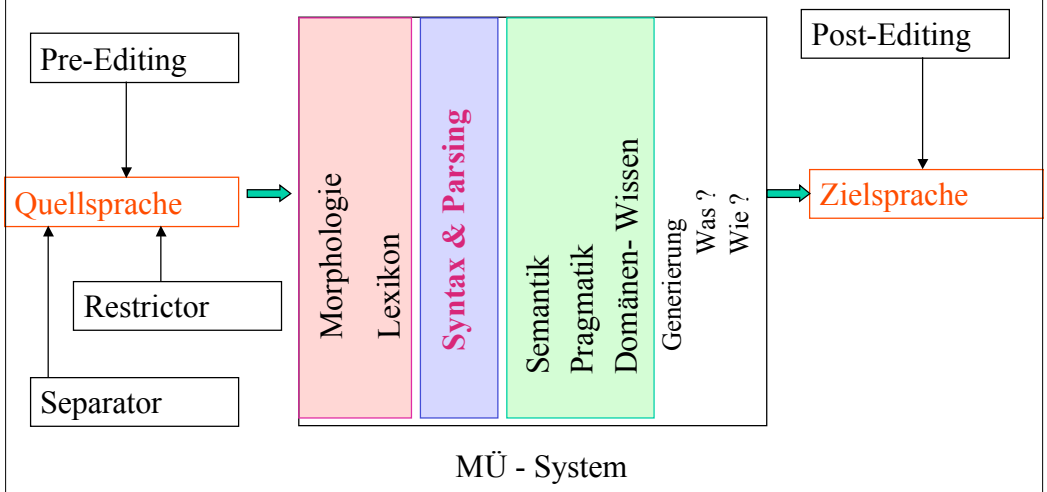
Syntax und Parsing



MÜ - System

Definitionen -1-

- **Syntax**: Beschreibung der Regeln zur korrekten Kombination der lexikalischen Elemente einer Sprache.
- **Parsing:** Maschinelle (formale) syntaktische Analyse

Definitionen -2- Konstituenten

- NP - used to refer to things, places, concepts, events, qualities
  - single pronouns
  - proper nouns
  - common nouns = head
  - head + specifiers and (or) qualifiers
    - » specifiers : ordinals, cardinals, determiners, possesives, quantifiers
    - » qualifiers : adjectives or nouns used as modifiers
- VP
  - verb = head
  - head andcomplemets
- AdjP
  - single adjectives
  - degree modifier
- PP
  - Preposition + other phrase
- AdvP : indicators of degree, locations, frequency etc.

Gliederung

- Syntax
- Parsing
- Syntax und Parsing in MÜ Systeme

Zwei Sichten auf die Syntax

1. Im Zentrum steht die syntaktische Struktur als abstrakte Beziehung zwischen präterminalen Symbolen. Diese Beziehung kann dargestellt werden als
   - Ersetzungsregel
   - Netzübergänge oder
   - Erkennungsprozeduren

   Ein Lexikon stellt die Verbindung her zwischen den "Satzmustern"und den Wörtern

   Regelgrammatik

2. Im Zentrum stehen Wortklassen oder Lexikoneinträge, die die Verbindbarkeit von Wörtern oder Wortklassen beschreiben. Durch Verbinden der Strukturinformation zu immer größeren Einheiten wird eine Satzbeschreibung erreicht

   Lexikalische Grammatik

In beiden Grammatiktypen können Dependenz- und Konstituenz-Strukturen dargestellt werden.

Grammatische Beschreibungstypen

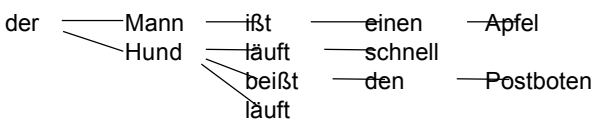
- Patterns
- Rekursive Netze (RTN)
- Kontextfreie Grammatik
- Dependenzgrammatik
- Konstituentenstrukturgrammatik
- Funktionale Grammatik
- Unifikationsgrammatik
- Kasusgrammatik

Patterns -1-

Patterns

- Direkte
  - wörtliche "die linke hintere Maschine jetzt anfahren"
  - offene `"(&c (das)(ist) $$)"` matcht alle Permutationen und akzeptiert ein beliebiges Folgewort: *das ist ein Fehler* und *ist das eine Maschine*
  - lexikalische "Det, Adj, Adv, Adj, N, V" *Der rote sehr helle Knopf leuchtet*
- mit Variablen "die 1 2 3 jetzt anfahren"

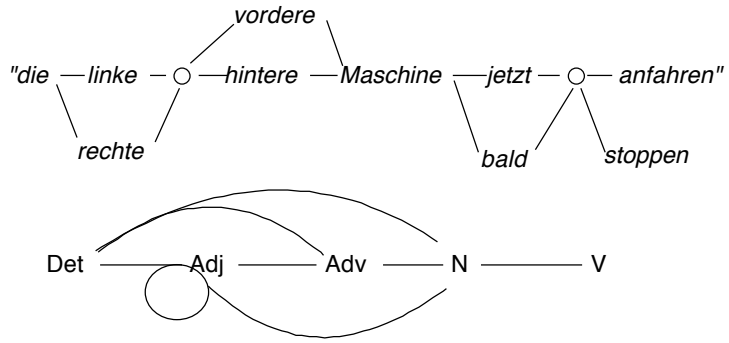
Wortübergangsgraph (Strukturierte Pattern)



Patterns -2-

Strukturierte Patterns

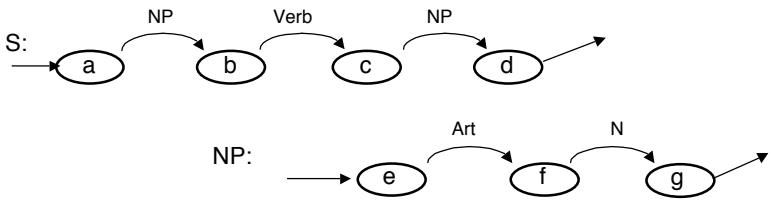
Übergangsnetz z (Endlicher Automat, Zustandsgraph)



Regulärer Ausdrück Det (Adj*) (Adv) N V

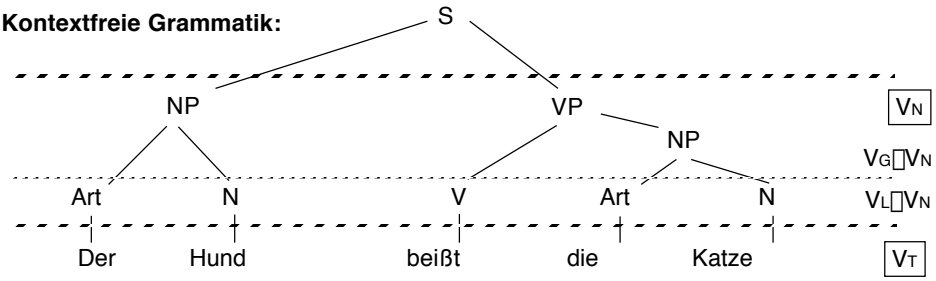
Rekursive Netze

Rekursives Netz (RTN)



Kontextfreie Grammatik

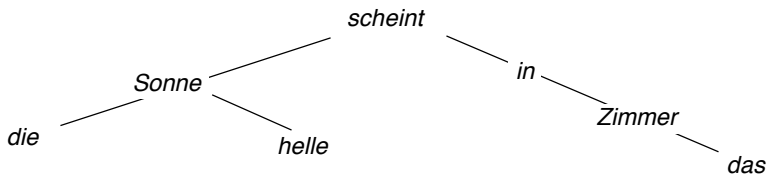
Kontextfreie Grammatik:



Verarbeitungsferne systematische Methode

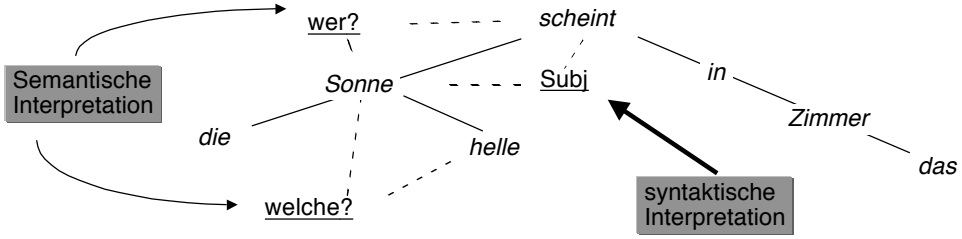
Dependenzgrammatik -1-

Dependenz (Beispiel: Head und Modifier-Grammatik):



Dependenzgrammatik-2-

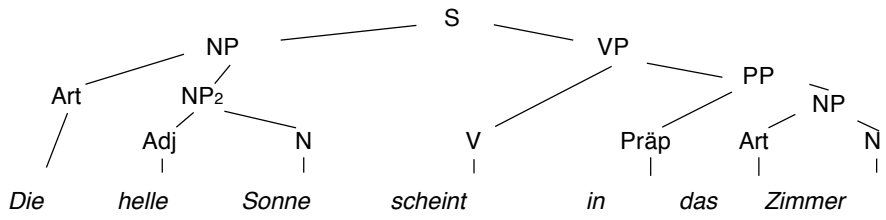
Dependenz (Beispiel: Head und Modifier-Grammatik):



Eigenschaften: Stellt die syntaktische Abhängigkeit vom Verb dar,
keine Projektion auf Oberflächenkette
basiert auf der Verbvalenz (mit obligatorischen und freien Ergänzungen)

Konstituentenstrukturgrammatik

Konstituenz (Beispiel IC-Analyse):



Eigenschaften: Unmittelbare Bestandteile des Satzes werden beschrieben,
Oberflächenstruktur bleibt sichtbar, Meist binäre Teilung,
über Teilbarkeit entscheidet Substitutionstest (Minimalpaare),
Probleme bei Diskontinuität von Elementen.

Konstituenten -Findungsprozeduren

- **Substitutionstest:** Elemente, die man für einander ersetzen kann, ohne daß sich an der Grammatikalität des Satzes etwas ändert, sind Konstituenten.

  Die langen Winterabende / Die Winterabende / Winterabende versetzen mich in Melancholie

- **Pronominalisierungstest:** Was sich zusammen pronominalisieren läßt, ist eine Konstituente

  Ede und Caro wohnen in der Maxstraße. Sie haben ein Kind.

  -" - *.Dort wohnen nur reiche Leute*

- **Fragetest:** Wonach man fragen kann, ist eine Konstituente

  Heute kommt meine Mutter zu Besuch. Wer kommt heute zu Besuch?

- **Koordinationstest:** Was sich koordinieren läßt, ist eine Konstituente

  Ede gab eine sehr kluge und keineswegs triviale Antwort

Funktionale Grammatiken - 1-

- Alle grammatischen Fakten können als constraints beschrieben werden:
  - *S* □ *NP VP* kann man lesen als die Bedingung , daß ein korrekter Satz aus NP und VP bestehen muß (top-down gelesen),
  - *Person = Nom Sg* kann man lesen als die Bedingung, daß das Wort *Person* nur eine syntaktische Stelle einnehmen kann, die die Eigenschaft *Nom Sg* hat (bottom-up gelesen).
  - für jede syntaktische Kategorie (klassisch: Nonterminale) gibt es eine Auswahl an Subkategorien und an relevanten Merkmalen.

Funktionale Grammatiken -2-

- Mit einem solchen constraint-Formalismus werden alle Stufen der Syntax beschrieben:
  - Satz,
  - Kategorien und
  - Lexeme.
- Ein constraint satisfaction system baut aus Teilen eine Gesamtinter- pretation zusammen oder generiert aus Teilen eine Gesamtstruktur

Unifikationsgrammatiken

- Beschreibung von syntaktischen Merkmalen durch Attribut-Wert-Matrix (attribute terms / feature terms)
- Zentrale Operation: Unifikation

- **Grammatik-Formalismen**:

	DCG	Pereira/Shieber
	PATR II	Shieber, Karttunen
Grammatik-Theorien:		
Functional Unification Grammar	FUG	Kay
Lexical-Functional Grammar	LFG	Kaplan/Bresnan
Generalized Phrase Structure Grammar	GPSG	Gazdar/Klein/ Pullum/Sag
Head-Driven Phrase Structure Grammar	HPSG	Pollard/Sag

Nach Görz

Vorteile unifikationsorientierter Grammatiken

einheitliche Operation für Merkmale aller Ebenen (Phonologie, Morphologie, Syntax, Semantik)

Deklarative Form daher lesbar ohne Kenntnis der Prozesse

Robustheit weil korrekte Teilstrukturen definiert und durch Vorhersagbarkeit von Wirkungen allein aus den im Grammatikformalismus notierten Regeln

Modularität durch hierarchische Strukturierung der Information

Homometrie d.h. kleine Änderungen in den Regeln haben geringe Wirkungen auf die Resultate, große Änderungen, große Wirkungen

nach Görz

Eigenschaften unifikationsbasierter Grammatiken

oberflächenorientiert d.h. es ist eine direkte Charakterisierung der faktischen Ordnung der Oberflächenstruktur - der aktuellen Anordnung der Zeichen in einem Satz - möglich

interpretativ in dem Sinne, daß mit den Zeichenketten wohldefinierte Beschreibungen assoziiert werden

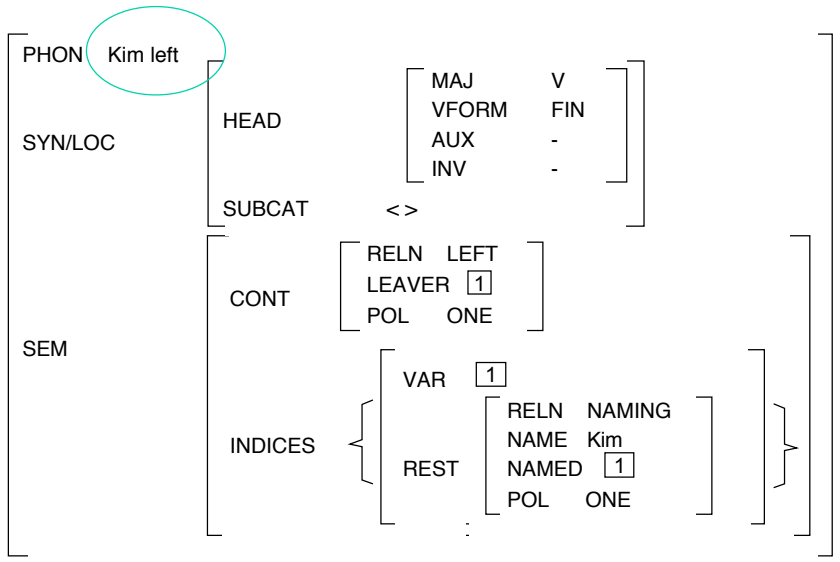
induktiv durch rekursive Definition der Assoziation von Beschreibungen mit Zeichenketten, so daß neue Beschreibungen aus Beschreibungen von Teilketten auf wohldefinierte Weise erzeugt werden, wobei die Kombinations- operationen von Zeichenketten vorgegeben sind

deklarativ die Assoziation zwischen Zeichenketten und Beschreibungen ist in einer Weise definiert, die nur angibt, welche Assoziationen zulässig sind, nicht, wie sie berechnet werden.

merkmalsbasiert durch Mengen von Paaren aus Merkmalsnamen und (ggf. strukturierten) Merkmalswerten (f-Strukturen)

HPSG

Carl Pollard
Ivan A. Sag:
Information-Based Syntax and Semantics
Stanford 1987. S. 93

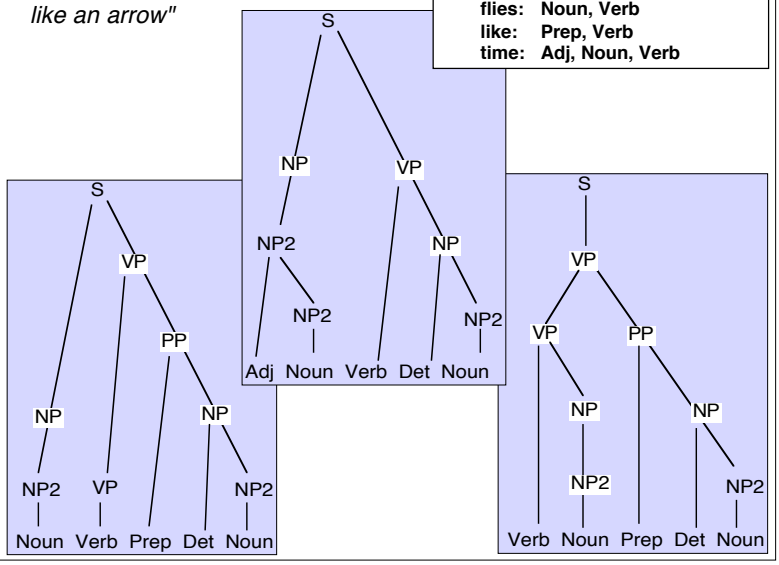


Syntaktische Mehrdeutigkeit

Grammatik
S ∅ NP VP
(Aussagen)
S ∅ VP (Befehle)
NP ∅ Det NP2
NP ∅ NP2
NP ∅ NP PP
NP2 ∅ Noun
NP2 ∅ Adj NP2
PP ∅ Prep NP
VP ∅ Verb
VP∅ Verb NP
VP ∅ VP PP

"time flies like an arrow"

Lexikon
an: Det
arrow: Noun
flies: Noun, Verb
like: Prep, Verb
time: Adj, Noun, Verb



Schwierigkeiten für deutsche Grammatiken

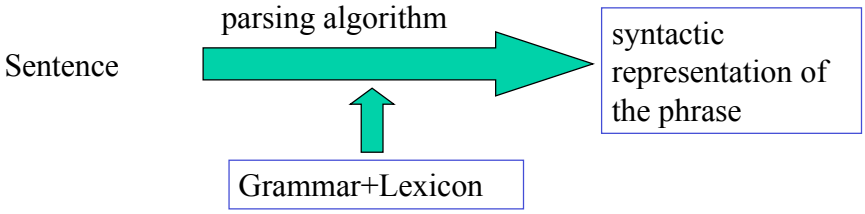
1. Flexions- und Kongruenzregeln
   - Person-Numerus-Kongruenz bei Verb und Subjekt: *"ich gebe"*
   - Kasus-Genus-Numerus-Kongruenz in NPs: *"einem großen Bären"*
   - Syntaktische Restriktionen des Verbs für Ergänzungen: *"jemanden einer Sache verdächtigen"*
   - artikelbestimmte Flexion des Adjektivs: *"ein groß**er** / der groß**e** ..*
2. Variable Satzgliedstellung für einige Teilstrukturen (NPs)
3. Diskontinuierliche Komponenten
   - Verbalklammer: *"er **wird** nächste Woche **ankommen**"*
   - Fernstellung des Nebensatzes: *"Die Meinung ist falsch, daß das einfach ist"*
4. Koordination
   - Präpositional-Attribute: *"Der Dicke in meinem Abteil mit dem riesigen Koffer auf dem Sitz neben sich"*
   - Adjektiv-Attribute: *"die phantastische, überwältigende, überzeugende Lösung"*
   - Nominalgruppen: *"Vorlesungen, Seminare und andere Veranstaltungen"*

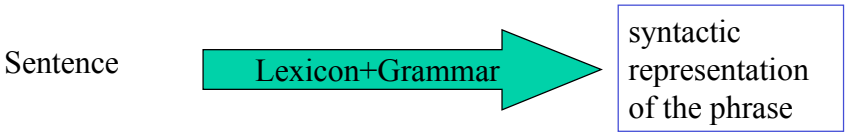
Gliederung

- Syntax
- Parsing 
- Syntax und Parsing in MÜ Systeme

Parser

Two types:





Grammatik und Parser

nach Hellwig

- Trennung von Grammatik und Parser
  Gründe:
  - Grammatik existiert unabhängig von Programm und Daten
  - Grammatik ist eine Beschreibung der Objektsprache in deklarativer Form (unabhängig vom Erkennungsprozeß)
  - Parser fußt auf der Syntax des Grammatikformalismus, nicht auf den Inhalten der Grammatik
  - ⇒ Interpretierender Parser

  Vorteile: Getrennte Erweiterbarkeit, Lesbarkeit der Grammatik durch Linguisten, Parserbau während des Grammatikaufbaus, Anwendung auf mehrere Sprachen, Verwendung der Grammatik beim Generieren.

- Integration von Grammatik und Parser
  Gründe:
  - Grammatik ist in das Programm eingebettet
  - Formulierung der Sprachbeschreibung mit Blick auf die Verarbeitung
  - Grammatik ist eine Sprachbeschreibung in prozeduraler Form (Erkennungsanweisungen)
  - ⇒ Prozeduraler Parser

  Vorteil: Effizienz, Formulierung ohne Grammatikformalismus möglich

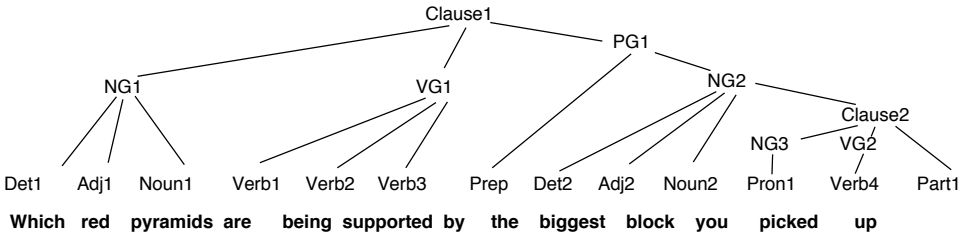
Parsingstrategien - 1 -

• **Architektur**	parallel	vs	sequentiell
• **Startpunkt**	top-down	vs	bottom-up
• **Regelanwendung**	deterministisch	vs	nicht-deterministisch
• **Auswahl**	first guess	vs	informierte Auswahl
• **Vorgehen**	Breitensuche	vs	Tiefensuche
• **Richtung**	rechts ---> links	vs	links ---> rechts
		vs	Insel-Parsing
• **Steuerung**	syntaktisch	vs	semantisch
		vs	semantisch-pragmatisch

Parsingstrategien - 2 -

• **Zielstruktur**	syntaktisch	vs	semantisch
• **Formalisiertheit**	ad hoc	vs	uniform
• **Ebenenspezifik**	ebenengebunden	vs	systemorientiert
• **Genauigkeit**	partiell genau	vs	formal beschreibbar
• **Informiertheit**	blind	vs	vorherschauend
• **Betrachtungsraum**	wortweise	vs	gruppenweise
• **Ambiguitäts-behandlung**	erfolgsorientiert	vs	exhaustiv

Parsing-Ergebnis: SHRDLU Syntakt. Relationen



Clause1 (*Which red pyramids are being supported by the biggest block you picked up?*) Clause Major Question Wh- Transitive Passive (Present in Present) Subject-Question Agent

NG1 (*Which red pyramids*) NG Determined Indefinite Question Plural Subject

- Det1 (*Which*) Determiner Question-Determiner Singular Plural
- Adj1 (*red*) Adjective
- Noun1 (*pyramids*) Noun Plural

VG1 (*are being supported*) VG Finite Passive (Present in Present)

- Verb1 (*are*) Verb Auxiliary Intensive Be Plural Present
- Verb2 (*being*) Verb Auxiliary Intensive Be Ing
- Verb3 (*supported*) Verb Transitive Past-Participle

Parsing-Ergebnis: GUS
Semantische Slot Filler

"I want to go to San Diego on May 28"

(Client Declare
(Case for *want* / e (Tense Present)
Agent = Dialog.Client.Person
Event = (Case for *go* (Tense Present)
Agent = Dialog.Client.Person
To-Place = (Case for City
Name = *San Diego*)
Date = (Case for Date
Month = *May*
Day = *28*))))

Nach Winograd

Parsing-Ergebnis: DB-Interface
Suchausdruck in einer DB-Sprache

"Liste die Namen aller Lieferanten, die mindestens alle die Teile liefern, die auch vom Lieferanten S2 geliefert werden"

```
SELECT          UNIQUE  S#
FROM            SP  SP X
WHERE           NOT EXISTS
                (SELECT *
                FROM SP SP Y
                WHERE S# = ´S2´
                AND             NOT EXISTS
                                (SELECT *
                                FROM SP
                                WHERE S# = SP X. S#
                                AND P# = SP Y. P#))
```


Parsing-Ergebnis: Tabelle

REF	SCALE	PHASE	YIELD	TEMP
para.1	small	solid	77%	-78 to 20
TIME	ENERGY	APPARATUS	FEATURES	
	cooling		IR. N MR. MS	
REG. NO	FUNCTION	AMT.	AUTHOR ID	
78624-62-1	product	2.70 g	7a	
78624-61-0	reactant		6a	
13274-48-6	reactant	1.24 g	N-methyltriazolinedione	
	solvent	80 ml	pentane	
	solvent	40 ml	ethyl acetate	

Formular-Repräsentation des Systems SIE für eine chemische Reaktionsbeschreibung

Parsing-Ergebnis: Semantische Repräsentation

"who is the man that leads the LOKI project?"

```
[  request (referent(_5747))
   presuppose (exists (_4340))  ]
   some (_4340)
[  unique (_4407)
   single (_4407)
   instance (_4407, person)
   propval (person,_4407,sex,male)
   [  some (_4725)
      [  unique (_5033)
         single (_5033)
         instance (_5033,project)
         propval (project,_5033,name,str (LOKI)) ]
```

```
      instance (_4725,leading)
         propval (leading,_4725,theta,_5033)
         propval (leading,_4725,alpha,_4407)
         topic (_4407)  ]  ]
instance (_4340,identity)
propval (identity,_4340,alpha,_4407)
propval (identity,_4340,theta,_5747)
topic (_4407)
[  some (_5747)
   single (_5747)
   instance (_5747,person) ]
```


Pattern matching -1-

- Pattern = ein syntaktischer Rahmen für lexikalisch-semantische Äquivalenzklassen.
- Patterns beschreiben häufige Ausdrücke einer Sprache.

Welche Übung Vorlesungen gibt es im Montag Grundstudium ?

Pattern matching -2-

- Sammlung von Patterns spezifiziern
  - eine feste Wortreihenfolge
  - Plätze für variable Wörter
- Ein Satz kann vearbeitet werden, wenn es (mindestens) ein gültiges Pattern für diesen Satz gibt.
- Für einen Satz kann man ein oder mehrere Patterns benutzen
- Probleme: Syntaktisch unkorrekte Sätze werden auch verarbeitet
  - z. B: Welche am Montag gibt es ?

Sämtliche im System SIR (Raphael 1967) verwendeten Patterns

- Is _ _?
- How many _ does _ have?
- How many _ does _ own?
- What is the _ of _?
- How many _ are parts of _?
- How many _ are there on _?
- _ has as a part one _.
- There is one _ on _.
- _ is just to the left of _.
- Is _ just to the left of _?
- _ is just to the right of _.
- Is _ just to the right of _?

- _ is _.
- _ has_.
- _ owns _.
- Where is _?
- Is _ part of _?
- Does _ own _?
- _ is _ part of _.
- There are _ on _.
- _ is to the left of _.
- Is _ to the left of _?
- _ is to the rigjt of _.
- Is _ to the right of _?

Pattern Operationen in DYPAR

• Optionales Element	?
• Disjunktion	|
• Wertaufnehmende Variable	:=
• Wertabgebende Variable	=
• Beliebiges Einzelwort	$
• Beliebige Zahl	$ n
• 0-n Wiederholungen	*
• 1-n Wiederholungen	+
• n Wiederholungen	≠
• Bis hin zu	&u
• Rest der Eingabe	$ r
• Alle Permutationen	&c
• konsumierende Negation	~
• nicht konsumierende Negation	&n
• Ersetzung von Ausdruck durch Funktionswert	&i

DYPAR Patterns

Beispiele:

(&u show) konsumiert die Eingabe bis zum ersten "show" (*"Could you please show"* fi show)

(&c (Das)(ist)(so)) matcht z.B.*"So ist das"* oder *"Ist das so"*.

(!varname := (&i true <positive Antwort>)) fi alle positiven Antworten werden durch true dargestellt.

(!word):= ~<article> bindet alles außer der Negation an word

BNF-Syntaxdefinition der in DYPAR möglichen Patterns

<pattern> ::= nil
<pattern> ::= (<pattern1>)
<pattern> ::= <term>
<pattern> ::= <term> <pattern1>
<term> ::= ATOMIC CONSTANT
<term>::= $
<term>::= NON-TERMINAL
<term>::= FUNCTION
<term> ::= (≠ < n ><pattern>)
<term>::= ? <pattern>
<term> ::= * <pattern>
<term> ::= + <pattern>
<term> ::= &u <pattern>
<term> ::= ! <pattern-list>
<term> ::= &c <pattern-list>
<term> ::= &i VALUE <pattern>
<term> ::= (%f LISP-FUNCTION (VARIABLE-NAME)*)
<term> ::= (:= VARIABLE-NAME <pattern>)
<term> ::= (= VARIABLE-NAME)
<term> ::= (&m <list>)
<n> ::= 1|2|3|
<list> ::= ATOMIC CONSTANT
<list> ::= ATOMIC CONSTANT <list>
<pattern-list> ::= (<pattern-list>)
<pattern-list> ::= (<pattern>)
<pattern-list> ::= (<pattern> <pattern-list>)

Vorteile und Nachteile von Pattern - Matchern

- Vorteile:
- schnell (für zeitkritische Anwendungen geeignet)
- einfach (intuitiv formulierbar, von BS oder anderen Anwendungen bekannt)
- bei kleinem Umfang gut wartbar
- lokale Effekte

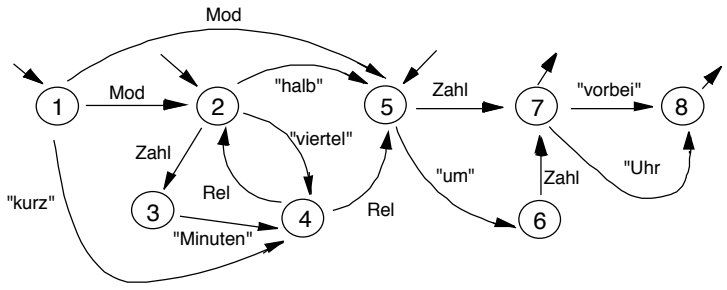
- Nachteile:
- liefern keine Strukturbeschreibung
- ungeeignet für einige syntaktische Phänomene (offene Wortfolge, Abhängigkeiten, …)
- umständlich für gleiche Strukturen an verschiedenen Stellen
- bei größerem Umfang unübersichtlich und schwer wartbar

Netzwerkgrammatiken

- Einfache Netzwerkgrammatiken (Endliche Automaten)
  Übergänge in **einem** Netz zwischen
  - Wörtern oder
  - lexikalischen Kategorien
- Rekursive Netzwerkgrammatiken (BTN / RTN)
  Übergänge in gestaffelten Netzen zwischen
  - Netzen
  - Wörtern oder
  - lexikalischen Kategorien
- Erweiterte Netzwerkgrammatiken (ATN)
  Übergänge in gestaffelten Netzen zwischen
  - Netzen
  - Wörtern oder
  - lexikalischen Kategorien

  unter :
  - Ausführung von Aktionen und
  - Aufbau einer Strukturbeschreibung

Endlicher Automat (BTN) für deutsche Zeitangaben



- Mod: eben, bald, gerade, gleich, genau
- Rel: vor nach
- Zahl: 1, 2, 3, ...59, 60.

CFG für deutsche Zeitangaben

S → Vor Zeit

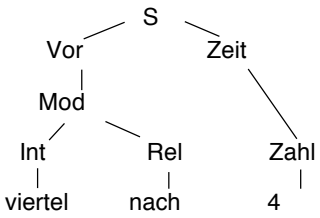
Zeit → Zahl (Uhr)

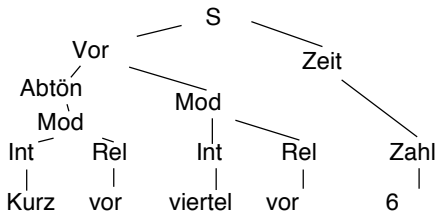
Vor → (Abtön) (Mod)

Mod → Frac | Int Rel | Mod

Abtön → (Hecke) Mod

Frac:	viertel, halb
Rel:	vor, nach
Int:	viertel, kurz
Nach:	vorbei
Uhr:	Uhr
Zahl:	1,2,3,4,...
Hecke:	fast, bald, gleich





Cocke / Kasami / Younger - Algorithmus - 1 -

Parser, die auf dem **Cocke**-Algorithmus beruhen, bearbeiten nur CFGs in Chomsky-Normalform und arbeiten

- von links nach rechts
- bottom-up
- breitenorientiert (parallele Verfolgung aller Analysepfade)
- in einem einzigen Durchgang (one-pass)

vor allem: binäre Regeln

ohne Zurücksetzen (backtracking)

Es wird für alle erreichten X mit allen zurückliegenden Y überprüft

Zunächst werden die einzelnen Wörter der Eingabekette von links nach rechts durchnummeriert.

nach Wahlster

Cocke / Kasami / Younger - Algorithmus - 2 -

Es wird eine Parsingtabelle (auch: Wegetabelle) aufgebaut:

- jedem Reduktionsschritt wird eindeutig eine Nummer zugeordnet,
- Das durch den Reduktionsschritt erzeugte nichtterminale Symbol und die angewandte Regel werden vermerkt,
- Mit LG (linke Grenze) und RG (rechte Grenze) wird relativ zur Nummerierung der Eingabekette jeweils der Teilstring der Eingabekette identifiziert, der von dem Reduktionsschritt erfaßt wird (nur bei lexikalischen Regeln ist RG = LG),
- Mit LK (linke Komponente) und RK (rechte Komponente) wird spezifiziert, aus welchen Komponenten eine neue syntaktische Kategorie gewonnen wurde, indem die entsprechenden Nummern der Reduktionsschritte angegeben werden.

Das ist nötig, weil höhere Nonterminale tiefere Nonterminale zusammenfassen können, die keine direkte Verbindung zu Terminalen haben können

06.11.2002

Cocke Algorithmus - 3 -

(1) Lies nächstes Wort w ein und notiere in der Parsingtabelle
- seine lexikalische Kategorie - seine Position z (RG = LG)
- setze RK = LK = 0

(2) Ab dem zweiten Eintrag der Parsingtabelle:
Solange es für die aktuelle Zeile in der Parsingtabelle mit der Kategorie X oder vorangehende Zeilen mit der Kategorie Y und RG (X) = Z solche Kategorien Y gibt, für die gilt
- RG (Y) = LG (X) -1 (d.h. Y steht unmittelbar links von X)
- es gibt eine Regel A □ Y X

ergänze als neue aktuelle Zeile in der Parsingtabelle
- die Kategorie A
- LG (A) = LG (Y), RG (A) = RG (X)
- LK (A) = < Nummer von Y in Parsing-Tabelle>
- RK (A) = < Nummer von X in Parsing-Tabelle>

Adjazenzregel

(3) Falls die neu eingetragene syntaktische Kategorie S ist und LG = 1 sowie RG = <Länge der Eingabekette>
dann: Stop sonst: gehe nach (1)

06.11.2002

46

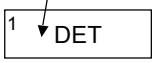
nach Wahlster

Cocke-Algorithmus - Schritt 1 -

Beispiel nach Wahlster

Det eintragen mit :
LG = I RG = I
LK = 0 RK = 0

(L1) "Det ∅ die, manche, einige"

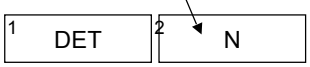


Die	*Mädchen*	*sahen*	*die*	*Kinder*	*dort*
I	*II*	*III*	*IV*	*V*	*VI*

Cocke-Algorithmus - Schritt 2 -

N eintragen mit :
LG = II RG = II
LK = 0 RK = 0

(L3) "N ∅ Mädchen, Kinder"

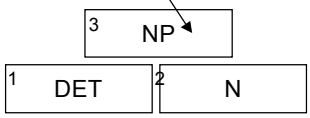


Die	*Mädchen*	*sahen*	*die*	*Kinder*	*dort*
I	*II*	*III*	*IV*	*V*	*VI*

Cocke-Algorithmus - Schritt 3 -

NP eintragen mit :
LG = I RG = II
LK = 1 RK = 2

(G10) "NP ∅ DET N"

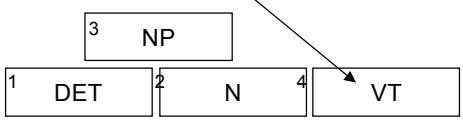


Die	*Mädchen*	*sahen*	*die*	*Kinder*	*dort*
I	*II*	*III*	*IV*	*V*	*VI*

Cocke-Algorithmus - Schritt 4 -

VT eintragen mit :
LG = III RG = III
LK = 0 RK = 0

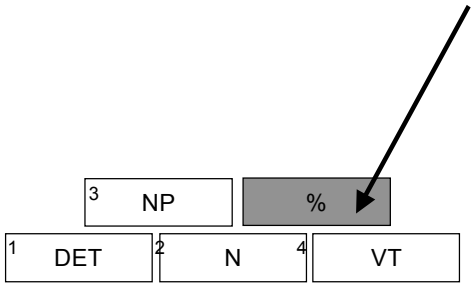
(L5) "VT ∅ sahen, ehrten"



Die	*Mädchen*	*sahen*	*die*	*Kinder*	*dort*
I	*II*	*III*	*IV*	*V*	*VI*

Cocke-Algorithmus - Schritt 4a -

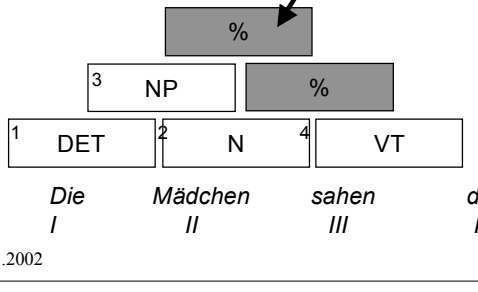
Weiterer Schritt mit N und VT ist nicht möglich:
4a Kat = % LG = II RG = III LK = 2 RK = 4 R =%



Die	*Mädchen*	*sahen*	*die*	*Kinder*	*dort*
I	*II*	*III*	*IV*	*V*	*VI*

Cocke-Algorithmus - Schritt 4b -

Weiterer Schritt mit NP und VT nicht möglich:
4b Kat = % LG = I RG = III LK = 3 RK = 4 R =%

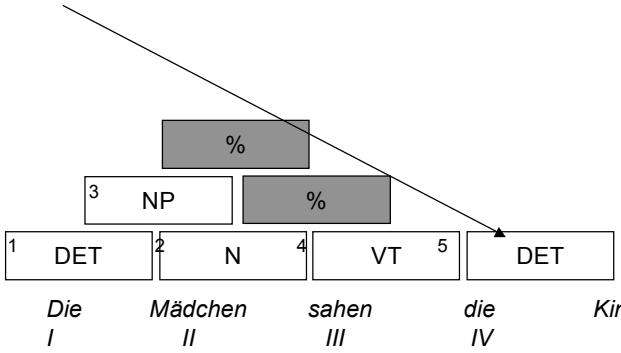


Die	*Mädchen*	*sahen*	*die*	*Kinder*	*dort*
I	*II*	*III*	*IV*	*V*	*VI*

Cocke-Algorithmus - Schritt 5 -

DET eintragen mit :
LG = IV RG = IV
LK = 0 RK = 0

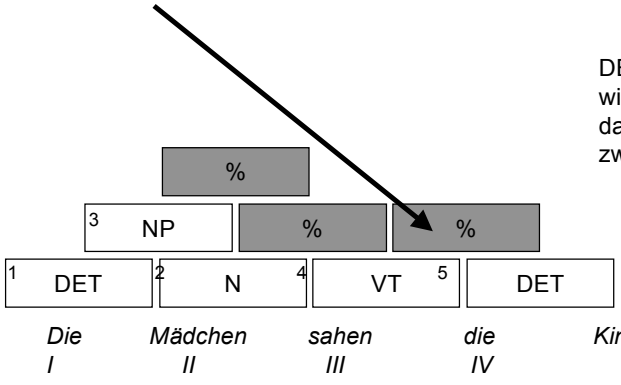
(L1) "Det ∅ die, manche, einige"



Cocke-Algorithmus - Schritt 5a -

Weiterer Schritt mit VT und DET nicht möglich:
5a Kat = % LG =IV RG = V LK = 0 RK = 0 R =%

DET mit N oder DET oder NP
wird nicht geprüft,
da verletzte Adjazenzregel
zwischen RG_i und LG_5

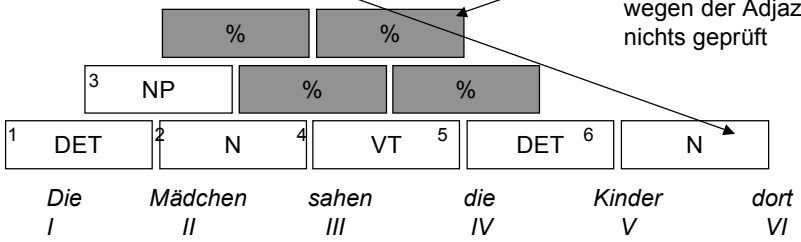


Cocke-Algorithmus - Schritt 6 -

N eintragen mit :
LG = V RG = V
LK = 0 RK = 0

(L3) "N ∅ Mädchen, Kinder"

Hier ist trivialerweise keine Regel vorhanden, sonst wird wegen der Adjazenzregel nichts geprüft

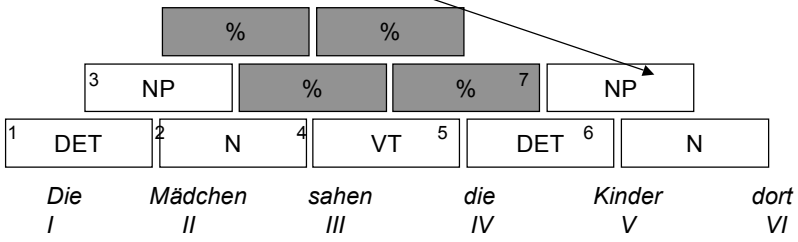


Die *Mädchen* *sahen* *die* *Kinder* *dort*
I *II* *III* *IV* *V* *VI*

Cocke-Algorithmus - Schritt 7 -

NP eintragen mit :
LG = IV RG = V
LK = 5 RK = 6

(G10) "NP ∅ DET N"



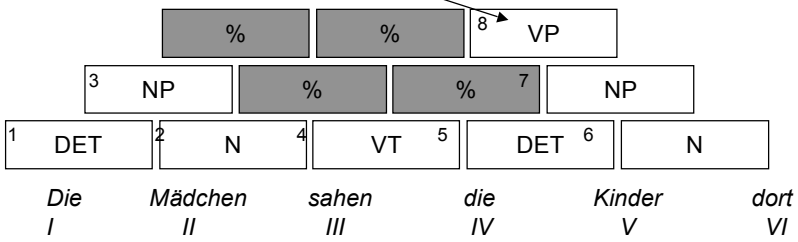
Die *Mädchen* *sahen* *die* *Kinder* *dort*
I *II* *III* *IV* *V* *VI*

Cocke-Algorithmus - Schritt 8 -

VP eintragen mit :
LG = III RG = V
LK = 4 RK = 7

(G9) "VP ∅ VT NP"

Weitere Schritte verstoßen gegen die Adjazenzregel



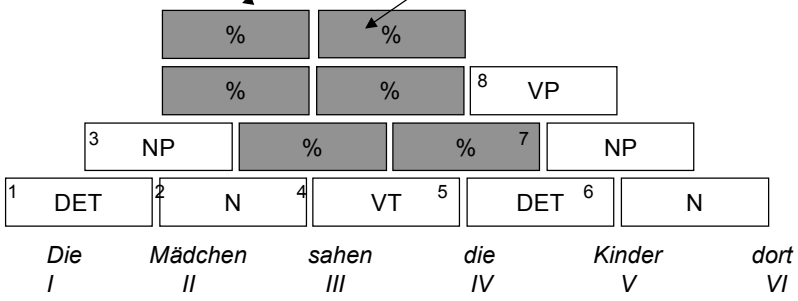
Die *Mädchen* *sahen* *die* *Kinder* *dort*
I *II* *III* *IV* *V* *VI*

06.11.2002

57

Cocke-Algorithmus - Schritt 8a und 8b -

VP mit N und DET
entsprechen keinen Regeln



Die *Mädchen* *sahen* *die* *Kinder* *dort*
I *II* *III* *IV* *V* *VI*

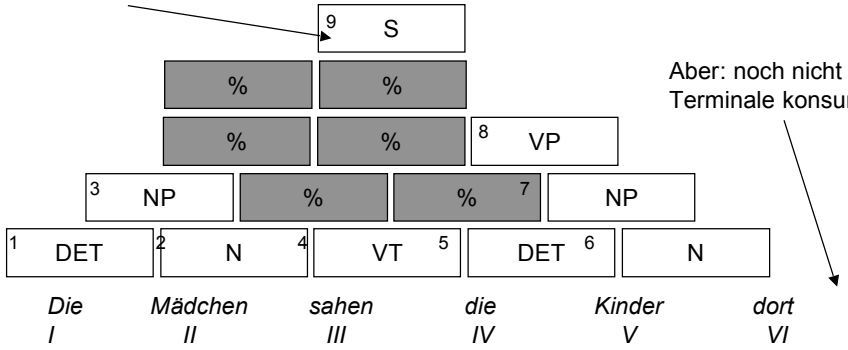
06.11.2002

58

Cocke-Algorithmus - Schritt 9 -

S eintragen mit :
LG = I RG = V
LK = 3 RK = 8

(G3) "S ∅ NP VP"

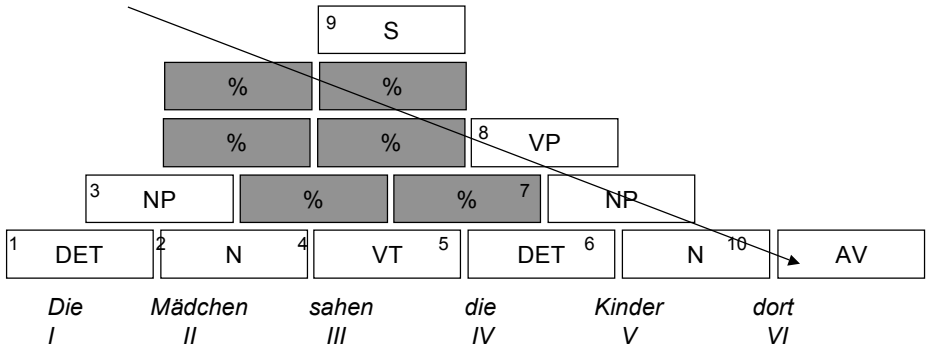


Aber: noch nicht alle Terminale konsumiert!

Cocke-Algorithmus - Schritt 10 -

AV eintragen mit :
LG = VI RG = VI
LK = 0 RK = 0

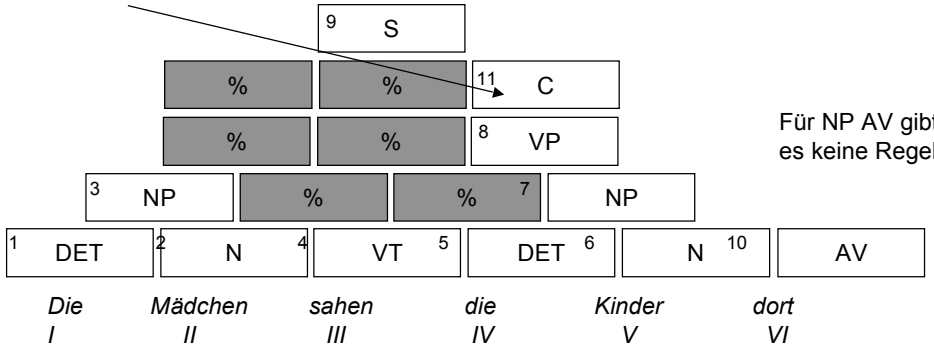
(L6) "AV ∅ dort, hier"



Cocke-Algorithmus - Schritt 11-

C eintragen mit :
LG = III RG = VI
LK = 8 RK = 10

(G8) "C ∅ VP AV"

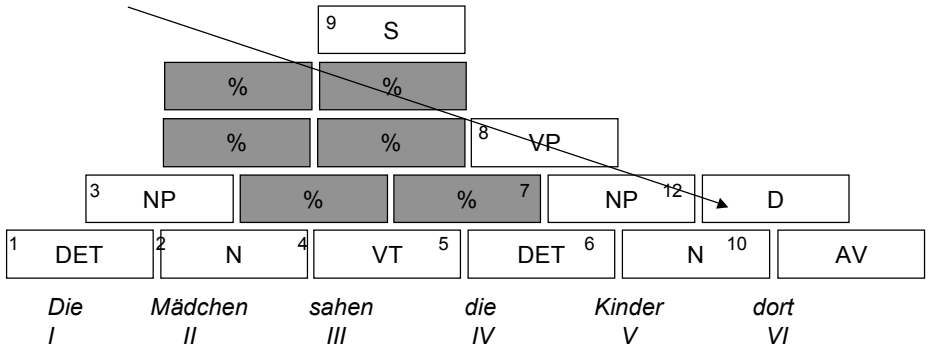


Für NP AV gibt es keine Regel

Cocke-Algorithmus - Schritt 12-

D eintragen mit :
LG = VI RG = VI
LK = 0 RK = 0

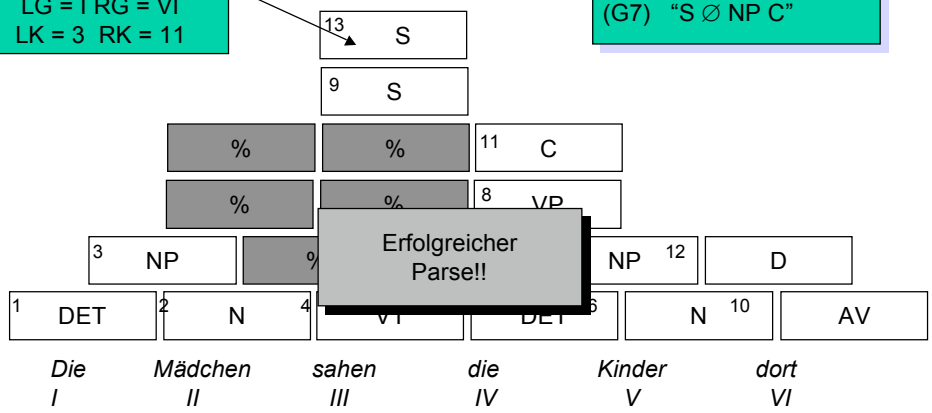
(G12) "D ∅ N AV"



Cocke-Algorithmus - Schritt 13 -

S eintragen mit :
LG = I RG = VI
LK = 3 RK = 11

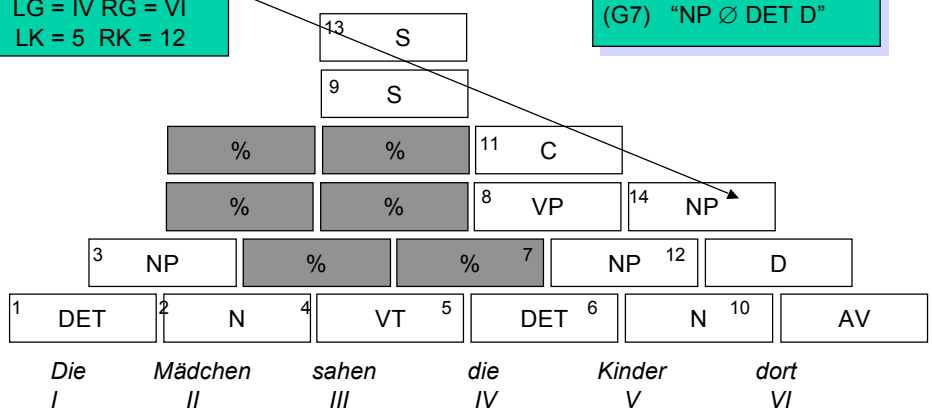
(G7) “S ∅ NP C”



Cocke-Algorithmus - Schritt 14 -

NP eintragen mit :
LG = IV RG = VI
LK = 5 RK = 12

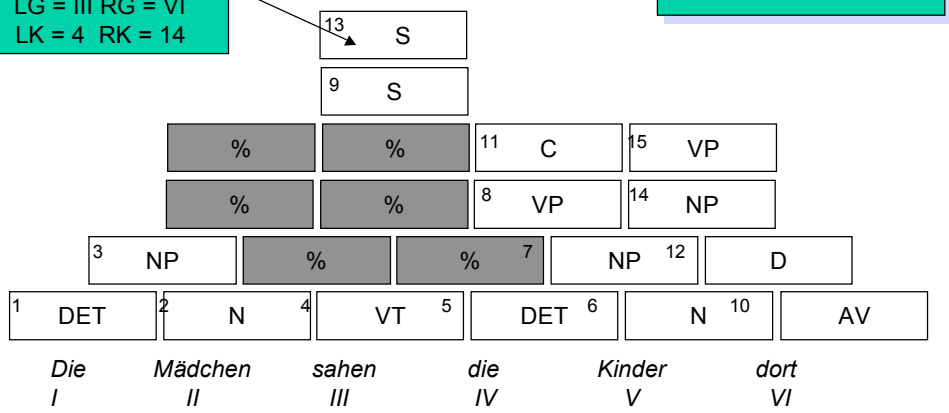
(G7) “NP ∅ DET D”



Cocke-Algorithmus - Schritt 15 -

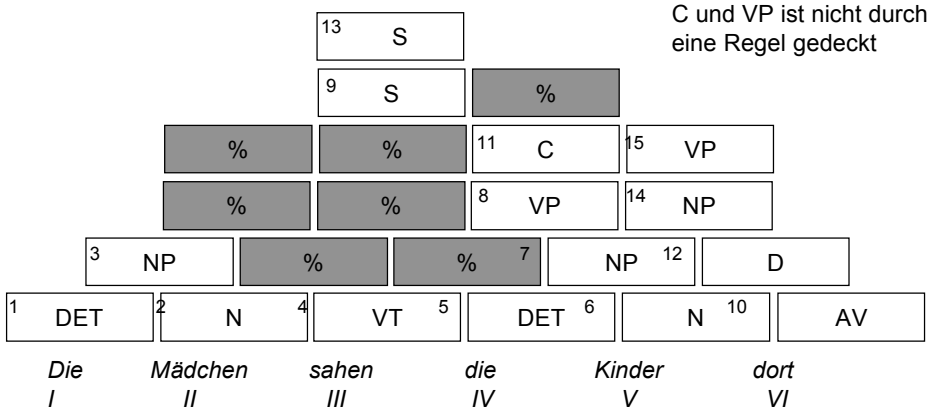
VP eintragen mit :
LG = III RG = VI
LK = 4 RK = 14

(G9) "VP ∅ VT NP"



Cocke-Algorithmus - Schritt 15a -

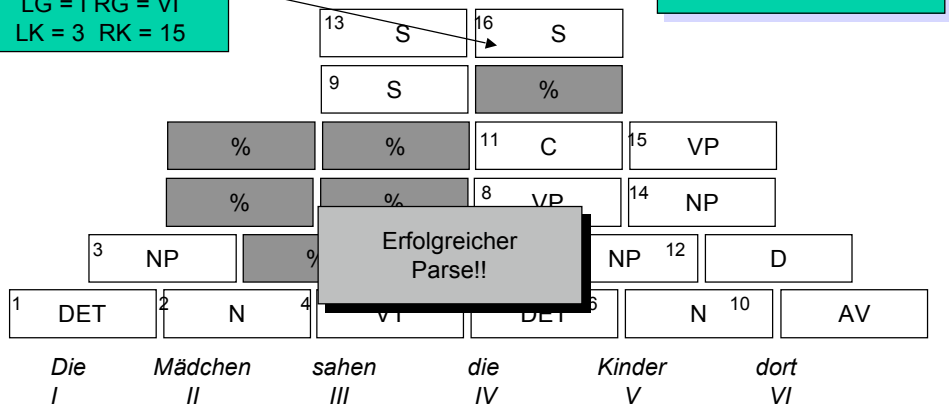
C und VP ist nicht durch eine Regel gedeckt



Cocke-Algorithmus - Schritt 16 -

S eintragen mit :
LG = I RG = VI
LK = 3 RK = 15

(G3) "S ∅ NP VP"



Die Parsingtabelle

Nr	Kat.	LG	RG	LK	RK	Regel
1	DET	I	I	0	O	L1
2	N	II	II	0	0	L3
3	NP	I	II	1	2	G10
4	VT	III	III	0	0	L5
5	DET	IV	IV	0	0	L1
6	N	V	V	0	0	L3
7	NP	IV	V	5	6	G10
8	VP	III	V	4	7	G9
9	S	I	V	3	8	G3
10	AV	VI	VI	0	0	L6
11	C	III	VI	8	10	G8
12	D	V	VI	6	10	G12
13	**S**	I	VI	3	11	G7
14	NP	IV	VI	5	12	G11
15	VP	III	VI	4	14	G9
16	**S**	I	VI	3	15	G3

(G1) S ∅ S A
(G2) A ∅ KON S
(G3) S ∅ NP VP
(G4) S ∅ NP VI
(G5) S ∅ NP B
(G6) B ∅ VI AV
(G7) S ∅ NP C
(G8 C ∅ VP AV
(G9) VP ∅ VT NP
(G10) NP ∅ DET N
(G11) NP ∅ DET D
(G12) D ∅ N AV

(L1) DET ∅ die, manche, einige,
(L2) KON ∅ und, oder, aber,
(L3) N ∅ Mädchen, Kinder,
(L4) VI ∅ strahlten, stanken,
(L5) VT ∅ sahen, ehrten,
(L6) AV ∅ dort, hier

Parsing Konstituentenstrukturgrammatik

- **top -down** : begins with the start symbol and searches through different paths to rewrite the symbols until the input sentence is generated or until all possibilities have been explored

- **bottom-up** : starts with the words in the sentence and uses the rewrite rules backward to reduce the sequence of symbols until it consists solely of S.

Chart-Parsing - 1 -

Von Martin Kay 1973 vorgeschlagen. Von Kaplan und Kay als General Syntactic Processor implementiert.

Eine Chart ist ein gerichteter bewerteter Graph, der im Verlauf des Parsing-Prozesses um immer weitere Kanten erweitert wird.

Zunächst wird die Chart initialisiert:

- Je einen Knoten an den Satzgrenzen und zwischen je zwei Wörtern der Eingabekette einsetzen, sowie
- Kanten einsetzen, die zwei Knoten links und rechts von einem terminalen Symbol verbinden und mit dem jeweils überspannten Wort markieren
- Kanten einsetzen, die zwei Knoten links und rechts von einem terminalen Symbol verbinden und mit der Wortklassenbezeichnung des durch sie überspannten Worts markieren. Können einem Wort mehrere alternative Kategorien zugeordnet werden, je eigene Kanten aufspannen.

Chart-Parsing - 2 -

Die Aufgabe des Parsers besteht darin, für jede im Verlauf der Analyse der Eingabe gefundene Konstituente eine mindestens zwei Knoten überspannende Kante in die Chart einzuführen.

Die Analyse gilt als abgeschlossen, wenn eine die ganze Chart überspannende Kante eingerichtet ist.

Die Chart enthält dann alle möglichen syntaktischen Zerlegungen der Eingabe.

Bei unvollständigen Eingaben wird zwar keine den ganzen Satz überspannende Kante gefunden, aber die Chart enthält meist einige der Grammatik entsprechenden Konstituenten als Fragmente, die dann z.B. von einer Komponente zur Rekonstruktion von Ellipsen weiterverarbeitet werden können.

Chart-Parsing - 3 -

Man unterscheidet zwei Typen von Kanten in einem Chart-Parser:

- Aktive Kanten zur Darstellung noch unvollständiger Konstitute
- Inaktive Kanten zur Darstellung bereits vollständiger Konstitute

Wenn das Ende einer aktiven Kante A und der Anfang einer inaktiven Kante I an einem Knoten zusammenstoßen, wird folgendes Verfahren zur Generierung einer neuen Kante N ausgelöst:

- der Anfang von N ist der von A,
- das Ende von N ist das von I,
- die Kategorie von N ist die von A,
- der Inhalt von N hängt vom Inhalt von A und der Kategorie sowie dem Inhalt von I ab,
- N ist entweder Aktiv oder inaktiv, je nachdem, ob A damit vervollständigt wird oder nicht.

Nach Wahlster

Chart-Parsing - Beispiel -

G1: S → NP VP	G6: NP2 → NP2 PP
G2: NP → Determiner NP2	G7: PP → Preposition
G3: NP → NP2	G8: VP → Verb
G4: NP2 → Noun	G9: VP → Verb NP
G5: NP2 → Adjective NP2	G10: VP → VP PP

"The rabbit with a saw nibbled on an orange"

Schritt: 1
Eintrag: $_1$S ∅ $_1$NP VP (aus Initialisierung)
Offene Kanten: $_1$NP ∅ $_1$Det NP2, $_1$NP ∅ NP2
Komplette Kanten: $_1$Det$_2$, ...

Schritt: 2
Eintrag: $_1$NP ∅ $_1$Det NP2 (aus Schritt 1)
Neue offene Kante: $_1$NP ∅ Det $_2$NP2
Komplette Kanten: $_1$Det$_2$, ...

Chart-Parsing - Beispiel - 2-

Schritt: 3
Eintrag: $_1$NP ∅ $_1$NP2 (aus Schritt 1)
Neue offene Kante: $_1$NP2 ∅ $_1$Noun, $_1$NP2 ∅ $_1$ADJ NP2,
$_1$NP2 ∅ $_1$NP2 PP
Komplette Kanten: $_1$Det$_2$, ...

Schritt: 4
Eintrag: Det $_2$NP2 (aus Schritt 2)
Neue offene Kante: $_2$NP2 ∅ $_2$Noun, $_2$NP2 ∅ $_2$ADJ NP2,
$_2$NP2 ∅ $_2$NP2 PP
Komplette Kanten: $_1$Det$_2$, ...

Schritt: 5
Eintrag: 1NP2 ∅ 1Noun (aus Schritt 3)
Neue offene Kante: keine, da der remainder weder ein Nonterminal ist, noch mit der Eingabe matcht.
Komplette Kanten: 1Det2, ...

Chart-Parsing - Beispiel - 3 -

Schritt: 6

Eintrag: $_1$NP2 ∅ $_1$Adj NP2 (aus Schritt 3)

Neue offene Kante: keine, da der remainder weder ein Nonterminal ist, noch mit der Eingabe matcht.

Komplette Kanten: $_1$Det$_2$, ...

Schritt:7

Eintrag: $_1$NP2 ∅ $_1$NP2 PP (aus Schritt 3)

Neue offene Kante: keine, da bereits ein $_1$NP2 in der Chart ist. Zur Blockierung von Links-Rekursion

Komplette Kanten: $_1$Det$_2$, ...

Schritt: 8

Eintrag: $_2$NP2 ∅ $_2$Noun (aus Schritt 4)

Neue offene Kante: $_2$NP2 ∅Noun$_3$ (nicht in der Graphik)

Komplette Kanten: $_1$Det$_2$, ...

Chart-Parsing - Beispiel - 4 -

Schritt: 9

Eintrag: $_2$NP2 ∅ $_2$Adj NP2 (aus Schritt 4)

Neue offene Kante: keine, da der remainder weder ein Nonterminal ist, noch mit der Eingabe matcht.

Komplette Kanten: $_1$Det$_2$, ...

Schritt: 10

Eintrag: 2NP2 ∅ 2NP2 PP (aus Schritt 4)

Neue offene Kante: keine, da bereits ein 1NP2 in der Chart ist. Zur Blockierung von Links-Rekursion

Komplette Kanten: 1Det2, ...

Chart-Parsing - Beispiel - 5 -

Schritt: 11
Eintrag: $_2$NP2 ∅ Noun$_3$ (aus Schritt 8)
Neue offene Kante: $_2$NP2 ∅ NP2 $_3$ PP, $_1$NP ∅ Det NP2$_3$
Komplette Kanten: $_1$Det$_2$, $_2$NP$_3$, ...

Schritt: 12
Eintrag: $_2$NP2 ∅ NP2$_3$ PP (aus Schritt 11)
Neue offene Kante: $_3$PP ∅ $_3$Prep NP
Komplette Kanten: $_1$Det$_2$, $_2$NP$_3$, ...

!

Schritt: 13
Eintrag: $_1$NP ∅ Det $_2$ NP2$_3$ (aus Schritt 11)
Neue offene Kante: $_1$S ∅ NP$_3$ VP
Komplette Kanten: $_1$NP$_3$, $_3$Prep$_4$, ...

Chart-Parsing - Beispiel - 6 -

Schritt: 14
Eintrag: $_3$PP ∅ $_3$Prep NP (aus Schritt 12)
Neue offene Kante: $_3$PP∅ Prep $_4$ NP
Komplette Kanten: $_1$NP$_3$, $_3$Prep$_4$, ...

Aufbau der Chart - Beispiel -

"The rabbit with a saw nibbled on an orange"

S ($_1$NPVP)

Die Initialisierung trägt die Zahlen zwischen den Terminalen ein und schreibt die erste S-Regel in die Chart: $_1$S ∅ $_1$NP VP
Offene Kanten: $_1$NP ∅ $_1$DET NP2, $_1$NP ∅$_1$ NP2

S ∅ NP VP
NP ∅ DET NP2
NP ∅ NP2
NP2 ∅ NOUN
NP2 ∅ ADJ NP2
NP2 ∅ NP2 PP
PP ∅ PREP NP
VP ∅ VERB
VP ∅ VERB NP
VP ∅ VP PP

Aufbau - 2 -

In die Chart wird als 1. mögliche Fortsetzung von S ($_1$NP VP) (= die 1. Offene Kante) eingetragen:
$_1$NP ∅ $_1$DET NP2
das paßt am linken Rand mit der Eingabekette.
Offene Kanten: ($_1$NP ∅$_1$NP2) + $_1$NP ∅ DET $_2$NP2
Komplette Kanten: $_1$DET$_2$, $_2$NOUN$_3$ usw. .....

S ($_1$NP VP)

$_1$NP ∅ $_1$DET NP2

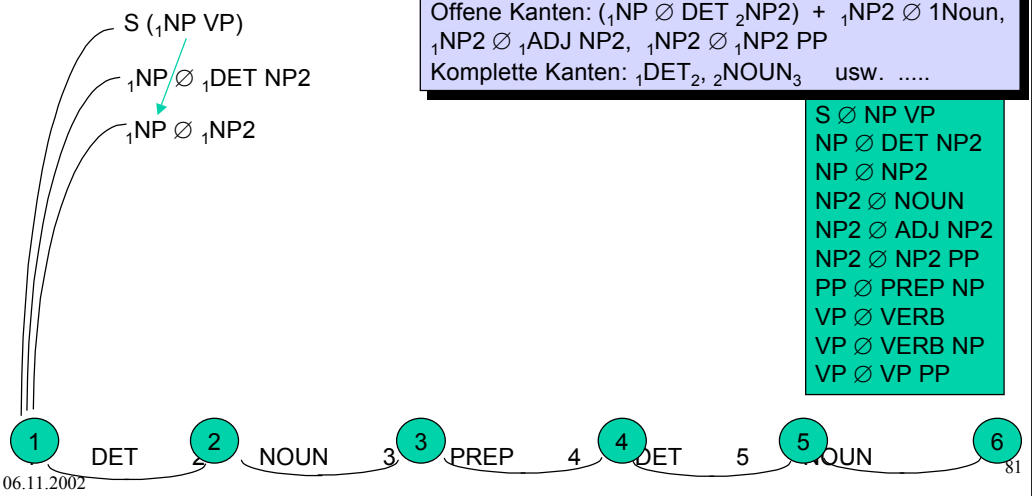
☺

In Klammern immer die noch nicht bearbeiteen offen Kanten früherer Schritte

S ∅ NP VP
NP ∅ DET NP2
NP ∅ NP2
NP2 ∅ NOUN
NP2 ∅ ADJ NP2
NP2 ∅ NP2 PP
PP ∅ PREP NP
VP ∅ VERB
VP ∅ VERB NP
VP ∅ VP PP

Aufbau - 3 -

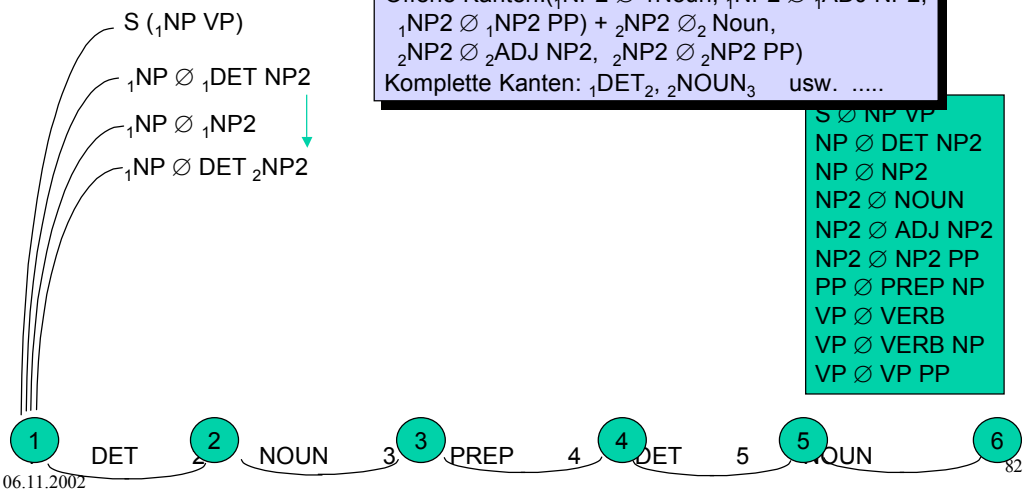
In die Chart wird als 2. mögliche Fortsetzung von
S ($_1$NP VP) (= älteste offene Kante) eingetragen:
$_1$NP ∅ $_1$NP2
Offene Kanten: ($_1$NP ∅ DET $_2$NP2) + $_1$NP2 ∅ 1Noun,
$_1$NP2 ∅ $_1$ADJ NP2, $_1$NP2 ∅ $_1$NP2 PP
Komplette Kanten: $_1$DET$_2$, $_2$NOUN$_3$ usw. .....



S ∅ NP VP
NP ∅ DET NP2
NP ∅ NP2
NP2 ∅ NOUN
NP2 ∅ ADJ NP2
NP2 ∅ NP2 PP
PP ∅ PREP NP
VP ∅ VERB
VP ∅ VERB NP
VP ∅ VP PP

Aufbau - 4 -

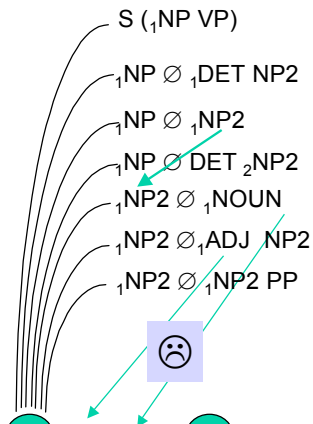
In die Chart wird als Folge von $_1$NP ∅ $_1$DET NP2
eingetragen: $_1$NP ∅ DET $_2$NP2
Offene Kanten:($_1$NP2 ∅ 1Noun, $_1$NP2 ∅ $_1$ADJ NP2,
$_1$NP2 ∅ $_1$NP2 PP) + $_2$NP2 ∅$_2$ Noun,
$_2$NP2 ∅ $_2$ADJ NP2, $_2$NP2 ∅ $_2$NP2 PP)
Komplette Kanten: $_1$DET$_2$, $_2$NOUN$_3$ usw. .....



S ∅ NP VP
NP ∅ DET NP2
NP ∅ NP2
NP2 ∅ NOUN
NP2 ∅ ADJ NP2
NP2 ∅ NP2 PP
PP ∅ PREP NP
VP ∅ VERB
VP ∅ VERB NP
VP ∅ VP PP

Aufbau - 5 -

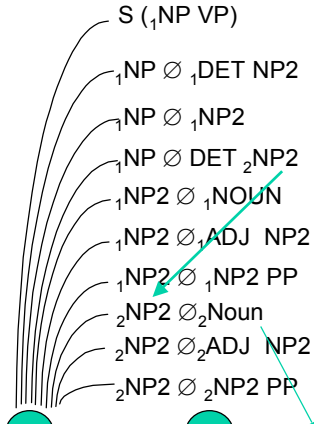
In die Chart wird als Auflösung von
$_1$NP ∅ $_1$NP2 eingetragen: $_1$NP2 ∅$_1$Noun,
danach $_1$NP2 ∅$_1$ADJ NP2 und $_1$NP2 ∅ $_1$NP2 PP
Neue offene Kanten: ($_2$NP2 ∅$_2$ Noun,
$_2$NP2 ∅ $_2$ADJ NP2, $_2$NP2 ∅ $_2$NP2 PP) keine neuen, weil Terminale nicht matchen und 1NP2 schon in der Chart ist.
Komplette Kanten: $_1$DET$_2$, $_2$NOUN$_3$ usw. .....



NP2
NP ∅ NP2
NP2 ∅ NOUN
NP2 ∅ ADJ NP2
NP2 ∅ NP2 PP
PP ∅ PREP NP
VP ∅ VERB
VP ∅ VERB NP
VP ∅ VP PP

Aufbau - 6 -

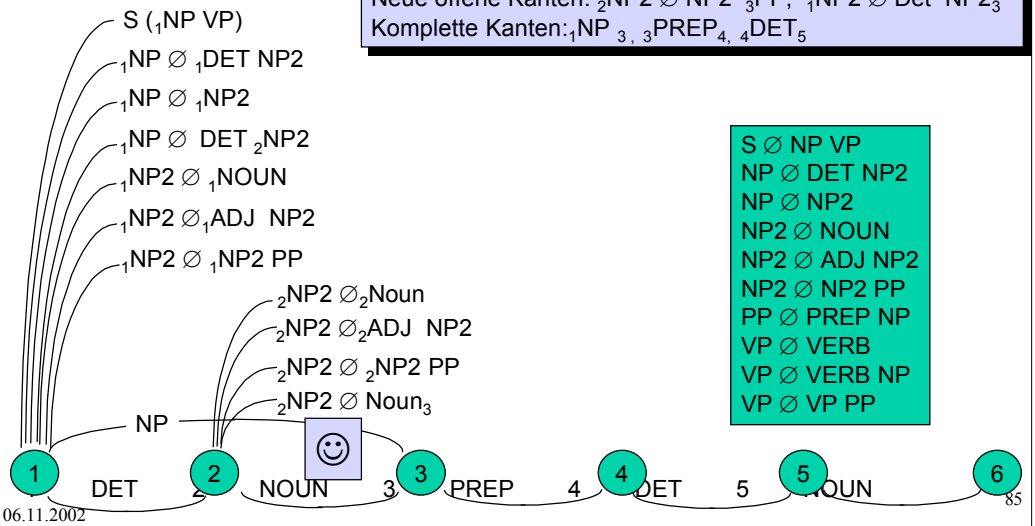
In die Chart wird als mögliche Auflösung von
$_1$NP ∅ DET $_2$NP2 eingetragen:
$_2$NP2 ∅$_2$ Noun, darauffolgend $_2$NP2 ∅ $_2$ADJ NP2, $_2$NP2 ∅ $_2$NP2 PP
Neue offene Kanten: $_2$NP2 ∅ Noun$_3$
Komplette Kanten: $_1$DET$_2$, $_2$NOUN$_3$ usw. .....



NP ∅ DET NP2
NP ∅ NP2
NP2 ∅ NOUN
NP2 ∅ ADJ NP2
NP2 ∅ NP2 PP
PP ∅ PREP NP
VP ∅ VERB
VP ∅ VERB NP
VP ∅ VP PP

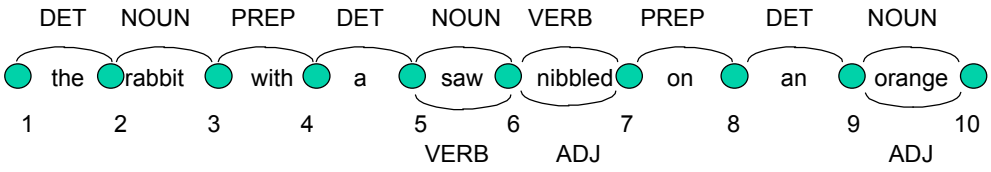
Aufbau - 7 -

In die Chart wird eingetragen: $_2$NP2 ∅ Noun$_3$
Neue offene Kanten: $_2$NP2 ∅ NP2 $_3$PP, $_1$NP2 ∅ Det NP2$_3$
Komplette Kanten:$_1$NP $_{3,}$ $_3$PREP$_{4,}$ $_4$DET$_5$



S ∅ NP VP
NP ∅ DET NP2
NP ∅ NP2
NP2 ∅ NOUN
NP2 ∅ ADJ NP2
NP2 ∅ NP2 PP
PP ∅ PREP NP
VP ∅ VERB
VP ∅ VERB NP
VP ∅ VP PP

Initialisierte Chart



Komplette Chart

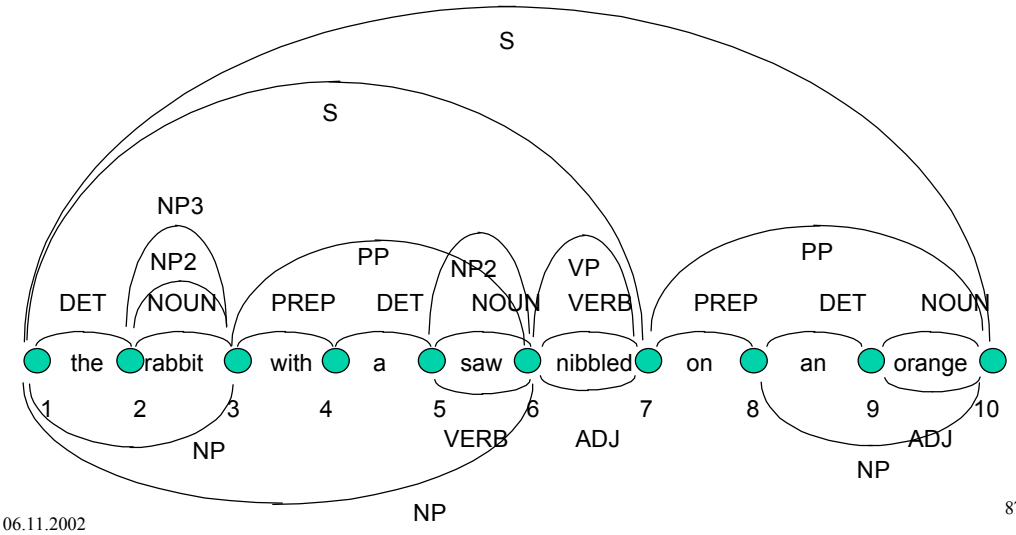


Chart-parsing vs. Traditionelle Methoden

- the same constituent is never used more than once
- in the worst case would build every possible constituent between every possible pair of positions
- more work in each step

Example: a sentence with 12 words :

brute force 10^{12} operations (1,000,000,000,000)

chart parser $1000*12^3$ (1,728,000)

on some examples, 500 000 times faster

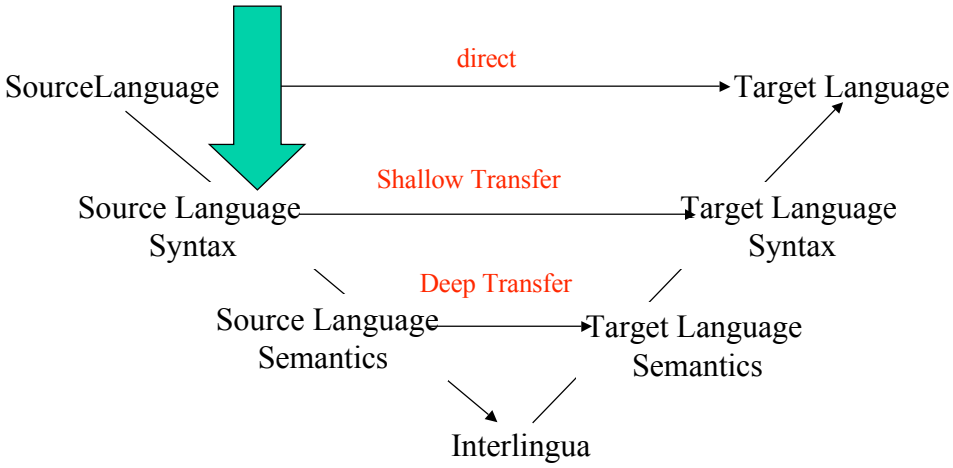
Parsing - problems

- partially incorrect phrases
  - Ex: An der Kupplung ist ein Kabelzug, der ist beschädigt.
    (looks like a relative clause, but wrong word order)
- ambiguities
  - Ex: Ich kann das rote Kabel am Vergaser nicht abziehen
- German separable verbs
  - Ex : Der Stecker, der an der Lampe ist, hängt lose herunter

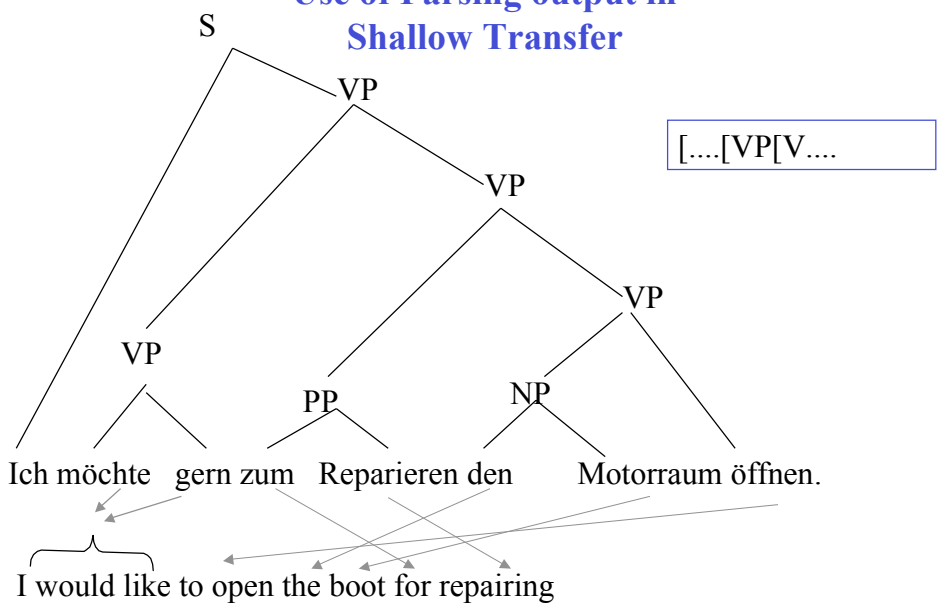
Gliederung

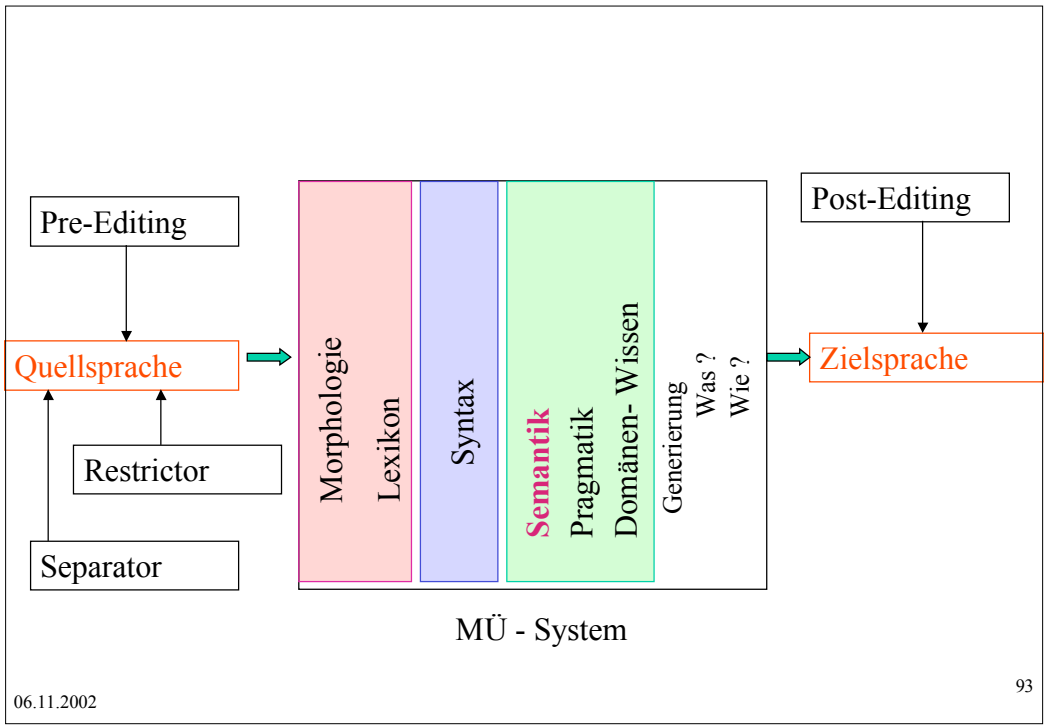
- Syntax
- Parsing
- Syntax und Parsing in MÜ Systeme

Parsing for Machine Translation



Use of Parsing output in Shallow Transfer




Pre-Editing
Quellsprache
Restrictor
Separator
Morphologie
Lexikon
Syntax
Semantik
Pragmatik
Domänen- Wissen
Generierung
Was ?
Wie ?
Post-Editing
Zielsprache
MÜ - System
06.11.2002
93