

Einführung in die Computerlinguistik

Parsing

Dozentin: Wiebke Petersen

WS 2004/2005

Nachtrag zu den Abschlußeigenschaften kontextfreier Sprachen

Kontextfreie Sprachen sind abgeschlossen bezüglich der Vereinigung von Mengen: Seien $G_1 = (N_1, T_1, S_1, P_1)$ und $G_2 = (N_2, T_2, S_2, P_2)$ zwei kontextfreie Grammatiken, die die Sprachen $L(G_1)$ und $L(G_2)$ generieren, dann ist $G = (N_1 \uplus N_2 \cup \{S\}, T_1 \cup T_2, S, P)$ mit $P = P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}$ die Grammatik, die $L(G_1) \cup L(G_2)$ generiert ($N_1 \uplus N_2$ ist die disjunkte Vereinigung von N_1 und N_2).

Nachtrag zu den Abschlußeigenschaften kontextfreier Sprachen

Kontextfreie Sprachen sind abgeschlossen bezüglich der Vereinigung von Mengen: Seien $G_1 = (N_1, T_1, S_1, P_1)$ und $G_2 = (N_2, T_2, S_2, P_2)$ zwei kontextfreie Grammatiken, die die Sprachen $L(G_1)$ und $L(G_2)$ generieren, dann ist $G = (N_1 \uplus N_2 \cup \{S\}, T_1 \cup T_2, S, P)$ mit $P = P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}$ die Grammatik, die $L(G_1) \cup L(G_2)$ generiert ($N_1 \uplus N_2$ ist die disjunkte Vereinigung von N_1 und N_2).

$G_1 = \langle \{S_1, A\}, \{a, b\}, S_1, P_1 \rangle$	$G_2 = \langle \{S_2, A\}, \{a, b\}, S_2, P_2 \rangle$
$P_1 = \left\{ \begin{array}{l} S_1 \rightarrow AS_1b \\ S_1 \rightarrow \epsilon \\ A \rightarrow aa \end{array} \right\}$	$P_2 = \left\{ \begin{array}{l} S_2 \rightarrow bS_2A \\ S_2 \rightarrow \epsilon \\ A \rightarrow Aa \\ A \rightarrow a \end{array} \right\}$
$L(G_1) = a^{2n}b^n$	$L(G_2) = b^n a^m \text{ mit } m \geq n$

Nachtrag zu den Abschlußeigenschaften kontextfreier Sprachen

Kontextfreie Sprachen sind abgeschlossen bezüglich der Vereinigung von Mengen: Seien $G_1 = (N_1, T_1, S_1, P_1)$ und $G_2 = (N_2, T_2, S_2, P_2)$ zwei kontextfreie Grammatiken, die die Sprachen $L(G_1)$ und $L(G_2)$ generieren, dann ist $G = (N_1 \uplus N_2 \cup \{S\}, T_1 \cup T_2, S, P)$ mit $P = P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}$ die Grammatik, die $L(G_1) \cup L(G_2)$ generiert ($N_1 \uplus N_2$ ist die disjunkte Vereinigung von N_1 und N_2).

$$\begin{array}{c}
 G_1 = \langle \{S_1, A\}, \{a, b\}, S_1, P_1 \rangle \quad \Bigg| \quad G_2 = \langle \{S_2, A\}, \{a, b\}, S_2, P_2 \rangle \\
 \text{Beispiel: } P_1 = \left\{ \begin{array}{l} S_1 \rightarrow AS_1b \\ S_1 \rightarrow \epsilon \\ A \rightarrow aa \end{array} \right\} \quad \Bigg| \quad P_2 = \left\{ \begin{array}{l} S_2 \rightarrow bS_2A \\ S_2 \rightarrow \epsilon \\ A \rightarrow Aa \\ A \rightarrow a \end{array} \right\} \\
 \hline
 L(G_1) = a^{2n}b^n \quad \Bigg| \quad L(G_2) = b^n a^m \text{ mit } m \geq n
 \end{array}$$

Dann ist G eine Grammatik, die $L(G_1) \cup L(G_2)$ generiert:

$$G = \langle \{S, A_1, A_2, S_1, S_2\}, \{a, b\}, S, P \rangle \quad P = \left\{ \begin{array}{ll} S \rightarrow S_1 & S \rightarrow S_2 \\ S_1 \rightarrow A_1 S_1 b & S_1 \rightarrow \epsilon \\ A_1 \rightarrow aa & \\ S_2 \rightarrow b S_2 A_2 & S_2 \rightarrow \epsilon \\ A_2 \rightarrow A_2 a & A_2 \rightarrow a \end{array} \right\}$$

Kompositionalität kontextfreier Grammatiken

Vorsicht, auch wenn kontextfreie Sprachen bezüglich der Vereinigung von Mengen abgeschlossen sind, heißt das nicht, daß man Grammatiken ohne weiteres modular aufbauen und dann durch “Vereinigung” zu einer größeren zusammenfügen kann.

$$G_1 = \langle \{S, V, VP, EN\}, \{Mary, sings\}, S, P_1 \rangle$$

$$P_1 = \left\{ \begin{array}{l} S \rightarrow EN VP \\ VP \rightarrow V \\ V \rightarrow sings \\ EN \rightarrow Mary \end{array} \right\}$$

$$G_2 = \langle \{S, V, VP, EN\}, \{Mary, John, loves\}, S, P_2 \rangle$$

$$P_2 = \left\{ \begin{array}{l} S \rightarrow EN VP \\ VP \rightarrow V EN \\ V \rightarrow loves \\ EN \rightarrow Mary \\ EN \rightarrow John \end{array} \right\}$$

Parsing

- *to parse* (grammatisch zerlegen) abgeleitet von *pars* (griechisch) Teil
- Ein Parser ist ein Automat, der einer Zeichenkette aufgrund einer Grammatik einen Derivationsbaum zuordnet.

Grammatik
+
Zeichenkette → Derivationsbaum

Unterschied Recognizer – Parser

Beides sind Automaten

Recognizer: stellt ausschließlich fest, ob eine Zeichenfolge ein Wort der von der Grammatik generierten Sprache ist oder nicht (Keller-automat).

Parser: erstellt den Derivationsbaum einer Zeichenfolge im Bezug auf die Grammatik.

Parsingstrategien

Parsingstrategien unterscheiden sich darin, in welcher Reihenfolge die Knoten eines Derivationsbaums erstellt werden.

Man unterscheidet zwei Hauptstrategien voneinander

- **inputgetriebenes Parsing (bottom up)**
- **theoriegetriebenes Parsing (top down)**

Zusätzlich charakterisiert man Parsingstrategien mit folgenden Begriffen:

depth-first \leftrightarrow breadth-first

left-to-right \leftrightarrow right-to-left

top-down, left-to-right, depth-first Parser

top-down:

- Parser **beginnt beim Startsymbol** S und versucht, durch sukzessive Regelanwendung schließlich bei der Eingabekette zu landen.
- Regelanwendungen (von links nach rechts) nennt man **Expansion**.
- Das Einlesen eines Elements der Eingabekette nennt man **Scan**.

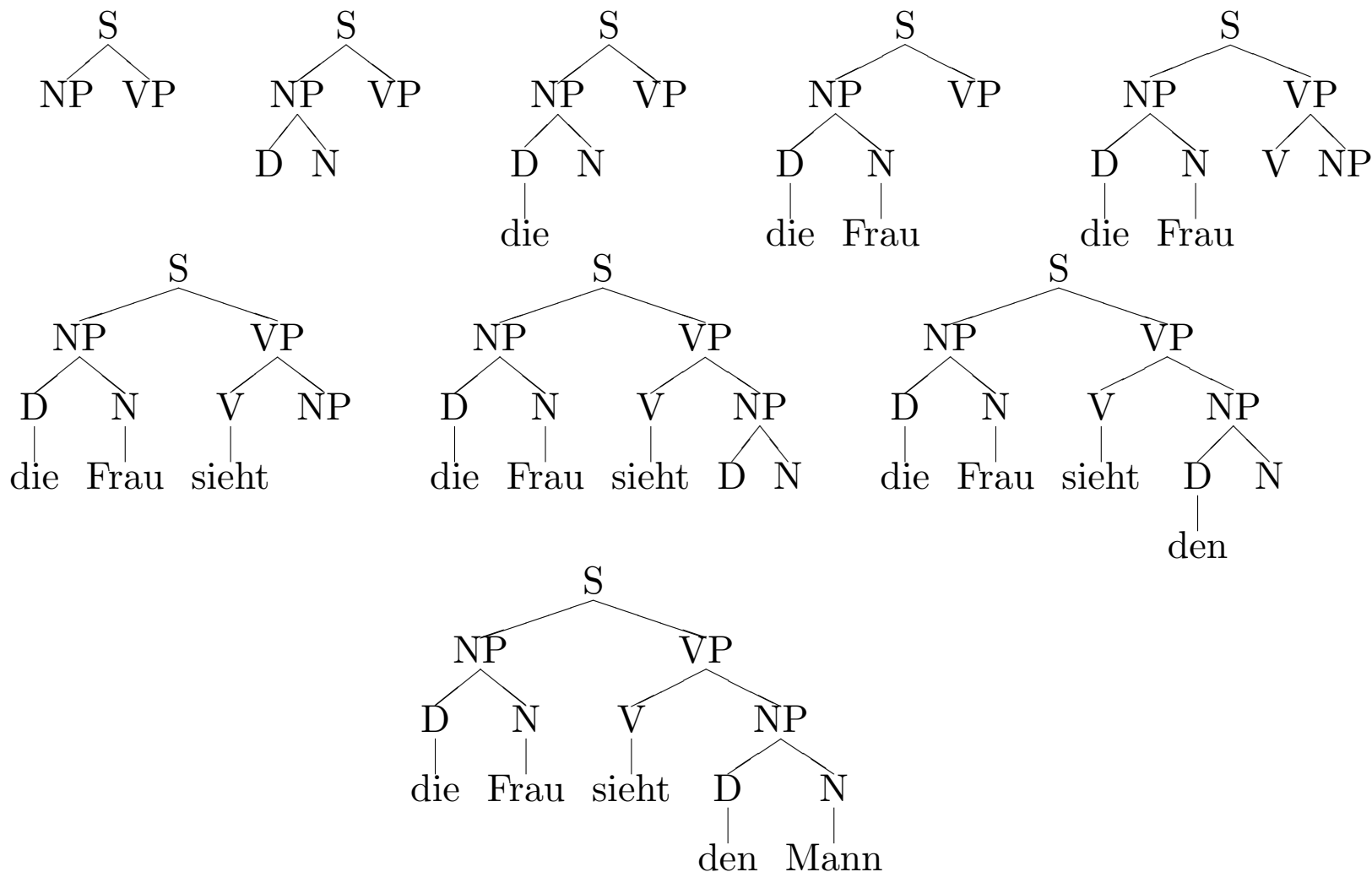
left-to-right: Der Parser versucht immer den am weitesten links stehenden Knoten des Ableitungsbaums zu expandieren oder mit diesem Knoten einen Scan durchzuführen.

depth-first: Der Parser versucht immer die am weitesten unten stehenden Knoten (das sind immer die zuletzt gebildeten) weiter zu expandieren oder hier einen Scan durchzuführen.

Beispiel:

$$P = \left\{ \begin{array}{l} S \rightarrow NP VP \quad VP \rightarrow V NP \quad NP \rightarrow D N \\ D \rightarrow \text{die} \quad D \rightarrow \text{den} \\ N \rightarrow \text{Frau} \quad N \rightarrow \text{Mann} \quad V \rightarrow \text{sieht} \end{array} \right\}$$

S



Linksrekursion

Top-down, left-to-right, depth-first Parser terminieren nicht bei Grammatiken, die linksrekursive Regeln beinhalten!

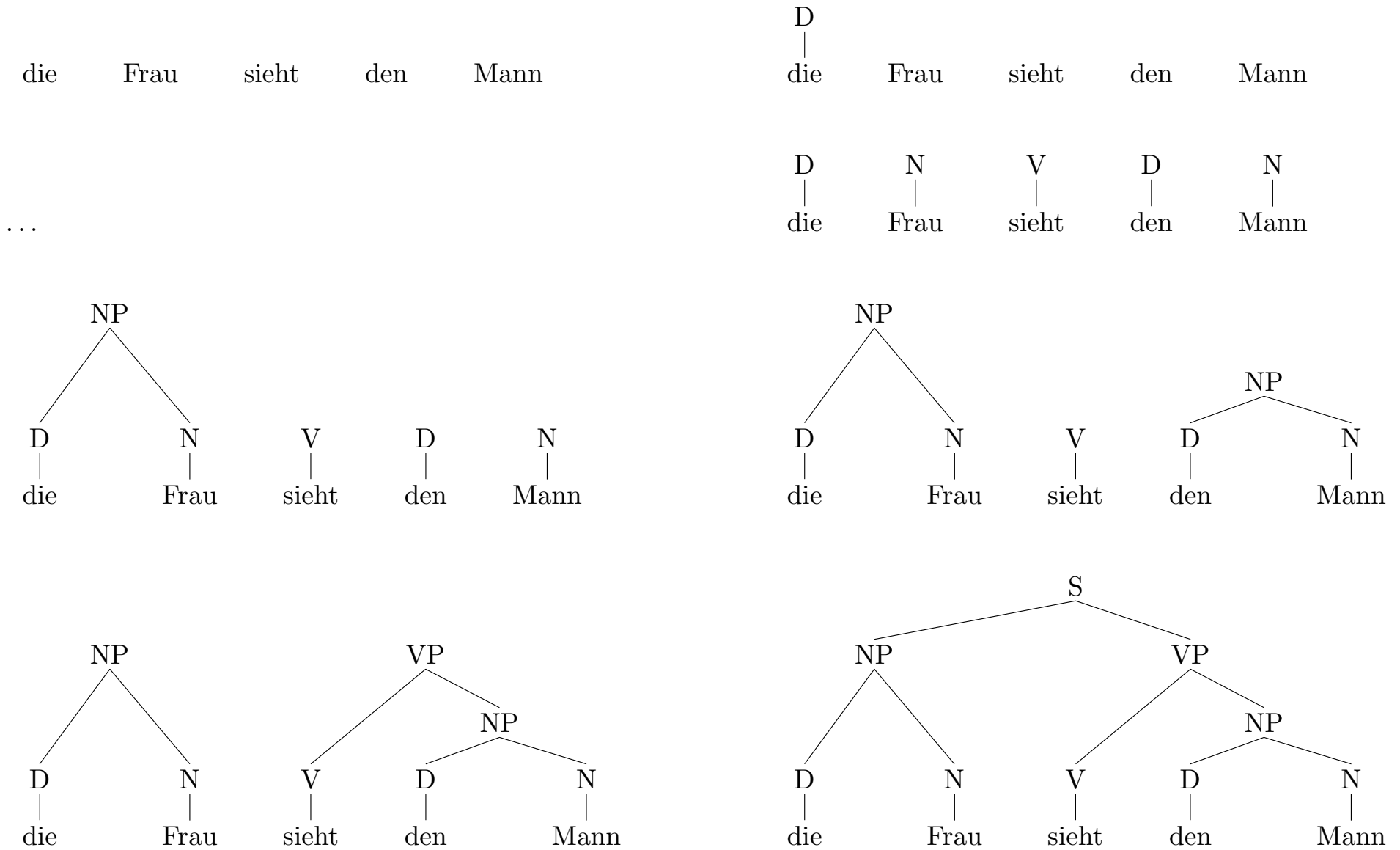
$$\begin{array}{l} S \rightarrow S \text{ und } S \\ NP \rightarrow NP \text{ PP} \\ T \rightarrow T \text{ O } T \end{array}$$

bottom-up, breadth-first, left-to-right Parser

bottom-up:

- Parser **beginnt bei der Eingabekette** und versucht, durch sukzessives **rückwärtiges** Anwenden der Regeln (von rechts nach links) schließlich bei dem Startsymbol S zu landen.

Beispiel:



ϵ -Regeln

Bottom-up, breadth-first, left-to-right Parser terminieren nicht bei Grammatiken, die ϵ -Regeln beinhalten, da eine solche Regel jederzeit anwendbar ist!

$$S \rightarrow \epsilon$$

(B.2) Backtracking

- Verschiedenen Aktionsmöglichkeiten des Parsers \Rightarrow Entscheidung
- Entscheidung kann falsch gewesen sein, d.h.
 - top-down: gewählte Regel führt nicht zur Ableitung der Eingabe
 - bottom-up: Eingabesatz kann nicht auf das Startsymbol reduziert werden
- Falsche Entscheidung kann sich erst zu späterem Zeitpunkt bemerkbar machen
- Fehlerbehebung nach dem Prinzip des **Backtracking**: Alle Schritte, bis zu der letzten Stelle, an der eine Wahlmöglichkeit bestand, rückgängig machen und dort eine andere Wahl treffen

(B.2) Backtracking: Beispiel

Ableitung des Satzes *der Hund sieht die Katze* (top-down, links-nach-rechts, Tiefe-Zuerst mit Backtracking):

Nr.	Satzform	Eingabe	Schritt
1	S	<i>der Hund sieht die Katze</i>	
2	NP VP	<i>der Hund sieht die Katze</i>	E
3	Det N VP	<i>der Hund sieht die Katze</i>	E
4	<i>der</i> N VP	<i>der Hund sieht die Katze</i>	E
5	N VP	<i>Hund sieht die Katze</i>	S
6	<i>Hund</i> VP	<i>Hund sieht die Katze</i>	E
7	VP	<i>sieht die Katze</i>	S
8	V	<i>sieht die Katze</i>	E
9	<i>bellt</i>	<i>sieht die Katze</i>	E
8'	V	<i>sieht die Katze</i>	B zu 8
9'	<i>sieht</i>	<i>sieht die Katze</i>	E
7''	VP	<i>sieht die Katze</i>	B zu 7
8''	V NP	<i>sieht die Katze</i>	E
9''	<i>bellt</i> NP	<i>sieht die Katze</i>	E
8'''	V NP	<i>sieht die Katze</i>	B zu 8''
9'''	<i>sieht</i> NP	<i>sieht die Katze</i>	E
10''''	NP	<i>die Katze</i>	S
11''''	Det N	<i>die Katze</i>	E
12''''	<i>der</i> N	<i>die Katze</i>	E
11'''''	Det N	<i>die Katze</i>	B zu 11''''
12'''''	<i>die</i> N	<i>die Katze</i>	E
13'''''	N	<i>Katze</i>	S
14'''''	<i>Hund</i>	<i>Katze</i>	E
13''''''	N	<i>Katze</i>	B zu 13'''''
14''''''	<i>Katze</i>	<i>Katze</i>	E
15''''''			S, akzept.

(E = Expandieren, S = Scannen, B = Backtracken)

Bottom-up-Parsing

Referat von Mara Keune

05.05.2004

Bottom-up-Parsing

- Bottom-up: eine Analyserichtung
(Gegenteil: Top-down)
- Analyse von unten nach oben
- Startzustand: der zu analysierende Satz, bestehend aus terminalen (lexikalische) Kategorien
- ***datengesteuerte Verarbeitung***: Satz wird mit Hilfe von Syntaxregeln und Lexikoneinträgen in mehreren Schritten auf das Startsymbol (S) zurückgeführt

Bottom-up-Parsing

- **Reduktion** ausgehend vom Wort: Regeln der Syntax werden **von rechts nach links** angewendet, die Symbole der rechten Seite werden durch das Symbol der linken Seite ersetzt
Beispiel: $S \rightarrow NP VP$
Reduktion: $[NP VP] \rightarrow [S]$
- Zielzustand: Startsymbol (S)

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

die

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

die

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

DET
|
die

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

DET
|
die

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

DET
|

die

Sonne

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

DET
|

die

Sonne

Bottom-up-Parsing

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

DET
|

die

N
|

Sonne

Bottom-up-Parsing

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

DET
|

die

N
|

Sonne

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V

DET
|

die

N
|

Sonne

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

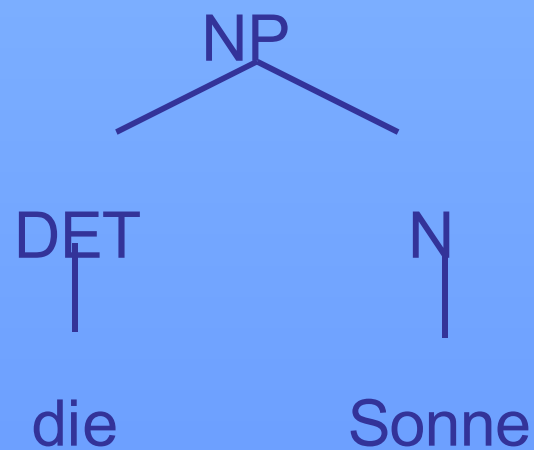
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

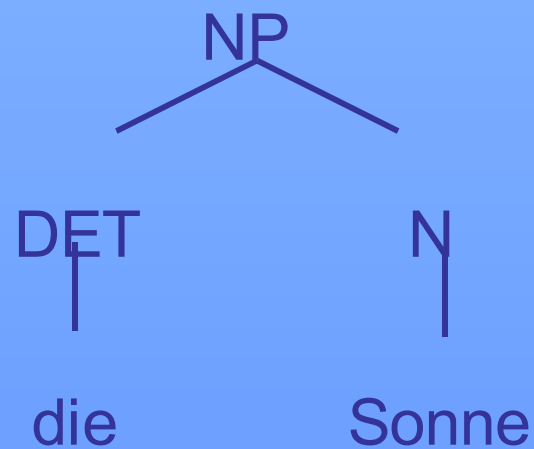
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

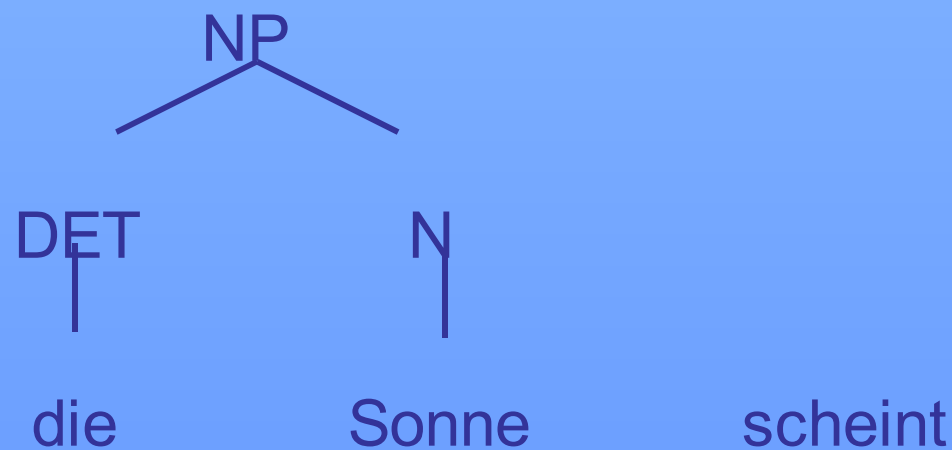
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

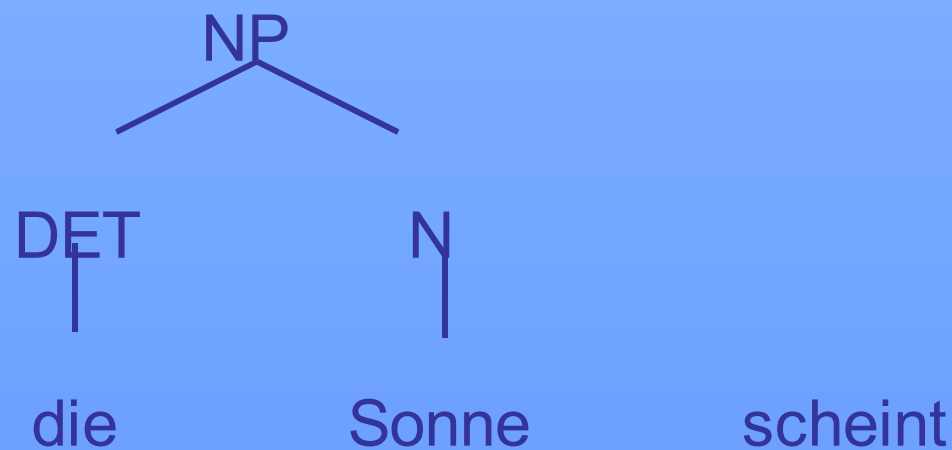
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

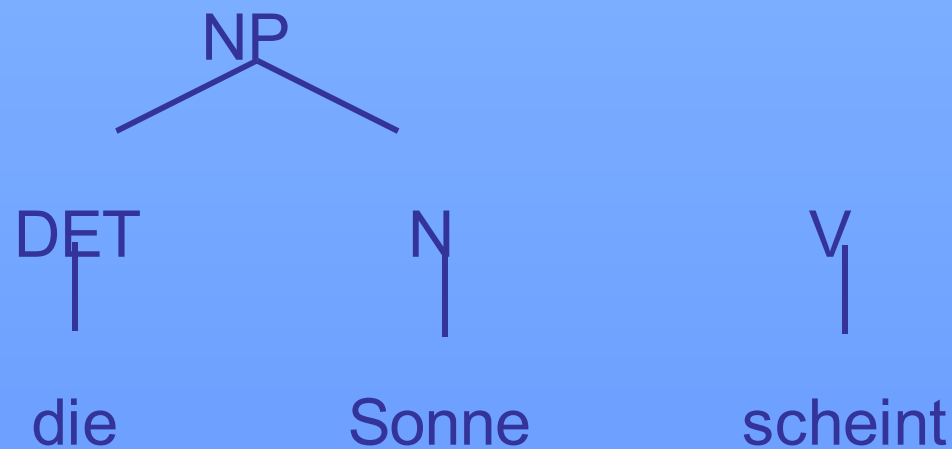
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

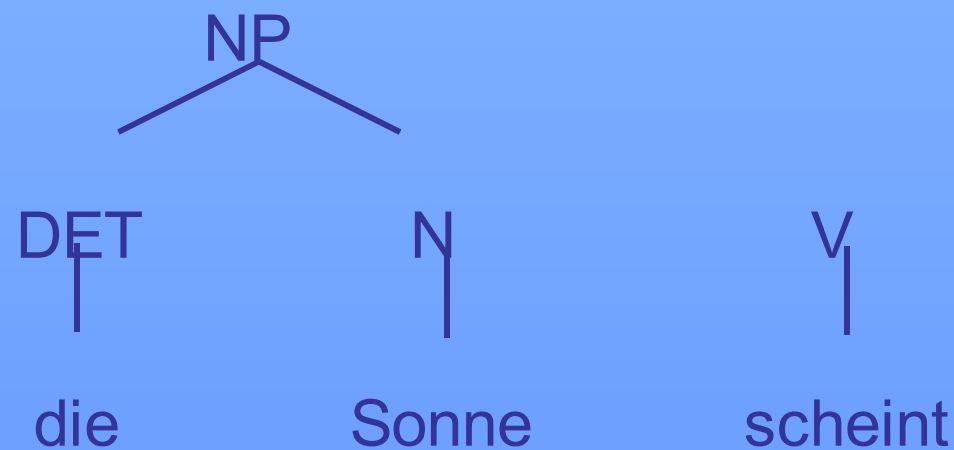
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

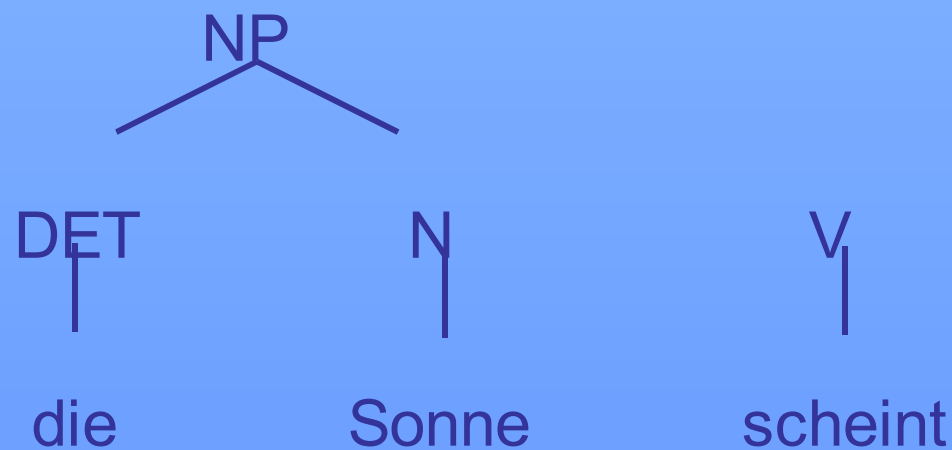
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

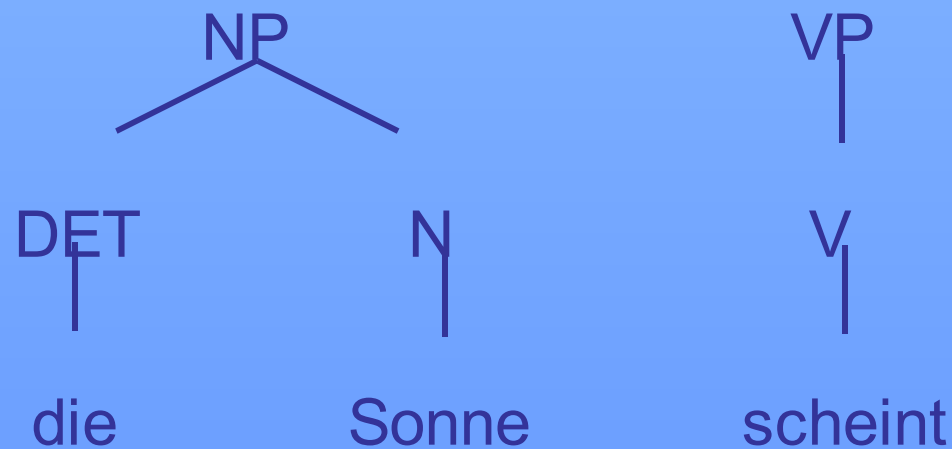
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

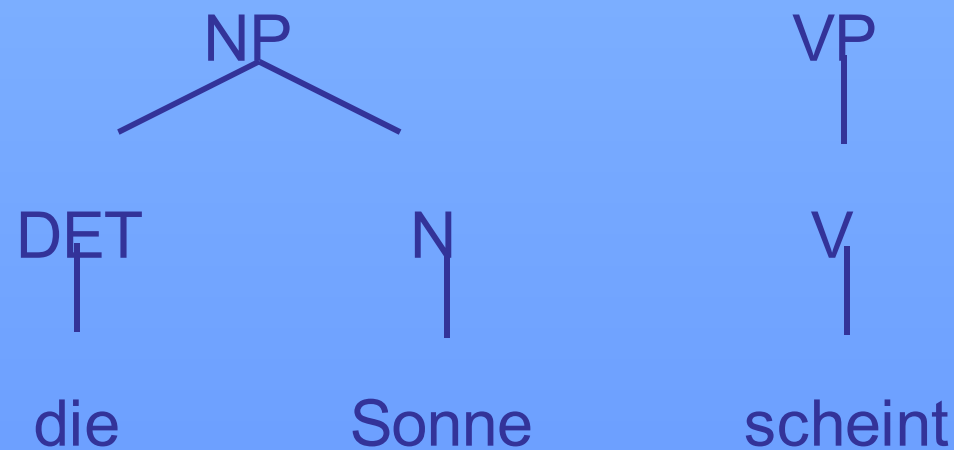
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

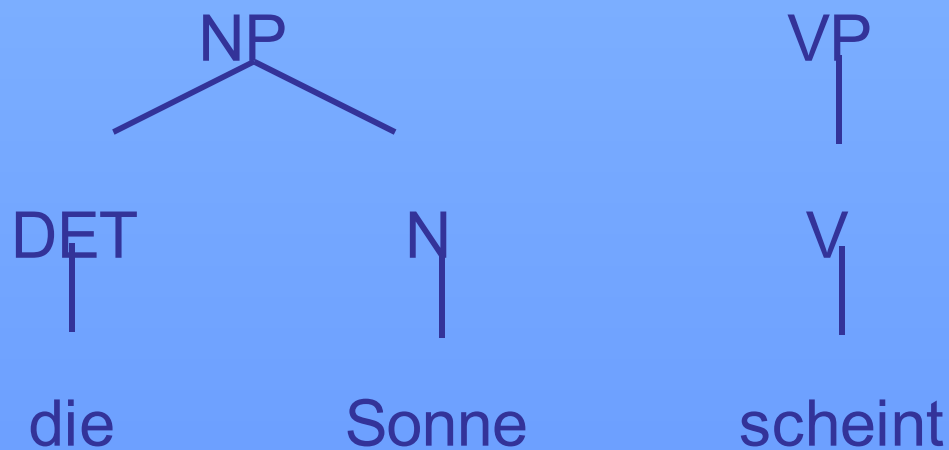
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

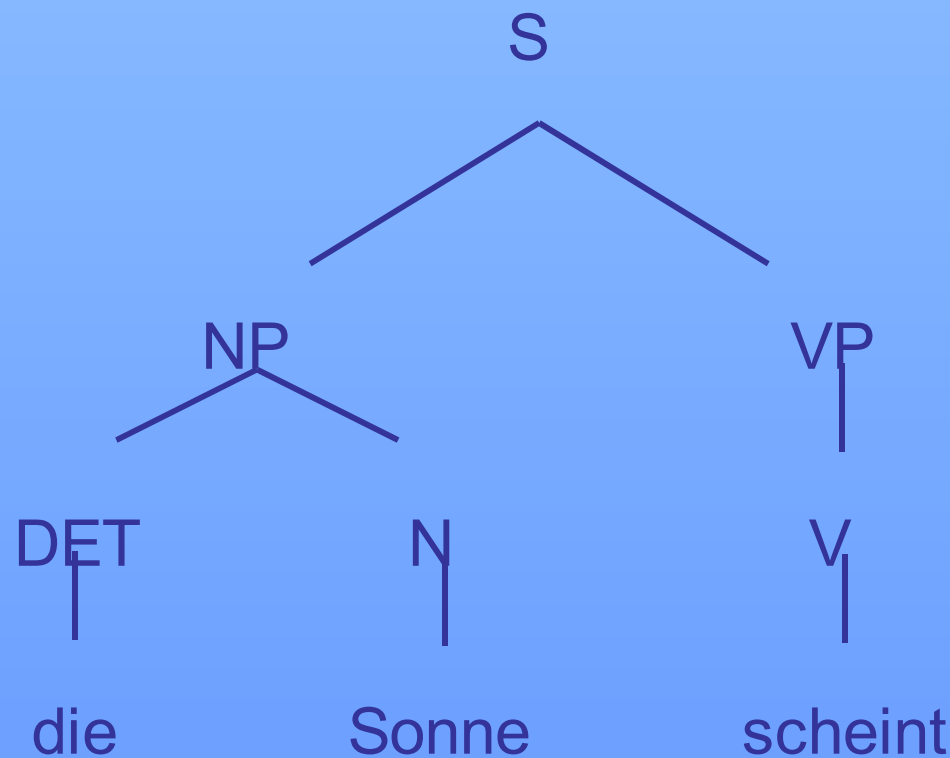
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Bottom-up-Parsing

Einfaches Beispiel: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$

$VP \rightarrow V$

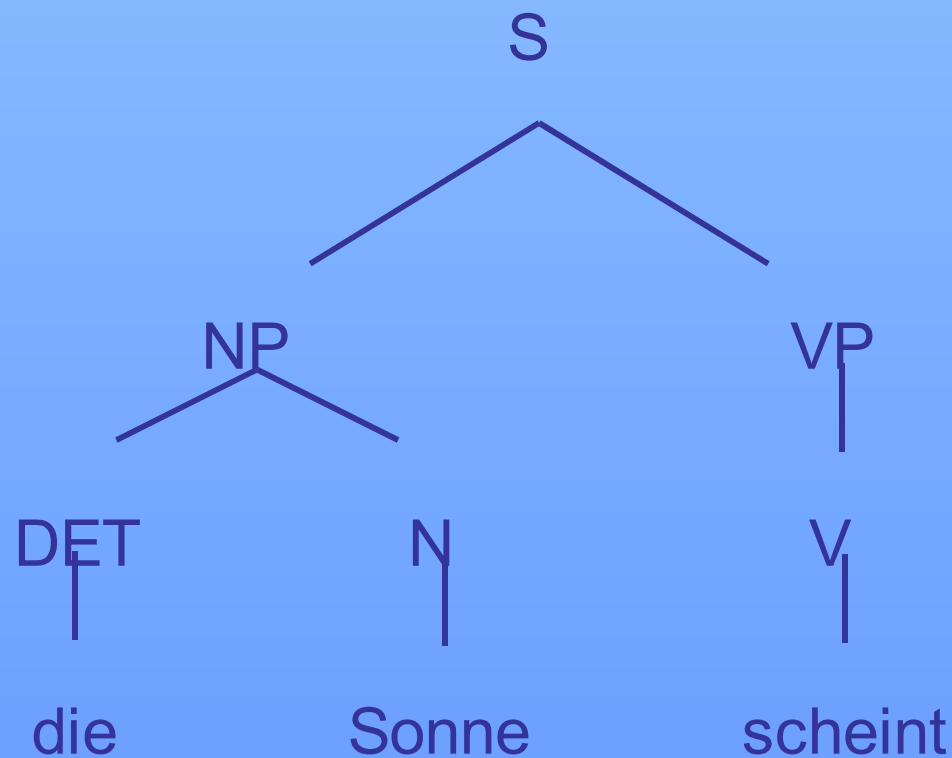
$NP \rightarrow DET N$

Lexikon:

die \in DET

Sonne \in N

scheint \in V



Bottom-up-Parsing

Unterschied Recognizer – Parser

Sind beide Automaten.

Recognizer

stellt ausschließlich fest, ob Satz in der Sprache möglich oder nicht möglich ist

Satz in der Sprache möglich → Ausgabe: TRUE

Satz in der Sprache nicht möglich → Ausgabe: FALSE

Parser

zusätzliches Erzeugen von Strukturbeschreibungen

Deterministischer Shift-Reduce-Recognizer

- Shift-Reduce: wichtigste Klasse der Bottom-up-Recognizer (bzw. -Parser)
- Verwendung von zwei Stacks
 1. Stack: noch nicht analysierter Satz(abschnitt)
 2. Stack: (Zwischen)Ergebnisse der Analyse
- **Operationen:**
 - Shift:** Wort wird von dem ersten auf den zweiten Stack übertragen, nur wenn Reduce nicht möglich
 - Reduce:** Inhalt des zweiten Stacks wird mit Hilfe einer Syntaxregel oder eines Lexikoneintrags reduziert

Deterministischer Shift-Reduce-Recognizer

Algorithmus

Wenn <Stack 2 nicht leer und Regel anwendbar>

Dann <*Reduce* (Stack 2)>

Sonst

 Wenn <Stack 1 nicht leer>

 Dann <*Shift* (Stack 1, Stack 2)>

 Sonst

 Wenn <Stack 2 == S>

 Dann <RETURN (True)>

 Sonst <RETURN (False)>

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation

Stack 1

Stack 2

Anfangszustand

[die, Sonne, scheint]

[]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

$die \in DET$ $Sonne \in N$ $scheint \in V$

Operation

Stack 1

Stack 2

Anfangszustand

[**die**, Sonne, scheint]

[]

Shift

[Sonne, scheint]

[**die**]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation

Stack 1

Stack 2

Anfangszustand

[die, Sonne, scheint]

[]

Shift

[Sonne, scheint]

[die]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in **DET** Sonne \in N scheint \in V

Operation

Stack 1

Stack 2

Anfangszustand

[die, Sonne, scheint]

[]

Shift

[Sonne, scheint]

[**die**]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

$die \in DET$ $Sonne \in N$ $scheint \in V$

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation

Stack 1

Stack 2

Anfangszustand

[die, Sonne, scheint]

[]

Shift

[Sonne, scheint]

[die]

Reduce

[Sonne, scheint]

[DET]

Shift

[scheint]

[Sonne, DET]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation

Stack 1

Stack 2

Anfangszustand

[die, Sonne, scheint]

[]

Shift

[Sonne, scheint]

[die]

Reduce

[Sonne, scheint]

[DET]

Shift

[scheint]

[Sonne, DET]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET **Sonne** \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne , DET]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET **Sonne** \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne , DET]
Reduce	[scheint]	[N , DET]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N **scheint** \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint , NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N **scheint** \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint , NP]
Reduce	[]	[V , NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]
Reduce	[]	[V, NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]
Reduce	[]	[V, NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]
Reduce	[]	[V, NP]
Reduce	[]	[VP, NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]
Reduce	[]	[V, NP]
Reduce	[]	[VP, NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]
Reduce	[]	[V, NP]
Reduce	[]	[VP, NP]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]
Reduce	[]	[V, NP]
Reduce	[]	[VP, NP]
Reduce	[]	[S]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]
Reduce	[]	[V, NP]
Reduce	[]	[VP, NP]
Reduce	[]	[S]

Beispiel Shift-Reduce-Recognizer: Die Sonne scheint.

Syntaxregeln:

$S \rightarrow NP VP$ $NP \rightarrow DET N$ $VP \rightarrow V$

Lexikon:

die \in DET Sonne \in N scheint \in V

Operation	Stack 1	Stack 2
Anfangszustand	[die, Sonne, scheint]	[]
Shift	[Sonne, scheint]	[die]
Reduce	[Sonne, scheint]	[DET]
Shift	[scheint]	[Sonne, DET]
Reduce	[scheint]	[N, DET]
Reduce	[scheint]	[NP]
Shift	[]	[scheint, NP]
Reduce	[]	[V, NP]
Reduce	[]	[VP, NP]
Reduce	[]	[S]

TRUE

Problem: Ambiguitäten

1. wenn verwendete Grammatik nicht deterministisch ist

Bsp.: $VP \rightarrow V$

$VP \rightarrow V NP$

2. wenn Lexikon Einträge enthält, die einem Wort verschiedene lexikalische Kategorien zuordnen

Bsp.: $\text{antworten} \in V$

$\text{antworten} \in N$

→ Algorithmus, der den richtigen Pfad auswählt wird benötigt

Problemlösung: Backtracking

Backtracking (Rücksetzen)

- ein Pfad wird so lange verfolgt, bis er sich als falsch erweist
- Verarbeitungsschritte bis zu der zuletzt getroffenen Entscheidung für einen bestimmten Pfad werden zurückgesetzt
- ein Alternativpfad wird gewählt
- jeder Syntaxregel und jedem Lexikoneintrag wird einer Zahl zugeordnet (Regel- bzw. Eintragsindex)
- 3. Stack wird implementiert, der Regel- und Eintragsindizes bzw. die Zahl 0 als Markierung für die Shift-Operation enthält

Shift-Reduce-Recognizer mit Backtracking

- Backtracking erforderlich, wenn in Situation weder Reduce noch Shift möglich und Endzustand noch nicht erreicht

Verschiedene Backtracking-Prozeduren

- nach Rücknahme einer Reduktion

Situation 1: es gibt eine noch nicht geprüfte Reduktionsmöglichkeit

→ andere Reduktionsmöglichkeit wird vorgenommen

Situation 2: es gibt keine noch nicht geprüfte Reduktionsmöglichkeit, aber ein neues Element kann geshiftet werden (Stack 1 enthält mind. 1 Element)

→ neues Element wird geshiftet

Shift-Reduce-Recognizer mit Backtracking

Situation 3: es gibt keine weitere Reduktionsmöglichkeit und keinen weiteren möglichen Shift

- die letzte Reduktion wird zurückgenommen
- mindestens ein weiteres Backtracking

- **nach Rücknahme eines Shifts:**

Situation 4: letzte Operation war ein Shift

- Shift-Operation wird zurückgenommen
- mindestens ein weiteres Backtracking

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation

Stack 1

Stack 2

Stack 3

Start

[c, e, a]

[]

[]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow$
N

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation

Stack 1

Stack 2

Stack 3

Start

[c, e, a]

[]

[]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$ 6: $N = \{\text{computer, antworten}\}$

Operation

Stack 1

Stack 2

Stack 3

Start

[c, e, a]

[]

[]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation

Stack 1

Stack 2

Stack 3

Start

[c, e, a]

[]

[]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation

Stack 1

Stack 2

Stack 3

Start

[c, e, a]

[]

[]

Shift

[e, a]

[c]

[0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation

Stack 1

Stack 2

Stack 3

Start

[c, e, a]

[]

[]

Shift

[e, a]

[c]

[0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation

Stack 1

Stack 2

Stack 3

Start

[c, e, a]

[]

[]

Shift

[e, a]

[c]

[0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow$
N

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$ 6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	[a, S]	[0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	[a, S]	[0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	[a , S]	[0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	R/R [a, S]	[0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	R/R [a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[V, S]	[5 ,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	R/R [a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[V, S]	[5,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	R/R [a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[V, S]	[5,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	R/R [a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[V, S]	[5,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	R/R [a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[V, S]	[5,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	R/R [a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[V, S]	[5,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	R/R [a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[V, S]	[5,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	R/R [a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[N, S]	[6,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	[a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[N, S]	[6,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	[a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[N, S]	[6,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	[a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[N, S]	[6,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	[a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[N, S]	[6,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]
Reduce	[a]	[VP, NP]	[2,5,0,4,6,0]
Reduce	[a]	[S]	[1,2,5,0,4,6,0]
Shift	[]	[a, S]	[0,1,2,5,0,4,6,0]
Reduce	[]	[N, S]	[6,0,1,2,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation

Stack 1

Stack 2

Stack 3

Start

[c, e, a]

[]

[]

Shift

[e, a]

[c]

[0]

Reduce

[e, a]

[N]

[6,0]

Reduce

[e, a]

[NP]

[4,6,0]

Shift

[a]

[e, NP]

[0,4,6,0]

Reduce

[a]

R/S [V, NP]

[5,0,4,6,0]

Reduce

[a]

[VP, NP]

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

Reduce

[a]

[S]

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

Shift

[]

[a, S]

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

Reduce

[]

[N, S]

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



Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$

2: $VP \rightarrow V$

3: $VP \rightarrow V NP$

4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	R/S [V, NP]	[5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	R/R [a, V, NP]	[0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	R/R [a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[V, V, NP]	[5,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	R/R [a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[V, V, NP]	[5,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	R/R [a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[V, V, NP]	[5,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	R/R [a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[V, V, NP]	[5,0,5,0,4,6,0]
Reduce	[]	[VP, V, NP]	[2,5,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	R/R [a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[V, V, NP]	[5,0,5,0,4,6,0]
Reduce	[]	[VP, V, NP]	[2,5,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	R/R [a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[V, V, NP]	[5,0,5,0,4,6,0]
Reduce	[]	[VP, V, NP]	[2,5,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation

Stack 1

Stack 2

Stack 3

Start

[c, e, a]

[]

[]

Shift

[e, a]

[c]

[0]

Reduce

[e, a]

[N]

[6,0]

Reduce

[e, a]

[NP]

[4,6,0]

Shift

[a]

[e, NP]

[0,4,6,0]

Reduce

[a]

[V, NP]

[5,0,4,6,0]

Shift

[]

R/R [a, V, NP] [0,5,0,4,6,0]

Reduce

[]

[V, V, NP]

[5,0,5,0,4,6,0]

Reduce

[]

[VP, V, NP]

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



Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	R/R [a, V, NP]	[0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	R/R [a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]
Reduce	[]	[NP, V, NP]	[4,6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]
Reduce	[]	[NP, V, NP]	[4,6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]
Reduce	[]	[NP, V, NP]	[4,6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]
Reduce	[]	[NP, V, NP]	[4,6,0,5,0,4,6,0]
Reduce	[]	[VP, NP]	[3,4,6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]
Reduce	[]	[NP, V, NP]	[4,6,0,5,0,4,6,0]
Reduce	[]	[VP, NP]	[3,4,6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]
Reduce	[]	[NP, V, NP]	[4,6,0,5,0,4,6,0]
Reduce	[]	[VP, NP]	[3,4,6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]
Reduce	[]	[NP, V, NP]	[4,6,0,5,0,4,6,0]
Reduce	[]	[VP, NP]	[3,4,6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]
Reduce	[]	[NP, V, NP]	[4,6,0,5,0,4,6,0]
Reduce	[]	[VP, NP]	[3,4,6,0,5,0,4,6,0]

Beispiel: Computer erzeugen Antworten.

Syntaxregeln:

1: $S \rightarrow NP VP$ 2: $VP \rightarrow V$ 3: $VP \rightarrow V NP$ 4: $NP \rightarrow N$

Lexikon:

5: $V = \{\text{antworten, erzeugen}\}$

6: $N = \{\text{computer, antworten}\}$

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0] TRUE
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP]	[4,6,0]
Shift	[a]	[e, NP]	[0,4,6,0]
Reduce	[a]	[V, NP]	[5,0,4,6,0]
Shift	[]	[a, V, NP]	[0,5,0,4,6,0]
Reduce	[]	[N, V, NP]	[6,0,5,0,4,6,0]
Reduce	[]	[NP, V, NP]	[4,6,0,5,0,4,6,0]
Reduce	[]	[VP, NP]	[3,4,6,0,5,0,4,6,0]

Vom Recognizer zum Parser

Möglichkeiten der Strukturbeschreibung

1. Möglichkeit:

die angewandten Regeln werden anschließend in umgekehrter Reihenfolge zur Konstruktion eines Phrasenstrukturbaums benutzt

2. Möglichkeit:

Stack 2 baut die Struktur parallel zur Ableitung auf und enthält nicht nur eine Ableitungszeile, sondern eine Liste von Strukturbeschreibungen

Vom Recognizer zum Parser

Stack 2 als Liste von Strukturbeschreibungen

Beispiel:

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP(N)]	[4,6,0]

Vom Recognizer zum Parser

Stack 2 als Folge von Listen

Beispiel:

Operation	Stack 1	Stack 2	Stack 3
Start	[c, e, a]	[]	[]
Shift	[e, a]	[c]	[0]
Reduce	[e, a]	[N]	[6,0]
Reduce	[e, a]	[NP(N)]	[4,6,0]
		...	
		[S(VP(NP(N)V)NP(N))]	

Literatur

- Allen, James 1995, *Natural Language Understanding*. Redwood City: Benjamin/Cummings, Kap. 3.
- Carstensen, Kai-Uwe et al. 2001, *Computerlinguistik und Sprachtechnologie*. Heidelberg/Berlin: Spektrum.
- Hellwig, Peter 1989, „Parsing natürlicher Sprache“, in: Bátori/Lenders/Putschke. *Computerlinguistik*. Handbücher zur Sprach- und Kommunikationswissenschaft. Band 4. Berlin: De Gruyter, S. 348-432.
- Naumann, Sven & Hagen Langer 1994, *Parsing. Eine Einführung in die maschinelle Analyse natürlicher Sprache*. Stuttgart: Teubner.

(B.3) Probleme beim Parsen

- Top-Down: Linksrekursion;
Bottom-Up: ϵ -Regeln

- Nichtdeterminismus

⇒ Echte und lokale Ambiguitäten müssen behandelt werden – z.B. durch Backtracking

⇒ Ineffizientes Arbeiten durch wiederholte Konstituentenanalyse