Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Sprachverarbeitung II / 2 FS 2017

Wortanalyse: Automaten und Parsing

Buch: Kapitel 6.3

Beat Pfister

Sprachverarbeitung II / 2

Vorlesung: **Sprachsynthese** (Teil II.2)

Grundlagen für die Anwendung linguist. Wissens >>>
(insbesondere für die Transkription)

- das Wortproblem
- Lösung für Typ-3-Sprachen
- Lösung für Typ-2-Sprachen

Übung: formale Sprachen

Das Wortproblem

Gegeben: Grammatik G, welche die Sprache $L(G)$ beschreibt

Frage: Ist das Wort w_i in $L(G)$ enthalten?

Problem: Frage ist nicht für beliebige G entscheidbar!

$\longrightarrow$ Sprachhierarchie nach Chomsky

Generelle Lösung des Wortproblems

Überlegung: Da es keine praktisch relevanten Sprachen gibt, die nicht zur Menge $\mathcal{L}_1$ gehören, reicht die Lösung des Wortproblems für Typ-1-Sprachen aus.

Behauptung: Für Typ-1-Sprachen gibt es einen Algorithmus, der das Wortproblem für jedes konkrete Wort w löst.

Begründung: $w \in L(G_1)$, falls $w \in U$ mit $U = \{u_i \mid S \Rightarrow^* u_i, |u_i| \leq |w|\}$

U ist begrenzt und in endlich vielen Schritten ableitbar, weil:

- konkretes Wort w gegeben $\longrightarrow |w|$ ist endlich
- Grammatik monoton $\longrightarrow$ alle $|u_i| \leq |w| \longrightarrow U$ ist begrenzt
- Menge der Regeln beschränkt $\longrightarrow$ alle Ableitungsfolgen $S \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_n$ mit $|u_n| \leq |w|$ sind endlich

Wortproblem-Algorithmus für Typ-1-Sprachen

INPUT: $w \in V_T^*$ und eine monotone Grammatik $G_1 = (V_N, V_T, P, S)$

OUTPUT: $w \in L(G_1)$ oder $w \notin L(G_1)$

BEGIN

$U := \{S\}$ (Menge der abgeleiteten Wörter)

REPEAT

$U := U \cup \{u_2 \mid u_1 \in U,\ u_1 \overset{r_i}{\Rightarrow} u_2,\ |u_2| \leq |w|\}$

UNTIL keine Veränderung

IF $w \in U$

THEN Ausgabe: $w \in L(G_1)$

ELSE Ausgabe: $w \notin L(G_1)$

END

END.

Erweiterung des Algorithmus

Es interessiert nicht nur eine ja/nein-Antwort, sondern vor allem auch die Ableitungsfolge. Nötige Erweiterung des Algorithmus:

- Für jedes erzeugte u_2 mit $u_1 \overset{r_i}{\Rightarrow} u_2$ das Tripel (u_2, u_1, r_i) in die Menge U aufnehmen

- Dann kann für $u_2 = w$ die Ableitungsfolge von hinten beginnend aufgeschrieben werden.

Wortproblem für Typ-1-Sprachen gelöst, . . .

. . . aber nur theoretisch, da Lösung praktisch nicht einsetzbar ist !

Grund: Menge U ist zwar begrenzt, kann aber sehr gross sein
(z.B. falls das Wort w ein Java-Programm ist).

- ♢ erforderliche Rechenleistung und
- ♢ benötigter Speicherplatz zu gross!

$\longrightarrow$ Spezielle Lösungen für Typ-2- und Typ-3-Sprachen!

Wortanalyse für Typ-3-Sprachen

Feststellung:	Grammatik G ist *erzeugendes* System für die Sprache $L(G)$
Frage:	Gibt es auch ein *analysierendes* System für $L(G)$ und wie erhält man es?
Antwort:	Jede Typ-3-Grammatik G_3 kann in einen endlichen Automaten A_E umgewandelt werden, wobei gilt: $L(A_E) = L(G_3)$.

Endlicher Automat ohne Ausgabe

Definiert durch ein Quintupel $A_E = (E, S_A, \delta, s_0, F)$ mit

$E = \{e_1, \ldots, e_n\}$	nichtleere Menge der Eingabesymbole
$S_A = \{s_0, \ldots, s_k\}$	nichtleere Menge der Zustände
$\delta : S_A \times E \mapsto S_A$	Überführungsfunktion für alle Zustände und Eingabesymbole
$s_0 \in S_A$	Anfangszustand
$F \subseteq S_A$	Menge der Endzustände

Anmerkung: Es gibt *deterministische* und *nichtdeterministische* Automaten, je nachdem, ob δ eindeutig ist oder nicht.

Algorithmus: Typ-3-Grammatik $\longrightarrow$ endl. Automat

`INPUT:` Rechtslineare Grammatik $G = (V_N, V_T, P, S)$ mit
Regeln der Formen: $Y \to xZ$, $Y \to x$ oder $Y \to \varepsilon$

`OUTPUT:` Endlicher Automat $A_E = (E, S_A, \delta, s_0, F)$ mit $L(A_E) = L(G)$

`BEGIN`

$E := V_T$	(Eingabealphabet)
$S_A := V_N \cup \{\Phi\}$	(Zustände)
$s_0 := S$	(Anfangszustand)
$F := \{Y \in V_N \mid (Y \to \varepsilon) \in P\} \cup \{\Phi\}$	(Endzustände)

`FOR EACH` Regel der Form $Y \to xZ$ (mit $Y, Z \in V_N$ und $x \in V_T$) `DO`
erweitere die Überführungsfunktion mit: $\delta(Y, x) \mapsto Z$
`END`
`FOR EACH` Regel der Form $Y \to x$ (mit $Y \in V_N$ und $x \in V_T$) `DO`
erweitere die Überführungsfunktion mit: $\delta(Y, x) \mapsto \Phi$
`END`
`END.`

Beispiel: Überführung $G_3 \longrightarrow A_E$

$L(G_3) = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$

rechtslineare Grammatik:

$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, S\}$

$P = \{\ S \to aS,$
$\quad S \to bA,$
$\quad A \to bS,$
$\quad A \to aA,$
$\quad A \to \varepsilon\ \}$

endlicher Automat:

$A_E = (E, S_A, \delta, s_0, F)$

Eingabealphabet: $E =$

Zustandsmenge: $S_A =$

Startzustand: $s_0 =$

Endzustände: $F =$

Überführungsfunktion:

Beispiel: Überführung $G_3 \longrightarrow A_E$

$L(G_3) = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$

rechtslineare Grammatik:

$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, S\}$

$P = \{\ S \to aS,$
$\quad S \to bA,$
$\quad A \to bS,$
$\quad A \to aA,$
$\quad A \to \varepsilon\ \}$

endlicher Automat:

$A_E = (E, S_A, \delta, s_0, F)$

Eingabealphabet: $E = V_T = \{a, b\}$

Zustandsmenge: $S_A =$

Startzustand: $s_0 =$

Endzustände: $F =$

Überführungsfunktion:

Beispiel: Überführung $G_3 \longrightarrow A_E$

$L(G_3) = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$

rechtslineare Grammatik:

$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, S\}$

$P = \{\ S \to aS,$
$\quad S \to bA,$
$\quad A \to bS,$
$\quad A \to aA,$
$\quad A \to \varepsilon\ \}$

endlicher Automat:

$A_E = (E, S_A, \delta, s_0, F)$

Eingabealphabet: $E = V_T = \{a, b\}$

Zustandsmenge: $S_A = V_N \cup \Phi = \{S, A, \Phi\}$

Startzustand: $s_0 =$

Endzustände: $F =$

Überführungsfunktion:

Beispiel: Überführung $G_3 \longrightarrow A_E$

$L(G_3) = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$

rechtslineare Grammatik:

$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, S\}$

$P = \{\ S \to aS,$
$\quad S \to bA,$
$\quad A \to bS,$
$\quad A \to aA,$
$\quad A \to \varepsilon\ \}$

endlicher Automat:

$A_E = (E, S_A, \delta, s_0, F)$

Eingabealphabet: $E = V_T = \{a, b\}$

Zustandsmenge: $S_A = V_N \cup \Phi = \{S, A, \Phi\}$

Startzustand: $s_0 = S$

Endzustände: $F =$

Überführungsfunktion:

Beispiel: Überführung $G_3 \longrightarrow A_E$

$L(G_3) = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$

rechtslineare Grammatik:

$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, S\}$

$P = \{\ S \to aS,$
$\quad S \to bA,$
$\quad A \to bS,$
$\quad A \to aA,$
$\quad A \to \varepsilon\ \}$

endlicher Automat:

$A_E = (E, S_A, \delta, s_0, F)$

Eingabealphabet: $E = V_T = \{a, b\}$

Zustandsmenge: $S_A = V_N \cup \Phi = \{S, A, \Phi\}$

Startzustand: $s_0 = S$

Endzustände: $F = \{A, \Phi\}$

Überführungsfunktion:

Beispiel: Überführung $G_3 \longrightarrow A_E$

$L(G_3) = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$

rechtslineare Grammatik:

$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, S\}$

$P = \{\ S \to aS,$
$\quad S \to bA,$
$\quad A \to bS,$
$\quad A \to aA,$
$\quad A \to \varepsilon\ \}$

endlicher Automat:

$A_E = (E, S_A, \delta, s_0, F)$

Eingabealphabet:	$E = V_T = \{a, b\}$
Zustandsmenge:	$S_A = V_N \cup \Phi = \{S, A, \Phi\}$
Startzustand:	$s_0 = S$
Endzustände:	$F = \{A, \Phi\}$
Überführungsfunktion:	$\delta(S, a) \mapsto S$

Beispiel: Überführung $G_3 \longrightarrow A_E$

$L(G_3) = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$

rechtslineare Grammatik:

$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, S\}$

$P = \{\ S \to aS,$
$\quad S \to bA,$
$\quad A \to bS,$
$\quad A \to aA,$
$\quad A \to \varepsilon\ \}$

endlicher Automat:

$A_E = (E, S_A, \delta, s_0, F)$

Eingabealphabet:	$E = V_T = \{a, b\}$
Zustandsmenge:	$S_A = V_N \cup \Phi = \{S, A, \Phi\}$
Startzustand:	$s_0 = S$
Endzustände:	$F = \{A, \Phi\}$
Überführungsfunktion:	$\delta(S, a) \mapsto S$
	$\delta(S, b) \mapsto A$

Beispiel: Überführung $G_3 \longrightarrow A_E$

$L(G_3) = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$

rechtslineare Grammatik:

$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, S\}$

$P = \{\ S \to aS,$
$\quad S \to bA,$
$\quad A \to bS,$
$\quad A \to aA,$
$\quad A \to \varepsilon\ \}$

endlicher Automat:

$A_E = (E, S_A, \delta, s_0, F)$

Eingabealphabet:	$E = V_T = \{a, b\}$
Zustandsmenge:	$S_A = V_N \cup \Phi = \{S, A, \Phi\}$
Startzustand:	$s_0 = S$
Endzustände:	$F = \{A, \Phi\}$
Überführungsfunktion:	$\delta(S, a) \mapsto S$
	$\delta(S, b) \mapsto A$
	$\delta(A, b) \mapsto S$

Beispiel: Überführung $G_3 \longrightarrow A_E$

$L(G_3) = \{w \in \{a,b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$

rechtslineare Grammatik:

$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, S\}$

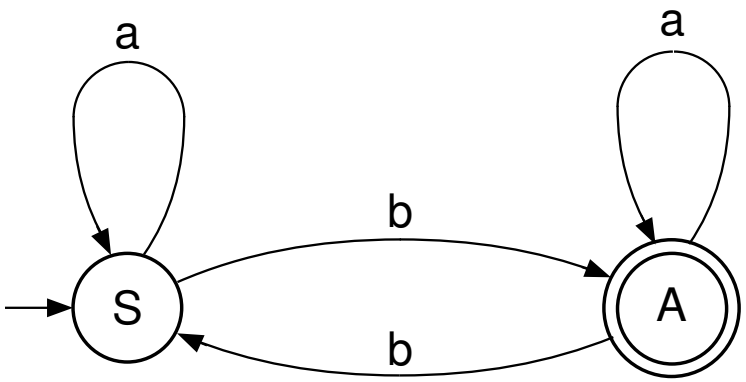
$P = \{\ S \to aS,$
$\quad S \to bA,$
$\quad A \to bS,$
$\quad A \to aA,$
$\quad A \to \varepsilon\ \}$

endlicher Automat:

$A_E = (E, S_A, \delta, s_0, F)$

Eingabealphabet:	$E = V_T = \{a, b\}$
Zustandsmenge:	$S_A = V_N \cup \Phi = \{S, A, \Phi\}$
Startzustand:	$s_0 = S$
Endzustände:	$F = \{A, \Phi\}$
Überführungsfunktion:	$\delta(S, a) \mapsto S$
	$\delta(S, b) \mapsto A$
	$\delta(A, b) \mapsto S$
	$\delta(A, a) \mapsto A$

Zustandsdiagramm und Zustandstabelle



	a	b
S	S	A
A	A	S

(äquivalente Darstellungen, abgesehen von den Start- und Endzuständen)

Wortanalyse mit endlichem Automaten

Deterministischer endlicher Automat:

A_{DE} benötigt $n = |w|$ Schritte um

a) zu entscheiden, ob $w \in L(A_E)$ und

b) eventuell die Ableitungsfolge zu ermitteln

Nichtdeterministischer endlicher Automat:

A_{NE} benötigt n Schritte, wobei i.a. $n \gg |w|$

Beispiel: Überführung $G_3 \longrightarrow A_{NE}$

$L(G_3) = \{w \in \{a,b\}^* \mid$ das letzte Zeichen ist vorher schon einmal vorgekommen$\}$

rechtslineare Grammatik:

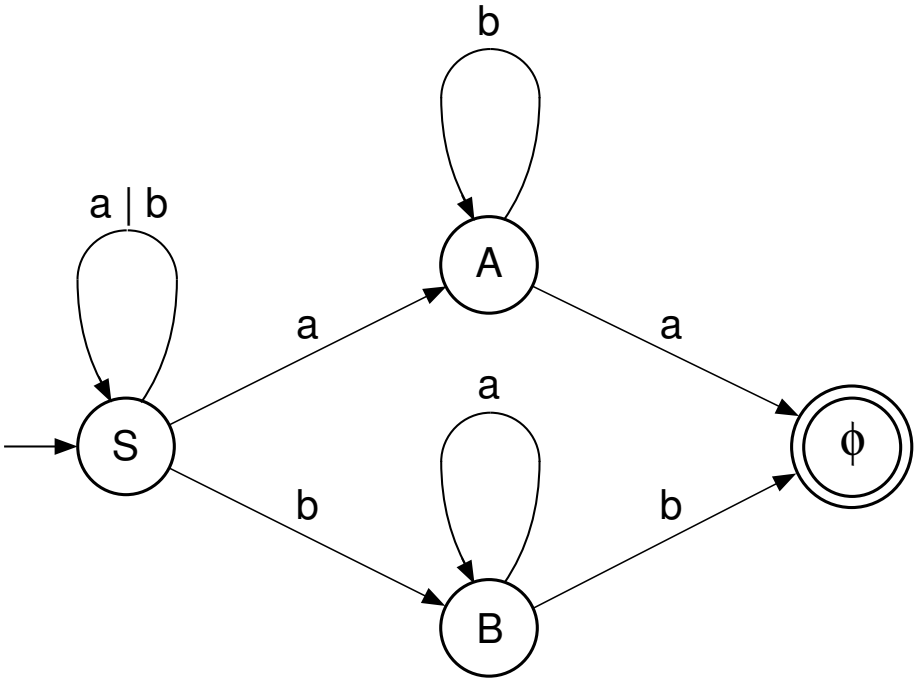
$G_3 = (V_N, V_T, P, S)$

$V_T = \{a, b\}$

$V_N = \{A, B, S\}$

$P = \{\ S \to aS,$
$\quad S \to bS,$
$\quad S \to aA,$
$\quad S \to bB,$
$\quad A \to bA,$
$\quad A \to a,$
$\quad B \to aB,$
$\quad B \to b\ \}$

nichtdeterministischer endlicher Automat: >>>



Nichtdeterministische endliche Automaten

Problem: Wortanalyse mit A_{NE} benötigt i.a. $n \gg |w|$ Schritte
und eine komplizierte Verwaltung der möglichen Zustandsfolgen
$\longrightarrow$ unpraktisch

Satz: Zu jedem nichtdeterministischen endlichen Automaten A_{NE}
lässt sich ein deterministischer endlicher Automat A_{DE}
angeben mit $L(A_{NE}) = L(A_{DE})$

Lösung: Umwandlung $A_{NE} \longrightarrow A_{DE}$

Algorithmus: $A_{NE} \longrightarrow A_{DE}$

Prinzip: Teilmengen der Zustände $\{s_0, s_1, s_2, \ldots\}$ von A_{NE} so zu Zuständen von A_{DE} zusammenfassen, dass δ eindeutig wird.

Vorgehen: Aufbau der Zustandstabelle von A_{DE} mit $S_{A_{DE}} = \{S_0, S_1, S_2, \ldots\}$:

a) Anfangszustand S_0 von A_{DE}: s_0 von A_{NE}

b) in der Zeile mit S_j für jedes Element $e_i \in E$ die Menge S_k in der Kolonne e_i eintragen, also $\delta(S_j, e_i) \mapsto S_k$

c) alle noch nicht in der ersten Kolonne stehenden S_k dort in je eine neue Zeile eintragen

d) für alle neu eingetragenen S_k die Schritte b) und c) ausführen, bis keine neuen S_k mehr entstehen

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

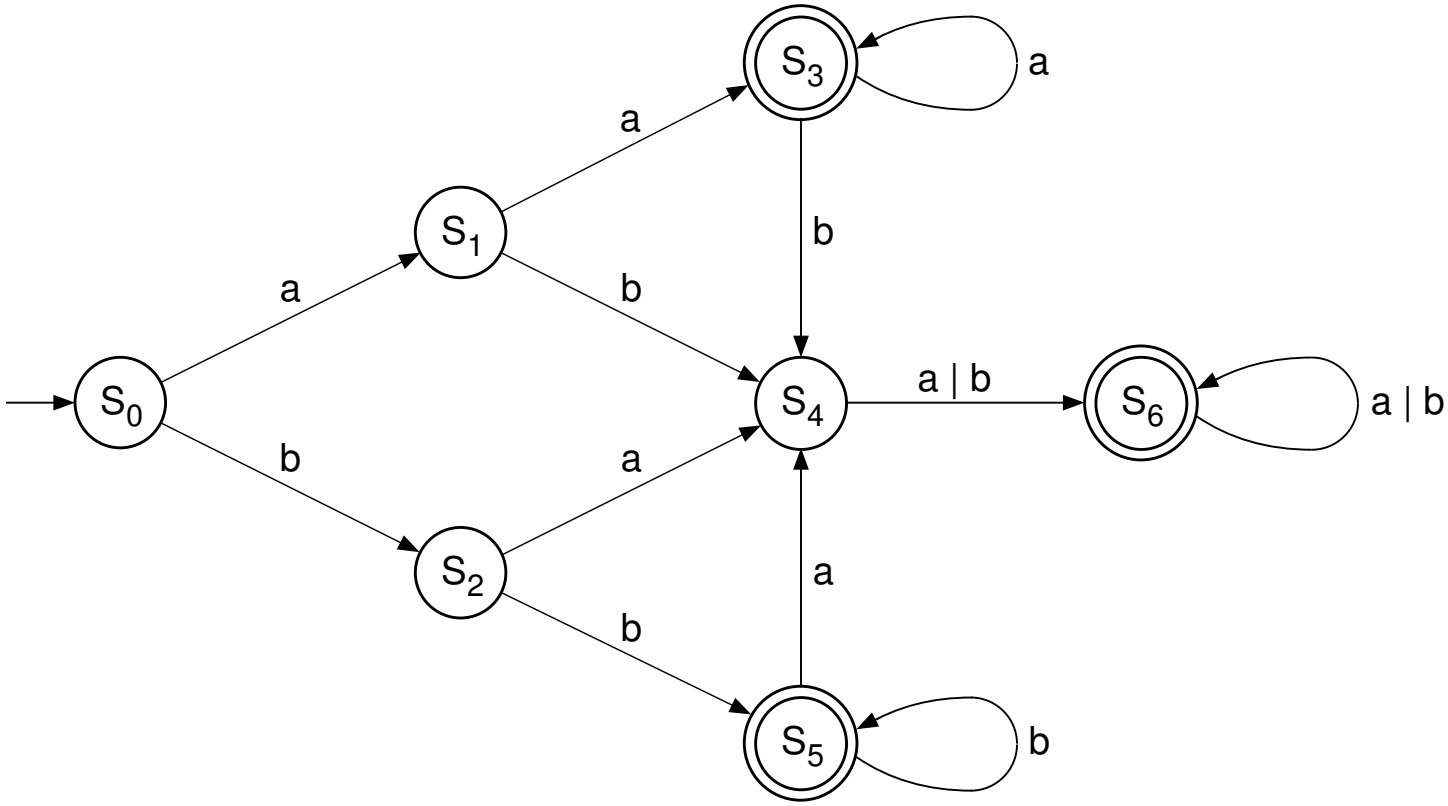
A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 := \{S, A\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_2 := \{S, B\}$	$\{S, A, B\}$	$\{S, B, \Phi\}$
$S_3 := \{S, A, \Phi\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_4 := \{S, A, B\}$	$\{S, A, B, \Phi\}$	$\{S, A, B, \Phi\}$
$S_5 := \{S, B, \Phi\}$	$\{S, A, B\}$	$\{S, B, \Phi\}$
$S_6 := \{S, A, B, \Phi\}$	$\{S, A, B, \Phi\}$	$\{S, A, B, \Phi\}$

Deterministischer Automat A_{NE}

$L(A_{NE}) = \{w \in \{a,b\}^* \mid$ das letzte Zeichen ist vorher schon einmal vorgekommen$\}$

	a	b
S_0	S_1	S_2
S_1	S_4	S_3
S_2	S_3	S_5
S_3	S_6	S_6
S_4	S_4	S_3
S_5	S_3	S_5
S_6	S_6	S_6



Grammatiken und Automaten für komplexe Sprachen

Praxis: Sprachen sind oft relativ komplex

Folge: Schreiben einer Grammatik ist schwierig

Ausweg: Häufig ist eine Sprache L_k zerlegbar, so dass

a) $L_k = L_{s_1} \cap L_{s_2} \cap \ldots$ (Schnittmenge)

b) $L_k = L_{s_1} \cup L_{s_2} \cup \ldots$ (Vereinigung)

Automat für Sprache L_k

Gegeben: Sprache $L_k = \{w \in \{a,b\}^* \mid$ w enthält eine ungerade Anzahl b und das letzte Zeichen ist vorher schon einmal vorgekommen$\}$.

Gesucht: Automat A_{Ek} mit $L(A_{Ek}) = L_k$

Lösung: weil
- $L_k = L_{s_1} \cap L_{s_2} = L(A_{E1}) \cap L(A_{E2})$
- A_{E1} und A_{E2} bekannt

>>>

$\longrightarrow$ Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

Algorithmus: A_{E1} und $A_{E2} \longrightarrow A_{Ek}$

A_{Ek} ist Automat mit Zustandspaaren (X, Y) mit X Zustand von A_{E1}
und Y Zustand von A_{E2}

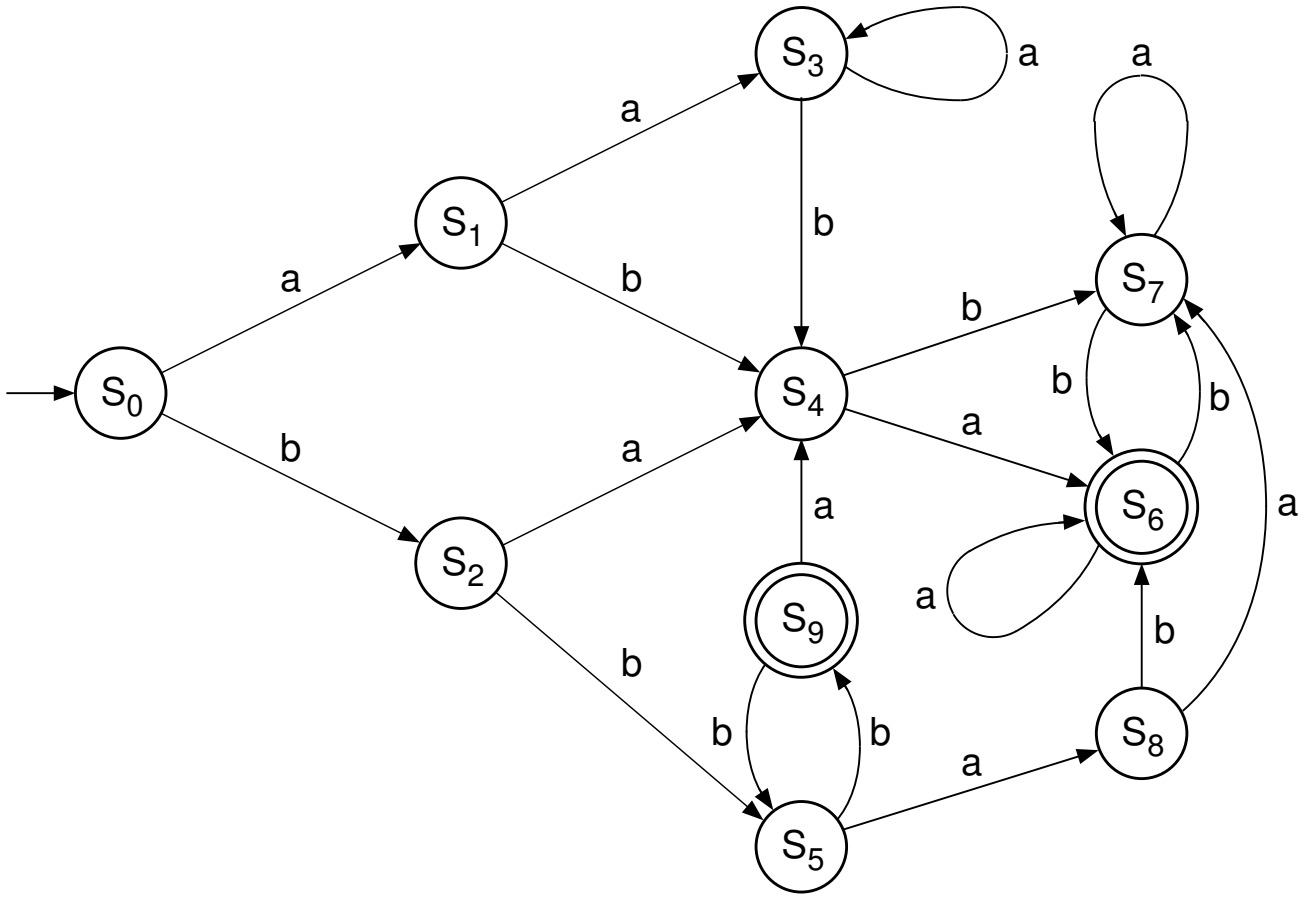
Anfangszustand: (S,S) = Anfangszustände von A_{E1} und A_{E2}

Folgezustände: $\delta((X_i, Y_j), e_r) \mapsto (X_m, Y_n)$

Endzustände: falls (X_m) Endzustand von A_{E1} und (Y_n) von A_{E2}
dann ist (X_m, Y_m) Endzustand von A_{Ek}

Neubezeichnung: $(X, Y) \longrightarrow S_i$ >>>

Automat A_{Ek} für Sprache L_k



	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	(A, D)
$S_2 := (A, B)$	(A, D)	(S, E)
$S_3 := (S, C)$	(S, C)	(A, D)
$S_4 := (A, D)$	(A, F)	(S, F)
$S_5 := (S, E)$	(S, D)	(A, E)
$\boldsymbol{S_6} := (A, F)$	(A, F)	(S, F)
$S_7 := (S, F)$	(S, F)	(A, F)
$S_8 := (S, D)$	(S, F)	(A, F)
$\boldsymbol{S_9} := (A, E)$	(A, D)	(S, E)

Wortanalyse mit Automaten

Typ-3-Sprachen: Es gibt zu jeder Typ-3-Sprache L_3 ein A_E mit $L(A_E) = L_3$ gibt und für jeden A_{NE} ein A_{DE}.
Also gilt: $\mathcal{L}_3 = \mathcal{L}_{3ndet} = \mathcal{L}_{3det}$

$\longrightarrow$ Wortanalyse für beliebige L_3 mittels A_{DE} möglich

Wortanalyse mit Automaten

Typ-3-Sprachen: Es gibt zu jeder Typ-3-Sprache L_3 ein A_E mit $L(A_E) = L_3$ gibt und für jeden A_{NE} ein A_{DE}.
Also gilt: $\mathcal{L}_3 = \mathcal{L}_{3ndet} = \mathcal{L}_{3det}$

$\longrightarrow$ Wortanalyse für beliebige L_3 mittels A_{DE} möglich

Typ-2-Sprachen: Gibt es zu Typ-2-Sprachen äquivalente Automaten ?

Wortanalyse mit Automaten

Typ-3-Sprachen: Es gibt zu jeder Typ-3-Sprache L_3 ein A_E mit $L(A_E) = L_3$ gibt und für jeden A_{NE} ein A_{DE}.
Also gilt: $\mathcal{L}_3 = \mathcal{L}_{3ndet} = \mathcal{L}_{3det}$

$\longrightarrow$ Wortanalyse für beliebige L_3 mittels A_{DE} möglich

Typ-2-Sprachen: Es gibt zu jeder Typ-2-Sprache L_2 einen Kellerautomaten A_K mit $L(A_K) = L_2$,
aber nicht für jeden A_{NK} ein A_{DK}.
Also ist: $\mathcal{L}_2 = \mathcal{L}_{2ndet} \supset \mathcal{L}_{2det}$

$\longrightarrow$ Wortanalyse mittels *Parsing*

Parsing-Algorithmus (Parser)

Aufgabe: Ermittelt für ein gegebenes Wort $w \in L(G_2)$ und eine gegebene Grammatik G_2 eine entsprechende Strukturbeschreibung (Ableitungsfolge oder -baum)

Prinzip: Direkte Anwendung der Grammatikregeln (kein Umweg über Automat)

Arbeitsweise:

a) *Verarbeitungsrichtung:* Verarbeitung des Wortes von links nach rechts oder umgekehrt oder ...

b) *Analyserichtung:* Start bei der Wurzel oder bei den Blättern des Syntaxbaumes

c) *Suchstrategie:* Tiefen- oder Breitensuche

Parsing-Beispiel

für den mathematischen Ausdruck: $a + b * (a + c)$

Links-rechts-Verarbeitung:	Beginn am Wortanfang
Top-down-Analyse:	Beginn bei der Wurzel
Tiefensuche:	zuerst den begonnenen Zweig expandieren

Parsing-Beispiel

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatik:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$

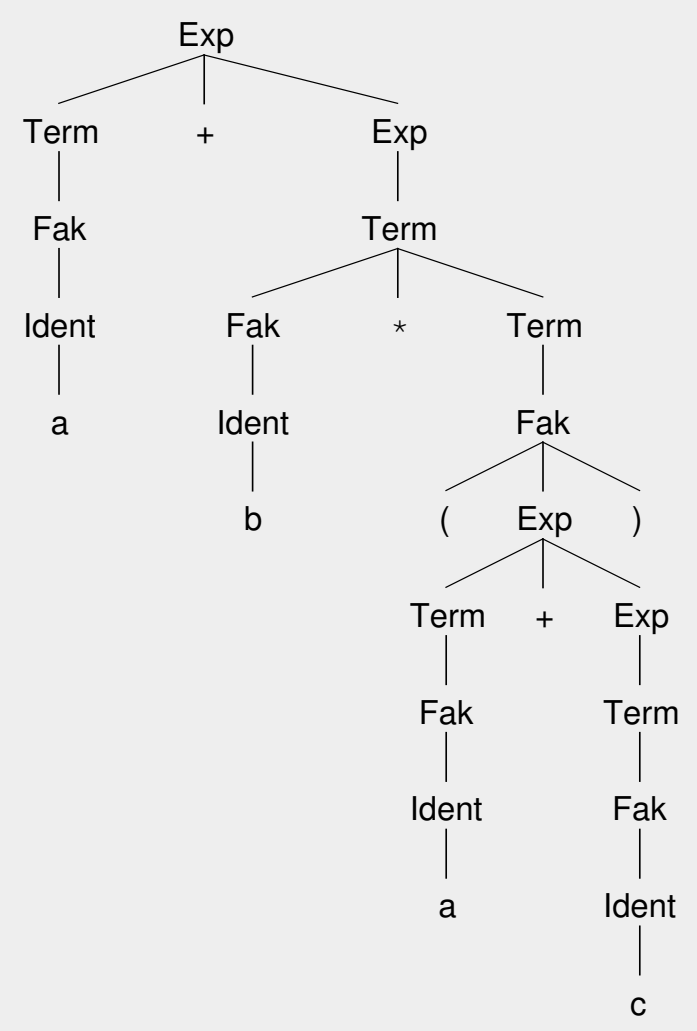
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

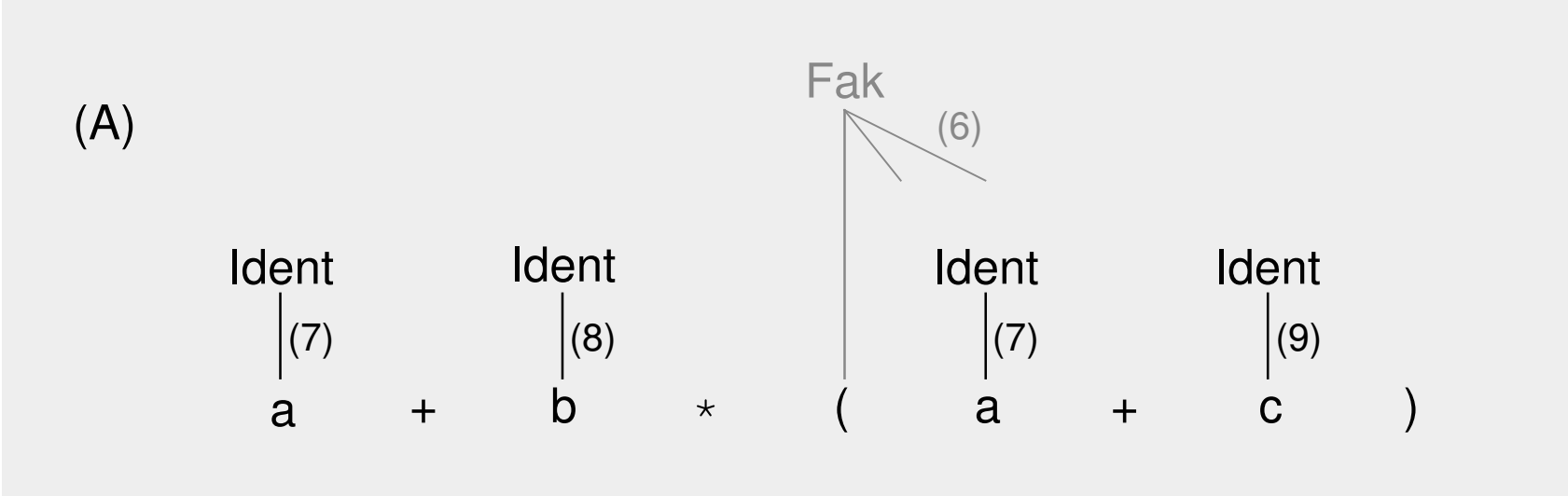
Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \mathit{Exp} \rightarrow \mathit{Term} && (1)\\
& \mathit{Exp} \rightarrow \mathit{Term} + \mathit{Exp} && (2)\\
& \mathit{Term} \rightarrow \mathit{Fak} && (3)\\
& \mathit{Term} \rightarrow \mathit{Fak} * \mathit{Term} && (4)\\
& \mathit{Fak} \rightarrow \mathit{Ident} && (5)\\
& \mathit{Fak} \rightarrow (\mathit{Exp}) && (6)\\
& \mathit{Ident} \rightarrow a && (7)\\
& \mathit{Ident} \rightarrow b && (8)\\
& \mathit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$

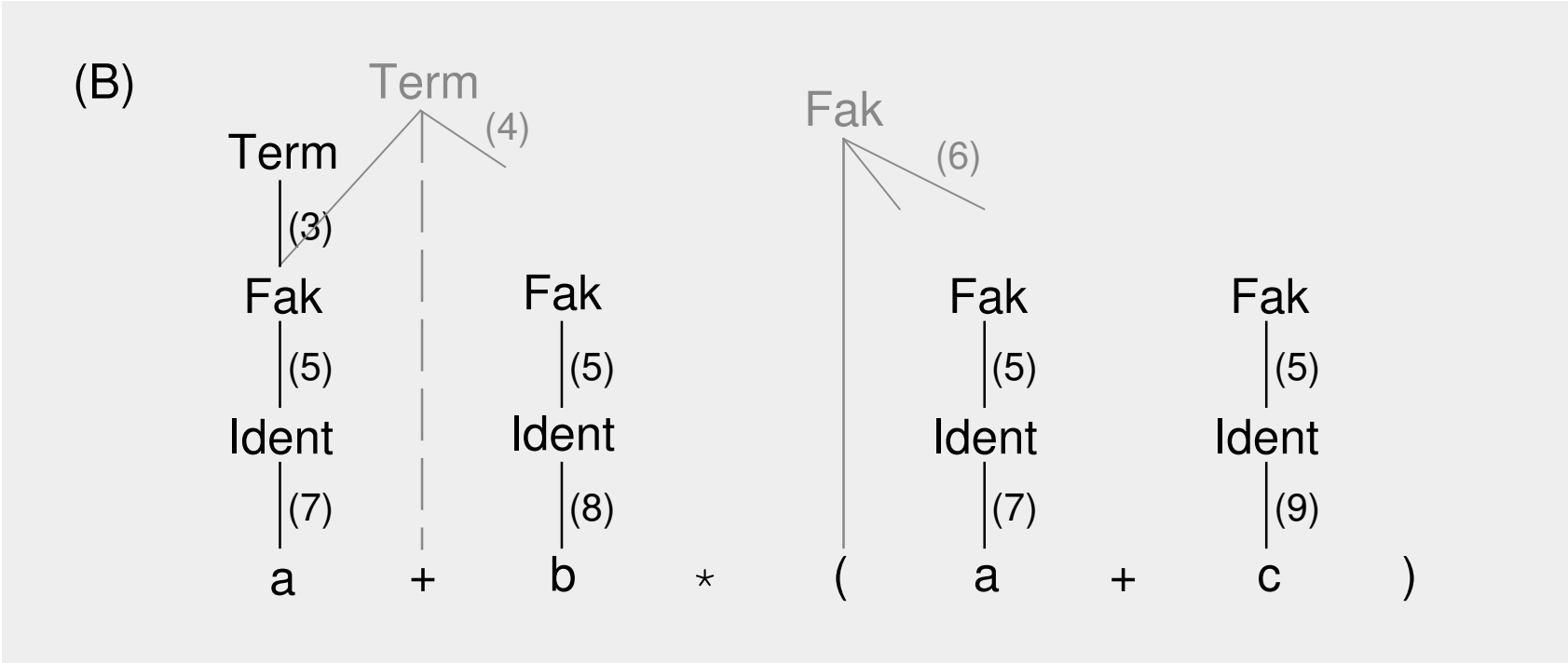


Parsing-Beispiel: Bottom-up-Analyse mit Breitensuche

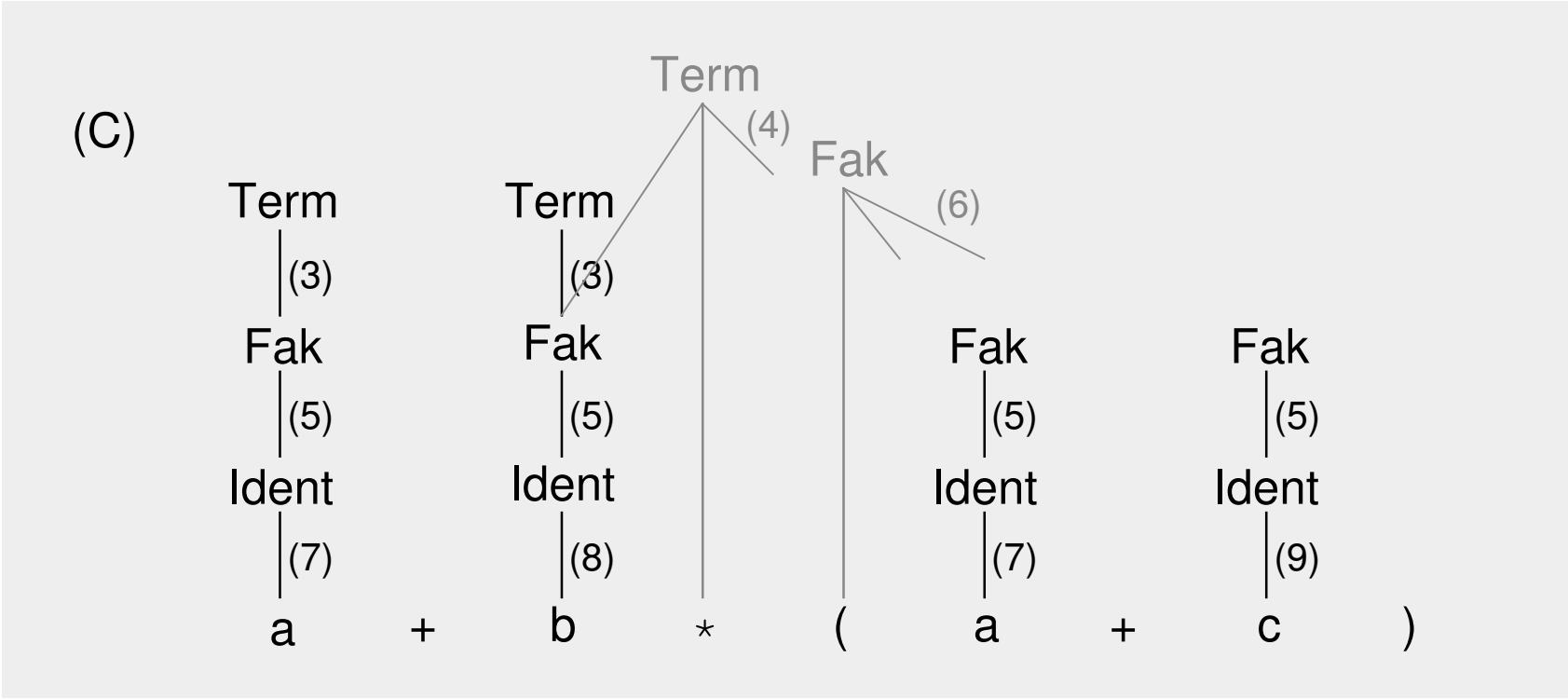
Beginn bei den Blättern des Syntaxbaumes



Parsing-Beispiel: Bottom-up-Analyse mit Breitensuche



Parsing-Beispiel: Bottom-up-Analyse mit Breitensuche



Wortanalyse für Typ-1-Sprachen

Theoretische Lösung: Wortproblem-Algorithmus für Typ-1-Sprachen

$\longrightarrow$ in der Praxis unbrauchbar

Grundsätzliche Lösung: Mittels Turing-Maschine A_T
(Automat mit unbegrenztem Speicherband):
Zu jeder Typ-1-Sprache L_1 gibt es eine
Turing-Maschine A_T, mit $L(A_T) = L_1$.

$\longrightarrow$ unpraktisch

Praktische Lösung: Mittels Parsing (manchmal problematisch!)

Zusammenfassung

- Gute und praktikable Lösung des “Wortproblems” vorhanden für Sprachen bzw. Grammatiken vom
  - Typ-3 $\longrightarrow$ deterministische endliche Automaten
  - Typ-2 $\longrightarrow$ (Chart-)Parsing
- Typ-1-Grammatiken werden in der Praxis vermieden, weil
  - Wortproblem schlecht entscheidbar
  - Ableitung i.a. nicht als Baum darstellbar
- Typ-2-Grammatiken sind ineffizient um die Syntax natürlicher Sprachen zu spezifizieren

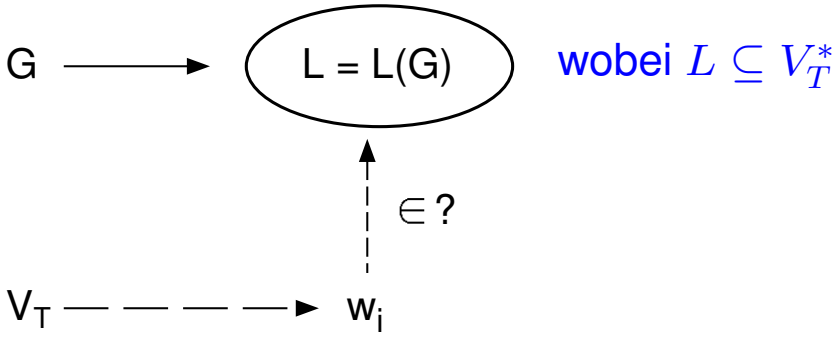
  $\longrightarrow$ Erweiterung zu DCG (*definite clause grammar*)

Thema der nächsten Lektion

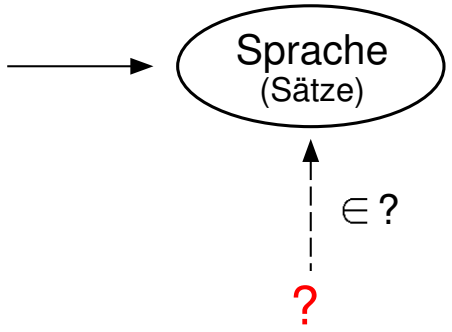
Grammatik für natürliche Sprache

Zur Übersicht der Vorlesung *Sprachverarbeitung II* >>>

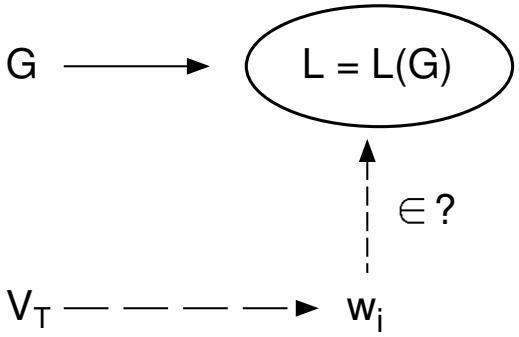
Formale Sprachen



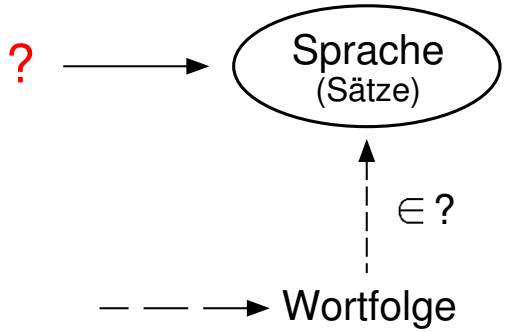
natürliche Sprachen



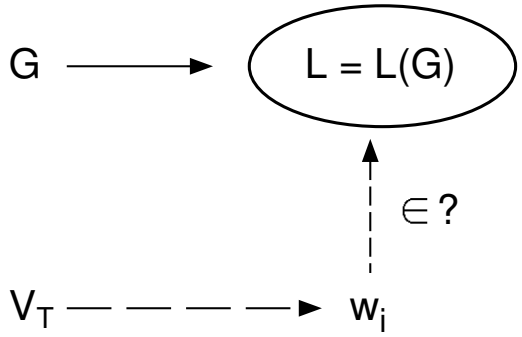
Formale Sprachen



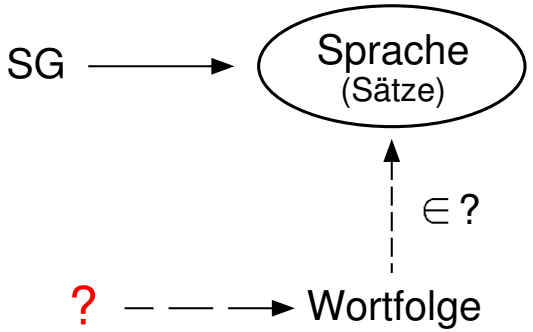
natürliche Sprachen



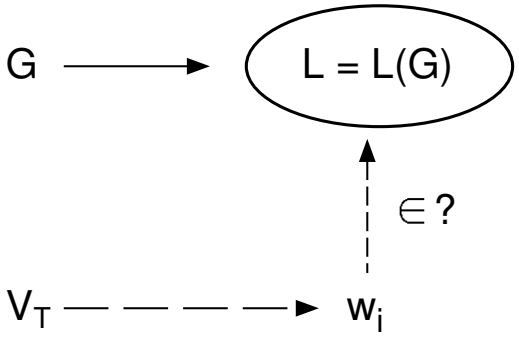
Formale Sprachen



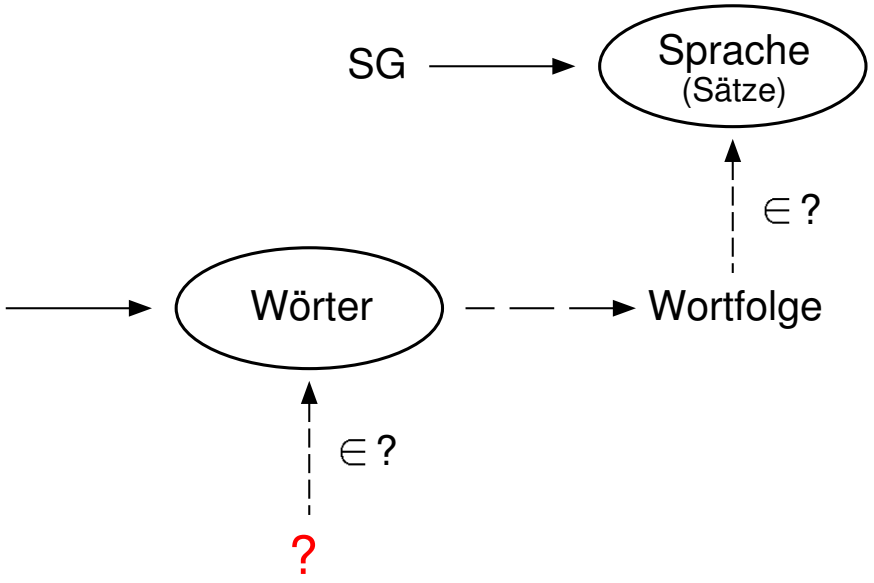
natürliche Sprachen



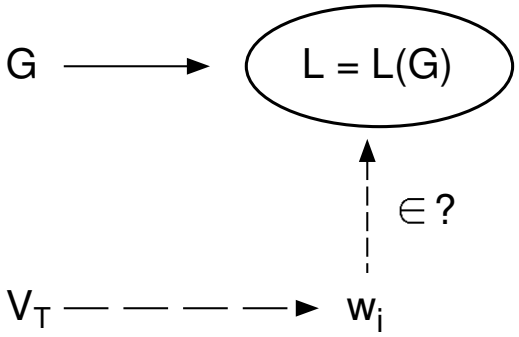
Formale Sprachen



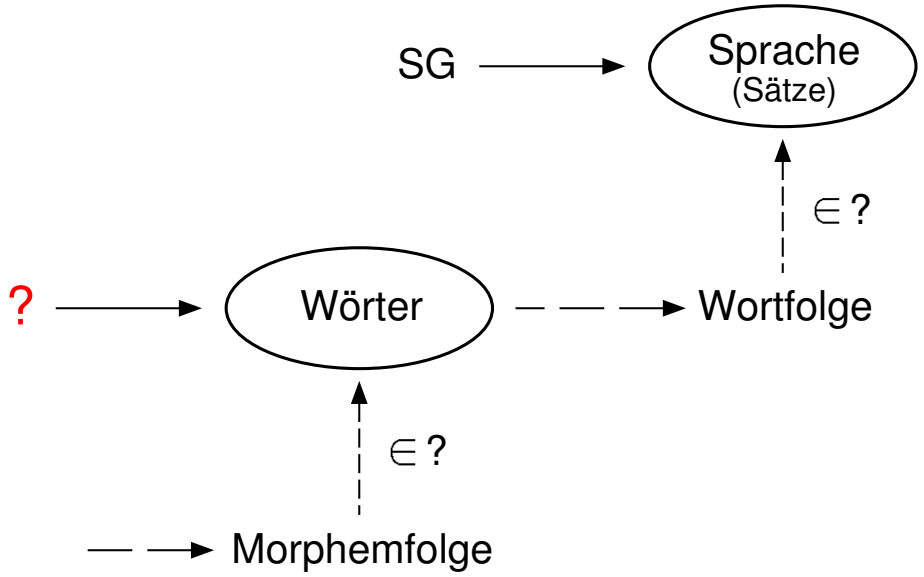
natürliche Sprachen



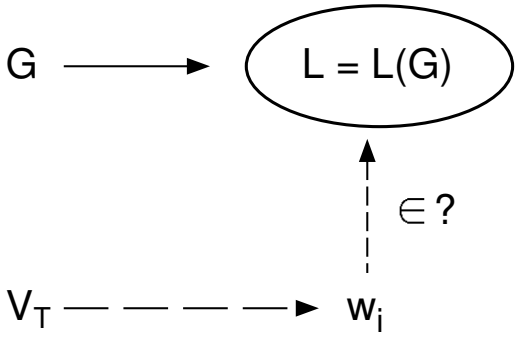
Formale Sprachen



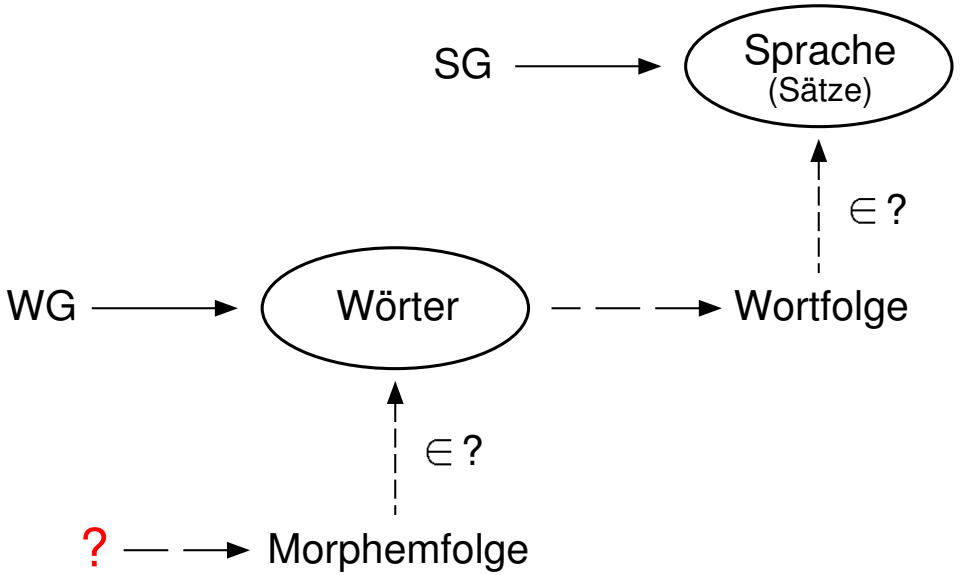
natürliche Sprachen



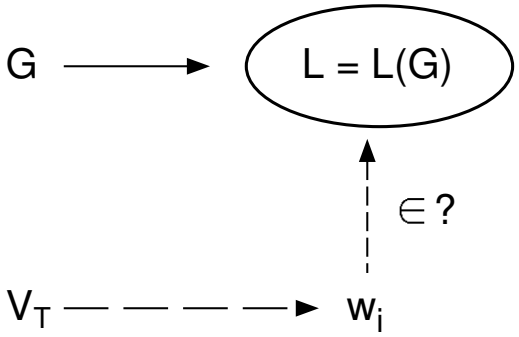
Formale Sprachen



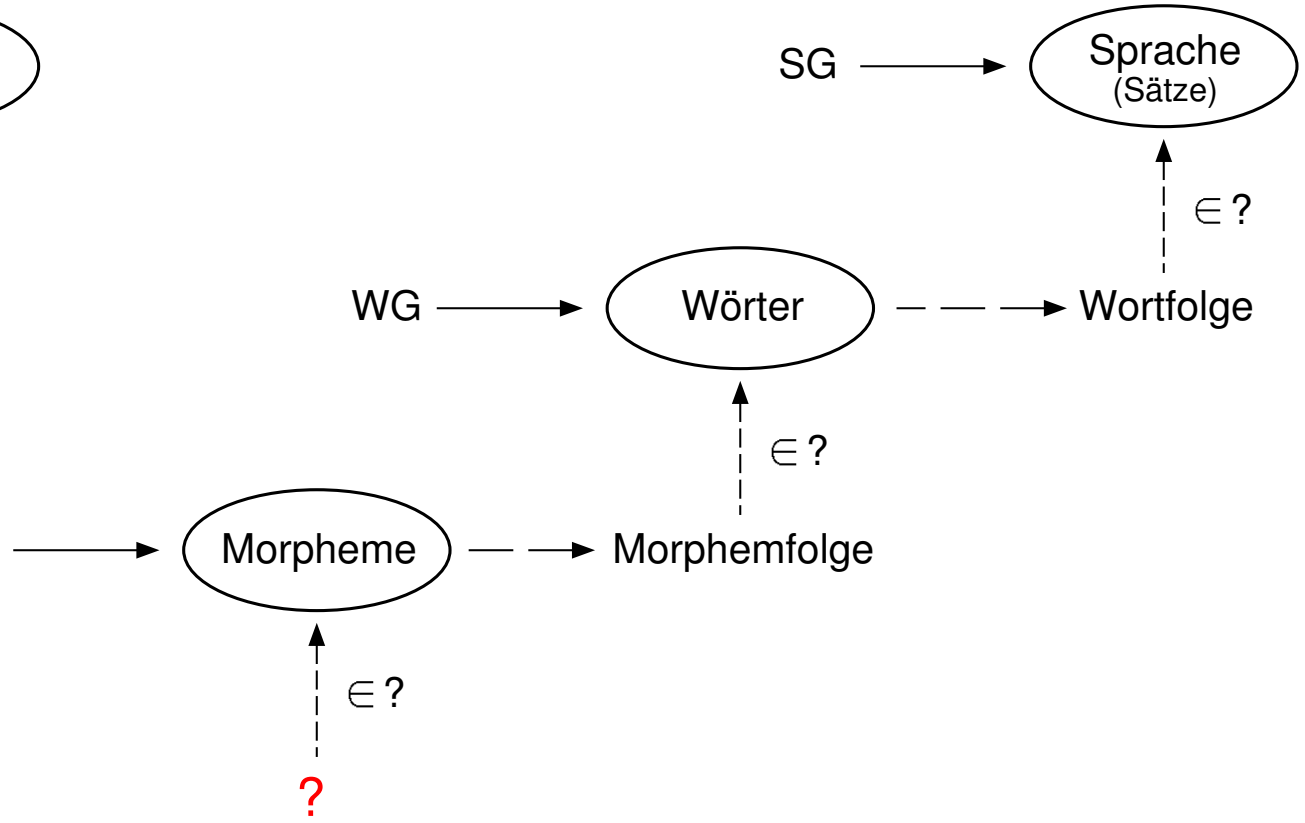
natürliche Sprachen



Formale Sprachen

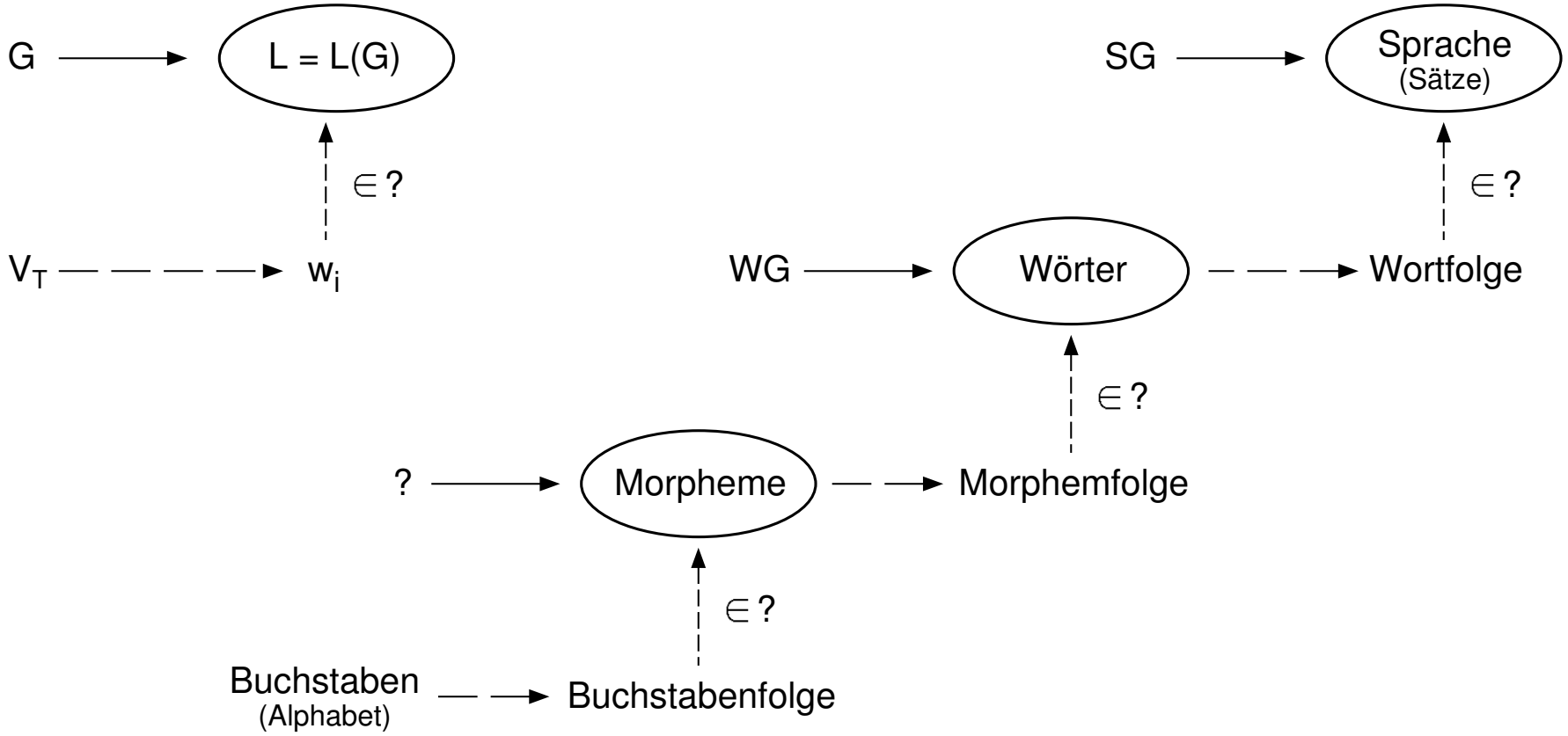


natürliche Sprachen

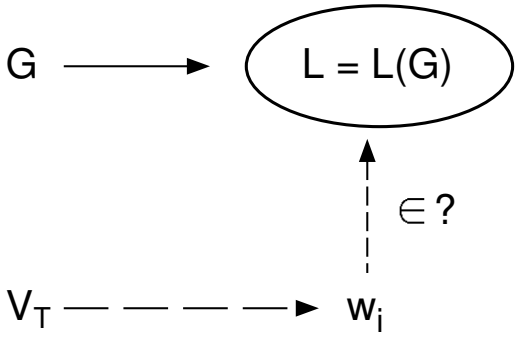


Formale Sprachen

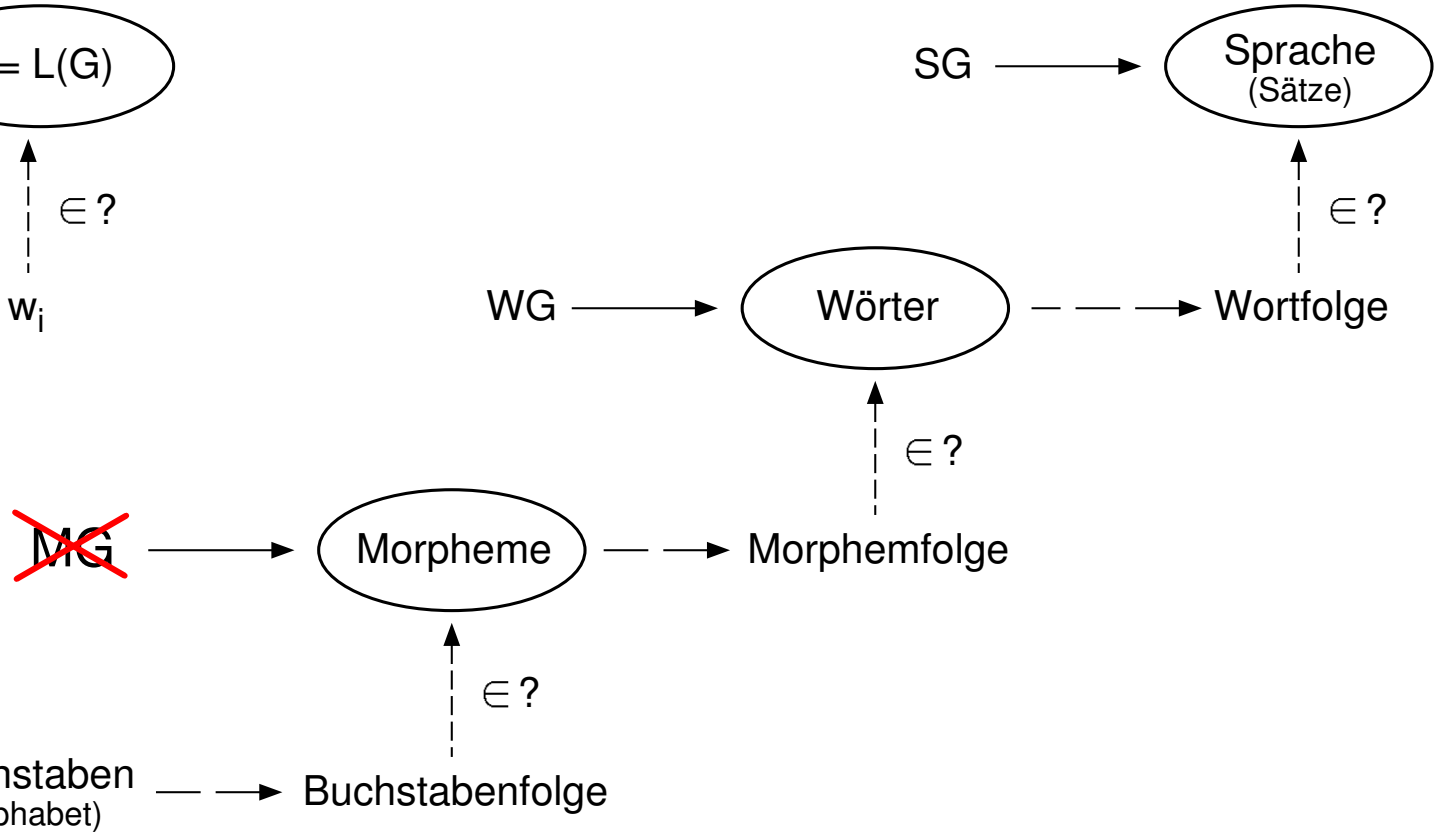
natürliche Sprachen



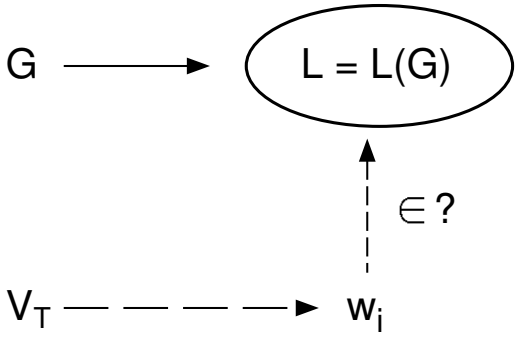
Formale Sprachen



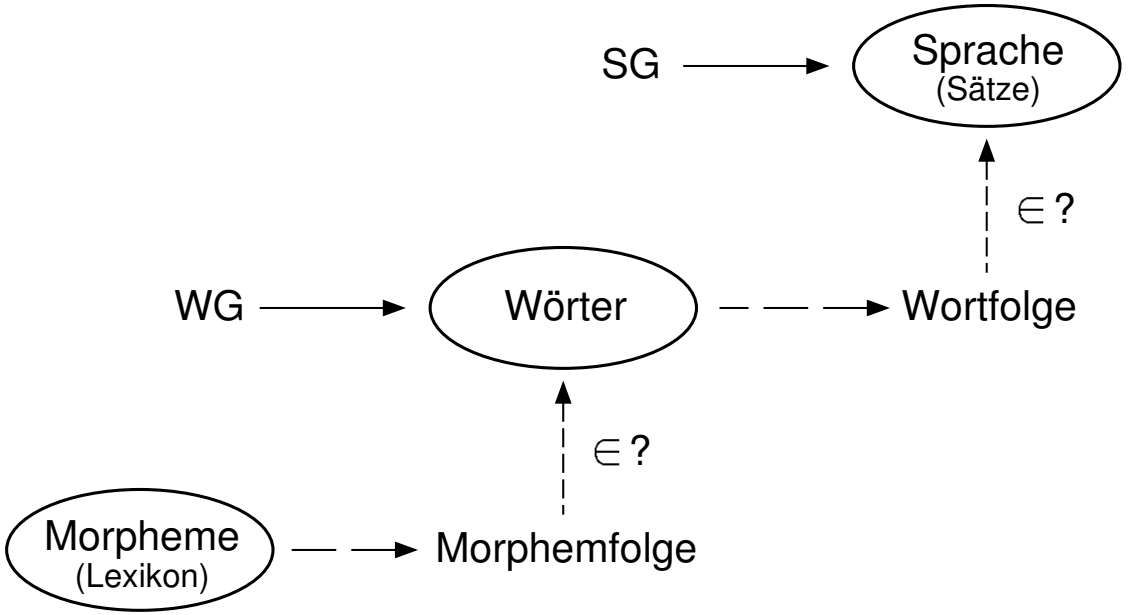
natürliche Sprachen



Formale Sprachen



natürliche Sprachen



<<<

Anmerkung

Für nichtdeterministische endliche Automaten A_{NE} gilt:

$w \in L(A_{NE})$, falls es mindestens eine Zustandsfolge $s_0, s_1, s_2, \ldots, s_{|w|}$ gibt, auf welcher der Automat das Wort verarbeitet und $s_{|w|}$ ein Endzustand ist.

<<<

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$		
$S_1 :=$		
$S_2 :=$		
$S_3 :=$		
$S_4 :=$		
$S_5 :=$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	
$S_1 :=$		
$S_2 :=$		
$S_3 :=$		
$S_4 :=$		
$S_5 :=$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 :=$		
$S_2 :=$		
$S_3 :=$		
$S_4 :=$		
$S_5 :=$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 := \{S, A\}$		
$S_2 := \{S, B\}$		
$S_3 :=$		
$S_4 :=$		
$S_5 :=$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 := \{S, A\}$	$\{S, A, \Phi\}$	
$S_2 := \{S, B\}$		
$S_3 := \{S, A, \Phi\}$		
$S_4 :=$		
$S_5 :=$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 := \{S, A\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_2 := \{S, B\}$		
$S_3 := \{S, A, \Phi\}$		
$S_4 := \{S, A, B\}$		
$S_5 :=$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 := \{S, A\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_2 := \{S, B\}$	$\{S, A, B\}$	
$S_3 := \{S, A, \Phi\}$		
$S_4 := \{S, A, B\}$		
$S_5 :=$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 := \{S, A\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_2 := \{S, B\}$	$\{S, A, B\}$	$\{S, B, \Phi\}$
$S_3 := \{S, A, \Phi\}$		
$S_4 := \{S, A, B\}$		
$S_5 := \{S, B, \Phi\}$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 := \{S, A\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_2 := \{S, B\}$	$\{S, A, B\}$	$\{S, B, \Phi\}$
$S_3 := \{S, A, \Phi\}$	$\{S, A, \Phi\}$	
$S_4 := \{S, A, B\}$		
$S_5 := \{S, B, \Phi\}$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 := \{S, A\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_2 := \{S, B\}$	$\{S, A, B\}$	$\{S, B, \Phi\}$
$S_3 := \{S, A, \Phi\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_4 := \{S, A, B\}$		
$S_5 := \{S, B, \Phi\}$		
$S_6 :=$		

>>>

Beispiel: $A_{NE} \longrightarrow A_{DE}$

A_{NE}

	a	b
S	$S \mid A$	$S \mid B$
A	Φ	A
B	B	Φ
Φ	–	–

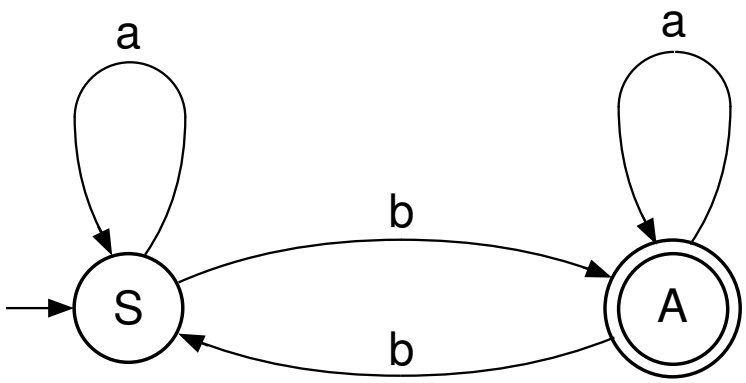
A_{DE}

	a	b
$S_0 := \{S\}$	$\{S, A\}$	$\{S, B\}$
$S_1 := \{S, A\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_2 := \{S, B\}$	$\{S, A, B\}$	$\{S, B, \Phi\}$
$S_3 := \{S, A, \Phi\}$	$\{S, A, \Phi\}$	$\{S, A, B\}$
$S_4 := \{S, A, B\}$	$\{S, A, B, \Phi\}$	
$S_5 := \{S, B, \Phi\}$		
$S_6 := \{S, A, B, \Phi\}$		

<<<

Automat A_{E1} für Sprache L_{s_1}

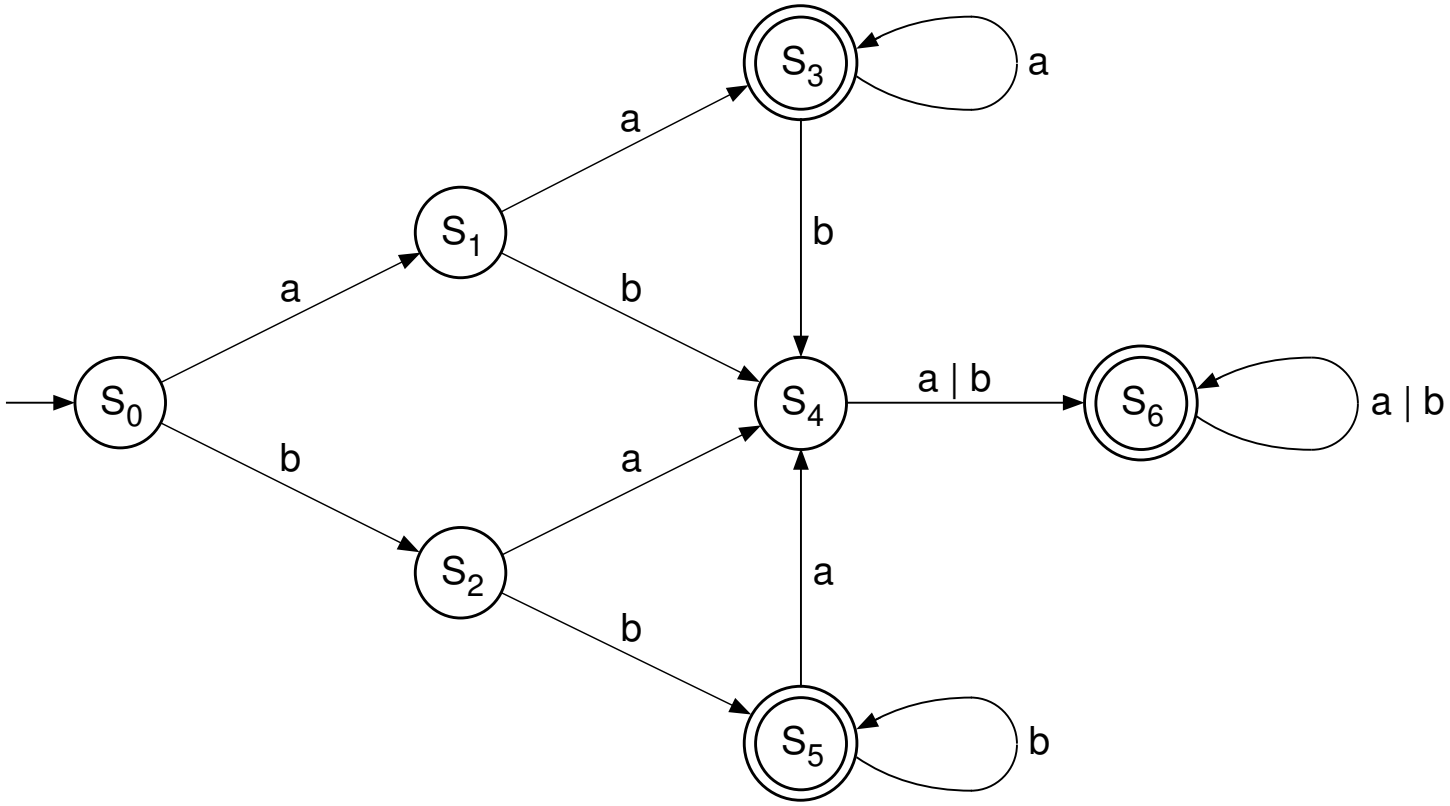
$L_{s_1} = \{w \in \{a, b\}^* \mid w$ enthält eine ungerade Anzahl $b\}$



	a	b
S	S	A
$\boldsymbol{A}$	A	S

Automat A_{E2} für Sprache L_{s2}

$L_{s2} = \{w \in \{a,b\}^* \mid$ das letzte Zeichen ist vorher schon einmal vorgekommen$\}$



	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

<<<

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	
$S_1 := (S, A)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$		
$S_2 := (A, B)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	
$S_2 := (A, B)$		
$S_3 := (S, C)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	(A, D)
$S_2 := (A, B)$		
$S_3 := (S, C)$		
$S_4 := (A, D)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	(A, D)
$S_2 := (A, B)$	(A, D)	
$S_3 := (S, C)$		
$S_4 := (A, D)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	(A, D)
$S_2 := (A, B)$	(A, D)	(S, E)
$S_3 := (S, C)$		
$S_4 := (A, D)$		
$S_5 := (S, E)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	(A, D)
$S_2 := (A, B)$	(A, D)	(S, E)
$S_3 := (S, C)$	(S, C)	
$S_4 := (A, D)$		
$S_5 := (S, E)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	(A, D)
$S_2 := (A, B)$	(A, D)	(S, E)
$S_3 := (S, C)$	(S, C)	(A, D)
$S_4 := (A, D)$		
$S_5 := (S, E)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	(A, D)
$S_2 := (A, B)$	(A, D)	(S, E)
$S_3 := (S, C)$	(S, C)	(A, D)
$S_4 := (A, D)$	(A, F)	
$S_5 := (S, E)$		
$S_6 := (A, F)$		

>>>

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	(A, D)
$S_2 := (A, B)$	(A, D)	(S, E)
$S_3 := (S, C)$	(S, C)	(A, D)
$S_4 := (A, D)$	(A, F)	(S, F)
$S_5 := (S, E)$	(S, D)	(A, E)
$S_6 := (A, F)$	(A, F)	(S, F)
$S_7 := (S, F)$	(S, F)	(A, F)
$S_8 := (S, D)$	(S, F)	(A, F)
$S_9 := (A, E)$	(A, D)	(S, E)

Endzustände?

Konstruktion von A_{Ek} aus A_{E1} und A_{E2}

A_{E1}:

	a	b
S	S	A
$\boldsymbol{A}$	A	S

A_{E2}:

	a	b
S	A	B
A	C	D
B	D	E
$\boldsymbol{C}$	C	D
D	F	F
$\boldsymbol{E}$	D	E
$\boldsymbol{F}$	F	F

A_{Ek}:

	a	b
$S_0 := (S, S)$	(S, A)	(A, B)
$S_1 := (S, A)$	(S, C)	(A, D)
$S_2 := (A, B)$	(A, D)	(S, E)
$S_3 := (S, C)$	(S, C)	(A, D)
$S_4 := (A, D)$	(A, F)	(S, F)
$S_5 := (S, E)$	(S, D)	(A, E)
$S_6 := (A, F)$	(A, F)	(S, F)
$S_7 := (S, F)$	(S, F)	(A, F)
$S_8 := (S, D)$	(S, F)	(A, F)
$S_9 := (A, E)$	(A, D)	(S, E)

Endzustände

<<<

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$

Exp

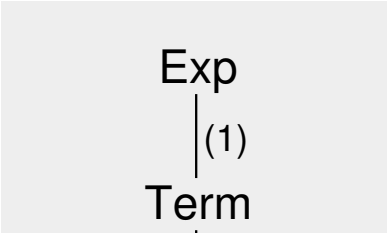
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} && \rightarrow && \textit{Term} && (1)\\
& \textit{Exp} && \rightarrow && \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} && \rightarrow && \textit{Fak} && (3)\\
& \textit{Term} && \rightarrow && \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} && \rightarrow && \textit{Ident} && (5)\\
& \textit{Fak} && \rightarrow && (\textit{Exp}) && (6)\\
& \textit{Ident} && \rightarrow && a && (7)\\
& \textit{Ident} && \rightarrow && b && (8)\\
& \textit{Ident} && \rightarrow && c\ \} && (9)
\end{aligned}
$$



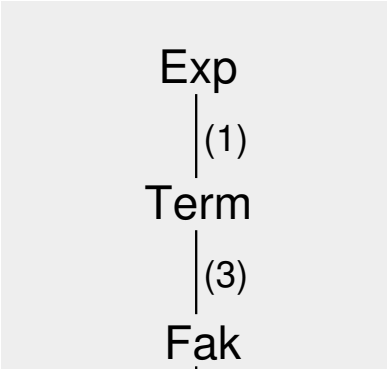
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$



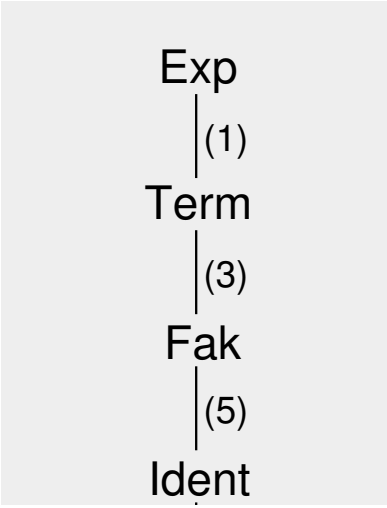
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$



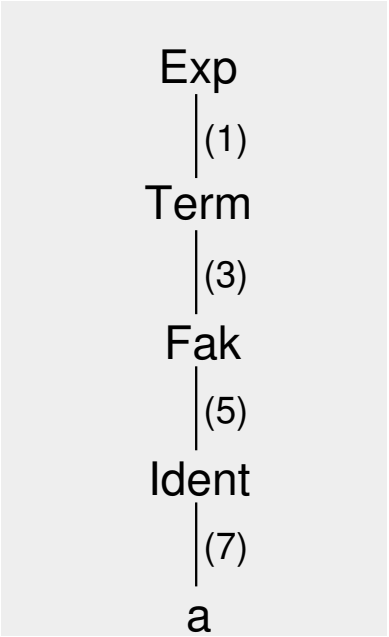
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$



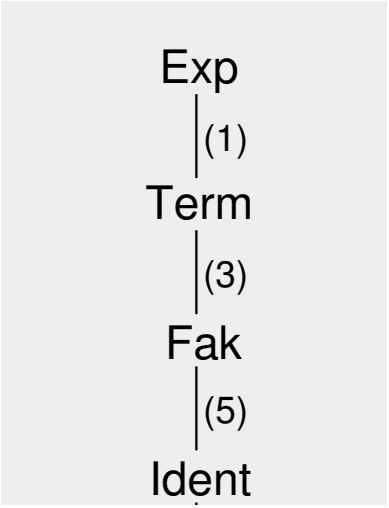
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$



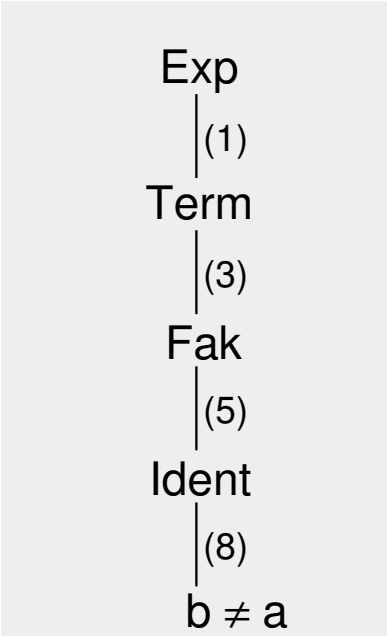
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$



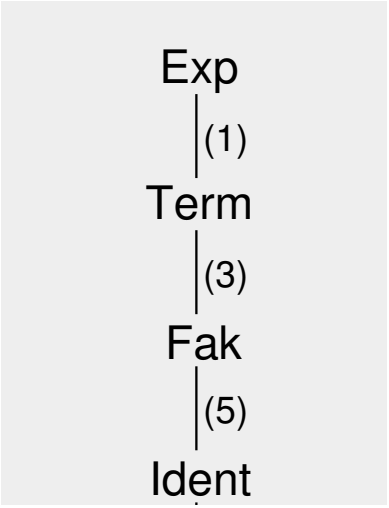
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$



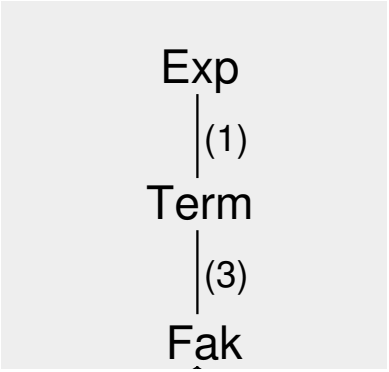
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$



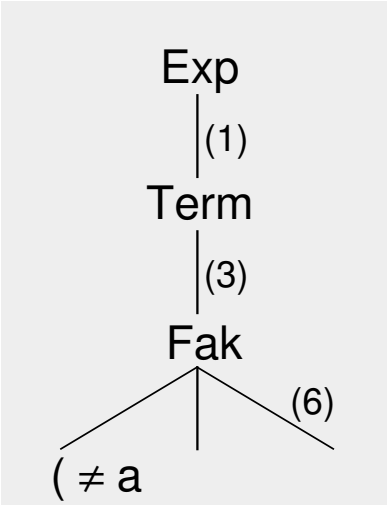
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} \rightarrow \textit{Term} && (1)\\
& \textit{Exp} \rightarrow \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} \rightarrow \textit{Fak} && (3)\\
& \textit{Term} \rightarrow \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} \rightarrow \textit{Ident} && (5)\\
& \textit{Fak} \rightarrow (\textit{Exp}) && (6)\\
& \textit{Ident} \rightarrow a && (7)\\
& \textit{Ident} \rightarrow b && (8)\\
& \textit{Ident} \rightarrow c\ \} && (9)
\end{aligned}
$$



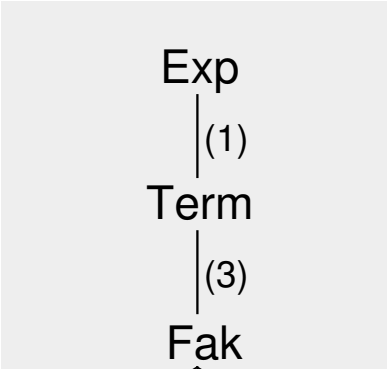
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} && \rightarrow && \textit{Term} && (1)\\
& \textit{Exp} && \rightarrow && \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} && \rightarrow && \textit{Fak} && (3)\\
& \textit{Term} && \rightarrow && \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} && \rightarrow && \textit{Ident} && (5)\\
& \textit{Fak} && \rightarrow && (\textit{Exp}) && (6)\\
& \textit{Ident} && \rightarrow && a && (7)\\
& \textit{Ident} && \rightarrow && b && (8)\\
& \textit{Ident} && \rightarrow && c\ \} && (9)
\end{aligned}
$$



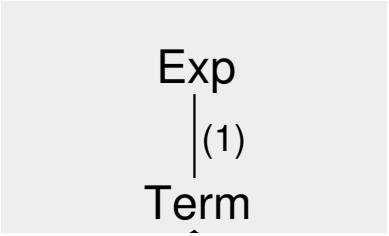
>>>

Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$$
\begin{aligned}
P = \{\ & \textit{Exp} && \rightarrow && \textit{Term} && (1)\\
& \textit{Exp} && \rightarrow && \textit{Term} + \textit{Exp} && (2)\\
& \textit{Term} && \rightarrow && \textit{Fak} && (3)\\
& \textit{Term} && \rightarrow && \textit{Fak} * \textit{Term} && (4)\\
& \textit{Fak} && \rightarrow && \textit{Ident} && (5)\\
& \textit{Fak} && \rightarrow && (\textit{Exp}) && (6)\\
& \textit{Ident} && \rightarrow && a && (7)\\
& \textit{Ident} && \rightarrow && b && (8)\\
& \textit{Ident} && \rightarrow && c\ \} && (9)
\end{aligned}
$$



>>>

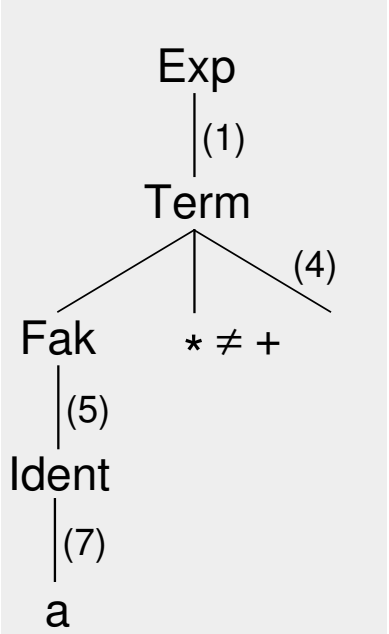
Parsing-Beispiel: Top-down-Analyse mit Tiefensuche

Gegebenes Wort: $w = a + b * (a + c)$

Gegebene Grammatikregeln:

$P = \{$

Exp	$\rightarrow$	*Term*	(1)
Exp	$\rightarrow$	*Term* $+$ *Exp*	(2)
Term	$\rightarrow$	*Fak*	(3)
Term	$\rightarrow$	*Fak* $*$ *Term*	(4)
Fak	$\rightarrow$	*Ident*	(5)
Fak	$\rightarrow$	(*Exp*)	(6)
Ident	$\rightarrow$	a	(7)
Ident	$\rightarrow$	b	(8)
Ident	$\rightarrow$	c $\}$	(9)



<<<