

Einführung in die Computerlinguistik

Kontextfreie Grammatiken

Alexander Fraser and Robert Zangenfeind

Center for Information and Language Processing

2020-01-20

Die Grundfassung dieses Foliensatzes wurde von Prof. Dr. Stefan Evert erstellt, und von Prof. Dr. Hinrich Schütze erweitert.

Fehler und Mängel sind ausschließlich meine Verantwortung.

- 1 Kontextfreie Grammatiken
- 2 Top-down parsing
- 3 CYK

1 Kontextfreie Grammatiken

2 Top-down parsing

3 CYK

- Mächtiger als reguläre Sprachen / Automaten
- Rekursive Strukturen
- Konstituentengrammatik

Definition kontextfreie Grammatik (CFG)

Eine kontextfreie Grammatik G über dem Alphabet Σ ist ein Quadrupel $G = (V, \Sigma, P, S)$. Die Elemente von V heißen **Variablen** oder **Nichtterminalsymbole**, entsprechend werden die Zeichen aus Σ auch als **Terminalsymbole** bezeichnet. Wir nehmen stets $V \cap \Sigma = \emptyset$ an. Üblicherweise verwenden wir für Terminalsymbole Kleinbuchstaben $a, b, c, \dots \in \Sigma$ und für Variablen Großbuchstaben $A, B, C, \dots \in V$. Zur Unterscheidung von Wörtern $u, v, w, \dots \in \Sigma^*$ bezeichnen wir Zeichenketten, die sowohl Variablen als auch Terminalsymbole enthalten können, als **Terme** und verwenden dafür griechische Kleinbuchstaben $\alpha, \beta, \gamma, \dots \in (V \cup \Sigma)^*$. $S \in V$ ist eine spezielle Variable, die **Startsymbol** genannt wird. $P \subseteq V \times (V \cup \Sigma)^*$ schließlich ist die Menge der **Produktionen**: jede Produktion ist von der Form $A \rightarrow \alpha$, wobei A eine Variable und α ein beliebiger Term ist.

Eine Produktion $A \rightarrow \alpha \in P$ wird auch als **Regel** bezeichnet, A als **linke Seite** und α als **rechte Seite** der Regel.

Zur Vereinfachung der Notation dürfen Regeln $A \rightarrow \alpha_1, A \rightarrow \alpha_2, \dots, A \rightarrow \alpha_n$ mit identischer linken Seite zusammengefasst werden:
 $A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$.

Ein **Ableitungsschritt** $\delta \Rightarrow_G \delta'$ überführt einen Term δ durch Ersetzung genau einer Variable in einen Term δ' . In der formalen Darstellung schreiben wir $\delta = \beta A \gamma$ und $\delta' = \beta \alpha \gamma$, wobei $A \in V$ die genannte Variable ist, die durch einen Term α ersetzt wird. Der Ableitungsschritt $\beta A \gamma \Rightarrow_G \beta \alpha \gamma$ ist **zulässig**, wenn es eine Produktion $A \rightarrow \alpha \in P$ gibt.

Eine **Ableitung** ist eine beliebige Folge von zulässigen Ableitungsschritten: $\alpha_1 \Rightarrow_G \alpha_2 \Rightarrow_G \dots \Rightarrow_G \alpha_n$. Wir schreiben kurz $\alpha_1 \Rightarrow_G^* \alpha_n$ und sagen, dass α_n aus α_1 **ableitbar** ist. Der Index G kann dabei ausgelassen werden, sofern klar ist, bezüglich welcher Grammatik G die Ableitung durchgeführt wird.

Definition kontextfreie Sprache

Die von G beschriebene **formale Sprache** $\mathcal{L}[G]$ ist die Menge aller Wörter w , die aus dem Startsymbol ableitbar sind:

$$\mathcal{L}[G] := \{w \in \Sigma^* \mid S \Rightarrow^* w\}.$$

Eine Sprache $L \subseteq \Sigma^*$ heißt **kontextfrei**, wenn sie durch eine kontextfreie Grammatik G beschrieben werden kann, d.h. wenn $L = \mathcal{L}[G]$ gilt.

Als Beispiel betrachten wir eine kontextfreie Grammatik G_1 für einfache arithmetische Ausdrücke über dem Alphabet $\Sigma_1 = \{0, 1, \dots, 9, +, *\}$. $G_1 = (V_1, \Sigma_1, P_1, S)$ ist folgendermaßen definiert:

$$V_1 := \{S, T\}$$

$$P_1 := \{S \rightarrow T,$$

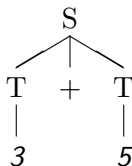
$$T \rightarrow T + T \mid T * T$$

$$T \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9\}.$$

Die Variable T repräsentiert dabei jeweils einen arithmetischen Term. Das Wort $w = 3 + 5$ gehört zu $\mathcal{L}[G_1]$, da es aus S ableitbar ist: $S \Rightarrow T \Rightarrow T + T \Rightarrow 3 + T \Rightarrow 3 + 5$, also kurz $S \Rightarrow^* 3 + 5$.

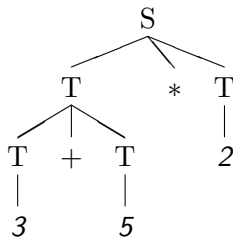
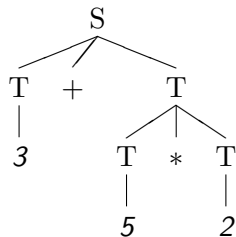
Ableitungsbaum für

$$S \Rightarrow T \Rightarrow T + T \Rightarrow 3 + T \Rightarrow 3 + 5$$



Die Ableitung von w bezüglich der Grammatik G_1 ist nicht eindeutig: eine andere mögliche Ableitung ist $S \Rightarrow T \Rightarrow T + T \Rightarrow T + 5 \Rightarrow 3 + 5$. Beide Varianten führen jedoch auf denselben Ableitungsbaum.

Ein Wort w kann bezüglich einer CFG G auch mehrere verschiedene Ableitungsbäume besitzen. Z.B. hat $w = 3 + 5 * 2 \in \mathcal{L}[G_1]$ die folgenden beiden Ableitungsbäume:



Die beiden Ableitungsbäume weisen w unterschiedliche **Struktur** zu, was für Anwendungen von großer Bedeutung ist. Man bezeichnet den Ableitungsbaum daher auch als **Analyse** von w durch die Grammatik. (Man denke z.B. an ein Taschenrechnerprogramm, das arithmetische Ausdrücke anhand ihres Ableitungsbaums auswertet. In diesem Fall wäre die linke Analyse die gewünschte, da $*$ stärker bindet als $+$). Eine Grammatik, in der es ein Wort w mit mehreren verschiedenen Ableitungsbäumen (bzw. Linksableitungen) gibt, heißt **mehrdeutig** / **ambig**.

Im obigen Beispiel wäre es wünschenswert, G_1 so abzuändern, dass $w = 3 + 5 * 2$ nur noch eine Analyse besitzt (nämlich die durch den linken Baum dargestellte). Eine solche Grammatik ist $G_2 = (V_2, \Sigma_1, P_1, S)$ mit

$$V_2 := \{S, P\}$$

$$P_2 := \{S \rightarrow S + S | P,$$

$$S \rightarrow 0|1|2|3|4|5|6|7|8|9,$$

$$P \rightarrow P * P,$$

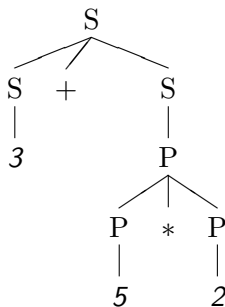
$$P \rightarrow 0|1|2|3|4|5|6|7|8|9\}.$$

Dabei stehen die Variablen S und P anschaulich für *Summe* und *Produkt*.

Verbesserte Grammatik:

Beispiel $w = 3 + 5 * 2$

Bezüglich G_2 besitzt w nur noch die folgende eindeutige Analyse:



- 1 Kontextfreie Grammatiken
- 2 Top-down parsing
- 3 CYK

Top-Down Parsing

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$

Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$

S

Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

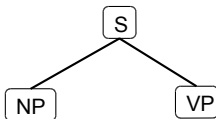
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

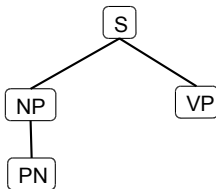
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

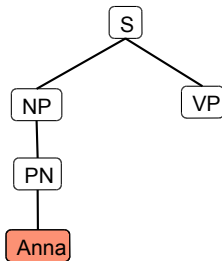
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

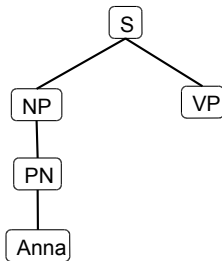
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

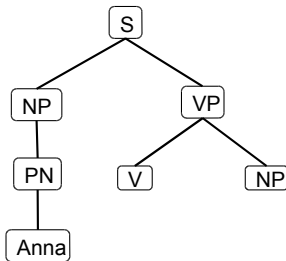
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

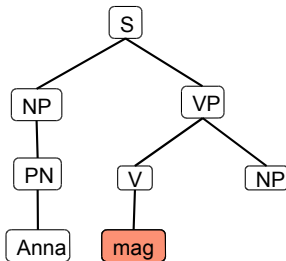
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna **mag** die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

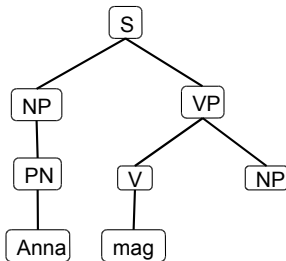
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

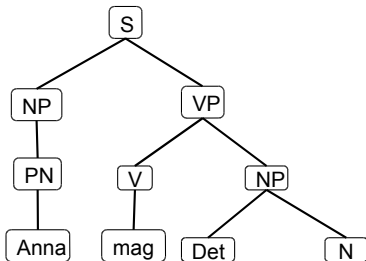
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

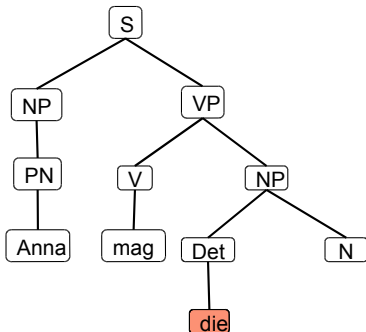
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag **die** Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

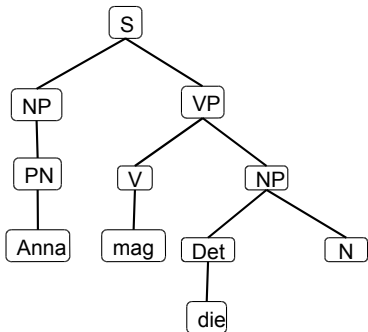
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow \textit{Katze}$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

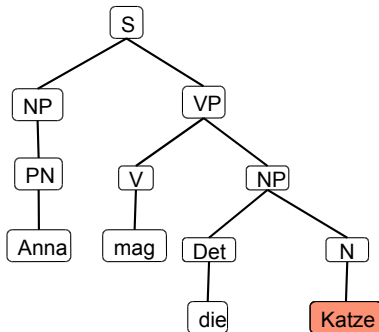
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



*Anna mag die **Katze***

Kontextfreier Top-Down Parser

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

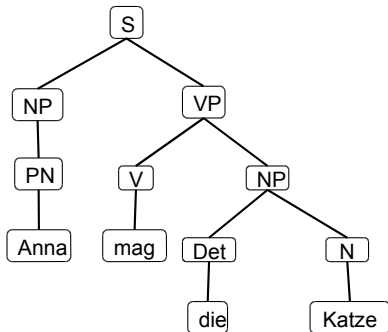
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze_

Nicht-Determinismus in der Regelanwendung

S → NP VP

NP → Det N

NP → PN

VP → V NP

Det → *die*

N → *Katze*

V → *mag*

PN → *Anna*

S

Anna mag die Katze

Nicht-Determinismus in der Regelanwendung

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

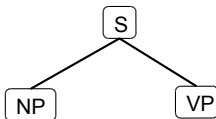
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Nicht-Determinismus in der Regelanwendung

$S \rightarrow NP VP$

$NP \rightarrow Det N$

$NP \rightarrow PN$

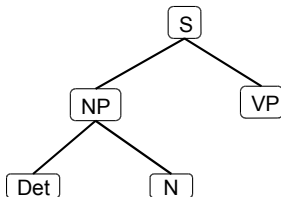
$VP \rightarrow V NP$

$Det \rightarrow die$

$N \rightarrow Katze$

$V \rightarrow mag$

$PN \rightarrow Anna$



Anna mag die Katze

Nicht-Determinismus in der Regelanwendung

S → NP VP

NP → Det N

NP → PN

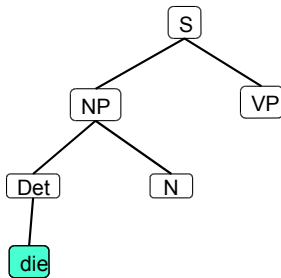
VP → V NP

Det → *die*

N → *Katze*

V → *mag*

PN → *Anna*



Anna mag die Katze

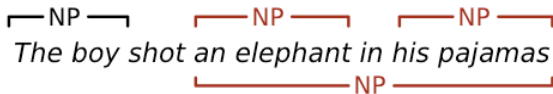
Top-Down-Parser, Eigenschaften

- Der Top-Down-Parser ist im Allgemeinen **nicht-deterministisch**: Für dasselbe Nichtterminal gibt es mehrere, eventuell sehr viele Ersetzungsregeln.
- Eine technische Lösung: Arbeiten mit einer Agenda, Last-In-First-Out, **Tiefensuche mit Backtracking**.
- Problem: Es werden viele Teilstrukturen erzeugt, die nie erfolgreich sein können, weil die Eingabekette keine passenden Wörter enthält.
 - Beispiel: Grammatik versucht, Subjekts-NP abzuleiten, der Satz fängt mit einer PP an.
- Im Falle „links-rekursiver“ Grammatiken geht der Parser in eine Endlosschleife.
 - Beispiel: $VP \rightarrow VP PP$
 - Links-rekursive Regeln kann man vermeiden, aber dann sind bestimmte Strukturen nicht mehr natürlich darstellbar.

- 1 Kontextfreie Grammatiken
- 2 Top-down parsing
- 3 CYK

Dynamische Programmierung

- Kontextfreie Grammatiken: die Anwendbarkeit einer Regel in einer Ableitung ist unabhängig vom Kontext.



diese Teile des Satzes können jeweils als NP-Konstituenten analysiert werden, unabhängig davon, wie der finale Parse-Baum aussieht.

One morning I shot an elephant in my pajamas.

How he got into my pajamas I don't know.

Chart-Parsing

- **Charts** sind kompakte Repräsentation aller (lokal) möglichen Teilkonstituenten der Eingabekette.
- **Charts können enthalten:**
 - Konstituenten, die bereits gefunden wurden
 - Hypothesen darüber, welche Konstituenten gefunden werden können (z.B. Earley-Algorithmus).
- Verschiedene Chart-Parser:
 - CYK, Earley, Bottom-up chart parser, ...
 - generieren alle möglichen Parse-Bäume bei Mehrdeutigkeiten

Cocke-Younger-Kasami (CYK) Algorithmus

- Chart-basierter Bottom-up-Parser, nutzt Prinzip der dynamischen Programmierung.
- Entwickelt von John Cocke, Tadao Kasami & Daniel Younger
- **Idee:**
 - finde alle Konstituenten mit Länge 1
 - finde alle Konstituenten mit Länge 2
 - finde alle Konstituenten mit Länge 3
 - ...
- Die Grammatik muss in **Chomsky-Normalform** vorliegen:
 - $A \rightarrow w$ (w Terminalsymbol)
 - $A \rightarrow BC$ (B und C Nichtterminalsymbole)
 - $S \rightarrow \varepsilon$ (S Startsymbol, nur wenn $\varepsilon \in L$)
 - hier nehmen wir an, dass $\varepsilon \notin L$

CYK: Chart als Matrix

Datenstruktur

Satzlänge $n=7 \rightarrow 7 \times 7$ Matrix
 \rightarrow wir nutzen die
obere Dreiecksmatrix
 \rightarrow Notation: [Zeile, Spalte]

N \rightarrow I | elephant | pajamas

V \rightarrow shot P \rightarrow in

DET \rightarrow an POSS \rightarrow my

NP \rightarrow DET N | POSS N | NP PP

PP \rightarrow P NP

S \rightarrow N VP | NP VP

VP \rightarrow V NP | VP PP

	I	shot	an	elephant	in	my	pajamas
	0	1	2	3	4	5	6
0	N						
1		V					
2			DET				
3				N			
4					P		
5						POSS	
6							N

Schritt 1: Diagonale ausfüllen

- \rightarrow Terminale zu Nichtterminalen
- \rightarrow Eintrag für Wort i in Zelle $[i, i]$
- \rightarrow benutze Lexikon

CYK: Chart als Matrix

Datenstruktur

Satzlänge $n=7 \rightarrow 7 \times 7$ Matrix
 \rightarrow wir nutzen die
obere Dreiecksmatrix
 \rightarrow Notation: [Zeile, Spalte]

N \rightarrow I | elephant | pajamas

V \rightarrow shot P \rightarrow in

DET \rightarrow an POSS \rightarrow my

NP \rightarrow DET N | POSS N | NP PP

PP \rightarrow P NP

S \rightarrow N VP | NP VP

VP \rightarrow V NP | VP PP

	I	shot	an	elephant	in	my	pajamas
	0	1	2	3	4	5	6
0	N	/					
1		V	/				
2			DET	NP			
3				N	/		
4					P	/	
5						POSS	NP
6							N

Schritt 2: Diagonale darüber ausfüllen

\rightarrow benutze Regeln der Form $A \rightarrow BC$

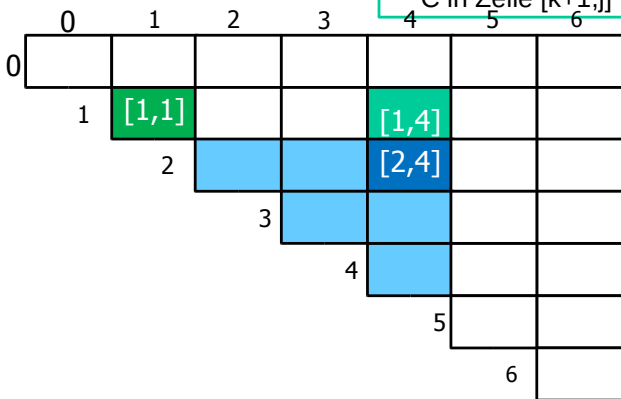
\rightarrow trage A in Zelle $[i-1, i]$ ein wenn B in Zelle $[i-1, i-1]$ und C in Zelle $[i, i]$

CYK Algorithmus: Beispiel

Schritt 2 bis Schritt n allgemein:
fülle jeweils die nächste Diagonale aus

trage A in Zelle $[i,j]$ ein, falls

- es eine Regel $A \rightarrow B C$ gibt und
- B in Zelle $[i,k]$ und
- C in Zelle $[k+1,j]$ $i \leq k < j$



CYK Algorithmus: Beispiel

Schritt 2 bis Schritt n allgemein:
fülle jeweils die nächste Diagonale aus

trage A in Zelle $[i,j]$ ein, falls

- es eine Regel $A \rightarrow B C$ gibt und
- B in Zelle $[i,k]$ und
- C in Zelle $[k+1,j]$ $i \leq k < j$

	0	1	2	3	4	5	6
0							
1			[1,2]		[1,4]		
2							
3					[3,4]		
4							
5							
6							

CYK Algorithmus: Beispiel

Schritt 2 bis Schritt n allgemein:
fülle jeweils die nächste Diagonale aus

trage A in Zelle $[i,j]$ ein, falls

- es eine Regel $A \rightarrow B C$ gibt und
- B in Zelle $[i,k]$ und
- C in Zelle $[k+1,j]$ $i \leq k < j$

	0	1	2	3	4	5	6
0							
1				[1,3]	[1,4]		
2							
3							
4					[4,4]		
5							
6							

CYK: Algorithmus

Datenstruktur

Satzlänge $n=7 \rightarrow 7 \times 7$ Matrix
 \rightarrow wir nutzen die
obere Dreiecksmatrix
 \rightarrow Notation: [Zeile, Spalte]

$N \rightarrow I \mid \text{elephant} \mid \text{pajamas}$

$V \rightarrow \text{shot} \quad P \rightarrow \text{in}$

$\text{DET} \rightarrow \text{an} \quad \text{POSS} \rightarrow \text{my}$

$\text{NP} \rightarrow \text{DET } N \mid \text{POSS } N \mid \text{NP } \text{PP}$

$\text{PP} \rightarrow P \text{ NP}$

$S \rightarrow N \text{ VP} \mid \text{NP } \text{VP}$

$\text{VP} \rightarrow V \text{ NP} \mid \text{VP } \text{PP}$

	I	shot	an	elephant	in	my	pajamas
	0	1	2	3	4	5	6
0	N	/					
1		V	/				
2			DET	NP			
3				N	/		
4					P	/	
5						POSS	NP
6							N

Schritt 2 bis Schritt n allgemein:

fülle jeweils die nächste Diagonale aus
trage A in Zelle $[i,j]$ ein, falls

- es eine Regel $A \rightarrow B C$ gibt und
- B in Zelle $[i,k]$ und
- C in Zelle $[k+1,i]$ $i \leq k < i$

CYK: Algorithmus

Datenstruktur

Satzlänge $n=7 \rightarrow 7 \times 7$ Matrix
 \rightarrow wir nutzen die
obere Dreiecksmatrix
 \rightarrow Notation: [Zeile, Spalte]

N \rightarrow I | elephant | pajamas

V \rightarrow shot P \rightarrow in

DET \rightarrow an POSS \rightarrow my

NP \rightarrow DET N | POSS N | NP PP

PP \rightarrow P NP

S \rightarrow N VP | NP VP

VP \rightarrow V NP | VP PP

	I	shot	an	elephant	in	my	pajamas
	0	1	2	3	4	5	6
0	N	/					
1		V	/				
2			DET	NP			
3				N	/		
4					P	/	
5						POSS	NP
6							N

Schritt 2 bis Schritt n allgemein:

fülle jeweils die nächste Diagonale aus
trage A in Zelle $[i,j]$ ein, falls

- es eine Regel $A \rightarrow B C$ gibt und
- B in Zelle $[i,k]$ und
- C in Zelle $[k+1,i]$ $i \leq k < i$

CYK: Algorithmus

Datenstruktur

Satzlänge $n=7 \rightarrow 7 \times 7$ Matrix
 \rightarrow wir nutzen die
obere Dreiecksmatrix
 \rightarrow Notation: [Zeile, Spalte]

$N \rightarrow I \mid \text{elephant} \mid \text{pajamas}$
 $V \rightarrow \text{shot} \quad P \rightarrow \text{in}$
 $\text{DET} \rightarrow \text{an} \quad \text{POSS} \rightarrow \text{my}$

$\text{NP} \rightarrow \text{DET } N \mid \text{POSS } N \mid \text{NP } \text{PP}$
 $\text{PP} \rightarrow P \text{ NP}$

$S \rightarrow N \text{ VP} \mid \text{NP } \text{VP}$
 $\text{VP} \rightarrow V \text{ NP} \mid \text{VP } \text{PP}$

	I	shot	an	elephant	in	my	pajamas
	0	1	2	3	4	5	6
0	N	—	—				
1		V	—				
2			DET	NP			
3				N	—		
4					P	—	
5						POSS	NP
6							N

Schritt 2 bis Schritt n allgemein:

fülle jeweils die nächste Diagonale aus
trage A in Zelle $[i,j]$ ein, falls

- es eine Regel $A \rightarrow B C$ gibt und
- B in Zelle $[i,k]$ und
- C in Zelle $[k+1,i]$ $i \leq k < i$

CYK: Algorithmus

Datenstruktur

Satzlänge $n=7 \rightarrow 7 \times 7$ Matrix
 \rightarrow wir nutzen die
 obere Dreiecksmatrix
 \rightarrow Notation: [Zeile, Spalte]

N \rightarrow I | elephant | pajamas
 V \rightarrow shot P \rightarrow in
 DET \rightarrow an POSS \rightarrow my

NP \rightarrow DET N | POSS N | NP PP
 PP \rightarrow P NP
 S \rightarrow N VP | NP VP
 VP \rightarrow V NP | VP PP

	I	shot	an	elephant	in	my	pajamas
	0	1	2	3	4	5	6
0	N	—	—				
1		V	—	VP			
2			DET	NP			
3				N	—		
4					P	—	
5						POSS	NP
6							N

Schritt 2 bis Schritt n allgemein:

fülle jeweils die nächste Diagonale aus
 trage A in Zelle $[i,j]$ ein, falls

- es eine Regel $A \rightarrow B C$ gibt und
- B in Zelle $[i,k]$ und
- C in Zelle $[k+1,i]$ $i \leq k < i$

CYK: Algorithmus

Datenstruktur

Satzlänge $n=7 \rightarrow 7 \times 7$ Matrix
 \rightarrow wir nutzen die
obere Dreiecksmatrix
 \rightarrow Notation: [Zeile, Spalte]

$N \rightarrow I \mid \text{elephant} \mid \text{pajamas}$

$V \rightarrow \text{shot} \quad P \rightarrow \text{in}$

$DET \rightarrow \text{an} \quad POSS \rightarrow \text{my}$

$NP \rightarrow DET N \mid POSS N \mid NP PP$

$PP \rightarrow P NP$

$S \rightarrow N VP \mid NP VP$

$VP \rightarrow V NP \mid VP PP$

	I	shot	an	elephant	in	my	pajamas
	0	1	2	3	4	5	6
0	N	/	/				
1		V	/	VP			
2			DET	NP			
3				N	/		
4					P	/	
5						POSS	NP
6							N

Schritt 2 bis Schritt n allgemein:

fülle jeweils die nächste Diagonale aus
trage A in Zelle $[i,j]$ ein, falls

- es eine Regel $A \rightarrow B C$ gibt und
- B in Zelle $[i,k]$ und
- C in Zelle $[k+1,i]$ $i \leq k < i$

CYK: Algorithmus

Datenstruktur

Satzlänge $n=7 \rightarrow 7 \times 7$ Matrix
 \rightarrow wir nutzen die obere Dreiecksmatrix
 \rightarrow Notation: [Zeile, Spalte]

$N \rightarrow I \mid \text{ele}$
 $V \rightarrow \text{shot}$
 $\text{DET} \rightarrow \text{an}$

Recognize/Erkennen:
Satz ist in der Sprache
wenn $S \in [0, n-1]$

$\text{NP} \rightarrow \text{DET } N \mid \text{POSS } N \mid \text{NP } \text{PP}$
 $\text{PP} \rightarrow P \text{ NP}$
 $S \rightarrow N \text{ VP} \mid \text{NP } \text{VP}$
 $\text{VP} \rightarrow V \text{ NP} \mid \text{VP } \text{PP}$

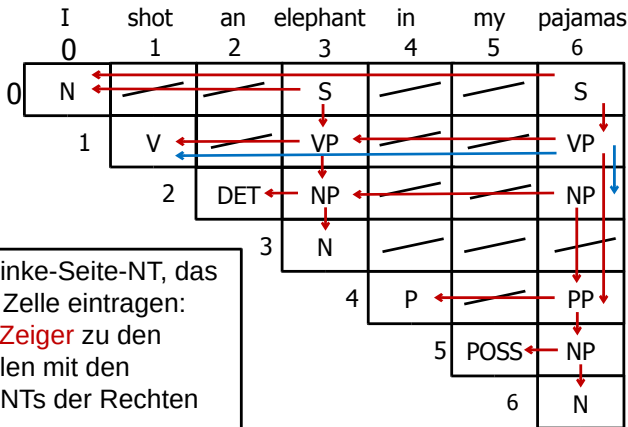
	I	shot	an	elephant	in	my	pajamas
	0	1	2	3	4	5	6
0	N	/	/	S	/	/	S
1		V	/	VP	/	/	VP
2			DET	NP	/	/	NP
3				N	/	/	/
4					P	/	PP
5						POSS	NP
6							N

Schritt 2 bis Schritt n allgemein:

fülle jeweils die nächste Diagonale aus
trage A in Zelle $[i, j]$ ein, falls

- es eine Regel $A \rightarrow B C$ gibt und
- B in Zelle $[i, k]$ und
- C in Zelle $[k+1, j]$ $i \leq k < j$

CYK: Algorithmus – Zeiger zu rechten Seiten



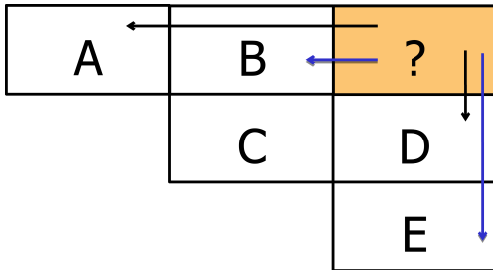
für jedes Linke-Seite-NT, das wir in eine Zelle eintragen: speichere **Zeiger** zu den beiden Zellen mit den jeweiligen NTs der Rechten Seite.

→ so können wir später die Parse-Bäume extrahieren, indem wir bei S in $[0, n-1]$

Bäume für dieses Beispiel:

an der Tafel

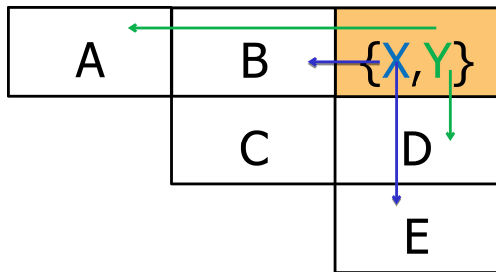
CYK: Mehrdeutigkeit in Zellen



Grammatik
 $X \rightarrow BE$
 $Y \rightarrow AD$

Eintrag für ?

CYK: Mehrdeutigkeit in Zellen



Grammatik

$X \rightarrow BE$

$Y \rightarrow AD$

Einträge in allen Zellen sind Mengen von Linke-Seite-Nichtterminalen (mit den jeweiligen Zeigern)

CYK Algorithmus: Eigenschaften

- **Korrekt:**

Wenn $S \in [0, n-1]$, dann $S \Rightarrow^* w_1 \dots w_n$
– n Tokens in der Eingabe

- **Vollständig:**

Wenn $S \Rightarrow^* w_1 \dots w_n$, dann $S \in [0, n-1]$

- **Laufzeit:**

Polynomiell in der Eingabelänge: $O(n^3)$

The CYK Parser

The CYK Membership Algorithm

Input:

- Grammar G in Chomsky Normal Form
- String w

Output:

find if $w \in L(G)$

The Algorithm

Input example:

• Grammar G : $S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

• String w : $aabbb$

aabbb

a a b b b

aa ab bb bb

aab abb bbb

aabb abbb

aabbb

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

a	a	b	b	b
A	A	B	B	B

aa	ab	bb	bb
----	----	----	----

aab	abb	bbb
-----	-----	-----

aabb	abbb
------	------

aabbb

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

a	a	b	b	b
A	A	B	B	B

aa	ab	bb	bb
	S,B	A	A

aab	abb	bbb
-----	-----	-----

aabb abbb

aabbb

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

a	a	b	b	b
A	A	B	B	B

aa	ab	bb	bb
	S,B	A	A

aab	abb	bbb
S,B	A	S,B

aabb	abbb
A	S,B

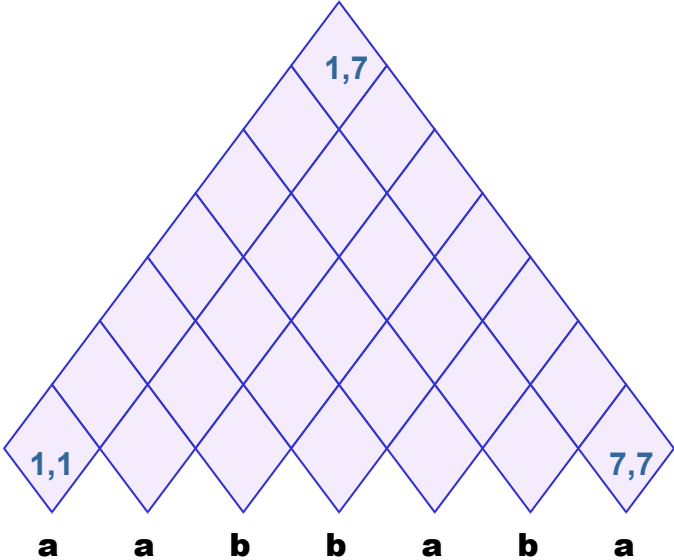
aabbb
S,B

Therefore: $aabbb \in L(G)$

Time Complexity: $|w|^3$

Observation: The CYK algorithm can be easily converted to a parser (bottom up parser)

The following slides are courtesy of
Professor Papp, University of Debrecen.



1,1

1,7

7,7

a

a

b

b

a

b

a

Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

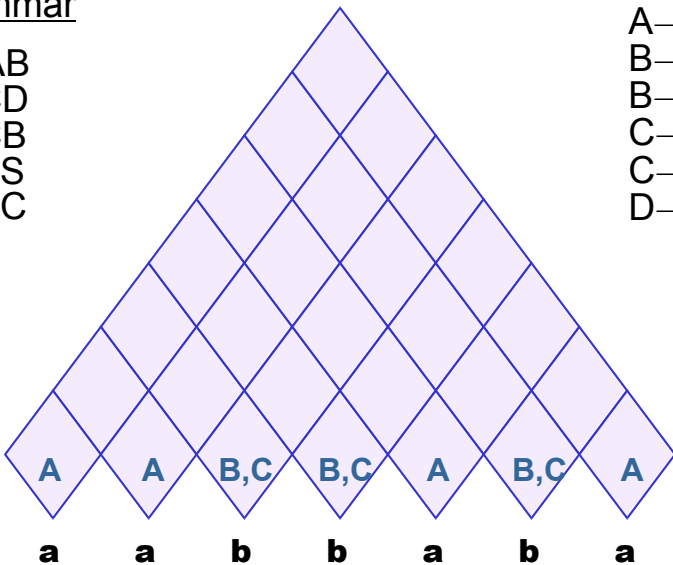
$B \rightarrow SC$

$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

$D \rightarrow BA$



Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

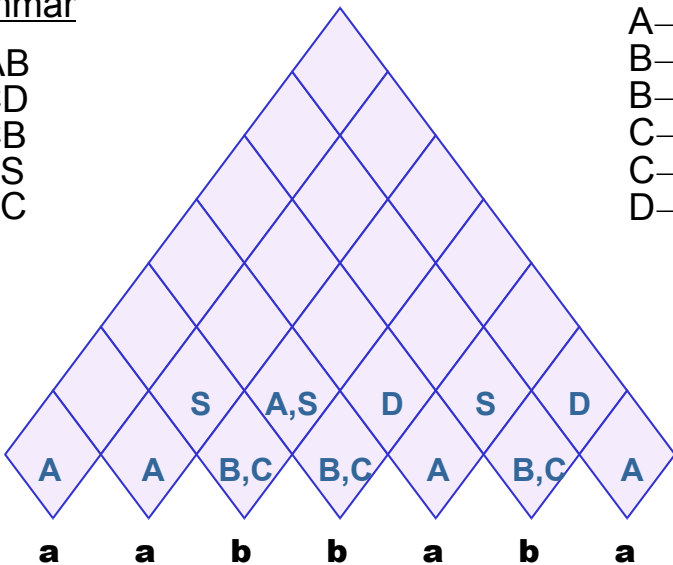
$B \rightarrow SC$

$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

$D \rightarrow BA$



Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

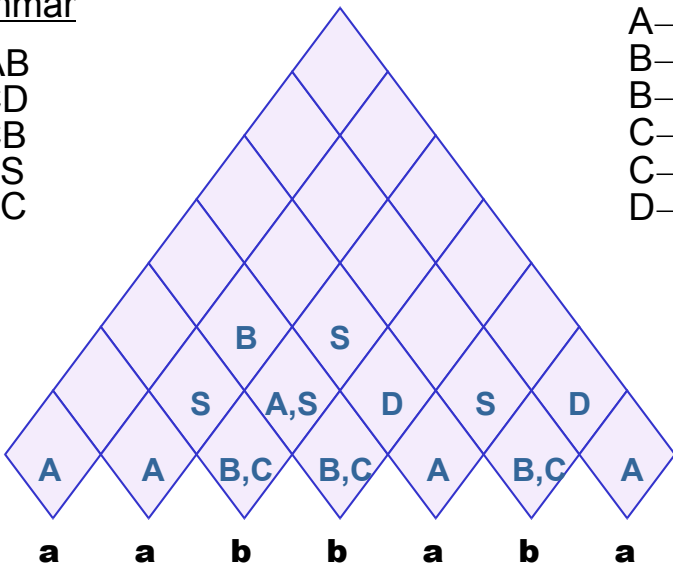
$B \rightarrow SC$

$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

$D \rightarrow BA$



Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

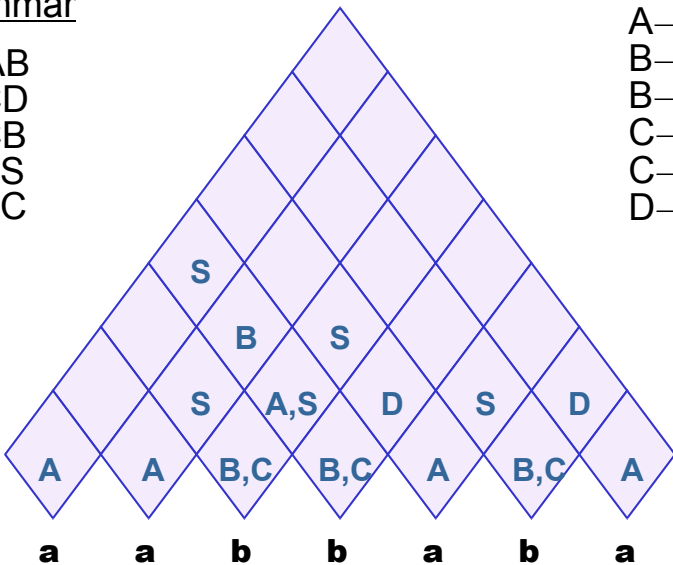
$B \rightarrow SC$

$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

$D \rightarrow BA$



Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

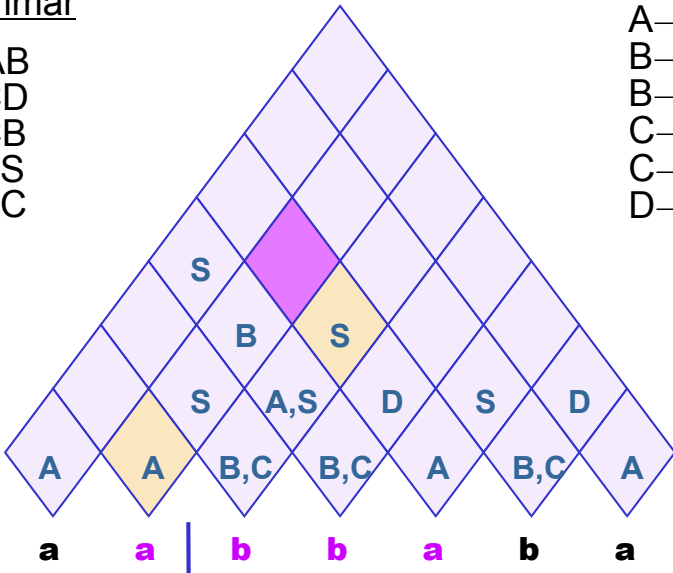
$B \rightarrow SC$

$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

$D \rightarrow BA$



Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

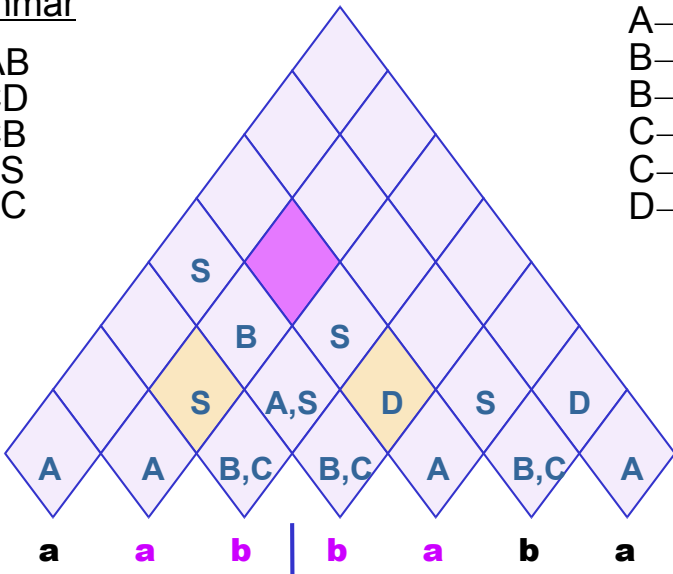
$B \rightarrow SC$

$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

$D \rightarrow BA$



Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

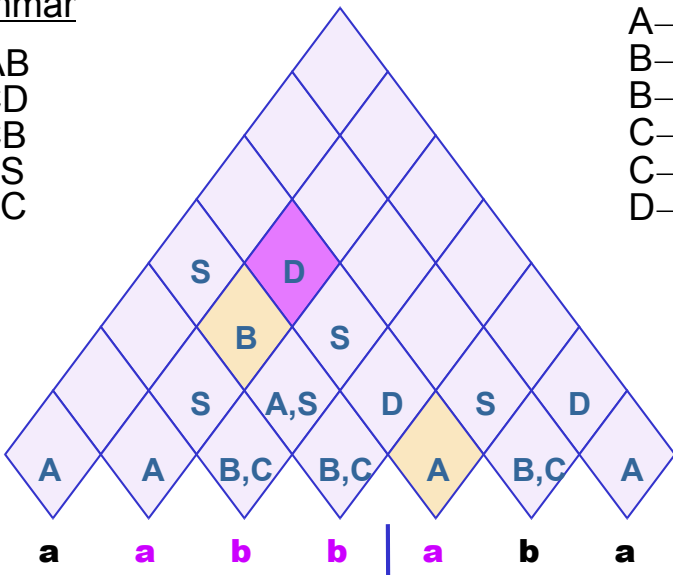
$B \rightarrow SC$

$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

$D \rightarrow BA$



Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

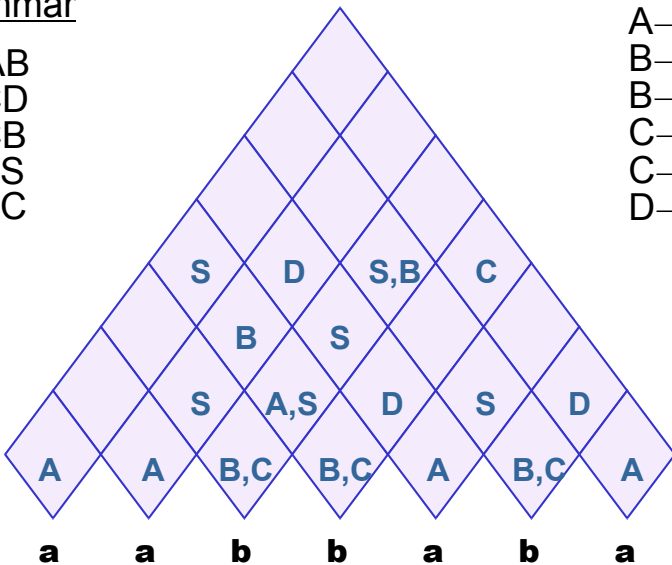
$B \rightarrow SC$

$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

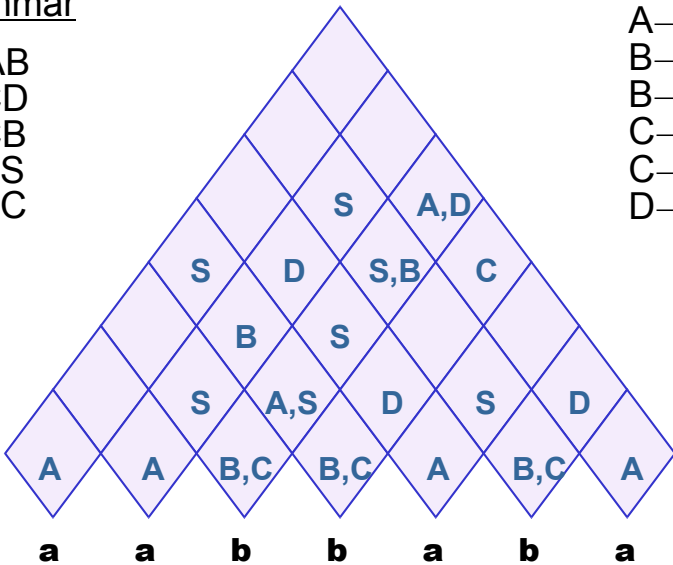
$D \rightarrow BA$



Grammar

$S \rightarrow AB$
 $S \rightarrow CD$
 $S \rightarrow CB$
 $S \rightarrow SS$
 $A \rightarrow BC$

$A \rightarrow a$
 $B \rightarrow SC$
 $B \rightarrow b$
 $C \rightarrow DD$
 $C \rightarrow b$
 $D \rightarrow BA$



Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

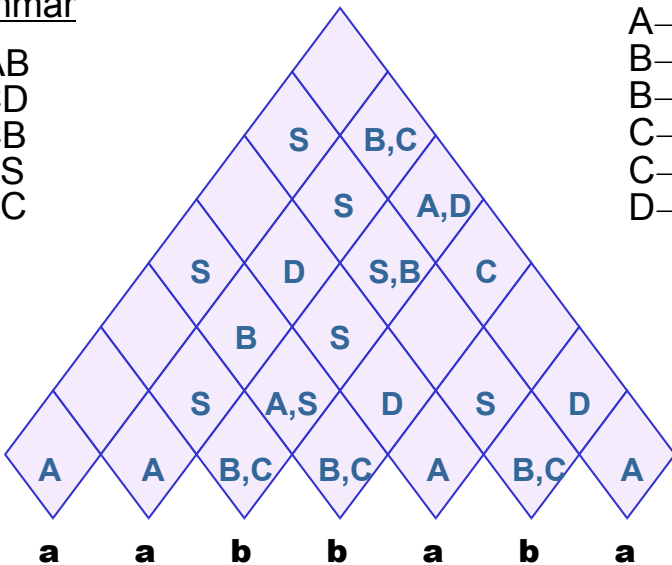
$B \rightarrow SC$

$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

$D \rightarrow BA$



Grammar

$S \rightarrow AB$

$S \rightarrow CD$

$S \rightarrow CB$

$S \rightarrow SS$

$A \rightarrow BC$

$A \rightarrow a$

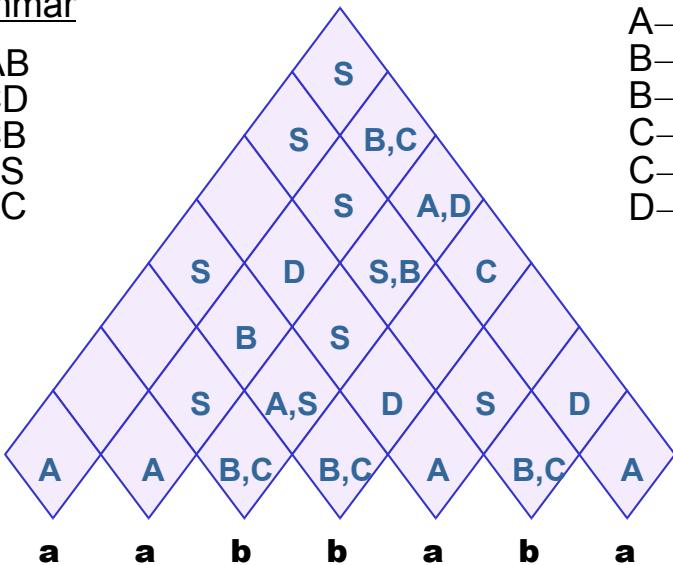
$B \rightarrow SC$

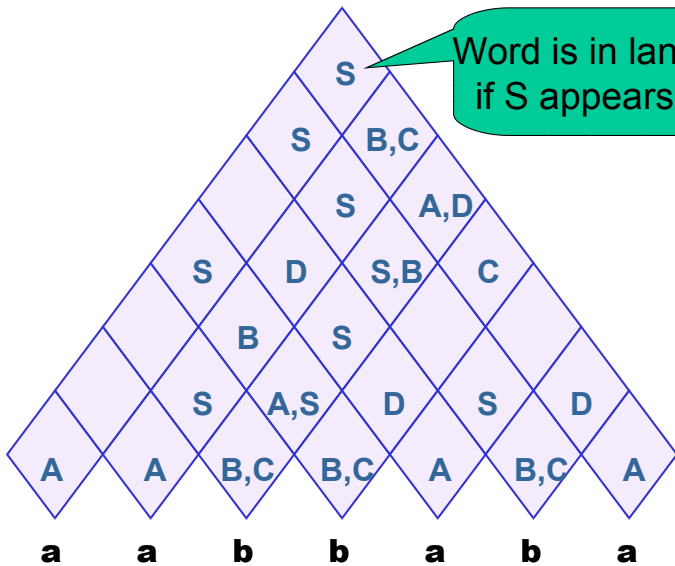
$B \rightarrow b$

$C \rightarrow DD$

$C \rightarrow b$

$D \rightarrow BA$





Word is in language if S appears here

Revisit complexity

exercise

Parse “baaba” for this grammar

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

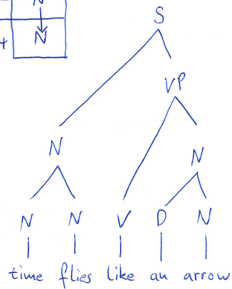
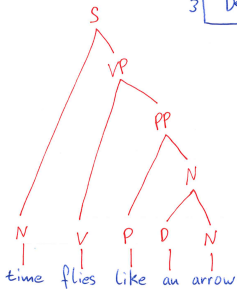
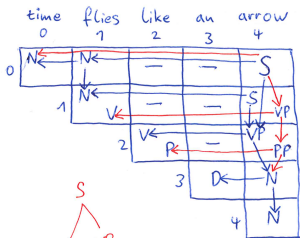
$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

(Hopcroft, Motwani, Ullman, p301)

$S \rightarrow N VP$
 $VP \rightarrow V N$
 $VP \rightarrow V PP$
 $N \rightarrow DN$
 $N \rightarrow NN$
 $PP \rightarrow PN$

$N \rightarrow \text{time}$
 $N \rightarrow \text{flies}$
 $N \rightarrow \text{like}$
 $V \rightarrow \text{flies}$
 $V \rightarrow \text{like}$
 $P \rightarrow \text{like}$
 $D \rightarrow \text{an}$



- Formale Definition CFG:
Terminale, Variablen, Startsymbol, Produktionen
- CYK
- Ableitungsbäume