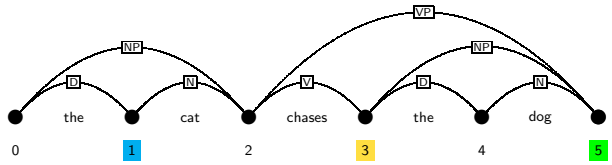
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 1
k = 3

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n



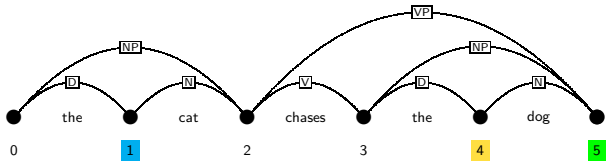
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 1
k = 4

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n



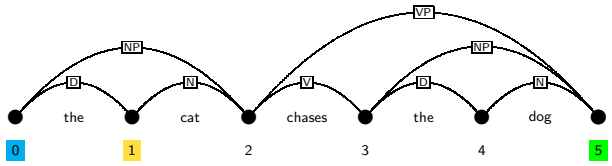
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 0
k = 1

	1	2	3	4	5
0	d	np			
1		n			
2			v		vp
3				d	np
4					n



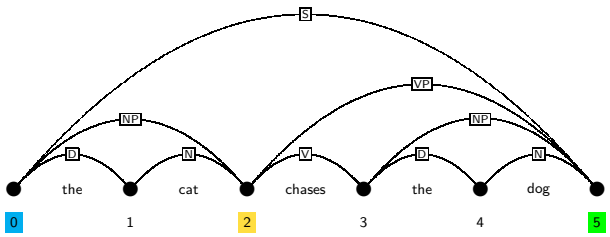
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 0
k = 2

	1	2	3	4	5
0	d	np			s
1		n			
2			v		vp
3				d	np
4					n



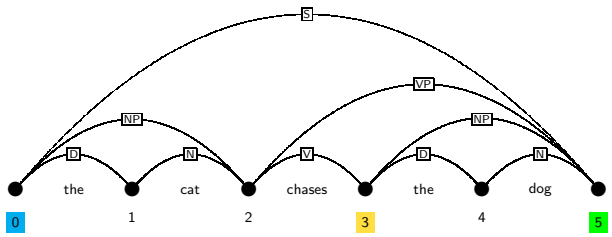
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 0
k = 3

	1	2	3	4	5
0	d	np			s
1		n			
2			v		vp
3				d	np
4					n



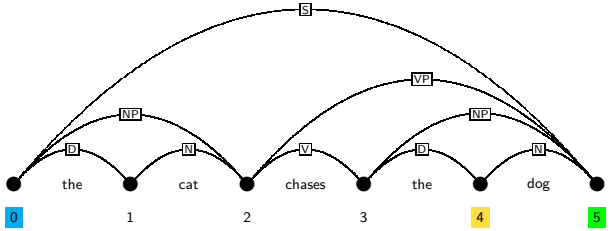
Beispielanwendung des CKY-Algorithmus

s → np vp
np → d n
vp → v np

d → the
n → dog
n → cat
v → chases

j = 5
i = 0
k = 4

	1	2	3	4	5
0	d	np			s
1		n			
2			v		vp
3				d	np
4					n



Mehrdeutigkeit

- Satz mit n Präpositionalphrasen nach dem Verb hat bei folgender Grammatik Catalan($n+2$) Parsebäume (1 2 5 14 42 132 469 1430 . . .).

  (42)
  np → tigger
  n → dog
  n → garden
  v → chases
  n → bone
  det → a

  s → np vp
  vp → vp pp
  np → np pp
  vp → v np
  np → det n
  pp → p np

- *chases a dog with a bone* hat zwei Parsemöglichkeiten als VP

  (43) a. chases + a dog with a bone (vp → v np)
  b. chases a dog + with a bone (vp → vp pp)

- übergeordnete Konstituenten, die diese VP nutzen, nicht zweimal in der Chart!

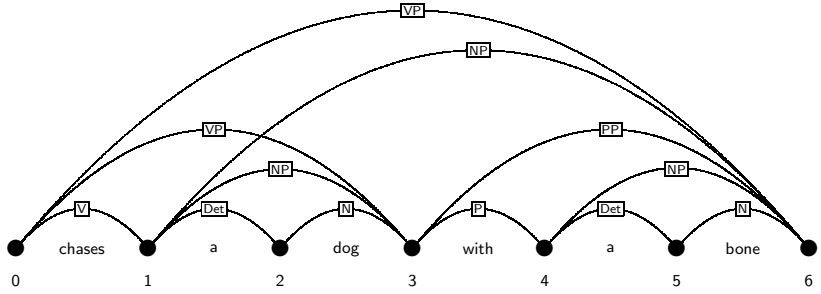
Repräsentation von Mehrdeutigkeit

(44) $_0$ chases $_1$ a $_2$ dog $_3$ with $_4$ a $_5$ bone $_6$

	1	2	3	4	5	6
0	v		vp			vp
1		det	np			np
2			n			
3				p		pp
4					det	np
5						n

Es gibt im Feld chart(0,6) nur eine VP, nicht zwei.
Die beiden Teilanalysen sind implizit in der Chart enthalten.

Chart für *chases a dog with a bone*



Obwohl es zwei Analysen gibt, gibt es nur eine Kante, die von 0 bis 6 geht!

Vom Erkenner zum Parser (I)

- bisher nicht genug Information, um Ableitungsgeschichte eines Satzes zu rekonstruieren
- mögliche Analysen in Form der einzelnen Teile repräsentieren
- statt einer Kategorie ein Tripel der Form $< \mathit{Category}, \mathit{Rule}, \mathit{Pos} >$
  Rule = Regel der Grammatik,
  die den Eintrag der Kategorie *Category* in die Chart rechtfertigt
  Pos = Position im String,
  an der sich die zwei Teilphrasen der rechten Regelseite treffen
- zusammen mit Anfangs- und Endposition von *Category* reicht das zur Rekonstruktion des Parsebaumes
- rekursives Auffinden aller Tabelleneinträge
- Generierung aller möglichen Analysen

Vom Erkenner zum Parser (II)

- $*$-Operation entsprechend definieren:
  $\mathit{chart}(i,k) * \mathit{chart}(k,j) =$
  $\{< \mathit{LHS}, R, k > |$
  Regel R der Grammatik ist $\mathit{LHS} \rightarrow \mathit{RHS}_1\ \mathit{RHS}_2$
  und es gibt ein Element $< \mathit{RHS}_1, _, _ >$ *in* $\mathit{chart}(i,k)$
  und es gibt ein Element $< \mathit{RHS}_2, _, _ >$ *in* $\mathit{chart}(k,j)$ $\}$
- mehrere Analysen einer Teilphrase (z. B. RHS_1) ergeben nicht mehrere Einträge für übergeordnete Phrasen:
  bei Konstruktion der Menge $< \mathit{LHS}, R, k >$ werden doppelte Elemente nicht berücksichtigt
- Grammatiken mit Regeln wie $S \rightarrow S$ sind verarbeitbar:
  Aufzählung der Analysen würde bei simpler Regelanwendung nie terminieren

Nachteile des Bottom-Up Parsens

- alle durch die Grammatik gerechtfertigten Konstituenten werden gebaut, egal ob sie in einer vollständigen Phrase auftauchen könnten oder nicht

  (45) a. The search for Spock was successful.
  b. search for Spock

  (45b) = N' und (45b) = VP
  the + VP geht nie
- Top-Down-Vorhersage würde Suchraum verkleinern

Active Chart Parsing – Regeln mit Punkten

- Verallgemeinerung des CKY-Algorithmus:
  auch Grammatiken mit mehr als zwei Nichtterminalen in rechten Regelseiten
- Punkt rechts von bereits gefundenen Konstituenten
  $\mathit{vp} \rightarrow .\mathit{v}\ \mathit{np}\ \mathit{pp}$
  $\mathit{vp} \rightarrow \mathit{v}.\ \mathit{np}\ \mathit{pp}$
  $\mathit{vp} \rightarrow \mathit{v}\ \mathit{np}.\ \mathit{pp}$
  $\mathit{vp} \rightarrow \mathit{v}\ \mathit{np}\ \mathit{pp}.$
- statt Verküpfung vollständiger Konstituenten,
  Verknüpfung teilweise vollständiger mit vollständigen
- Kantenrepräsentation = Regel mit Punkt und Anfangs- und Endpunkte
  ${}_3[\mathit{vp} \rightarrow \mathit{v}\ \mathit{np}\ {}_8\ \mathit{pp}]$

  Zahl der Endposition entspricht dem Punkt in der Regel

Active Chart Parsing (II)

- Beispiele für Kombination von Regeln mit Punkten (Kanten):
  ${}_2[vp \rightarrow v\ {}_3\ np\ pp] * {}_3[np \rightarrow \ldots{}_5] = {}_2[vp \rightarrow v\ np\ {}_5\ pp]$
  ${}_2[vp \rightarrow v\ np\ {}_5\ pp] * {}_5[pp \rightarrow \ldots{}_8] = {}_2[vp \rightarrow v\ np\ pp\ {}_8]$
- aktive und inaktive Kanten – Parser = *Active-Chart-Parser*
- Multiplikationsregel für Mengen von Kanten:
  $S_1 * S_2 = \{{}_i[L \rightarrow AX_jB] \mid {}_i[L \rightarrow A_k\ X\ B] \in S_1 \text{ und } {}_k[X \rightarrow \ldots\ {}_j] \in S_2\}$
- Regel ist unabhängig von der Grammatik
- Kombination zweier Kanten:

  (46) ${}_i[L \rightarrow A_k\ X\ B] * {}_k[X \rightarrow \ldots\ {}_j] = {}_i[L \rightarrow AX_jB]$

  Dabei ist L ein Nichtterminal, X ein Terminal- oder Nichtterminalsymbol und A und B sind eventuell leere Strings aus Nichtterminalen.

  (46) = *Fundamentalregel* des Chart-Parsens

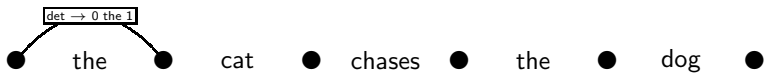
Bottom-Up-Einbeziehung – (Gazdar und Mellish, 1989)

- Wie kommen aktive Kanten in die Chart?
- wenn Wort eingetragen wird, werden auch Regeln mit Punkt für all die Grammatikregeln eingetragen, von denen das Wort eine linke Ecke ist
- Punkt steht nach der Kategorie des entsprechenden Wortes
  Verb zwischen Position 4 und 5, für die Regel $vp \rightarrow v\ np\ pp$ ergibt sich:
  ${}_4[vp \rightarrow v\ {}_5\ np\ pp]$

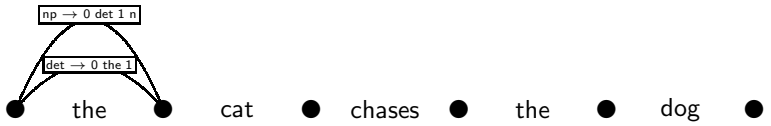
  Bottom-Up-Regel
  Wenn eine inaktive Kante der Form $C \rightarrow W1.$, die von i nach j geht, in die Chart eingetragen wird,
  dann nimm für jede Regel der Form $B \rightarrow C\ W2$ eine Kante $B \rightarrow C.\ W2$ von i nach j in die Chart auf.

  Dabei sind W1 und W2 eventuell leere Folgen von Terminal- und Nichtterminalsymbolen.

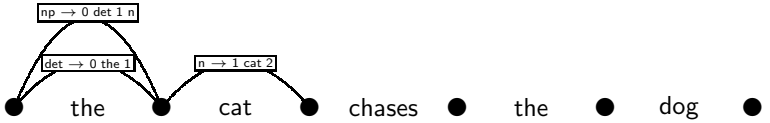
Beispielanalyse mit GM-Chart-Parser (Vervollständigung)



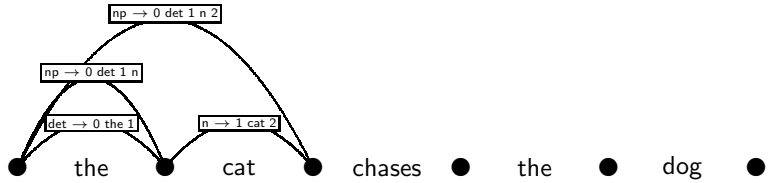
Beispielanalyse mit GM-Chart-Parser (Vorhersage)



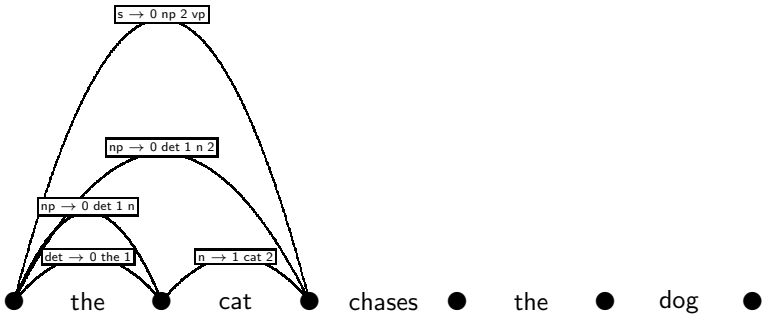
Beispielanalyse mit GM-Chart-Parser (Vervollständigung)



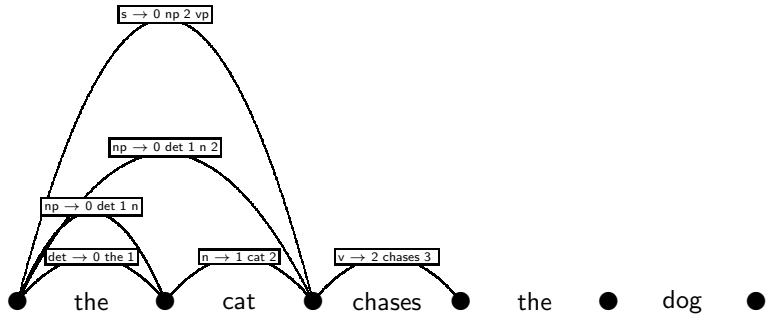
Beispielanalyse mit GM-Chart-Parser (Vervollständigung)



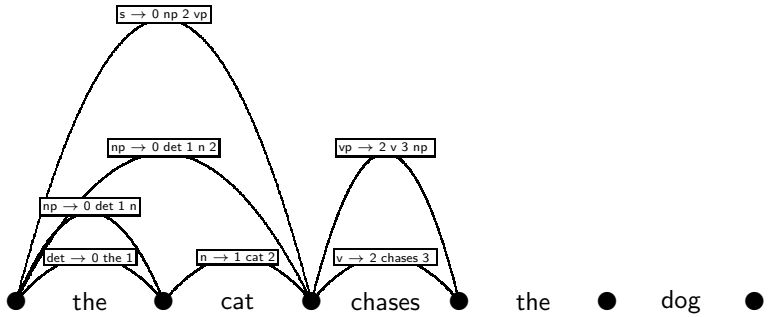
Beispielanalyse mit GM-Chart-Parser (Vorhersage)



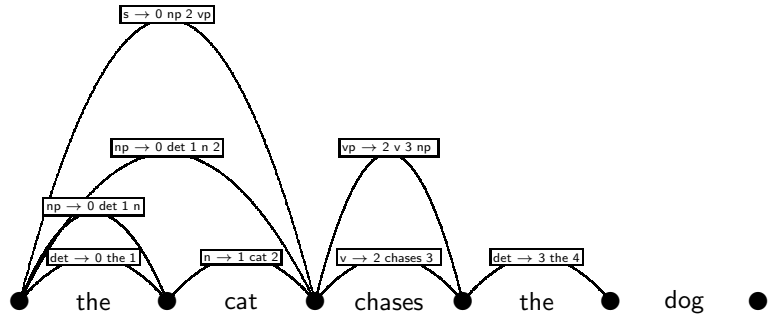
Beispielanalyse mit GM-Chart-Parser (Vervollständigung)



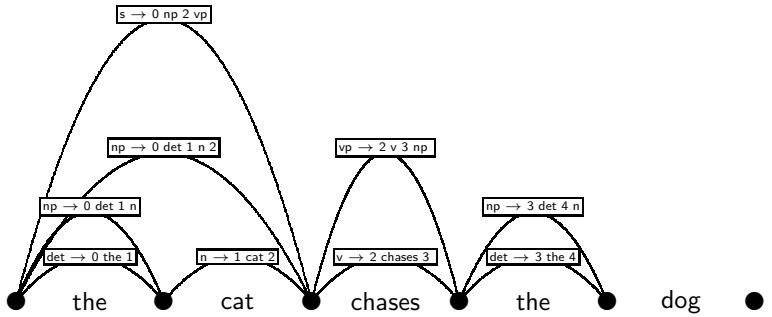
Beispielanalyse mit GM-Chart-Parser (Vorhersage)



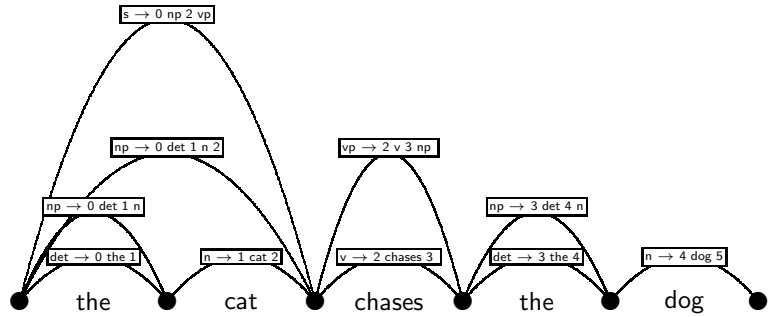
Beispielanalyse mit GM-Chart-Parser (Vervollständigung)



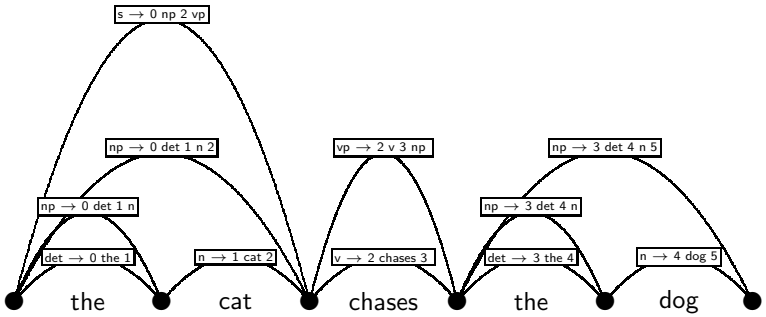
Beispielanalyse mit GM-Chart-Parser (Vorhersage)



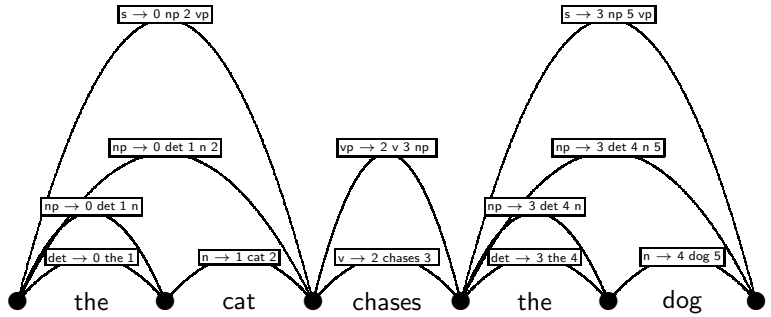
Beispielanalyse mit GM-Chart-Parser (Vervollständigung)



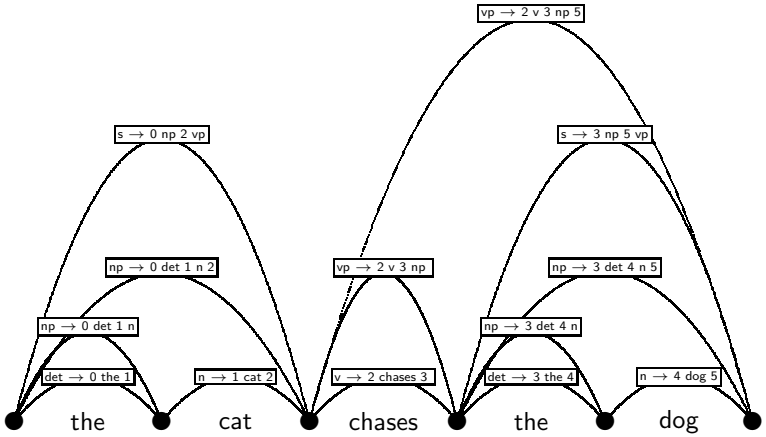
Beispielanalyse mit GM-Chart-Parser (Vervollständigung)



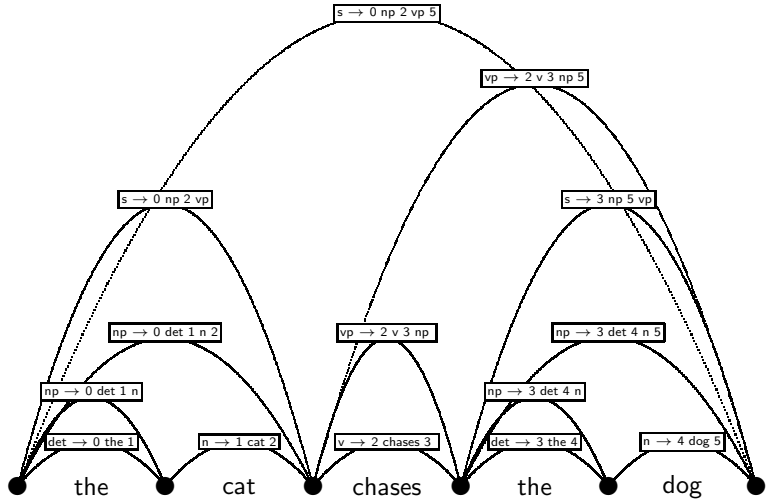
Beispielanalyse mit GM-Chart-Parser (Vorhersage)



Beispielanalyse mit GM-Chart-Parser (Vervollständigung)



Beispielanalyse mit GM-Chart-Parser (Vervollständigung)



Top-Down-Einbeziehung – der Earley Algorithmus

- Top-Down-Active-Chart-Parser von Jay Earley 1970 entwickelt
- effektivste Algorithmus zum Parsen von kontextfreien Grammatiken
- Top-Down-Vorhersage:
  Eintrag in der Diagonale = Phrase der Länge Null (von i zu i)
  entspricht Hypothese, daß bei i eine Phrase beginnt

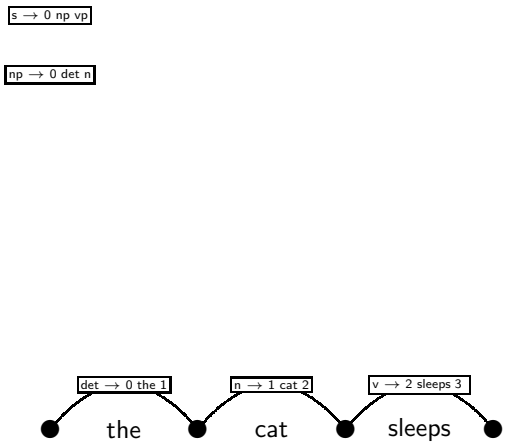
  Zum Beispiel:
  ${}_0[s \rightarrow_0 np\ vp]$

  Anzahl der sinnlosen Konstituenten, die gebaut werden,
  ist gegenüber dem CKY-Algorithmus gering.

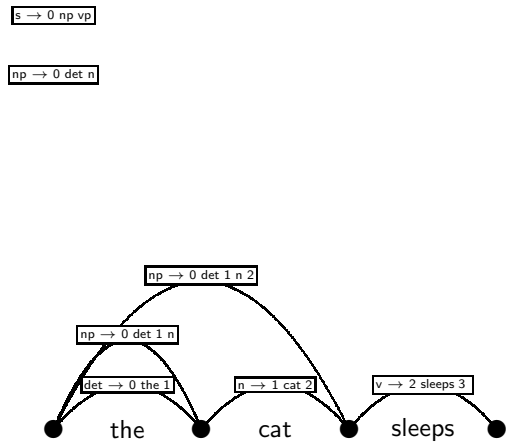
Vorhersage, Scannen und Vervollständigung

1. *Vorhersage* Die Kante, die eingetragen wird, ist aktiv, aber es gibt keine Kante in der Chart, mit der sie kombiniert werden könnte. In diesem Fall werden alle Hypothesen, also Regeln mit Punkt ganz am Anfang, für die benötigte Kante eingetragen.
2. *Scannen* Die Kante, die eingetragen wird, ist aktiv, und es gibt bereits Kanten in der Chart, die mit ihr kombiniert werden können. Für jede dieser Kanten wird die Multiplikationsregel angewendet. Dadurch entstehen weitere Kanten.
3. *Vervollständigung* Die Kante, die eingetragen wird, ist vollständig.
   Für alle Kanten, mit denen die eingetragene Kante kombiniert werden kann, wird die Multiplikationsregel angewendet. Dadurch entstehen weitere Kanten.
   Man beachte, daß es entsprechende aktive Kanten in der Chart geben muß.
   Diese aktiven Kanten sind durch Vorhersage entstanden.

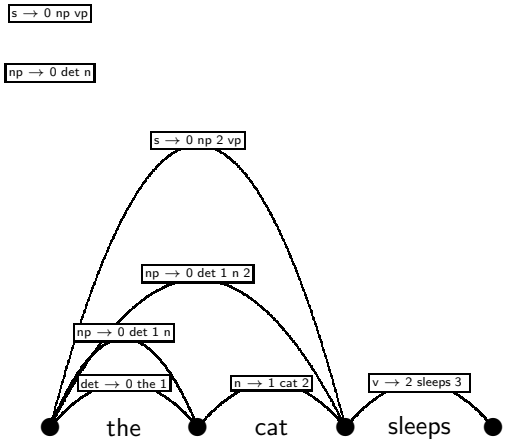
Beispielanalyse mit Early-Parser (Vorhersage)



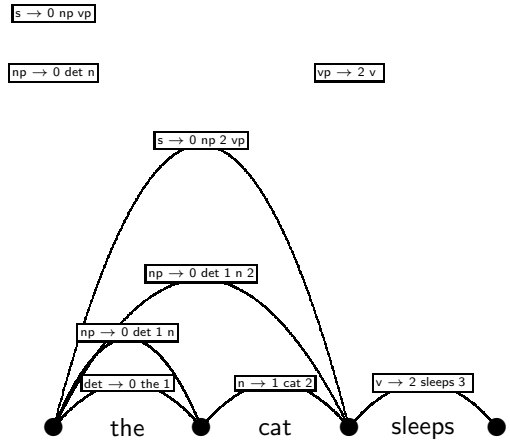
Beispielanalyse mit Early-Parser (Scannen)



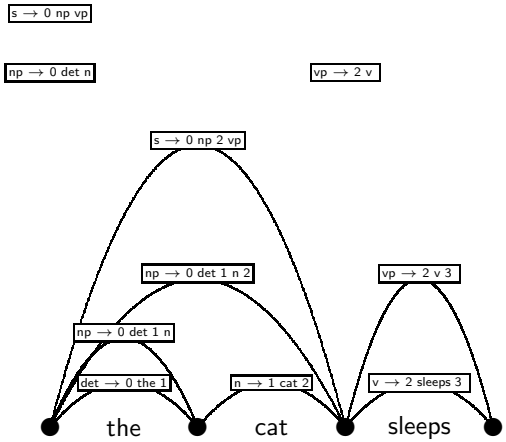
Beispielanalyse mit Early-Parser (Vervollständigung)



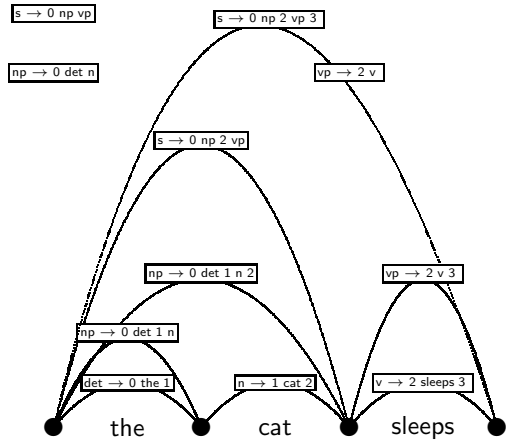
Beispielanalyse mit Early-Parser (Vorhersage)



Beispielanalyse mit Early-Parser (Scannen)



Beispielanalyse mit Early-Parser (Vervollständigung)



Initialisierung der Chart

- zuerst Vorhersage für Startsymbol oder erst lexikalische Kanten eintragen?
- bei Vorhersage des Startsymbols werden vollständige Kanten erst eingetragen, wenn alle Vorhersagen bis zum Anfang dieser Kante gemacht wurden. → Scannen tritt nie auf

Matrix

- Verwenden Tabellen zum Nachschlagen von Kombinationsmöglichkeiten
- dreidimensionale Matrix von Boolean-Werten
- zweite und dritte Dimension sind Anfangs- und Endpunkte von Kanten, erste Dimension entspricht möglichen Zuständen
- für Zustand s und Chart-Positionen i und j ist $chart(s, i, j)$ *true* oder *false*. Beispiel:
  s → np vp
  np → det n
  vp → v np np

  Zustände:

s → .np vp	s → np.vp	s → np vp.
np → .det n	np → det.n	np → det n
vp → .v np np	vp → v.np np	vp → v np.np
vp → v np np.		

predictions, left_sister und right_sister

predictions(s → np. vp) = {vp → .v np np }
predictions(vp → v.np np) = { np → .det n }

`left_sister(x,y)` gibt für zwei Zustände x und y *true* zurück, falls x nach links mit y kombiniert werden kann:

```
left_sister((s --> . np vp), (np --> det n .)) = true
left_sister((np --> det. n), (np --> det n .)) = false
left_sister((s --> .np vp),  (np --> det. n )) = false
```

`right_sister(x,y)` funktioniert analog für die Kombination nach rechts:

```
right_sister((np --> det n .), (s --> . np vp)) = true
right_sister((np --> det n .), (np --> det. n)) = false
right_sister((np --> det. n ), (s --> .np vp )) = false
```


Der Algorithmus

```
procedure chartparse:
   for state in predictions(... --> .s) do
       enter_edge(state,0,0)
   for j from 1 to n do
       for state in {(A --> . | A --> wordj} do
           enter_edge(state,j-1,j)


procedure enter_edge(state1,i,j):
   if chart(state1,i,j) = false then
      set chart(state1,i,j) to true

      % predict
      for state2 in predictions(state1) do
          enter_edge(state2,j,j)

      % complete
      for each k, state2 such that chart(state2,k,i) is true do
          if left_sister(state2,state1) then
             enter_edge(state2*state1,k,j)

      % scan
      for each k, state2 such that chart(state2,j,k) is true do
          if right_sister(state2,state1) then
             enter_edge(state1*state2,i,k)
```

Konstruktion eines Parsers

- Positionen im String mit in die Regel mit Punkt aufnehmen → Analyse rekonstruierbar

  vp → (2) . *v np np*
  vp → (2) *v* (3) . *np np*
  vp → (2) *v* (3) *np* (5) . *np*
  vp → (2) *v* (3) *np* (5) *np* (7) .

- Gibt es mehrere Analysen für NPs zwischen 5 und 7 dann gibt es trotzdem nur eine Repräsentation für die VP, die diese NPs benutzt.
  → Effizenz

Kategorialgrammatik (I)

Problem der Verarbeitung einer Grammatik bei Top-Down-Einbeziehung wie beim Earley-Algorithmus & Subkategorisierung über Merkmal:

- ein Merkmal der Lexikoneinträge wählt Phrasenstrukturregel
  vp → v(ditrans) np np
- bei Top-Down-Parser ohne Vorausschau viele VP-Hypothesen mit Punkt am Anfang
- Lexikonlookup + Multiplikationsregel → nur wenige dieser Hypothesen brauchbar
  vp → v(ditrans).np np
  vp → v(np_and_pp).np pp(to)

Kategorialgrammatik (II)

- Left-Corner-Strategie für Englisch (rechtsverzweigend), aber Deutsch
  s(fin,final) → np v(fin,intr)
  s(fin,final) → np np v(fin,trans)
  s(fin,final) → np np np v(fin,ditrans)
- Bottom-Up-Chart-Parser ohne Vorhersage → 10 unnütze Hypothesen

  (47) weil Karl Maria das Buch gibt.

- Jede NP kann Anfang oder Mittelstück einer Verbalprojektion sein.
- Lösung:
  Lexikon bestimmt, welche aktiven Kanten in die Chart aufgenommen werden.
- Elemenierung der Phrasenstrukturregeln

Motivation der Kategorialgrammatik

- komplexe Kategorien ersetzen die Subcat-Merkmale wie *ditrans* oder *np_and_pp*

Regel	Kategorie im Lexikon
vp → v(ditrans).np np	(vp/np)/np
vp → v(np_and_pp).np pp(to)	(vp/pp)/np

- Kategorien sind wie aktive Kanten
- *Multiplikationsregel*
  Regel1 : $X/Y * Y = X$

chased	Mary
vp/np	np
vp	

- Kategorie v wird nicht mehr benötigt.

Kategorialgrammatik

- vp kann auch eleminiert werden: vp = s\np
  Regel2 : $Y * X\backslash Y = X$

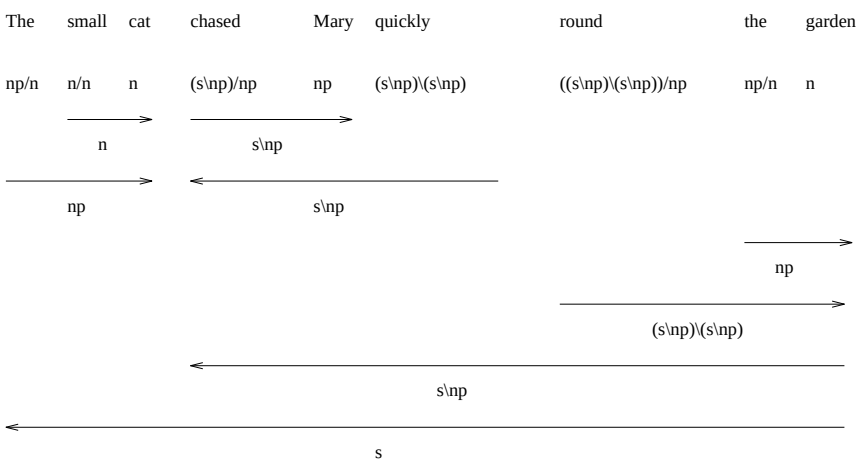
the	cat	chased	Mary
np/n	n	(s\np)/np	np
np		s\np	
s			

- kein expliziter Unterschied zwischen Phrasen und Wörtern:
  - intransitives Verb = Verbphrase = $(s\backslash np)$
  - genauso Eigennamen = Nominalphrasen = np

Modifikation

- optionale Modifkatoren:
  vp → vp pp
  noun → noun pp
- beliebig viele PPen nach einer VP bzw. einem Nomen
- Modifikatoren allgemein haben Form: $X\backslash X$ bzw. X/X
- Prämodifikator für Nomina:
  noun → adj noun
  Adjektive = n/n
- Postmodifier für Nomina: $n\backslash n$
- vp-Modifikator → X = $s\backslash np$
- vp-Modifikator: $(s\backslash np)\backslash(s\backslash np)$.

Ableitung mit einer Kategorialgrammatik



Übung

Überlegen Sie, wie man (48) in der Kategorialgrammatik analysieren kann:

(48) Ein dicker Mann lacht laut.

Semantische Verarbeitung

- **Eingabe:** Syntaktische Repräsentation eines Satzes. Repräsentation des bisherigen Diskurses.
- **Verarbeitung:** Auflösung der Referenzen von (Nominal)-Phrasen, Anaphern. Komposition der Bedeutungen der Satzteile zur Bedeutungsrepräsentation des Satzes.
  Interpretation der Bedeutung des Satzes im Kontext.
- **Ausgabe:** Semantische Repräsentation der Satzes in einer Version der Prädikatenlogik oder in einem anderen Bedeutungs- bzw. Wissensrepräsentationsformalismus.

Semantische Relationen zwischen Wörtern

- Synonymie
  Orange – *Apfelsine*
  Telefon – *Fernsprecher*
  Stockwerk – *Etage*
  Ostzone – *DDR*
  Morgenstern – *Abendstern* = Venus
- Antonymie
  heiß – *kalt*
  Höhen – *Tiefen*
- Hyponymie, Hyperonymie
  Früchte – *Äpfel*
  arbeiten – *schuften*

Semantische Relationen zwischen Sätzen

- Äquivalenz
  Peter ist kein Papagei. – *Es ist nicht wahr, daß Peter ein Papagei ist.*
  Wir wählten Klaus. – *Klaus wurde von uns gewählt.*
- Kontradiktion
  Peter ist kein Papagei. – *Peter ist ein Papagei.*
  Kein Baby kann sprechen. – *Es gibt einen sprechenden Säugling.*
- Folgerung, Enthaltensein
  Sowohl Karl als auch Richard hat Email. – *Karl hat Email.*
  Das Glas war rot. – *Das Glas war farbig.*

Bedeutungsrepräsentation

Zwei Traditionen in der Repräsentation der Bedeutung:

- Logische Tradition:
  Die Bedeutung wird in einer Sprache der Logik repräsentiert.
  (z. B. Prädikatenlogik, Temporallogik, intensionale Logik)
  Montague-Semantik, Situationstheorie, Diskursrepräsentationstheorie
- KI-Tradition:
  Die Bedeutung wird in einem Wissensrepräsentationsformalismus der KI repräsentiert. (z. B. Skripte, semantische Netze, Frame-Sprachen)
  KL-ONE, Conceptual Graphs
- Gegenwärtige Entwicklung: Verschmelzung der Traditionen

Prädikatenlogik erster Stufe (PL 1)

- Wie kann man die Bedeutung von (49) repräsentieren?

  (49) Karl kennt Maria.

- Individuenkonstanten: *clara*, *karl*, *maria*, *max*, *peter*, ...
- Prädikatkonstanten: *schlafen*, *kennen*, ...
- Formel: *kennen*(*karl*, *maria*)

  (50) a. Peter kennt Maria. → *kennen*(*peter*, *maria*)
  b. Max kennt Maria. → *kennen*(*max*, *maria*)
  c. Clara kennt Peter. → *kennen*(*clara*, *peter*)

Repräsentation und Abfrage von Wissen in PL 1

- bekannte Fakten:
  kennen(*karl*, *maria*)
  kennen(*peter*, *maria*)
  kennen(*max*, *maria*)
  kennen(*clara*, *peter*)
- Frage: Wer kennt Maria?
  kennen(X, *maria*)
- Frage: Wer kennt wen?
  kennen(X, Y)

Negation, Konjunktion und Disjunktion

- Negation: ¬

  (51) Maria kennt Peter nicht.

  ¬*kennen*(*maria*, *peter*)
- Konjunktion: ∧

  (52) Peter kennt Maria und Max kennt Maria.

  kennen(*peter*, *maria*) ∧ *kennen*(*max*, *maria*)
- Disjunktion: ∨

  (53) Peter kennt Maria oder Max kennt Maria.

  kennen(*peter*, *maria*) ∨ *kennen*(*max*, *maria*)
- Vorsicht!
  Natürliche Sprache entspricht nicht eins zu eins diesen logischen Ausdrücken.
- Suche: Person, die Maria und Peter kennt: *kennen*(X, *maria*) ∧ *kennen*(X, *peter*)

Quantoren

- Determinatoren wie *alle* und *eine* werden mittels Variablen und Quantoren repräsentiert:
  - Existenzquantor: ∃

    (54) a. Karl liest ein Buch.
    b. Es gibt ein X für das gilt: X ist ein Buch und Karl liest X.

    ∃X *buch*(X) ∧ *lesen*(*karl*, X)
  - Allquantor: ∀

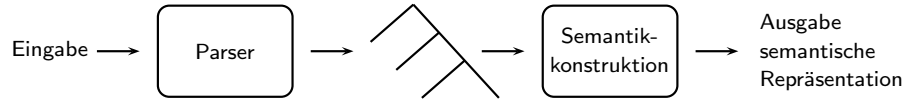
    (55) a. Alle Studenten sind begeistert.
    b. Für alle X gilt, daraus das gilt, daß X ein Student ist, folgt, daß X begeistert ist.

    ∀X *student*(X) → *begeistert*(X)

Syntax der Prädikatenlogik erster Stufe

Formel	→	AtomareFormel
	|	Formel Konnektor Formel
	|	Quantor Variable, ... Formel
	|	¬ Formel
	|	(Formel)
AtomareFormel	→	Prädikat(Term, ...)
Term	→	Funktion(Term, ...)
	|	Konstante
	|	Variable
Konnektor	→	$\wedge$ | $\vee$ | $\Rightarrow$
Quantor	→	$\forall$ | $\exists$
Konstante	→	a | karl | ...
Variable	→	X | Y | ...
Prädikat	→	kennen | neben | ...
Funktion	→	LokationVon | BruderVon | ...

Semantikkonstruktion



- Erzeugung syntaktischer Strukturen mit statistischen Verfahren oder aus Baumbänken
- Konstruktion einer Bedeutungsrepräsentation mit Hilfe eines semantischen Lexikons
- reicht (je nach Syntaxkomponente) nicht aus, um Fernabhängigkeiten aufzulösen

  (56) a. Wem glaubst Du, daß er geholfen hat. (mehrdeutig)
  b. Wem glaubst Du, daß er dem Mann das Buch gegeben hat. (eindeutig)
- Benutzung eines Valenzlexikons
- Beispiel Verb*mobil*: Schielen, 2000

Korrespondenz syntaktischer und semantischer Regeln

- Annahme, daß Konstruktion der Bedeutungsrepräsentation parallel zum Aufbau syntaktischer Strukturen verläuft
- parallel zur Ableitung von S auch Bedeutung von S berechnen

  (57) S → V NP_1 NP_2
- Wie kann man die Bedeutung von NP_1 mit der Bedeutung von V verknüpfen?

λ-Abstraktion und -Konversion

- Abstraktion: Konstante wird durch Variable ersetzt
  - $schlafen(peter) \rightarrow schlafen(X)$
- Variable greifbar machen:
  - $\lambda X\ schlafen(X)$
  - $\lambda Y \lambda X\ kennen(X, Y)$
- λ-Konversion: Variable wird ersetzt
  - $\lambda X\ schlafen(X)(peter) = schlafen(peter)$
  - $\lambda Y \lambda X\ kennen(X, Y)(peter) = \lambda X\ kennen(X, peter)$
    $\lambda X\ kennen(X, peter)(maria) = kennen(maria, peter)$
- Genaueres zum λ-Kalkül in der Veranstaltung *Logik für Linguisten*

Syntaktische Regeln mit Semantikteil

allgemein:

(58) $A \rightarrow \alpha_1 \ldots \alpha_n \quad \{f(\alpha_j.sem, \ldots, \alpha_k.sem)\}$

Sätze und Verben:

(59) a. s → v np { v.sem(np.sem) }
b. s → v np_1 np_2 { (v.sem(np_2.sem) (np_1.sem) }
c. v → *schläft* $\{\lambda X\ schlafen(X)\}$
d. v → *kennt* $\{\lambda Y \lambda X\ kennen(X, Y)\}$

Nominalphrasen, Artikel und Nomina:

(60) a. np → det n' $\{det.sem(n'.sem)\}$
b. det → *ein* $\{ \lambda P \lambda Q \exists X(P(X) \wedge Q(X)) \}$
c. n' → noun $\{noun.sem\}$
d. n → *Buch* { λX*buch*(X) }

Teil III

Technologien und Anwendungen

Gliederung Teil III

- Sprachtechnologien
- Motivationen für die Modellierung menschlicher Sprache

Sprachtechnologien

Texttechnologien	Sprechtechnologien
Text Technologies	Speech Technologies

Human-Machine Interactivity
Information & Knowledge Management

Speech Technologies

Spoken Dialogue Systems

Speech Translation Systems

Voice Recognition/
Speaker Identification

Language Indentification

Speech Verification

Speech Recognition

Voice Modelling

Speech Synthesis

Speech Production

Speech Applications

Voice Control Systems
Dication Systems
Text-to-Speech Systems
Identification and Verification Systems
Information Access
Spoken Dialogue Systems
Speech Translation Systems

Text Technologies

Written Dialogue Systems
Text Translation Systems
Language Indentification
Information Retrieval
Document Categorization
Document Clustering
Text Summarization
Information Extraction
Spell Checking
Grammar/Style Checking
Abstract Generation
Report Generation
Text Generation
Document Production

Text Applications

Spell Checkers
Machine-Assisted Human Translation
Indicative Machine Translation
Grammar Checkers
Human Assisted Machine Translation
High Quality Text Translation
Text Generation Systems

Freie Universität Berlin

Sprachtechnologie für das Informationsmanagement

- Grundbegriffe: Daten, Information, Wissen
- Informationsmanagement
- Wissensmanagement
- Verteilte wissensverarbeitende Systeme
- Transformation von Information in Wissen
- Ontologien und Inferenzen
- Praktische Konsequenzen

Freie Universität Berlin

Grundbegriffe

Daten:	Muster, die gespeichert und verarbeitet werden können = potentielle Information
Information:	Muster, die die Zustandsübergänge in zustandsverändernden Systemen beeinflussen. (Daten, die so dargebracht worden sind, daß sie ein Informationsbedürfnis erfüllen, ð, daß sie bei relevanten Entscheidung helfen)
Wissen:	Information, die so repräsentiert und mit anderen Informationen vernetzt ist, daß sie unmittelbar zugreifbar ist (und im Kontext interpretiert und für die Interpretation weiterer Informationen genutzt werden kann).

Freie Universität Berlin

Das Wesen der Information

Informationen sind Muster, die die Zustandsübergänge eines zustandsverändernden Systems beeinflussen. (Dabei kann einer der möglichen Folgezuständen durchaus gleich dem Ausgangszustand sein.)

Nicht die Übergänge werden durch Information bewirkt, sondern lediglich die Auswahl zwischen mindestens zwei Folgezuständen. (Kausalität)

Vorbedingung ist, daß das System bei der Auswahl auf die möglichen Muster eingestellt ist.

Freie Universität Berlin

Eigenschaften von Wissen

- unmittelbare Zugreifbarkeit
- enge Vernetzung
- Fundierung
- bei propositionalem Wissen:
  - Inferenzfähigkeit
  - Wahrheit bzw. soziale Akzeptanz, Nützlichkeit

Aufgaben des Informationsmanagements (I)

Verwaltung und Nutzbarmachung von sehr großen Informationsmengen, wie wir sie heute bereits auf dem WWW, in Intranets und in großen Text-Datenbanken finden.

Das Netz macht sie erst einmal nur verfügbar.

Im Gegensatz zu herkömmlichen Datenbanken ist die Information viel weniger vorstrukturiert (in Sinne der Strukturierung von Computerdaten). Auf der anderen Seiten sind die relevanten inhaltlichen Strukturen natürlich weitaus komplexer.

Aufgaben des Informationsmanagements (II)

Information wird

- gewonnen
- kategorisiert
- gefiltert
- zusammengeführt
- strukturiert
- dem Benutzer zugeführt
- adäquat präsentiert

Probleme des Informationsmanagements

- Distributivität: Information liegt auf verschiedenen Maschinen
- Heterogenität:
  - Vielzahl von Dokumentformaten
  - Multilingualität
  - Multimedialität (z. B. Sprache, Bilder, Klänge)
  - Multimodalität (z. B. geschr. u. gesprochene Sprache, Filmdateien o. Realzeitübertragungen)
- Unstrukturiertheit:
  - keine einheitliche Klassifikation
  - keine einheitliche interne Strukturierung
  - keine einheitliche und verläßliche Hypertextverknüpfung
- Redundanz: Viele Informationen sind mehrfach vorhanden.

Einige Technologien

Sammeln (*gathering*)

Indizieren (*indexing*)

Kategorisierung (*categorization*)

Gruppierung (*clustering*)

Zusammenfassung (*summarization*)

Informationsextraktion (*information extraction*)

Informationsfusion (*information fusion*)

Berichtsgenerierung (*report generation*)

Textübersetzung (*text translation*)

Informationsgewinnung

Sammeln (gathering)

Data Mining auch Text Mining

Konversion z. B. Einscannen, OCR, Transkription

Agenten z. B. WebBots

Sprache im WWW

Sprache ist nur ein Medium auf dem WWW.

Aber unter den verschiedenen Medien hat die Sprache einen besonderen Status.

Bücher, Filme, Bilder, Musikstücke und Computerprogramme beschreiben und finden wir am besten mit Sprache.

Nur mithilfe der Sprache können wir Wissen strukturieren und sinnvoll vernetzen.

Die Sprache ist das Gewebe des World Wide Web

Menschliche Sprache

Die Sprache hat Seiten, die dem Menschen leichtfallen,
dem Computer hingegen schwer. Insbesondere:

- Ambiguität: viele Wörter und Phrasen haben mehrere Bedeutungen
- Paraphrasen: es gibt viele Möglichkeiten, das Gleiche auszudrücken
- Ungenauigkeit: oft ist die Bedeutung von Ausdrücken unscharf

Heutige Suchtechnologie

- Wort-Index
- Boolsche Kombinationen
- verschiedene Indexierungsverfahren
- eingeschränkte Morphologie
- Sortierung nach Relevanz
- Suche in mehreren Sprachen

Probleme für heutige Suchmaschinen

Sie finden nicht genug!

- andere Wortformen
  der Herzog, *des Herzogs*, *die Herzöge*
- Unter- und Überbegriffe
  Alfa Romeo Zagato > roadster > sports car > car > motor vehicle > vehicle
- Paraphrasen
  steuerliche Gründe, *Steuergründe*, *steuerliche Erwägungen*,
  steuerliche Überlegungen, *fiskalische Erwägungen*, *um Steuern zu sparen*, ...

Paraphrasen: Ein kleines Experiment (I)

Nehmen wir an, Sie suchten nach Automobilfirmen

und gäben daher der Suchmaschine (z.B. HOTBOT) den Suchbegriff

„Automobilfirmen“

Im Englischen suchten Sie nach:

„automobile companies“

Paraphrasen: Ein kleines Experiment (II)

automobile companies 704 Automobilfirmen 55

Paraphrasen: Ein kleines Experiment (III)

automobile companies 704	Automobilfirmen 55
car builders 233	Autohersteller 320
car makers 1846	Autobauer 131
auto makers 2307	Autoproduzenten 26
automobile makers 181	Autofabrikant 89
car companies 3046	Autofirmen 86
cars companies 14	Pkw Hersteller 15
motor companies 194	Automobilunternehmen 57
auto companies 1345	Automobilhersteller 602
car manufacturers 3056	Kfz-Hersteller 42
motor manufacturers 582	Autounternehmen 9
automobile manufacturers 4263	Automobilkonzerne 83
manufacturers of cars 151	Unternehmen der Automobilbranche 4
manufacturers of autos 15	Hersteller von Autos 4
manufacturers of automobiles 165	Hersteller von Automobilen 13
manufacturers of motor vehicles 55	Hersteller von Kraftfahrzeugen 3

Weitere Probleme

Sie finden zu viel!

- Ambiguität
  deutsch: Zug, Bahn, Leitung, Schalter
  englisch: terminal, line, engine
- Polysemie
  Buch, Schule, printer
- Eigennamen
  Personennamen: Maurer, Washington, Chase
  Ortsbezeichnungen: Essen, Halle, Bismarck

Das Web ist mutlilingual

Das WWW war anfangs vorherrschend monolingual

(1994: 96 % aller WWW Seiten englisch)

Nicht-englische Inhalte nehmen schneller zu.

(1996: 91 % englisch, 2000 ca. 85 %)

Relevante Faktoren

Entwicklung vom Avantgardemedium zum Massenmedium

Ausbreitung in neue Regionen (Lateinamerika, Asien, arabische Welt)

Digitalisierung großer Bibliotheken in vielen Ländern

Rolle des WWW als globaler Handelsplatz

Rolle des WWW als Medium für politische Information und Propaganda

Zunahme sozialer und kultureller Inhalte

Die Zukunft des WWW ist vielsprachig.

Noch mehr Probleme!

- Andere Schriftsysteme müssen kodiert und dargestellt werden:
  Chinesisch, Japanisch, Arabisch, Griechisch, ...
- Die Wortbildungsregeln der Sprachen geraten sich ins Gehege:
  Skat vs. skating
  Limes vs. lime
- sprachübergreifende Ambiguität stört bei der Suche:
  Brief vs. brief overview
  Post vs. post messages
  Porto vs. Porto travel information
  Haut vs. Haut Barr
  cute vs. cute girls

Multilingualität als Herausforderung

Eine große Chance tut sich auf:

Es wird möglich sein,
durch das niedergeschriebene Wissen der Menschheit zu navigieren,
ohne an der Sprachgrenze stehenbleiben zu müssen.

Diese technologische Herausforderung erfordert aber
Fortschritte auf den folgenden Gebieten:

- lexikalische Semantik
- konzeptuelle Strukturierung
- Verbesserungen in maschineller Übersetzung

Maschinelle Übersetzung

Die vollautomatische maschinelle Übersetzung (fully automatic maschine translation = FAMT) beliebiger Texte ist heute nicht möglich.

Das liegt nicht an der linguistischen Verarbeitung der Texte,
sondern am fehlenden Wissen der Maschine über die Inhalte.

Für sehr eingeschränkte Gegenstandsbereiche und Textarten können aber brauchbare Übersetzungen geliefert werden.

Ansonsten dient die maschinelle Übersetzung heute erfolgreich als Vorstufe für menschliche Übersetzung (machine-assisted human translation = MAHT).

MÜ ist dennoch brauchbar

Eine zufriedenstellende automatische Übersetzung beliebiger Texte ist heute also nicht möglich.

Aber die Technologie liefert Übersetzungen, die den Leser sehr wohl das Thema und die wesentlichsten Inhalte erkennen lassen.

DFKI arbeitet mit dem Übersetzungssystem LOGOS.

Andere große Übersetzungssysteme (SYSTRAN, METAL) werden ebenfalls für WWW Anwendungen eingesetzt.

Die Übersetzungen nennen wir indikative Übersetzungen.

Ausblick

Die Strukturierung des digitalen menschlichen Wissens ist eine der großen Herausforderungen des nächsten Jahrhunderts.

Die Sprachtechnologie ist eine Schlüsseltechnologie für dieses ehrgeizige Vorhaben, denn die Sprache ist das Gewebe des Wissens.

Informationsextraktion (I)

In der IE werden gezielt relevante Informationen aus Texten herausgesucht und strukturiert.

Bremen, 14.10.1997, wiwo: Lagersoftware weiter im Aufwind

Die Bremer Firma Trade Consult hat auf einer Pressekonferenz in Hannover die Version 2.0 ihrer erfolgreichen Lagerverwaltungssoftware Store Age vorgestellt. Die neue Version ermöglicht jetzt auch ...

Auf der Pressekonferenz gab Geschäftsführer Franz Merleback auch die Umsatzzahlen der Softwareschmiede für das 3. Quartal bekannt. Wurden im zweiten Quartal bereits über 30 Millionen Mark umgesetzt, so konnte Merleback jetzt das stolze Ergebnis von 42,5 M ...

Informationsextraktion (II)

In der IE werden gezielt relevante Informationen aus Texten herausgesucht und strukturiert.

Bremen, **14.10.1997**, wiwo: Lagersoftware weiter im Aufwind

Die Bremer **Firma** *Trade Consult* hat auf einer Pressekonferenz in Hannover die Version 2.0 ihrer erfolgreichen Lagerverwaltungssoftware Store Age vorgestellt. Die neue Version ermöglicht jetzt auch ...

Auf der Pressekonferenz gab Geschäftsführer Franz Merleback auch die **Umsatzzahlen** der Softwareschmiede für das 3. Quartal bekannt. Wurden im *zweiten Quartal* bereits über *30 Millionen Mark* **umgesetzt**, so konnte Merleback *jetzt* das stolze Ergebnis von *42,5 M* ...

Ausgabe in tabellarischer Form

Firma	96Q4	1996	97Q1	97Q2	97Q3	97Q4	1997	Diff.
ComSoft		120Mio				110Mio		
Trade Consult				**30 Mio**	**42,5Mio**			
Z&M					71,0Mio			

Ajdukiewicz, Kasimir. 1934. Sprache und Sinn. *Erkenntnis* 4, 100–138.

Bar-Hillel, Yehoshua, Perles, Micha A. und Shamir, Eliahu. 1961. On Formal Properties of Simple Phrase-Structure Grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 14(2), 143–172.

Barwise, Jon und Perry, John. 1987. *Situationen und Einstellungen – Grundlagen der Situationssemantik*. Berlin, New York: de Gruyter.

Berman, Judith. 2003. *Clausal Syntax of German*. Studies in Constraint-Based Lexicalism, Stanford, CA: CSLI Publications.

Berman, Judith und Frank, Anette. 1996. *Deutsche und französische Syntax im Formalismus der LFG*. Linguistische Arbeiten, Nr. 344, Tübingen: Max Niemeyer Verlag.

Borsley, Robert D. 1999. *Syntactic Theory: A Unified Approach*. London: Edward Arnold, zweite Auflage.

Bresnan, Joan (Hrsg.). 1982. *The Mental Representation of Grammatical Relations*. MIT Press Series on Cognitive Theory and Mental Representation, Cambridge, MA/London: MIT Press.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Oxford, UK/Cambridge, USA: Blackwell.

Bußmann, Hadumod (Hrsg.). 1990. *Lexikon der Sprachwissenschaft*. Kröners Taschenausgabe, Nr. 452, Stuttgart: Alfred Kröner Verlag, zweite Auflage.

Carstensen, Kai-Uwe, Ebert, Christian, Endriss, Cornelia, Jekat, Susanne, Klabunde, Ralf und Langer, Hagen (Hrsg.). 2001. *Computerlinguistik und Sprachtechnologie. Eine Einführung*. Spektrum Lehrbuch, Heidelberg/Berlin: Spektrum Akademischer Verlag.

Chomsky, Noam. 1981. *Lectures on Government and Binding*. Dordrecht: Foris Publications.

Engel, Ulrich. 1977. *Syntax der deutschen Gegenwartssprache*, Band 22 von *Grundlagen der Germanistik*. Berlin: Erich Schmidt Verlag.

Fleischer, Wolfgang und Barz, Irmhild. 1995. *Wortbildung der deutschen Gegenwartssprache*. Tübingen: Max Niemeyer Verlag, zweite Auflage.

Fourquet, Jean. 1970. *Prolegomena zu einer deutschen Grammatik*. Sprache der Gegenwart – Schriften des Instituts für deutsche Sprache in Mannheim, Nr. 7, Düsseldorf: Pädagogischer Verlag Schwann.

Gazdar, Gerald, Klein, Ewan, Pullum, Geoffrey K. und Sag, Ivan A. 1985. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard University Press.

Gazdar, Gerald und Mellish, Christopher. 1989. *Natural Language Processing in Prolog*. Wokingham: Addison-Wesley. http://www.informatics.susx.ac.uk/research/nlp/gazdar/nlp-in-prolog/, 17.10.2004.

Gibson, Edward. 1998. Linguistic Complexity: Locality of Syntactic Dependencies. *Cognition* 68(1), 1–76.

Görz, Günther, Rollinger, Claus-Rainer und Schneeberger, Josef (Hrsg.). 2000. *Handbuch der Künstlichen Intelligenz*. München, Wien: Oldenburg Verlag, dritte Auflage.

Haider, Hubert. 1990. Pro-bleme? In Gisbert Fanselow und Sascha W. Felix (Hrsg.), *Strukturen und Merkmale syntaktischer Kategorien*, Studien zur deutschen Grammatik, Nr. 39, Seiten 121–143, Tübingen: original Gunter Narr Verlag jetzt Stauffenburg Verlag.

Heid, Ulrich. 2000. Morphologie und Lexikon. In Görz u. a. (2000), Seiten 665–709.

Herzog, Otthein und Rollinger, Claus-Rainer (Hrsg.). 1991. *Text Understanding in LILOG*. Lecture Notes in Artificial Intelligence, Nr. 546, Berlin/Heidelberg/New York, NY: Springer Verlag.

Jackendoff, Ray S. 1977. $\overline{\text{X}}$ *Syntax: A Study of Phrase Structure*. Cambridge, MA/London, England: MIT Press.

Jurafsky, Daniel und Martin, James H. 2000. *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. New Jersey:

Prentice Hall.

Kamp, Hans und Reyle, Uwe. 1993. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Studies in Linguistics and Philosophy, Nr. 42, Dordrecht/Boston/London: Kluwer Academic Publishers.

Kay, Martin. 1984. Functional Unification Grammar: A Formalism for Machine Translation. In Yorick Wilks (Hrsg.), *Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, Seiten 75–78, Stanford University, California: Association for Computational Linguistics.

Kolb, Hans-Peter und Thiersch, Craig L. 1991. Levels and Empty Categories in a Principles and Parameters Based Approach to Parsing. In Hubert Haider und Klaus Netter (Hrsg.), *Representation and Derivation in the Theory of Grammar*, Studies in Natural Language and Linguistic Theory, Nr. 22, Dordrecht/Boston/London: Kluwer Academic Publishers.

Konieczny, Lars. 1996. *Human Sentence Processing: a Semantics-Oriented Parsing Approach*. Dissertation, Universität Freiburg, iIG-Berichte 3/96. http://www.researchgate.net/publication/36150321_Human_sentence_processing_a_semantics-oriented_parsing_approach_/, 20.04.2013.

Kunze, Jürgen. 1975. *Abhängigkeitsgrammatik*. studia grammatica, Nr. 12, Berlin: Akademie Verlag.

Manning, Christopher D. und Schütze, Hinrich. 2000. *Foundations of Statistical Natural Language Processing*. Cambridge, MA/London, England: MIT Press, dritte Auflage.

Müller, Stefan. 1994. *Prolog und Computerlinguistik – Teil 1 Syntax*. Vorlesungsskripte Computerlinguistik, Humboldt-Universität zu Berlin. http://hpsg.fu-berlin.de/~stefan/Pub/prolog.html, 24.11.2013.

Müller. Stefan. 1999. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Linguistische Arbeiten, Nr. 394, Tübingen: Max Niemeyer Verlag. http://hpsg.fu-berlin.de/~stefan/Pub/hpsg.html, 24.11.2013.

Müller, Stefan. 2013. *Head-Driven Phrase Structure Grammar: Eine Einführung*. Stauffenburg Einführungen, Nr. 17, Tübingen: Stauffenburg Verlag, dritte Auflage. http://hpsg.fu-berlin.de/~stefan/Pub/hpsg-lehrbuch.html, 24.11.2013.

Pinkal, Manfred. 2000. Semantikformalismen für die Sprachverarbeitung. In Görz u. a. (2000), Seiten 739–782.

Pollard, Carl J. und Sag, Ivan A. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics, Chicago, IL/London: The University of Chicago Press.

Sag, Ivan A., Wasow, Thomas und Bender, Emily M. 2003. *Syntactic Theory: A Formal Introduction*. CSLI Lecture Notes, Nr. 152, Stanford, CA: CSLI Publications, zweite Auflage.

Schielen, Michael. 2000. Semantic Construction. In Wahlster (2000), Seiten 200–215.

Shieber, Stuart M., Uszkoreit, Hans, Pereira, Fernando, Robinson, Jane und Tyson, Mabry. 1983. The Formalism and Implementation of PATR-II. In *Research on Interactive Acquisition and Use of Knowledge*, Seiten 39–79, Menlo Park, CA: Artificial Intelligence Center, SRI International.

Steedman, Mark J. 2000. *The Syntactic Process*. Language, Speech, and Communication, Cambridge, MA/London, England: MIT Press.

Tarvainen, Kalevi. 2000. *Einführung in die Dependenzgrammatik*. Reihe germanistische Linguistik, Nr. 35, Tübingen: Max Niemeyer Verlag.

Tesnière, Lucien. 1980. *Grundzüge der strukturalen Syntax*. Stuttgart: Klett-Cotta.

Trost, Harald. 1991. Recognition and Generation of Word Forms for Natural Language Understanding Systems: Integrating Two-Level Morphology and Feature Unification. *Applied Artificial Intelligence* 5, 411–457.

Uszkoreit, Hans. 2000. Foliensatz zur Einführung in die Computerlinguistik.

Völler, Reinhard. 2001. Automatentheorie und formale Sprachen. Vorlesungsskript Universität Hamburg. http://www.informatik.haw-hamburg.de/~voeller/autom.pdf, 01.11.2002.

Wahlster, Wolfgang (Hrsg.). 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*. Artificial Intelligence, Berlin/Heidelberg/New York, NY: Springer Verlag.

Wood, Mary McGee. 1993. *Categorial Grammars*. London, New York: Routledge.