

# Syntax natürlicher Sprachen

## Vorlesung 9: Technische Aspekte und Parsing-Algorithmen

Martin Schmitt

Ludwig-Maximilians-Universität München

11.12.2018

# Arten von Parsing-Algorithmen

## Top-Down

- Recursive Descent
- LL (Left-to-right Leftmost (derivation))
- LL(k)
- L(\*)
- Earley

## Bottom-Up

- Recursive Ascent
- GLR (Generalized Left-to-right Rightmost (derivation))
- Shift-Reduce
- CYK

# Arten von Parsing-Algorithmen

## Top-Down

- Recursive Descent
- LL (Left-to-right Leftmost (derivation))
- LL(k)
- L(\*)
- Earley

## Bottom-Up

- Recursive Ascent
- GLR (Generalized Left-to-right Rightmost (derivation))
- Shift-Reduce
- CYK

# Themen der heutigen Vorlesung

- 1 Top-Down-Parsing: Recursive Descent
- 2 Bottom-Up-Parsing: Shift Reduce
- 3 Chart Parsing: Earley Algorithmus
- 4 Parsing mit Merkmalstrukturen
  - Modifizierter Earley Parser
  - Komplexität
- 5 Ausblick: Statistisches Parsing

# Nächstes Thema

- 1 Top-Down-Parsing: Recursive Descent
- 2 Bottom-Up-Parsing: Shift Reduce
- 3 Chart Parsing: Earley Algorithmus
- 4 Parsing mit Merkmalstrukturen
  - Modifizierter Earley Parser
  - Komplexität
- 5 Ausblick: Statistisches Parsing

# Recursive Descent Parser

## Top-Down-Parsing (dt. *Abwärtsparsen*)

Parsing-Strategie, bei der man von der höchsten Ebene eines Syntaxbaums (Startsymbol der Grammatik) ausgeht und sich mithilfe der Ersetzungsregeln (Produktionsregeln) einer Grammatik bis zu den Terminalen (Lexemen) vorarbeitet.

## Recursive Descent Parsing (dt. *rekursiver Abstieg*)

- Form von Top-Down-Parsing
- probiert jede anwendbare Regel aus
- benutzt *Backtracking* im Problemfall
- am intuitivsten „händisch“ zu programmieren
- kann je nach Grammatik zu exponentieller Laufzeit führen (oder sogar zu unendlich langer Laufzeit)

# Recursive Descent Parser: Beispiel

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

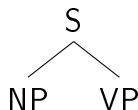
S

Chomsky kennt das Buch



# Recursive Descent Parser: Beispiel

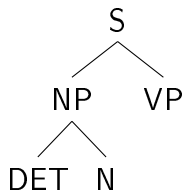
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

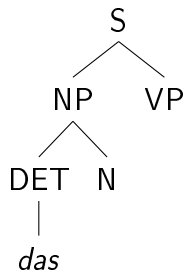
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

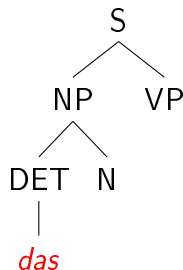
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

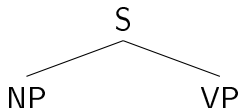


Chomsky kennt das Buch



# Recursive Descent Parser: Backtracking

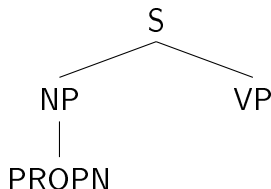
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

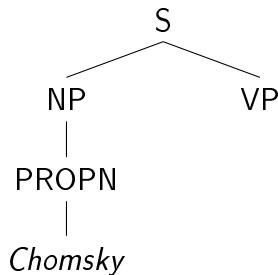
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

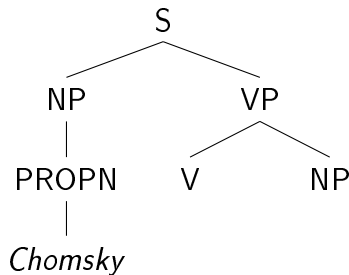
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

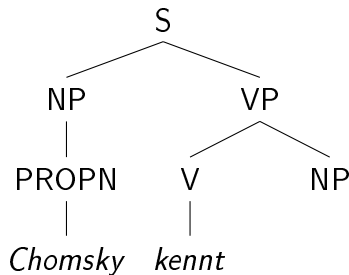


Chomsky kennt das Buch



# Recursive Descent Parser: Beispiel

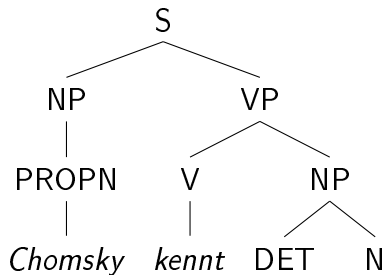
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

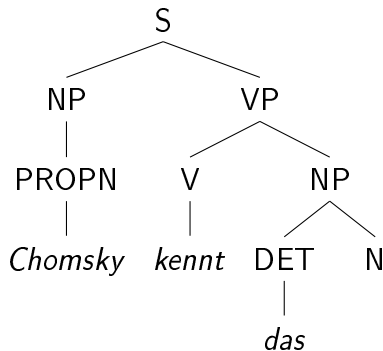
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

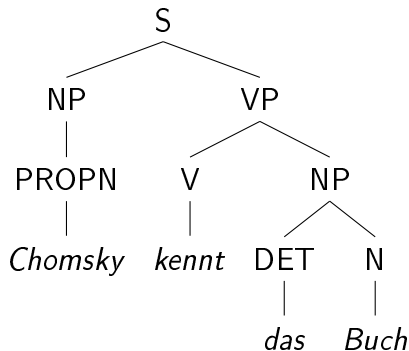
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  **$DET \rightarrow \text{das}$**
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

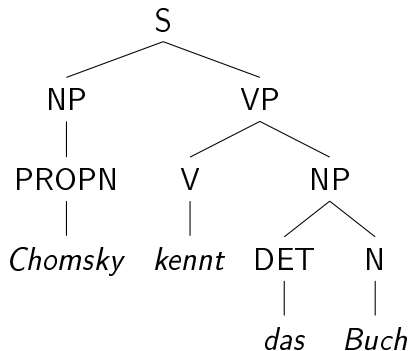
- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Recursive Descent Parser: Beispiel

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch



# Recursive Descent Parsing: Anmerkungen

## Probleme

- Es kann zu jeder Zeit für ein Nichtterminal viele verschiedene Ersetzungsregeln geben.
- Im schlimmsten Fall müssen alle diese Regeln ausprobiert werden (exponentieller Blow-up).
- Viele Teilstrukturen werden erzeugt, obwohl sie nie erfolgreich sein können.  
→ Bsp.: Eingabesatz enthält gar nicht die passenden Wörter.

## Gefahr der Endlosschleife

- *Links-rekursive* führen (bei naiver Ausführung) zu unendlicher Laufzeit!
- Beispiel: ADJP → ADJP ADJ

# Recursive Descent Parsing: Anmerkungen

## Probleme

- Es kann zu jeder Zeit für ein Nichtterminal viele verschiedene Ersetzungsregeln geben.
- Im schlimmsten Fall müssen alle diese Regeln ausprobiert werden (exponentieller Blow-up).
- Viele Teilstrukturen werden erzeugt, obwohl sie nie erfolgreich sein können.  
→ Bsp.: Eingabesatz enthält gar nicht die passenden Wörter.

## Gefahr der Endlosschleife

- *Links-rekursive* führen (bei naiver Ausführung) zu unendlicher Laufzeit!
- Beispiel: ADJP → ADJP ADJ

# Recursive Descent Parsing: Anmerkungen

## Probleme

- Es kann zu jeder Zeit für ein Nichtterminal viele verschiedene Ersetzungsregeln geben.
- Im schlimmsten Fall müssen alle diese Regeln ausprobiert werden (exponentieller Blow-up).
- Viele Teilstrukturen werden erzeugt, obwohl sie nie erfolgreich sein können.

→ Bsp.: Eingabesatz enthält gar nicht die passenden Wörter.

## Gefahr der Endlosschleife

- *Links-rekursive* führen (bei naiver Ausführung) zu unendlicher Laufzeit!
- Beispiel: ADJP → ADJP ADJ



# Recursive Descent Parsing: Anmerkungen

## Probleme

- Es kann zu jeder Zeit für ein Nichtterminal viele verschiedene Ersetzungsregeln geben.
- Im schlimmsten Fall müssen alle diese Regeln ausprobiert werden (exponentieller Blow-up).
- Viele Teilstrukturen werden erzeugt, obwohl sie nie erfolgreich sein können.
  - Bsp.: Eingabesatz enthält gar nicht die passenden Wörter.

## Gefahr der Endlosschleife

- *Links-rekursive* führen (bei naiver Ausführung) zu unendlicher Laufzeit!
- Beispiel: ADJP → ADJP ADJ

## Probleme

- Es kann zu jeder Zeit für ein Nichtterminal viele verschiedene Ersetzungsregeln geben.
- Im schlimmsten Fall müssen alle diese Regeln ausprobiert werden (exponentieller Blow-up).
- Viele Teilstrukturen werden erzeugt, obwohl sie nie erfolgreich sein können.
  - Bsp.: Eingabesatz enthält gar nicht die passenden Wörter.

## Gefahr der Endlosschleife

- *Links-rekursive* führen (bei naiver Ausführung) zu unendlicher Laufzeit!
- Beispiel: ADJP → ADJP ADJ

# Recursive Descent Parsing: Anmerkungen

## Probleme

- Es kann zu jeder Zeit für ein Nichtterminal viele verschiedene Ersetzungsregeln geben.
- Im schlimmsten Fall müssen alle diese Regeln ausprobiert werden (exponentieller Blow-up).
- Viele Teilstrukturen werden erzeugt, obwohl sie nie erfolgreich sein können.  
→ Bsp.: Eingabesatz enthält gar nicht die passenden Wörter.

## Gefahr der Endlosschleife

- *Links-rekursive* führen (bei naiver Ausführung) zu unendlicher Laufzeit!
- Beispiel: ADJP → ADJP ADJ

# Nächstes Thema

- 1 Top-Down-Parsing: Recursive Descent
- 2 Bottom-Up-Parsing: Shift Reduce
- 3 Chart Parsing: Earley Algorithmus
- 4 Parsing mit Merkmalstrukturen
  - Modifizierter Earley Parser
  - Komplexität
- 5 Ausblick: Statistisches Parsing

# Shift Reduce Parsing

## Bottom-Up-Parsing (dt. *Aufwärtsparsen*)

Parsing-Strategie, bei der man von den kleinsten vorgefundenen Einheiten (Token, Lexeme, Terminale) ausgeht und versucht, diese nach und nach zu größeren syntaktischen Strukturen zu verbinden, bis man beim Startsymbol der Grammatik angelangt ist.

## Shift Reduce Parsing (dt. *Verschieben – Zurückführen*)

- Form von Bottom-Up-Parsing (*datengeleitetes Parsing*)
- gebraucht die Datenstruktur *Stack* (dt. *Stapel*)
- **verschiebt** Token auf den Stapel, um sie auf Grammatikregeln **zurückzuführen**

# Shift Reduce Parser: Beispiel

Grammatik

Stapel

Ableitungsbaum

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

Grammatik

Stapel

Ableitungsbaum

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Shift!

Grammatik

Stapel

Ableitungsbaum

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Chomsky

*Chomsky*

Chomsky kennt das Buch



# Reduce!

Grammatik

Stapel

Ableitungsbaum

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

PROPN

PROPN

|  
*Chomsky*

Chomsky kennt das Buch

# Reduce!

Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Stapel

NP

Ableitungsbaum



Chomsky kennt das Buch

# Shift!

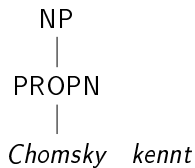
Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Stapel



Ableitungsbaum



Chomsky kennt das Buch

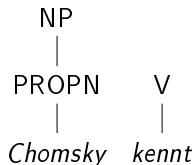
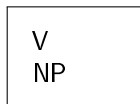
# Reduce!

Grammatik

Stapel

Ableitungsbaum

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$



Chomsky kennt das Buch

# Shift!

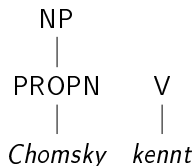
Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Stapel

das  
V  
NP

Ableitungsbaum



*das*

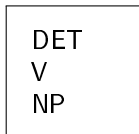
Chomsky kennt das Buch

# Reduce!

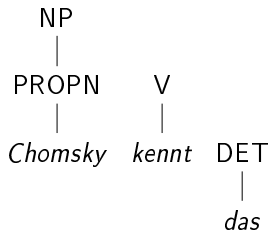
Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Stapel



Ableitungsbaum



Chomsky kennt das Buch

# Shift!

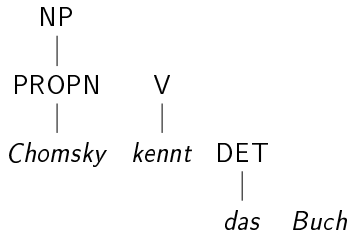
Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Stapel

Buch  
DET  
V  
NP

Ableitungsbaum



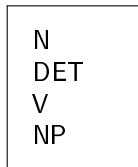
Chomsky kennt das Buch

# Reduce!

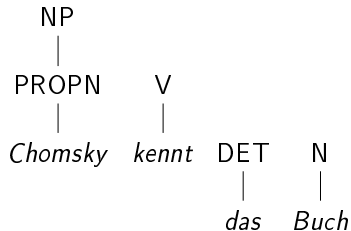
## Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

## Stapel



## Ableitungsbaum



Chomsky kennt das Buch

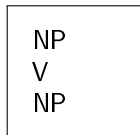


# Reduce!

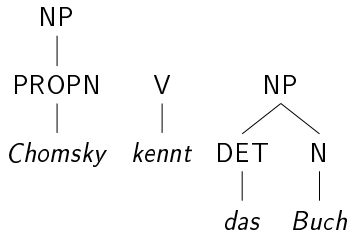
Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Stapel



Ableitungsbaum



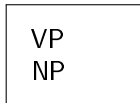
Chomsky kennt das Buch

# Reduce!

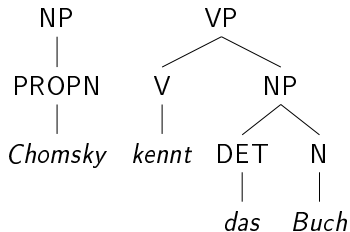
Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Stapel



Ableitungsbaum



Chomsky kennt das Buch

# Reduce!

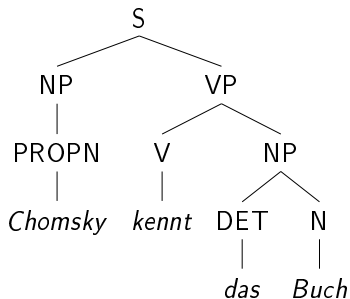
Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Stapel



Ableitungsbaum



Chomsky kennt das Buch



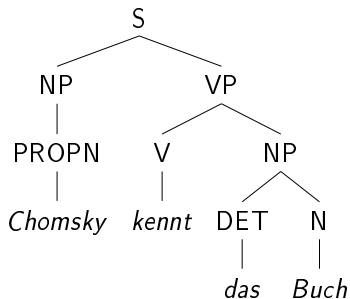
Grammatik

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Stapel



Ableitungsbaum



Chomsky kennt das Buch

# Shift Reduce Parsing: Anmerkungen

## Vorteile

- arbeitet abhängig von der Eingabe
- ist daher effizienter als ein Top-Down-Parser

## Probleme

- erzeugt auch Teilstrukturen, die zu keinem Ergebnis führen
  - benötigt also im Allgemeinen auch Backtracking
- potentiell exponentielle Laufzeit

# Shift Reduce Parsing: Anmerkungen

## Vorteile

- arbeitet abhängig von der Eingabe
- ist daher effizienter als ein Top-Down-Parser

## Probleme

- erzeugt auch Teilstrukturen, die zu keinem Ergebnis führen
  - benötigt also im Allgemeinen auch Backtracking
- potentiell exponentielle Laufzeit

# Shift Reduce Parsing: Anmerkungen

## Vorteile

- arbeitet abhängig von der Eingabe
- ist daher effizienter als ein Top-Down-Parser

## Probleme

- erzeugt auch Teilstrukturen, die zu keinem Ergebnis führen
  - benötigt also im Allgemeinen auch Backtracking
- potentiell exponentielle Laufzeit

# Shift Reduce Parsing: Anmerkungen

## Vorteile

- arbeitet abhängig von der Eingabe
- ist daher effizienter als ein Top-Down-Parser

## Probleme

- erzeugt auch Teilstrukturen, die zu keinem Ergebnis führen
  - benötigt also im Allgemeinen auch Backtracking
- potentiell exponentielle Laufzeit



# Shift Reduce Parsing: Anmerkungen

## Vorteile

- arbeitet abhängig von der Eingabe
- ist daher effizienter als ein Top-Down-Parser

## Probleme

- erzeugt auch Teilstrukturen, die zu keinem Ergebnis führen
  - benötigt also im Allgemeinen auch Backtracking
- potentiell exponentielle Laufzeit

# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
  - alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
  - geht besser mit POS Ambiguitäten um
  - baut Strukturen öfter als benötigt
  - verbringt viel Zeit mit unmöglichen Ableitungen
- startet die Analyse beim Beginn der Eingabe
  - alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
  - muss alle lexikalische Ambiguitäten berücksichtigen
  - baut jede Struktur nur einmal
  - verbringt viel Zeit mit unnötigen Strukturen

# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
  - alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
  - geht besser mit POS Ambiguitäten um
  - baut Strukturen öfter als benötigt
  - verbringt viel Zeit mit unmöglichen Ableitungen
- startet die Analyse beim Beginn der Eingabe
  - alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
  - muss alle lexikalische Ambiguitäten berücksichtigen
  - baut jede Struktur nur einmal
  - verbringt viel Zeit mit unnötigen Strukturen

# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
  - alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
  - geht besser mit POS Ambiguitäten um
  - baut Strukturen öfter als benötigt
  - verbringt viel Zeit mit unmöglichen Ableitungen
- startet die Analyse beim Beginn der Eingabe
  - alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
  - muss alle lexikalische Ambiguitäten berücksichtigen
  - baut jede Struktur nur einmal
  - verbringt viel Zeit mit unnötigen Strukturen

# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
  - alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
  - geht besser mit POS Ambiguitäten um
  - baut Strukturen öfter als benötigt
  - verbringt viel Zeit mit unmöglichen Ableitungen
- startet die Analyse beim Beginn der Eingabe
  - alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
  - muss alle lexikalische Ambiguitäten berücksichtigen
  - baut jede Struktur nur einmal
  - verbringt viel Zeit mit unnötigen Strukturen

# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
- alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
- geht besser mit POS Ambiguitäten um
- baut Strukturen öfter als benötigt
- verbringt viel Zeit mit unmöglichen Ableitungen

- startet die Analyse beim Beginn der Eingabe
- alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
- muss alle lexikalische Ambiguitäten berücksichtigen
- baut jede Struktur nur einmal
- verbringt viel Zeit mit unnötigen Strukturen

# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
  - alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
  - geht besser mit POS Ambiguitäten um
  - baut Strukturen öfter als benötigt
  - verbringt viel Zeit mit unmöglichen Ableitungen
- startet die Analyse beim Beginn der Eingabe
  - alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
  - muss alle lexikalische Ambiguitäten berücksichtigen
  - baut jede Struktur nur einmal
  - verbringt viel Zeit mit unnötigen Strukturen

# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
  - alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
  - geht besser mit POS Ambiguitäten um
  - baut Strukturen öfter als benötigt
  - verbringt viel Zeit mit unmöglichen Ableitungen
- startet die Analyse beim Beginn der Eingabe
  - alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
  - muss alle lexikalische Ambiguitäten berücksichtigen
  - baut jede Struktur nur einmal
  - verbringt viel Zeit mit unnötigen Strukturen



# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
  - alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
  - geht besser mit POS Ambiguitäten um
  - baut Strukturen öfter als benötigt
  - verbringt viel Zeit mit unmöglichen Ableitungen
- startet die Analyse beim Beginn der Eingabe
  - alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
  - muss alle lexikalische Ambiguitäten berücksichtigen
  - baut jede Struktur nur einmal
  - verbringt viel Zeit mit unnötigen Strukturen

# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
  - alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
  - geht besser mit POS Ambiguitäten um
  - baut Strukturen öfter als benötigt
  - verbringt viel Zeit mit unmöglichen Ableitungen
- startet die Analyse beim Beginn der Eingabe
  - alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
  - muss alle lexikalische Ambiguitäten berücksichtigen
  - baut jede Struktur nur einmal
  - verbringt viel Zeit mit unnötigen Strukturen

# Top-Down vs. Bottom-Up

- startet die Analyse beim Startsymbol
  - alterniert zwischen Regelanwendung (*Predict*) und Abgleich mit der Eingabe (*Scan*)
  - geht besser mit POS Ambiguitäten um
  - baut Strukturen öfter als benötigt
  - verbringt viel Zeit mit unmöglichen Ableitungen
- startet die Analyse beim Beginn der Eingabe
  - alterniert zwischen Einlesen der Eingabe (*Shift*) und „Rückwärtsanwendung“ der Regeln (*Reduce*)
  - muss alle lexikalische Ambiguitäten berücksichtigen
  - baut jede Struktur nur einmal
  - verbringt viel Zeit mit unnötigen Strukturen

# Nächstes Thema

- 1 Top-Down-Parsing: Recursive Descent
- 2 Bottom-Up-Parsing: Shift Reduce
- 3 Chart Parsing: Earley Algorithmus**
- 4 Parsing mit Merkmalstrukturen
  - Modifizierter Earley Parser
  - Komplexität
- 5 Ausblick: Statistisches Parsing

## Chart Parsing

- *Dynamische Programmierung* vermeidet doppelte Berechnungen.

## Earley Parsing

## Chart Parsing

- *Dynamische Programmierung* vermeidet doppelte Berechnungen.
- Zwischenergebnisse werden in Datenstruktur (*Chart*) gespeichert

## Earley Parsing

## Chart Parsing

- *Dynamische Programmierung* vermeidet doppelte Berechnungen.
- Zwischenergebnisse werden in Datenstruktur (*Chart*) gespeichert

## Earley Parsing

- Top-Down-Parser (ohne Backtracking)

## Chart Parsing

- *Dynamische Programmierung* vermeidet doppelte Berechnungen.
- Zwischenergebnisse werden in Datenstruktur (*Chart*) gespeichert

## Earley Parsing

- Top-Down-Parser (ohne Backtracking)
- Algorithmus kann eigentlich nur Grammatikalität entscheiden.



## Chart Parsing

- *Dynamische Programmierung* vermeidet doppelte Berechnungen.
- Zwischenergebnisse werden in Datenstruktur (*Chart*) gespeichert

## Earley Parsing

- Top-Down-Parser (ohne Backtracking)
  - Algorithmus kann eigentlich nur Grammatikalität entscheiden.
- Zur Baumerstellung müssen zusätzliche Verweise gespeichert werden.

## Chart Parsing

- *Dynamische Programmierung* vermeidet doppelte Berechnungen.
- Zwischenergebnisse werden in Datenstruktur (*Chart*) gespeichert

## Earley Parsing

- Top-Down-Parser (ohne Backtracking)
  - Algorithmus kann eigentlich nur Grammatikalität entscheiden.
- Zur Baumerstellung müssen zusätzliche Verweise gespeichert werden.
- Komplexität:  $\mathcal{O}(n^3)$

## Chart Parsing

- *Dynamische Programmierung* vermeidet doppelte Berechnungen.
- Zwischenergebnisse werden in Datenstruktur (*Chart*) gespeichert

## Earley Parsing

- Top-Down-Parser (ohne Backtracking)
  - Algorithmus kann eigentlich nur Grammatikalität entscheiden.
- Zur Baumerstellung müssen zusätzliche Verweise gespeichert werden.
- Komplexität:  $\mathcal{O}(n^3)$
  - funktioniert nur mit  $\varepsilon$ -freien Grammatiken!

## $\varepsilon$ -Regel

- Regel der Form  $A \rightarrow \varepsilon$  (Nichtterminal  $A$  wird gelöscht)
- Im n1tk-Format so gar nicht zu formulieren, aber theoretisch erlaubt (z. B. für optionale Elemente)

## Eliminierungsalgorithmus

- 1 Wähle ein Nichtterminal  $A$  mit einer  $\varepsilon$ -Regel
- 2 Entferne die  $\varepsilon$ -Regel
- 3 Für jede Regel  $p$  mit  $A$  auf der rechten Seite:  
dupliziere die Regel für jede mögliche Kombination mit/ohne  $A$
- 4 Falls es immer noch  $\varepsilon$ -Regeln gibt, gehe zurück zu Schritt 1.

## $\varepsilon$ -Regel

- Regel der Form  $A \rightarrow \varepsilon$  (Nichtterminal  $A$  wird gelöscht)
- Im nltk-Format so gar nicht zu formulieren, aber theoretisch erlaubt (z. B. für optionale Elemente)

## Eliminierungsalgorithmus

- 1 Wähle ein Nichtterminal  $A$  mit einer  $\varepsilon$ -Regel
- 2 Entferne die  $\varepsilon$ -Regel
- 3 Für jede Regel  $p$  mit  $A$  auf der rechten Seite:  
dupliziere die Regel für jede mögliche Kombination mit/ohne  $A$   
(2 „Anzahl der Vorkommen von  $A$  in  $p$ “ neue Regeln)
- 4 Falls es immer noch  $\varepsilon$ -Regeln gibt, gehe zurück zu Schritt 1.

## Beispiel (Leeres Subjekt bei Imperativ)

- 1  $S \rightarrow NP \ VP$
- 2  $NP \rightarrow \varepsilon$
- 3  $NP \rightarrow DET \ N$
- 4  $VP \rightarrow V$
- 5  $V \rightarrow \text{"schlaf"} \mid \text{"schläft"}$
- 6  $DET \rightarrow \text{"der"}$
- 7  $N \rightarrow \text{"Hund"}$

Zur Vermeidung von Übergenerierung fehlen noch entsprechende Bedingungen! (s. Hausaufgabe)

## Beispiel (nach Eliminierung)

- 1 S  $\rightarrow$  NP VP
- 2 S  $\rightarrow$  VP
- 3 NP  $\rightarrow$  DET N
- 4 VP  $\rightarrow$  V
- 5 V  $\rightarrow$  "schlaf" | "schläft"
- 6 DET  $\rightarrow$  "der"
- 7 N  $\rightarrow$  "Hund"

Zur Vermeidung von Übergenerierung fehlen noch entsprechende Bedingungen! (s. Hausaufgabe)

# Earley Algorithmus I

## Gegeben

Eingabesequenz  $s = s_1, \dots, s_n$ ; Grammatik  $G = (T, N, P, S)$

## Datenstrukturen

- Position := Tokengrenze  
(z. B. zwischen  $s_1$  und  $s_2$  etc.)
- Zu jeder Pos. Menge  $Q$  von *Zuständen*
- Zustand :=  $(X \rightarrow \alpha \cdot \beta, i)$   
bestehend aus
  - der aktuellen Produktionsregel  $X \rightarrow \alpha\beta \in P$ ,
  - der aktuellen Position in dieser Regel (der Punkt  $\cdot$ ),
  - der Ursprungsposition  $i$  in der Eingabe, an der das Abgleichen dieser Regel begann.



## Operationen

**P** Prediction (dt. *Voraussage*)

falls  $(A \rightarrow \dots \cdot B \dots, j) \in Q_i$  mit  $B \in N$ , dann für jede Regel  $B \rightarrow \alpha \in P$  :

setze  $(B \rightarrow \cdot \alpha, i) \in Q_i$

**S** Scanning (dt. *Überprüfung*)

falls  $(A \rightarrow \dots \cdot a \dots, j) \in Q_i$  mit  $a \in T$  und  $a = s_{i+1}$ , dann

setze  $(A \rightarrow \dots a \cdot \dots, j) \in Q_{i+1}$

**C** Completion (dt. *Vervollständigung*)

falls  $(A \rightarrow \dots \cdot, j) \in Q_i$ , dann für alle Zustände

$(B \rightarrow \dots \cdot A \dots, k) \in Q_j$  :

setze  $(B \rightarrow \dots A \cdot \dots, k) \in Q_i$

## Algorithm

- 1 Initialisiere  $Q_0$  mit dem Zustand  $(S' \rightarrow \cdot S, 0)$  mit  $S'$  frisches nichtterminales Symbol
- 2 Führe je nach Situation eine der drei Operationen (P, S, C) aus, bis keine weiteren Zustände mehr hinzugefügt werden können.
- 3 Wiederhole Schritt 2 bis keine neuen Zustände mehr hinzugefügt werden können.
- 4 Akzeptiere die Eingabesequenz  $s$  genau dann, wenn  $(S' \rightarrow S \cdot, 0) \in Q_{|s|}$

$\implies$  Beispiel auf der nächsten Folie

## Grammatik:

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Pos. Zustände

---

$Q_0$

$(S' \rightarrow \cdot S, 0)$

---

$Q_1$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Pos.    Zustände

---

$Q_0$

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

---

$Q_1$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Pos. Zustände

---

$Q_0$

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

---

$Q_1$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Pos. Zustände

---

$Q_0$

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROPN, 0)$

---

$Q_1$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Pos. Zustände

---

$Q_0$

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROPN, 0)$

$(DET \rightarrow \cdot \text{das}, 0)$

---

$Q_1$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Pos. Zustände

---

$Q_0$

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROPN, 0)$

$(DET \rightarrow \cdot \text{das}, 0)$

$(PROPN \rightarrow \cdot \text{Chomsky}, 0)$

---

$Q_1$

---

0 Chomsky 1 kennt 2 das 3 Buch 4



## Grammatik:

- ①  $S \rightarrow NP VP$
- ②  $NP \rightarrow DET N$
- ③  $NP \rightarrow PROPN$
- ④  $VP \rightarrow V NP$
- ⑤  $DET \rightarrow \text{das}$
- ⑥  $N \rightarrow \text{Buch}$
- ⑦  $PROPN \rightarrow \text{Chomsky}$
- ⑧  $V \rightarrow \text{kennt}$

Pos. Zustände

---

$Q_0$

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROPN, 0)$

$(DET \rightarrow \cdot \text{das}, 0)$

$(PROPN \rightarrow \cdot \text{Chomsky}, 0)$

---

$Q_1$

$(PROPN \rightarrow \text{Chomsky} \cdot, 0)$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(4)  $VP \rightarrow V NP$

(8)  $V \rightarrow \text{kennt}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROPN, 0)$

$(DET \rightarrow \cdot \text{das}, 0)$

$(PROPN \rightarrow \cdot \text{Chomsky}, 0)$

Pos.    Zustände

---

$Q_1$

$(PROPN \rightarrow \text{Chomsky } \cdot, 0)$

---

$Q_2$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(4)  $VP \rightarrow V NP$

(8)  $V \rightarrow \text{kennt}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROP N, 0)$

$(DET \rightarrow \cdot \text{das}, 0)$

$(PROP N \rightarrow \cdot \text{Chomsky}, 0)$

Pos. Zustände

---

$Q_1$

$(PROP N \rightarrow \text{Chomsky} \cdot, 0)$

$(NP \rightarrow PROP N \cdot, 0)$

---

$Q_2$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(4)  $VP \rightarrow V NP$

(8)  $V \rightarrow \text{kennt}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROPN, 0)$

$(DET \rightarrow \cdot \text{das}, 0)$

$(PROPN \rightarrow \cdot \text{Chomsky}, 0)$

Pos. Zustände

---

$Q_1$

$(PROPN \rightarrow \text{Chomsky } \cdot, 0)$

$(NP \rightarrow PROPN \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

---

$Q_2$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(4)  $VP \rightarrow V NP$

(8)  $V \rightarrow \text{kennt}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROPN, 0)$

$(DET \rightarrow \cdot \text{das}, 0)$

$(PROPN \rightarrow \cdot \text{Chomsky}, 0)$

Pos. Zustände

---

$Q_1$

$(PROPN \rightarrow \text{Chomsky} \cdot, 0)$

$(NP \rightarrow PROPN \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

---

$Q_2$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(4)  $VP \rightarrow V NP$

(8)  $V \rightarrow \text{kennt}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROPN, 0)$

$(DET \rightarrow \cdot \text{das}, 0)$

$(PROPN \rightarrow \cdot \text{Chomsky}, 0)$

Pos. Zustände

---

$Q_1$

$(PROPN \rightarrow \text{Chomsky } \cdot, 0)$

$(NP \rightarrow PROPN \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$(V \rightarrow \cdot \text{kennt}, 1)$

---

$Q_2$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(4)  $VP \rightarrow V NP$

(8)  $V \rightarrow \text{kennt}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$(S \rightarrow \cdot NP VP, 0)$

$(NP \rightarrow \cdot DET N, 0)$

$(NP \rightarrow \cdot PROPN, 0)$

$(DET \rightarrow \cdot \text{das}, 0)$

$(PROPN \rightarrow \cdot \text{Chomsky}, 0)$

Pos. Zustände

---

$Q_1$

$(PROPN \rightarrow \text{Chomsky } \cdot, 0)$

$(NP \rightarrow PROPN \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$(V \rightarrow \cdot \text{kennt}, 1)$

---

$Q_2$

$(V \rightarrow \text{kennt } \cdot, 1)$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(2)  $NP \rightarrow DET\ N$

(3)  $NP \rightarrow PROPN$

(5)  $DET \rightarrow \text{das}$

(7)  $PROPN \rightarrow \text{Chomsky}$

## $Q_1$ :

$(PROPN \rightarrow \text{Chomsky } \cdot, 0)$

$(NP \rightarrow PROPN \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V\ NP, 1)$

$(V \rightarrow \cdot \text{kennt}, 1)$

Pos. Zustände

---

$Q_2$

$(V \rightarrow \text{kennt } \cdot, 1)$

---

$Q_3$

---

0 Chomsky 1 kennt 2 das 3 Buch 4



## Grammatik:

(2)  $NP \rightarrow DET\ N$

(3)  $NP \rightarrow PROPN$

(5)  $DET \rightarrow \text{das}$

(7)  $PROPN \rightarrow \text{Chomsky}$

## $Q_1$ :

$(PROPN \rightarrow \text{Chomsky} \cdot, 0)$

$(NP \rightarrow PROPN \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V\ NP, 1)$

$(V \rightarrow \cdot \text{kennt}, 1)$

## Pos. Zustände

---

$Q_2$

$(V \rightarrow \text{kennt} \cdot, 1)$

$(VP \rightarrow V \cdot NP, 1)$

---

$Q_3$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(2) NP  $\rightarrow$  DET N

(3) NP  $\rightarrow$  PROPN

(5) DET  $\rightarrow$  das

(7) PROPN  $\rightarrow$  Chomsky

Q<sub>1</sub>:

(PROPN  $\rightarrow$  Chomsky  $\cdot$ , 0)

(NP  $\rightarrow$  PROPN  $\cdot$ , 0)

(S  $\rightarrow$  NP  $\cdot$  VP, 0)

(VP  $\rightarrow \cdot$  V NP, 1)

(V  $\rightarrow \cdot$  kennt, 1)

Pos. Zustände

---

Q<sub>2</sub>

(V  $\rightarrow$  kennt  $\cdot$ , 1)

(VP  $\rightarrow$  V  $\cdot$  NP, 1)

(NP  $\rightarrow \cdot$  DET N, 2)

---

Q<sub>3</sub>

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(2)  $NP \rightarrow DET\ N$

(3)  $NP \rightarrow PROP$

(5)  $DET \rightarrow \text{das}$

(7)  $PROP \rightarrow \text{Chomsky}$

## $Q_1$ :

$(PROP \rightarrow \text{Chomsky} \cdot, 0)$

$(NP \rightarrow PROP \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V\ NP, 1)$

$(V \rightarrow \cdot \text{kennt}, 1)$

## Pos. Zustände

$Q_2$

$(V \rightarrow \text{kennt} \cdot, 1)$

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET\ N, 2)$

$(NP \rightarrow \cdot PROP, 2)$

$Q_3$

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(2)  $NP \rightarrow DET\ N$

(3)  $NP \rightarrow PROPN$

(5)  $DET \rightarrow \text{das}$

(7)  $PROPN \rightarrow \text{Chomsky}$

## $Q_1$ :

$(PROPN \rightarrow \text{Chomsky} \cdot, 0)$

$(NP \rightarrow PROPN \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V\ NP, 1)$

$(V \rightarrow \cdot \text{kennt}, 1)$

## Pos. Zustände

$Q_2$

$(V \rightarrow \text{kennt} \cdot, 1)$

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET\ N, 2)$

$(NP \rightarrow \cdot PROPN, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

$Q_3$

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(2)  $NP \rightarrow DET\ N$

(3)  $NP \rightarrow PROPN$

(5)  $DET \rightarrow \text{das}$

(7)  $PROPN \rightarrow \text{Chomsky}$

## $Q_1$ :

$(PROPN \rightarrow \text{Chomsky} \cdot, 0)$

$(NP \rightarrow PROPN \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V\ NP, 1)$

$(V \rightarrow \cdot \text{kennt}, 1)$

## Pos. Zustände

$Q_2$

$(V \rightarrow \text{kennt} \cdot, 1)$

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET\ N, 2)$

$(NP \rightarrow \cdot PROPN, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

$(PROPN \rightarrow \cdot \text{Chomsky}, 2)$

$Q_3$

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(2)  $NP \rightarrow DET\ N$

(3)  $NP \rightarrow PROPN$

(5)  $DET \rightarrow \text{das}$

(7)  $PROPN \rightarrow \text{Chomsky}$

## $Q_1$ :

$(PROPN \rightarrow \text{Chomsky} \cdot, 0)$

$(NP \rightarrow PROPN \cdot, 0)$

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V\ NP, 1)$

$(V \rightarrow \cdot \text{kennt}, 1)$

## Pos. Zustände

$Q_2$

$(V \rightarrow \text{kennt} \cdot, 1)$

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET\ N, 2)$

$(NP \rightarrow \cdot PROPN, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

$(PROPN \rightarrow \cdot \text{Chomsky}, 2)$

$Q_3$

$(DET \rightarrow \text{das} \cdot, 2)$

0 Chomsky 1 kennt 2 **das** 3 Buch 4

# Grammatik:

(6)  $N \rightarrow \text{Buch}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$Q_1$ :

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$Q_2$ :

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET N, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

Pos. Zustände

---

$Q_3$

$(DET \rightarrow \text{das} \cdot, 2)$

---

$Q_4$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(6)  $N \rightarrow \text{Buch}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$Q_1$ :

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$Q_2$ :

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot \text{DET } N, 2)$

$(\text{DET} \rightarrow \cdot \text{das}, 2)$

Pos.    Zustände

---

$Q_3$

$(\text{DET} \rightarrow \text{das} \cdot, 2)$

$(NP \rightarrow \text{DET} \cdot N, 2)$

---

$Q_4$

---

0 Chomsky 1 kennt 2 das 3 Buch 4



## Grammatik:

(6)  $N \rightarrow \text{Buch}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$Q_1$ :

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$Q_2$ :

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET N, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

Pos. Zustände

---

$Q_3$

$(DET \rightarrow \text{das} \cdot, 2)$

$(NP \rightarrow DET \cdot N, 2)$

$(N \rightarrow \cdot \text{Buch}, 3)$

---

$Q_4$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(6)  $N \rightarrow \text{Buch}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$Q_1$ :

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$Q_2$ :

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET N, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

Pos. Zustände

---

$Q_3$

$(DET \rightarrow \text{das} \cdot, 2)$

$(NP \rightarrow DET \cdot N, 2)$

$(N \rightarrow \cdot \text{Buch}, 3)$

---

$Q_4$

$(N \rightarrow \text{Buch} \cdot, 3)$

---

0 Chomsky 1 kennt 2 das 3 **Buch** 4

## Grammatik:

(6)  $N \rightarrow \text{Buch}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$Q_1$ :

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$Q_2$ :

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET N, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

Pos. Zustände

---

$Q_3$

$(DET \rightarrow \text{das} \cdot, 2)$

$(NP \rightarrow DET \cdot N, 2)$

$(N \rightarrow \cdot \text{Buch}, 3)$

---

$Q_4$

$(N \rightarrow \text{Buch} \cdot, 3)$

$(NP \rightarrow DET N \cdot, 2)$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(6)  $N \rightarrow \text{Buch}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$Q_1$ :

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$Q_2$ :

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET N, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

Pos. Zustände

---

$Q_3$

$(DET \rightarrow \text{das} \cdot, 2)$

$(NP \rightarrow DET \cdot N, 2)$

$(N \rightarrow \cdot \text{Buch}, 3)$

---

$Q_4$

$(N \rightarrow \text{Buch} \cdot, 3)$

$(NP \rightarrow DET N \cdot, 2)$

$(VP \rightarrow V NP \cdot, 1)$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(6)  $N \rightarrow \text{Buch}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$Q_1$ :

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$Q_2$ :

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET N, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

Pos. Zustände

---

$Q_3$

$(DET \rightarrow \text{das} \cdot, 2)$

$(NP \rightarrow DET \cdot N, 2)$

$(N \rightarrow \cdot \text{Buch}, 3)$

---

$Q_4$

$(N \rightarrow \text{Buch} \cdot, 3)$

$(NP \rightarrow DET N \cdot, 2)$

$(VP \rightarrow V NP \cdot, 1)$

$(S \rightarrow NP VP \cdot, 0)$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(6)  $N \rightarrow \text{Buch}$

$Q_0$ :

( $S' \rightarrow \cdot S, 0$ )

$Q_1$ :

( $S \rightarrow NP \cdot VP, 0$ )

( $VP \rightarrow \cdot V NP, 1$ )

$Q_2$ :

( $VP \rightarrow V \cdot NP, 1$ )

( $NP \rightarrow \cdot DET N, 2$ )

( $DET \rightarrow \cdot \text{das}, 2$ )

Pos. Zustände

---

$Q_3$

( $DET \rightarrow \text{das} \cdot, 2$ )

( $NP \rightarrow DET \cdot N, 2$ )

( $N \rightarrow \cdot \text{Buch}, 3$ )

---

$Q_4$

( $N \rightarrow \text{Buch} \cdot, 3$ )

( $NP \rightarrow DET N \cdot, 2$ )

( $VP \rightarrow V NP \cdot, 1$ )

( $S \rightarrow NP VP \cdot, 0$ )

( $S' \rightarrow S \cdot, 0$ )

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Grammatik:

(6)  $N \rightarrow \text{Buch}$

$Q_0$ :

$(S' \rightarrow \cdot S, 0)$

$Q_1$ :

$(S \rightarrow NP \cdot VP, 0)$

$(VP \rightarrow \cdot V NP, 1)$

$Q_2$ :

$(VP \rightarrow V \cdot NP, 1)$

$(NP \rightarrow \cdot DET N, 2)$

$(DET \rightarrow \cdot \text{das}, 2)$

Pos. Zustände

---

$Q_3$

$(DET \rightarrow \text{das} \cdot, 2)$

$(NP \rightarrow DET \cdot N, 2)$

$(N \rightarrow \cdot \text{Buch}, 3)$

---

$Q_4$

$(N \rightarrow \text{Buch} \cdot, 3)$

$(NP \rightarrow DET N \cdot, 2)$

$(VP \rightarrow V NP \cdot, 1)$

$(S \rightarrow NP VP \cdot, 0)$

$(S' \rightarrow S \cdot, 0) \checkmark$

---

0 Chomsky 1 kennt 2 das 3 Buch 4

## Top-Down-Parsing mit Extras



## Top-Down-Parsing mit Extras

- Zwischenergebnisse werden in Datenstruktur (Chart) gespeichert (→ Chart-Parsing, Dynamische Programmierung)

## Top-Down-Parsing mit Extras

- Zwischenergebnisse werden in Datenstruktur (Chart) gespeichert (→ Chart-Parsing, Dynamische Programmierung)
- Zustände werden mit Positionen in der Eingabesequenz abgeglichen (Elemente des Bottom-Up-Parsings)

## Top-Down-Parsing mit Extras

- Zwischenergebnisse werden in Datenstruktur (Chart) gespeichert (→ Chart-Parsing, Dynamische Programmierung)
  - Zustände werden mit Positionen in der Eingabesequenz abgeglichen (Elemente des Bottom-Up-Parsings)
- Komplizierter als *Recursive Descent* und *Shift Reduce*

## Top-Down-Parsing mit Extras

- Zwischenergebnisse werden in Datenstruktur (Chart) gespeichert (→ Chart-Parsing, Dynamische Programmierung)
- Zustände werden mit Positionen in der Eingabesequenz abgeglichen (Elemente des Bottom-Up-Parsings)
- Komplizierter als *Recursive Descent* und *Shift Reduce*
- Dafür wesentlich schneller

## Top-Down-Parsing mit Extras

- Zwischenergebnisse werden in Datenstruktur (Chart) gespeichert (→ Chart-Parsing, Dynamische Programmierung)
- Zustände werden mit Positionen in der Eingabesequenz abgeglichen (Elemente des Bottom-Up-Parsings)
- Komplizierter als *Recursive Descent* und *Shift Reduce*
- Dafür wesentlich schneller

## Komplexität

## Top-Down-Parsing mit Extras

- Zwischenergebnisse werden in Datenstruktur (Chart) gespeichert (→ Chart-Parsing, Dynamische Programmierung)
- Zustände werden mit Positionen in der Eingabesequenz abgeglichen (Elemente des Bottom-Up-Parsings)
- Komplizierter als *Recursive Descent* und *Shift Reduce*
- Dafür wesentlich schneller

## Komplexität

- Laufzeit in  $\mathcal{O}(n^3)$  im schlimmsten Fall

## Top-Down-Parsing mit Extras

- Zwischenergebnisse werden in Datenstruktur (Chart) gespeichert (→ Chart-Parsing, Dynamische Programmierung)
- Zustände werden mit Positionen in der Eingabesequenz abgeglichen (Elemente des Bottom-Up-Parsings)
- Komplizierter als *Recursive Descent* und *Shift Reduce*
- Dafür wesentlich schneller

## Komplexität

- Laufzeit in  $\mathcal{O}(n^3)$  im schlimmsten Fall
- Für unambige Grammatiken sogar  $\mathcal{O}(n^2)$

## Top-Down-Parsing mit Extras

- Zwischenergebnisse werden in Datenstruktur (Chart) gespeichert (→ Chart-Parsing, Dynamische Programmierung)
- Zustände werden mit Positionen in der Eingabesequenz abgeglichen (Elemente des Bottom-Up-Parsings)
- Komplizierter als *Recursive Descent* und *Shift Reduce*
- Dafür wesentlich schneller

## Komplexität

- Laufzeit in  $\mathcal{O}(n^3)$  im schlimmsten Fall
- Für unambige Grammatiken sogar  $\mathcal{O}(n^2)$
- Für bestimmte Typen von Grammatiken (LR) sogar  $\mathcal{O}(n)$



# Earley Parser: Zusammenfassung

## Top-Down-Parsing mit Extras

- Zwischenergebnisse werden in Datenstruktur (Chart) gespeichert (→ Chart-Parsing, Dynamische Programmierung)
- Zustände werden mit Positionen in der Eingabesequenz abgeglichen (Elemente des Bottom-Up-Parsings)
- Komplizierter als *Recursive Descent* und *Shift Reduce*
- Dafür wesentlich schneller

## Komplexität

- Laufzeit in  $\mathcal{O}(n^3)$  im schlimmsten Fall
- Für unambige Grammatiken sogar  $\mathcal{O}(n^2)$
- Für bestimmte Typen von Grammatiken (LR) sogar  $\mathcal{O}(n)$
- Funktioniert am besten mit *links-rekursiven* Regeln

# Nächstes Thema

- 1 Top-Down-Parsing: Recursive Descent
- 2 Bottom-Up-Parsing: Shift Reduce
- 3 Chart Parsing: Earley Algorithmus
- 4 Parsing mit Merkmalstrukturen
  - Modifizierter Earley Parser
  - Komplexität
- 5 Ausblick: Statistisches Parsing

# Earley Algorithmus mit Merkmalen

## 1. Möglichkeit

## 1. Möglichkeit

- Parsen wie bisher und am Ende versuchen, zu unifizieren

## 1. Möglichkeit

- Parsen wie bisher und am Ende versuchen, zu unifizieren
- Unschön: **Zahl von möglichen Analysen wird nicht so früh wie möglich beschränkt**

## 1. Möglichkeit

- Parsen wie bisher und am Ende versuchen, zu unifizieren
  - Unschön: **Zahl von möglichen Analysen wird nicht so früh wie möglich beschränkt**
- Optimierungspotential

# Earley Algorithmus mit Merkmalen

## 1. Möglichkeit

- Parsen wie bisher und am Ende versuchen, zu unifizieren
  - Unschön: **Zahl von möglichen Analysen wird nicht so früh wie möglich beschränkt**
- Optimierungspotential

## 2. Möglichkeit

# Earley Algorithmus mit Merkmalen

## 1. Möglichkeit

- Parsen wie bisher und am Ende versuchen, zu unifizieren
  - Unschön: **Zahl von möglichen Analysen wird nicht so früh wie möglich beschränkt**
- Optimierungspotential

## 2. Möglichkeit

- Merkmalstruktur zu jedem Earley-Zustand hinzufügen



# Earley Algorithmus mit Merkmalen

## 1. Möglichkeit

- Parsen wie bisher und am Ende versuchen, zu unifizieren
  - Unschön: **Zahl von möglichen Analysen wird nicht so früh wie möglich beschränkt**
- Optimierungspotential

## 2. Möglichkeit

- Merkmalstruktur zu jedem Earley-Zustand hinzufügen
- Complete-Operation unifiziert die Merkmalstrukturen der beiden Zustände

# Earley Algorithmus mit Merkmalen

## 1. Möglichkeit

- Parsen wie bisher und am Ende versuchen, zu unifizieren
  - Unschön: **Zahl von möglichen Analysen wird nicht so früh wie möglich beschränkt**
- Optimierungspotential

## 2. Möglichkeit

- Merkmalstruktur zu jedem Earley-Zustand hinzufügen
- Complete-Operation unifiziert die Merkmalstrukturen der beiden Zustände
- Predict-Operation fügt neuen Zustand nur hinzu, wenn er von keinem vorhandenen *subsumiert* wird

# Earley Algorithmus mit Merkmalen

## 1. Möglichkeit

- Parsen wie bisher und am Ende versuchen, zu unifizieren
  - Unschön: **Zahl von möglichen Analysen wird nicht so früh wie möglich beschränkt**
- Optimierungspotential

## 2. Möglichkeit

- Merkmalstruktur zu jedem Earley-Zustand hinzufügen
- Complete-Operation unifiziert die Merkmalstrukturen der beiden Zustände
- Predict-Operation fügt neuen Zustand nur hinzu, wenn er von keinem vorhandenen *subsumiert* wird
- Nicht-destruktive Unifikation einsetzen! (Kopien machen!)

# Merkmalbasiertes Parsing: Komplexität

Unterschied gegenüber ursprünglichem Earley Parser

# Merkmalbasiertes Parsing: Komplexität

## Unterschied gegenüber ursprünglichem Earley Parser

- Zustandsmenge nach Zuständen durchsuchen, deren Merkmalstrukturen mit gegebener Merkmalstruktur unifizieren

# Merkmalbasiertes Parsing: Komplexität

## Unterschied gegenüber ursprünglichem Earley Parser

- Zustandsmenge nach Zuständen durchsuchen, deren Merkmalstrukturen mit gegebener Merkmalstruktur unifizieren
- Häufiges Kopieren von Merkmalstrukturen (nicht-destruktive Unifikation)

# Merkmalbasiertes Parsing: Komplexität

## Unterschied gegenüber ursprünglichem Earley Parser

- Zustandsmenge nach Zuständen durchsuchen, deren Merkmalstrukturen mit gegebener Merkmalstruktur unifizieren
- Häufiges Kopieren von Merkmalstrukturen (nicht-destruktive Unifikation)

## Komplexität

# Merkmalbasiertes Parsing: Komplexität

## Unterschied gegenüber ursprünglichem Earley Parser

- Zustandsmenge nach Zuständen durchsuchen, deren Merkmalstrukturen mit gegebener Merkmalstruktur unifizieren
- Häufiges Kopieren von Merkmalstrukturen (nicht-destruktive Unifikation)

## Komplexität

- im Allgemeinen ist Unifikationsparsen „relativ teuer“



# Merkmalbasiertes Parsing: Komplexität

## Unterschied gegenüber ursprünglichem Earley Parser

- Zustandsmenge nach Zuständen durchsuchen, deren Merkmalstrukturen mit gegebener Merkmalstruktur unifizieren
- Häufiges Kopieren von Merkmalstrukturen (nicht-destruktive Unifikation)

## Komplexität

- im Allgemeinen ist Unifikationsparsen „relativ teuer“
- NP-vollständig in manchen Versionen

# Merkmalbasiertes Parsing: Komplexität

## Unterschied gegenüber ursprünglichem Earley Parser

- Zustandsmenge nach Zuständen durchsuchen, deren Merkmalstrukturen mit gegebener Merkmalstruktur unifizieren
- Häufiges Kopieren von Merkmalstrukturen (nicht-destruktive Unifikation)

## Komplexität

- im Allgemeinen ist Unifikationsparsen „relativ teuer“
- NP-vollständig in manchen Versionen
- mit sehr umfangreichen Constraints sogar Turing-vollständig (!)

# Merkmalbasiertes Parsing: Komplexität

## Unterschied gegenüber ursprünglichem Earley Parser

- Zustandsmenge nach Zuständen durchsuchen, deren Merkmalstrukturen mit gegebener Merkmalstruktur unifizieren
- Häufiges Kopieren von Merkmalstrukturen (nicht-destruktive Unifikation)

## Komplexität

- im Allgemeinen ist Unifikationsparsen „relativ teuer“
- NP-vollständig in manchen Versionen
- mit sehr umfangreichen Constraints sogar Turing-vollständig (!)
- Zahlreiche Varianten existieren

# Merkmalbasiertes Parsing: Komplexität

## Unterschied gegenüber ursprünglichem Earley Parser

- Zustandsmenge nach Zuständen durchsuchen, deren Merkmalstrukturen mit gegebener Merkmalstruktur unifizieren
- Häufiges Kopieren von Merkmalstrukturen (nicht-destruktive Unifikation)

## Komplexität

- im Allgemeinen ist Unifikationsparsen „relativ teuer“
- NP-vollständig in manchen Versionen
- mit sehr umfangreichen Constraints sogar Turing-vollständig (!)
- Zahlreiche Varianten existieren
  - Quasi-destruktive Unifikation (Hideto Tomabechi)

# Merkmalbasiertes Parsing: Komplexität

## Unterschied gegenüber ursprünglichem Earley Parser

- Zustandsmenge nach Zuständen durchsuchen, deren Merkmalstrukturen mit gegebener Merkmalstruktur unifizieren
- Häufiges Kopieren von Merkmalstrukturen (nicht-destruktive Unifikation)

## Komplexität

- im Allgemeinen ist Unifikationsparsen „relativ teuer“
- NP-vollständig in manchen Versionen
- mit sehr umfangreichen Constraints sogar Turing-vollständig (!)
- Zahlreiche Varianten existieren
  - Quasi-destruktive Unifikation (Hideto Tomabechi)
  - Tractable HPSG (Gerald Penn)

# Nächstes Thema

- 1 Top-Down-Parsing: Recursive Descent
- 2 Bottom-Up-Parsing: Shift Reduce
- 3 Chart Parsing: Earley Algorithmus
- 4 Parsing mit Merkmalstrukturen
  - Modifizierter Earley Parser
  - Komplexität
- 5 **Ausblick: Statistisches Parsing**

## Orakel

Eine Funktion, die immer die richtige, nächste Operation liefert.

# Orakel aus Daten ableiten I

## Orakel

Eine Funktion, die immer die richtige, nächste Operation liefert.

## Probabilistisches Modell als Orakel

Eine Wahrscheinlichkeitsverteilung über Operationen gegeben der aktuelle Zustand (des Stapels, des Lesebuffers, der bisherigen Analyse):  $P(op \mid state)$



## Ableiten eines probabilistischen Modells aus Daten

- zu nutzende Features (Merkmale) des Parser-Zustandes festlegen

## Ableiten eines probabilistischen Modells aus Daten

- zu nutzende Features (Merkmale) des Parser-Zustandes festlegen
- gegeben ein Korpus aus Sätzen und Syntaxbäumen, erstelle für jeden Satz Paare von Features und korrekten Operationen ( $\Rightarrow$  Trainingsdaten)

## Ableiten eines probabilistischen Modells aus Daten

- zu nutzende Features (Merkmale) des Parser-Zustandes festlegen
- gegeben ein Korpus aus Sätzen und Syntaxbäumen, erstelle für jeden Satz Paare von Features und korrekten Operationen ( $\Rightarrow$  Trainingsdaten)
- Optimierte die Parameter eines statistischen Wahrscheinlichkeitsmodells dahingehend, dass es so oft wie möglich, die richtige Vorhersage macht.

# Kodierung von Merkmalen

## Binäre Merkmale

Ein binäres Merkmal ist entweder vorhanden (1) oder nicht (0).

## Beispiele

- $f_1 \equiv \text{stack} = [\dots | \text{in} | \text{München}]$
- $f_2 \equiv \text{stack} = [\dots | \text{München}]$  und  $\text{buffer} = [\text{gewesen} | \dots]$

## Merkmalsvektoren

- repräsentieren alle möglichen Merkmale als Folge von 0 und 1
- ermöglichen Methoden der linearen Algebra ( $\Rightarrow$  Modell)
- Bsp.:  $\vec{x} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$

## Einfaches lineares Modell

Sei Matrix  $\mathbf{W} \in \mathbb{R}^{|Op| \times |F|}$  mit  $Op$  Operationen und  $F$  Features.  
Jede Zeile (Operation) gewichtet die Merkmale anders.

Wir berechnen Scores für jede Operation durch Multiplikation:  $\mathbf{W}\vec{x}$

# Softmax-Regression

## Einfaches lineares Modell

Sei Matrix  $\mathbf{W} \in \mathbb{R}^{|Op| \times |F|}$  mit  $Op$  Operationen und  $F$  Features.  
Jede Zeile (Operation) gewichtet die Merkmale anders.

Wir berechnen Scores für jede Operation durch Multiplikation:  $\mathbf{W}\vec{x}$

## Normalisierung durch Softmax

$$softmax(\vec{v})_i = \frac{e^{v_i}}{\sum_j e^{v_j}}$$

# Softmax-Regression

## Einfaches lineares Modell

Sei Matrix  $\mathbf{W} \in \mathbb{R}^{|Op| \times |F|}$  mit  $Op$  Operationen und  $F$  Features.  
Jede Zeile (Operation) gewichtet die Merkmale anders.

Wir berechnen Scores für jede Operation durch Multiplikation:  $\mathbf{W}\vec{x}$

## Normalisierung durch Softmax

$$softmax(\vec{v})_i = \frac{e^{v_i}}{\sum_j e^{v_j}}$$

- Werte liegen zwischen 0 und 1

# Softmax-Regression

## Einfaches lineares Modell

Sei Matrix  $\mathbf{W} \in \mathbb{R}^{|Op| \times |F|}$  mit  $Op$  Operationen und  $F$  Features.  
Jede Zeile (Operation) gewichtet die Merkmale anders.

Wir berechnen Scores für jede Operation durch Multiplikation:  $\mathbf{W}\vec{x}$

## Normalisierung durch Softmax

$$softmax(\vec{v})_i = \frac{e^{v_i}}{\sum_j e^{v_j}}$$

- Werte liegen zwischen 0 und 1
- $\sum_j softmax(\vec{v})_j = 1$



# Softmax-Regression

## Einfaches lineares Modell

Sei Matrix  $\mathbf{W} \in \mathbb{R}^{|Op| \times |F|}$  mit  $Op$  Operationen und  $F$  Features.  
Jede Zeile (Operation) gewichtet die Merkmale anders.

Wir berechnen Scores für jede Operation durch Multiplikation:  $\mathbf{W}\vec{x}$

## Normalisierung durch Softmax

$$softmax(\vec{v})_i = \frac{e^{v_i}}{\sum_j e^{v_j}}$$

- Werte liegen zwischen 0 und 1
- $\sum_j softmax(\vec{v})_j = 1$
- $softmax(\vec{v})_{op}$  ist die Wahrscheinlichkeit für Operation  $op$

## Kosten-/Gütefunktion

Negative log likelihood

$$\mathcal{L}(\mathbf{W}) = - \sum_{i=1}^n \log P(y^i | \vec{x}^i)$$

- Trainiert wird durch Minimierung der Kostenfunktion
- Maximierung von Güte entspricht der Minimierung von Kosten
- Der Unterschied ist ein Minuszeichen
- Zahlreiche Bibliotheken existieren, die die automatische Optimierung einer solchen Funktion ermöglichen: z. B. Pytorch, Tensorflow, Dynet.

- liefert immer ein Ergebnis

# Statistik vs. formale Grammatik

- liefert immer ein Ergebnis
- findet häufig eine gute (wahrscheinliche) Lösung

- liefert immer ein Ergebnis
- findet häufig eine gute (wahrscheinliche) Lösung
- Annotatoren können sich auf echte linguistische Beispiele konzentrieren

- liefert immer ein Ergebnis
- findet häufig eine gute (wahrscheinliche) Lösung
- Annotatoren können sich auf echte linguistische Beispiele konzentrieren
- kann effizient berechnet werden

# Statistik vs. formale Grammatik

- liefert immer ein Ergebnis
- findet häufig eine gute (wahrscheinliche) Lösung
- Annotatoren können sich auf echte linguistische Beispiele konzentrieren
- kann effizient berechnet werden
- erkennt ungrammatische Sätze; ist aber auch naturgemäß unvollständig

# Statistik vs. formale Grammatik

- liefert immer ein Ergebnis
- findet häufig eine gute (wahrscheinliche) Lösung
- Annotatoren können sich auf echte linguistische Beispiele konzentrieren
- kann effizient berechnet werden
- erkennt ungrammatische Sätze; ist aber auch naturgemäß unvollständig
- findet immer alle korrekte(n) Lösung(en)



# Statistik vs. formale Grammatik

- liefert immer ein Ergebnis
- findet häufig eine gute (wahrscheinliche) Lösung
- Annotatoren können sich auf echte linguistische Beispiele konzentrieren
- kann effizient berechnet werden
- erkennt ungrammatische Sätze; ist aber auch naturgemäß unvollständig
- findet immer alle korrekte(n) Lösung(en)
- Generalisierung muss vom Grammatikschreiber eingeplant werden

# Statistik vs. formale Grammatik

- liefert immer ein Ergebnis
- findet häufig eine gute (wahrscheinliche) Lösung
- Annotatoren können sich auf echte linguistische Beispiele konzentrieren
- kann effizient berechnet werden
- erkennt ungrammatische Sätze; ist aber auch naturgemäß unvollständig
- findet immer alle korrekte(n) Lösung(en)
- Generalisierung muss vom Grammatikschreiber eingeplant werden
- exakte Algorithmen benötigen polynomiale Laufzeit

# Rückblick auf heutige Themen

- 1 Top-Down-Parsing: Recursive Descent
- 2 Bottom-Up-Parsing: Shift Reduce
- 3 Chart Parsing: Earley Algorithmus
- 4 Parsing mit Merkmalstrukturen
  - Modifizierter Earley Parser
  - Komplexität
- 5 Ausblick: Statistisches Parsing