Formale Sprachen

Inhaltsübersicht

1 Die Idee

Eine formale Sprache wird als Menge von endlichen Zeichenketten über einem endlichen Alphabet bzw. über einem endlichen (Basis-)Lexikon verstanden. Zu dem Alphabet {a, b} ist z.B. {ab} eine Sprache, aber auch die Menge aller endlichen Ketten, die aus einer beliebigen Anzahl von a's und b's bestehen (einschließlich aller Ketten ohne a's und aller Ketten ohne b's). Die Symbole (Zeichen) in der Ausgangsmenge werden jeweils als nicht weiter zerlegbar betrachtet. Bei der Beschreibung der formalen Eigenschaften von Sprachen ist es üblich, nur den ersten Fall zu betrachten und die Elemente der Sprache daher als Wörter zu bezeichnen. Betrachtet man die Ausgangsmenge als Lexikon, so bezeichnet man die Elemente der Sprache als Sätze.

Eine formale Sprache wird durch eine formale Grammatik festgelegt („erzeugt"), d.h. die Aufgabe der Grammatik ist es, die Menge der Zeichenketten, die zu der Sprache gehören, von der Menge der Zeichenketten, die nicht zu der Sprache gehören, abzugrenzen. Diesem Abgrenzungsproblem entspricht das sog. Wortproblem, ein Entscheidbarkeitsproblem: Kann für jedes Wort (jede Zeichenkette) über dem vorausgesetzten Alphabet durch einen Algorithmus[1] entschieden werden, ob es zu der Sprache gehört oder nicht? Die wichtigste Unterscheidung von Sprachtypen – die Chomsky-Hierarchie – betrifft daher den 'Aufwand' für die Lösung des Entscheidbarkeitsproblem.

Existiert zu einer Sprache *L* eine formale Grammatik – ein System von Ersetzungsregeln bestimmter Art –, so kann man zumindest für alle Elemente von *L* entscheiden („berechnen"), *dass* sie zu *L* gehören. Allerdings kann man bei einigen Sprachen dieser Art für die übrigen Zeichenketten nicht entscheiden, dass

[1] Ein Algorithmus ist ein allgemeines, endlich beschreibbares Verfahren, ein Problem bestimmter Art in endlich vielen Schritten zu lösen.

sie *nicht* zu der Sprache gehören. Solche Sprachen sind zumindest semi-entscheidbar: Zu ihnen selbst gibt es zwar eine formale Grammatik, nicht aber zu ihrem Komplement (= der Menge der Zeichenketten über dem vorausgesetzten Alphabet abzüglich der Elemente der Sprache selbst). Dieses Komplement ist eine formale Sprache, die nicht einmal semi-entscheidbar ist. Abgesehen von dem Nachweis der Existenz solcher Sprachen, interessiert man sich in der Theorie der formalen Sprachen nur für Sprachen, zu denen es eine Grammatik gibt, d.h. die zumindest semi-entscheidbar sind. Von besonderem Interesse sind dabei die Sprachen, die eine 'möglichst einfache' Grammatik haben. Nach der Klärung einiger Grundbegriffe (Abschn. 2) werden wir uns einen Überblick über die verschiedenen Typen 'relativ einfacher' Sprachen verschaffen (Abschn. 3).

An dieser Stelle halten wir noch fest, dass das Wortproblem auf das engste mit der Berechenbarkeit eines Problems durch Turing-Maschinen (das ist das allgemeinste Modell eines Computers) zusammenhängt und daher ein zentraler Gegenstand der Theoretischen Informatik ist. Darüber hinaus dient die Theorie der formalen Sprachen in der Informatik dazu, Programmiersprachen durch Regeln (Grammatiken) für das Erzeugen von syntaktisch korrekten Programmen zu beschreiben. Diese werden benötigt, um aus einem Programm-Quelltext ein lauffähiges Programm zu erzeugen.

In der Sprachwissenschaft spielt die Theorie der formalen Sprachen eine zentrale Rolle in allen Ansätzen, die natürliche Sprachen als formale Sprachen (re)konstruieren wie die seit 1956 von Chomsky begründete Generative Grammatik, etwa gemäß Montagues Programm von „English as a formal language" in seinem Aufsatz gleichen Titels von 1970. Ansätze dieser Art sind vor allem für die Computerlinguistik – bei der automatischen Sprachverarbeitung – auch von großer praktischer Bedeutung.

Einen kurzen Überblick zur Geschichte der formalen Sprachen, die mit einem Aufsatz von Axel Thue 1914 beginnt, gibt Klenk (1995: 1576), detaillierter Levelt (1974 [Bd. 1]: 131–134).

2 Grundbegriffe

Ein ***Alphabet*** Σ ist eine endliche Menge von Symbolen. Ein ***Wort*** w über einem Alphabet Σ ist eine endliche Folge von Symbolen aus Σ. ***Das leere Wort*** ε := die leere Folge $\varnothing$ ($\varepsilon \notin \Sigma$).

Beispiele:

Sei $\Sigma = \{0, 1\}$. Dann ist $w = 0111011$ ein Wort (über Σ).

Sei $\Sigma = \{A, B, C, \ldots, Z, a, b, c, \ldots, z\}$. Dann gehören zu den Wörtern (über Σ) z.B. w_1 = Abend, w_2 = sprudeln, w_3 = abzdx.

Notationskonventionen:

Seien w, w' Wörter über Σ, $a \in \Sigma$ und $k \in \mathbb{N}_0$.

wa := die Verkettung von w mit der Einerfolge von a.

ww' := die Verkettung von w mit w'.

$w^0 := \varepsilon$,

$w^{k+1} := ww^k$ (beachte: $w^1 = w$).

$\varepsilon^R := \varepsilon$,

$(aw)^R := w^R a$ (also gilt insbesondere: $a^R = a$; „R“ erinnert an „Rückwärts“).

$|w|$:= die Länge von w (offensichtlich gilt: $|w^k| = k \cdot |w|$).

$\Sigma^0 := \{\varepsilon\}$ (beachte: $\Sigma^0 \neq \varnothing$!),

$\Sigma^{k+1} := \{wa \mid w \in \Sigma^k \text{ und } a \in \Sigma\}$

= die Menge aller Wörter über Σ mit k+1 Buchstaben (beachte: $\Sigma^1 = \Sigma$), d.h. k+1 ist die Länge der Wörter in Σ^{k+1}.

$\Sigma^* := \bigcup_{n=0}^{\infty} \Sigma^n$ = die Menge aller Wörter über Σ („*“ heißt auch „Kleene-Stern“).[2]

$\Sigma^+ := \bigcup_{n=1}^{\infty} \Sigma^n$ = die Menge aller Wörter über Σ ohne das leere Wort.

Übungen:

Sei $\Sigma = \{a, b, c\}$. Sei $w = ab$, sei $w' = c$, $w'' = bac$.

Beachte: Kursivbuchstaben sind Variablen, normal gesetzte lateinische Buchstaben sind Konstanten (hier: Symbole aus dem Alphabet).

(1) $wb =$

(2) $ww' =$

(3) $w'^3w^2 =$

(4) $|ww''| =$

(5) $\Sigma^2 =$

(6) $\varnothing^* =$

(7) Überprüfen Sie, ob gilt:

(a) $w \in \Sigma^2$

(b) $w' \in \Sigma^2$

(c) $w'^2 \in \Sigma^2$

(d) $w'b \in \Sigma^2$

(e) $w'' \in \Sigma^*$

(f) $abcd \in \Sigma^*$

(8) Überprüfen Sie, ob gilt:

(a) $ww' \in \Sigma^*$ und $w'w \in \Sigma^*$

(b) $(ww')w'' = w(w'w'')$

(c) $\varepsilon w = w = w\varepsilon$

(d) $ww' = w'w$

---

[2] $\bigcup_{n=0}^{\infty} \Sigma^n$ ist die Vereinigung aller Σ^n mit $n \in \mathbb{N}_0$ (d.h. n ‘läuft von 0 bis unendlich’).

Aus der Verallgemeinerung von (8a–c) folgt, dass $(\Sigma^*, \cdot)$ – der Kleene-Abschluss von Σ mit der Verkettung $\cdot$ – eine Halbgruppe mit neutralem Element ist.

Eine ***formale Sprache*** über Σ ist eine Teilmenge $L \subseteq \Sigma^*$. Während Σ^* abzählbar unendlich ist (also die Mächtigkeit von $\mathbb{N}$ hat) und somit jede Sprache höchstens abzählbar unendlich ist, gibt es bereits zu dem Alphabet {0, 1} überabzählbar viele Sprachen $L \subseteq \{0,1\}^*$, genauer: $2^{\aleph_0}$ viele.[3] Außer durch die mengentheoretischen Operationen können Sprachen auch 'verkettet' und durch den Stern 'abgeschlossen' werden:

$L_1L_2 := \{ww' \mid w \in L_1 \text{ und } w' \in L_2\}$
$L^0 := \{\varepsilon\}$
$L^{n+1} := LL^n$
$L^* = \bigcup_{n=0}^{\infty} L^n$

Dies verallgemeinert die Definition des Kleene-Abschlusses für Alphabete in naheliegender Weise.

Von besonderem Interesse sind diejenigen Sprachen, die durch eine Allgemeine Regelgrammatik (Semi-Thue-System[4], Typ-0-Grammatik) festgelegt werden. Eine solche Grammatik besteht aus (1) einer Menge Σ von Terminalsymbolen, z.B. einer Menge von Wortformen, (2) einer Menge V von Variablen, z.B. einer Menge von Symbolen für syntaktische Kategorien, (3) einem Startsymbol S und (4) einer Menge P von Ableitungsregeln („Produktionen"), in denen erfaßt wird, wie man ausgehend vom Startsymbol durch schrittweises Ersetzen eine Folge von Terminalsymbolen erhält. Z.B. ist $G = \langle \{$**Petra**, **Anna**, **kauft**, **verkauft**, **ein**, **das**, **Auto**, **Haus**$\}$, $\{S, VP, NP, EN, V, D, N\}$, S, $P\rangle$ mit P =

(1) $S \rightarrow NP\ VP$
(2) $VP \rightarrow V\ NP$
(3) $NP \rightarrow EN$
(4) $NP \rightarrow D\ N$
(5) $EN \rightarrow$ **Petra**
(6) $EN \rightarrow$ **Anna**
(7) $V \rightarrow$ **kauft**
(8) $V \rightarrow$ **verkauft**
(9) $D \rightarrow$ **das**
(10) $D \rightarrow$ **ein**
(11) $N \rightarrow$ **Auto**
(12) $N \rightarrow$ **Haus**

---

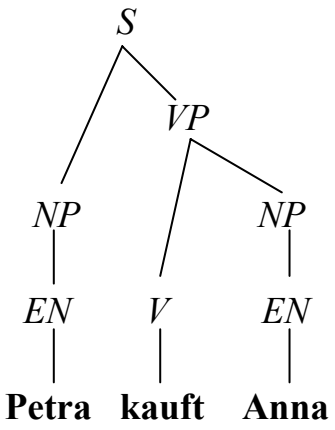
[3] $\aleph_0$ – lies: „aleph-Null" – ist die Mächtigkeit von $\mathbb{N}$, $2^{\aleph_0}$ die Mächtigkeit der Menge der reellen Zahlen. Da es nur abzählbar unendlich viele Turingmaschinen gibt, denn jede Turingmaschine ist durch eine sog. Gödelnummer aus $\mathbb{N}$ eindeutig charakterisierbar, folgt hieraus bereits, dass es Sprachen (Probleme) gibt, die nicht durch eine Turingmaschine berechenbar (entscheidbar) sind (vgl. z.B. Partee u.a. 1993: 433 oder Wegener 1996: 14, 21).

[4] Produktionssysteme für Zeichenketten wurden 1914 von dem norwegischen Mathematiker Axel Thue eingeführt und 1956 von Chomsky mit natürlichen Sprachen in Verbindung gebracht (vgl. Klenk 1995: 1576).

eine Grammatik, die die Sprache

$L =$ {**Petra kauft Anna, Petra kauft ein Auto, Petra kauft das Auto, Petra kauft ein Haus, Petra kauft das Haus, Petra verkauft Anna, Petra verkauft ein Auto, Petra verkauft das Auto, Petra verkauft ein Haus, Petra verkauft das Haus, Anna kauft Petra, Anna kauft ein Auto, Anna kauft das Auto, Anna kauft ein Haus, Anna kauft das Haus, Anna verkauft Petra, Anna verkauft ein Auto, Anna verkauft das Auto, Anna verkauft ein Haus, Anna verkauft das Haus, ein Auto kauft Petra, …**}

erzeugt.[5] Z.B. läßt sich **Petra kauft Anna** aus S mittels der Produktionen (1)–(3) und (5)–(7) folgendermaßen ableiten: $S \Rightarrow_G NP\ VP \Rightarrow_G NP\ V\ NP \Rightarrow_G EN\ V\ NP \Rightarrow_G EN\ V\ EN \Rightarrow_G$ **Petra** $V\ EN \Rightarrow_G$ **Petra kauft** $EN \Rightarrow$ **Petra kauft Anna.** Ableitungen dieser Art lassen sich durch Baumdiagramme darstellen:



Verallgemeinern wir das Beispiel:

$G = \langle \Sigma, V, S, R \rangle$ ist eine (***Typ-0-*)*Grammatik*** gdwg:

(1) Σ ist eine endliche Menge von (Terminal-)Symbolen,

(2) V ist eine endliche Menge von Variablen (Nicht-Terminal-Symbolen), wobei $\Sigma \cap V = \varnothing$,

(3) $S \in V$ (das Startsymbol),

(4) $R \subseteq (\Sigma \cup V)^+ \times (\Sigma \cup V)^*$ ist eine endliche Menge von Ableitungsregeln $\langle l,r \rangle$ („Ersetzungsregeln“, „Produktionen“; notiert auch als „$l \rightarrow r$“).[6]

D.h. die linke Seite l einer Ableitungsregel darf ein beliebiges, nicht-leeres Wort aus dem Gesamtalphabet enthalten, und die rechte Seite ein beliebiges Wort (einschließlich ε). Die Regel $l \rightarrow r$ besagt, dass l – wo immer es vorkommt – durch

---

[5] Abweichend von den sprachwissenschaftlichen Gepflogenheiten kennzeichen wir hier objektsprachliche Beispiele durch Fettdruck, da wir Kursivdruck bereits für Variablen einschließlich der Nicht-Terminal-Symbole verwenden.

[6] Das Kreuzprodukt $M_1 \times M_2$ zweier Mengen M_1 und M_2 ist die Menge $\{\langle x, x_2 \rangle \mid x_1 \in M_1$ und $x_2 \in M_2\}$.

r ersetzt werden kann. Dabei ist zunächst ganz unerheblich, ob durch die Regel Terminal- oder Nicht-Terminal-Symbole ersetzt werden.[7] D.h. Grammatiken erzeugen Sprachen, die u.U. abzählbar unendlich sind, mit *endlichen* Mitteln durch Rekursion: Ein Teilwort auf der linken Seite wird – ggf rekursiv – durch ein Teilwort auf der rechten Seite ersetzt. Damit sind die durch Grammatiken erzeugbaren Sprachen gerade die rekursiv aufzählbaren Sprachen,[8] und dies sind gerade die semi-entscheidbaren Sprachen, d.h. diejenigen Sprachen, deren Elemente durch einen Algorithmus bestimmbar sind, wenngleich für ein Element aus ihrem Komplement u.U. nicht durch einen Algorithmus entscheidbar ist, dass es nicht zu der Sprache gehört. Wichtig ist außerdem noch die Zerlegung des Gesamtalphabets in Terminal- und Nicht-Terminal-Symbole und die Auszeichung genau eines Nicht-Terminal-Symbols als Startsymbol.

w' ist in $G = \langle \Sigma, V, S, R)$ ***direkt ableitbar*** aus w ($w \Rightarrow_G w'$, kurz auch: $w \Rightarrow w'$) genau dann, wenn es $x, y, y', z \in (\Sigma \cup V)^*$ gibt, so dass gilt: $w = xyz$ und $w' = xy'z$ und $y \rightarrow y' \in R$. Eine ***Ableitung*** in G der Länge n ist eine Folge $w_0 \ldots w_n$, für die gilt: $w_i \in (\Sigma \cup V)^*$ für alle $i = 0, \ldots, n$ und $x_{i-1} \Rightarrow_G x_i$ für alle $i = 1, \ldots, n$. Wir schreiben auch kurz:

$$w_0 \Rightarrow_G w_1 \Rightarrow_G \ldots \Rightarrow_G w_n.$$

Die Ableitungslänge ist also die Anzahl der Ableitungsschritte. Wenn es auf die Anzahl der Schritte nicht ankommt schreiben wir auch kurz:

$$w_0 \Rightarrow^*_G w_n$$

und sagen: w_n ist in G ***(indirekt) ableitbar*** aus w_0. Damit haben wir unser Ziel erreicht und können definieren:

$L(G) := \{w \in \Sigma^* \mid S \Rightarrow^*_G w\}$ heißt die von G ***erzeugte Sprache***.

---

[7] Die Einschränkung auf linke Seiten mit mindestens einem Nicht-Terminal-Symbol sorgt bei Partee u.a. (1993: 436) sowie Klenk (1995: 1580) dafür, dass die linke Seite nicht das leere Wort sein kann. Die Begründung von Klabunde (1998: 43, FN) für diese Einschränkung zusätzlich zum expliziten Ausschluss des leeren Wortes, die er wohl von Klenk (1995: 1580) übernommen hat, ist rein linguistisch motiviert: Wenn aus einer Kette von Terminal-Symbolen eine weitere Kette von Terminalsymbolen ableitbar ist, ist das bei einer allgemeinen Regelgrammatik unerheblich. Die linguistische Motivation für diese Forderung ist, dass die nicht-terminalen Symbole Kategoriensymbole und die terminalen Symbole Lexikoneinträge bzw. Einzellaute sein sollen (s. Klenk 1995: 1580). In der theoretischen Informatik findet man neben Definitionen ohne diese Einschränkung (z.B. Schöning 2001: 13) gelegentlich auch solche mit dieser Einschränkung (z.B. Verbeek 2000: Kap. 7.1). Entscheidend ist, dass in beiden Fällen dieselben Sprachen erzeugt werden: die rekursiv aufzählbaren.

[8] Zu beachten ist der Unterschied zwischen abzählbar und rekursiv aufzählbar: alle Sprachen sind abzählbar, d.h. eindeutig (injektiv) auf die natürlichen Zahlen abbildbar, aber nicht alle Sprachen sind rekursiv aufzählbar (s.u., Abschn. 3). Statt „rekursiv aufzählbar“ wird in der Literatur oft auch abkürzend nur von „aufzählbar“ gesprochen.

D.h. L(G) ist die Menge der aus dem Startsymbol S ableitbaren Wörter über dem Alphabet Σ der Terminalsymbole.

Beispiele:

$G_1 = \langle \{0, 1\}, \{S\}, S, P \rangle$ mit
$P = \{\langle S, 0\rangle, \langle S, 1\rangle, \langle S, 00\rangle, \langle S, 11\rangle, \langle S, 0S0\rangle, \langle S, 1S1\rangle\}$
erzeugt die Sprache der nicht-leeren Palindrome über $\{0, 1\}$, z.B. lässt sich ableiten: $S \Rightarrow 1S1 \Rightarrow 10S01 \Rightarrow 101101$

Hinweis: Die Produktionen von G_1 – allgemein: Regeln mit derselben linken Seite – lassen sich zusammenfassen (BNF = Backus-Naur-Form):
$S \rightarrow 0 \mid 1 \mid 00 \mid 11 \mid 0S0 \mid 1S1$

$G_2 = \langle \{$**Anton, Emil, Pudel, beobachtet, den**$\}, \{S, VP, NP, EN, V, D, N\}, S, P\}$ mit $P =$

(1) $S \rightarrow NP\ \ VP$
(2) $VP \rightarrow V\ \ NP$
(3) $NP \rightarrow EN \mid D\ \ N$
(4) $V \rightarrow$ **beobachtet**
(5) $EN \rightarrow$ **Anton** | **Emil**
(6) $D \rightarrow$ **den**
(7) $N \rightarrow$ **Pudel**

erzeugt Wörter (Sätze) wie **Anton beobachtet den Pudel**.

Wenn man die Regeln durchnummeriert, kann man bei den Ableitungsschritten auch die Nummer der jeweiligen Regel angeben.

Übungen:

(1) $\{w \in \{0, 1\}^3 \mid w$ beginnt mit 0 $\} =$
(2) Geben Sie einige Elemente der folgenden Sprachen an:
$\{0^n1^n \mid n \in \mathbb{N}\}$
$\{w \in \{0, 1\}^* \mid w^R = w\}$ (die Sprache der Palindrome)
$L_1 = \{0^n1^n \mid n \in \mathbb{N}\} \cdot \{2^n \mid n \in \mathbb{N}\}$
$L_2 = \{2^n \mid n \in \mathbb{N}\} \cdot \{0^n1^n \mid n \in \mathbb{N}\}$
$L_3 = (\{0^n1^n \mid n \in \mathbb{N}\} \cdot \{2^n \mid n \in \mathbb{N}\})^*$
$L_4 = (\{2^n \mid n \in \mathbb{N}\} \cdot \{0^n1^n \mid n \in \mathbb{N}\})^*$
Überprüfen Sie: $L_1 = L_2$? $L_3 = L_4$?
(3) Kann eine nicht-semi-entscheidbare Sprache wohlbestimmt (eindeutig abgegrenzt) sein?
(4) Geben Sie Ableitungen für 01100100110 in G_1 und für **den Pudel beobachtet Anton** in G_2 an.
(5) Wandeln Sie G_1 so ab, dass auch das leere Palindrom erzeugt wird.
(6) Wieviele Ableitungen für **Anton beobachtet Emil** gibt es in G_2?
(7) Lässt sich in G_2 auch **Anton beobachtet den Emil** ableiten?

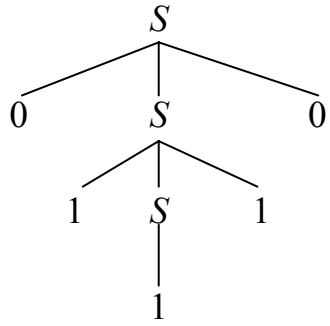
(8) Überprüfen Sie, ob L(G_2) = L(G_3), wobei
G_3 = ⟨{**Anton, Emil, Pudel, beobachtet, den**}, {*S, V, NP, EN, V, D, N*}, *S*, *P*}
mit *P* =

(1) $S \rightarrow NP\ \ V\ \ NP$
(2) –
(3) $NP \rightarrow EN \mid D\ \ N$
(4) $V \rightarrow$ **beobachtet**
(5) $EN \rightarrow$ **Anton | Emil**
(6) $D \rightarrow$ **den**
(7) $N \rightarrow$ **Pudel**

D.h. G_2 und G_3 unterscheiden sich nur bei den beiden ersten Ableitungsregeln.
(*Hinweis:* Gleichheit von Mengen überprüft man am besten, indem man die beiden Beziehungen $\subseteq$ und $\supseteq$ überprüft. Anstelle *alle* Elemente von L(G_2) bzw. L(G_3) zu betrachten, können Sie sich auch auf die relevanten Ableitungen beschränken.)

(9) Überprüfen Sie, ob bei dem Ausgangsbeispiel tatsächlich L(G) = L gilt. Ist L eine Teilmenge des Deutschen?

Bei Grammatiken, in denen keine Mehrfachersetzungen vorkommen, wo also in jeder Regel nur ein Symbol der linken Seite ersetzt wird, können zu den Ableitungen ***Ableitungsbäume*** (***Syntaxbäume***) konstruiert werden:[9] Bei jedem Ableitungsschritt wird das zu ersetzende Symbol auf der linken Seite der Regel, die in diesem Schritt angewendet wird, als Ausgangsknoten ('Vaterknoten') und die ersetzenden Symbole auf der rechten Seite als 'Söhne' (oder: 'Töchter') aufgefasst. Z.B. entspricht der Ableitung $S \Rightarrow 0S0 \Rightarrow 01S10 \Rightarrow 01110$ in G_1 der folgende Ableitungsbaum:



D.h. durch eine Ableitung wird dem resultierenden Wort – einer Kette von Terminalsymbolen – eine Struktur aufgeprägt; und umgekehrt lässt sich jeder Ableitungsschritt als 'Zerlegung in Konstituenten des resultierenden Wortes' auffassen. Ein ***Baum*** wird dabei graphentheoretisch verstanden als Paar ⟨*K*, *E*⟩, wobei *K* eine

[9] Auf diese Voraussetzung weisen z.B. Partee u.a. (1993: 513) hin. Ableitungsbäume werden daher oft nicht allgmein, sondern im Zusammenhang mit Grammatiken eingeführt, bei denen diese Voraussetzung immer erfüllt ist (s.u., Abschn. 3).

endliche Menge von (etikettierten) Knoten und $E \subseteq K \times K$ eine endliche Menge von Knotenpaaren („Kanten“) ist, für die gilt: Es gibt genau einen Knoten ohne Vorgänger (die ***Wurzel*** des Baumes), und für jeden anderen Knoten *k* gibt es genau einen Pfad von der Wurzel zu *k* (vgl. z.B. Klenk 1995: 1587). Das Beispiel zeigt, dass der Baum von oben (der Wurzel) nach unten ‘wächst’, wobei die Richtung der Kanten üblicherweise nur durch die vertikale Orientierung der Graphik und nicht durch Pfeile dargestellt wird.

Ein Baum lässt sich immer ohne überkreuzende Kanten darstellen und an den ***Blättern*** (= den Knoten ohne Nachfolger) kann man dann von links nach rechts das abgeleitete Wort ablesen. D.h. in einem Baum in überkreuzungsfreier Darstellung sind drei Arten von Informationen repräsentiert:
– der hierarchische Aufbau des resultierenden Wortes aus Teilen
– der Typ der Teile
– die Reihenfolge der Teile des Wortes

Übungen:

(1) Warum können bei Grammatiken mit Mehrfachersetzungen keine Ableitungsbäume konstruiert werden? – Betrachten Sie z.B. eine Grammatik, die eine Regel wie $ABC \rightarrow AD$ enthält.

(2) Welche Information geht beim Übergang von den Ableitungen zu den Ableitungsbäumen verloren? – Betrachten sie z.B. die Bäume zu den Ableitungen in Aufg. 6 aus dem vorhergehenden Übungsblock.

(3) Betrachte die Grammatik $G = \{\{a\}, \{S\}, S, \{S \rightarrow SS, S \rightarrow a\}\}$. Welche Bäume können der Ableitung $S \Rightarrow SS \Rightarrow SSS \Rightarrow aSS \Rightarrow aaS \Rightarrow aaa$ in *G* zugeordnet werden?[10]

3 Die Chomsky-Hierarchie

Allgemeine Regelsprachen, d.h. Sprachen, die durch eine allgemeine Regelgrammatik erzeugt werden können, sind ***rekursiv aufzählbar***: Alle Wörter der Sprache können von einer Turingmaschine in endlich vielen Schritten erkannt („akzeptiert“) werden. Allerdings ist das Komplement einer solchen Sprache u.U. nicht rekursiv aufzählbar, d.h. die Wörter über dem zugrundeliegenden Alphabet, die nicht zu der Sprache gehören, werden von keiner Turingmaschine erkannt.[11] In diesem Fall ist das Wortproblem für diese Sprache und damit diese Sprache selbst ***nicht entscheidbar***. Psycholinguistische und methodologische Argumente dafür, dass natürliche Sprachen als entscheidbar angesehen werden sollten, führt Levelt

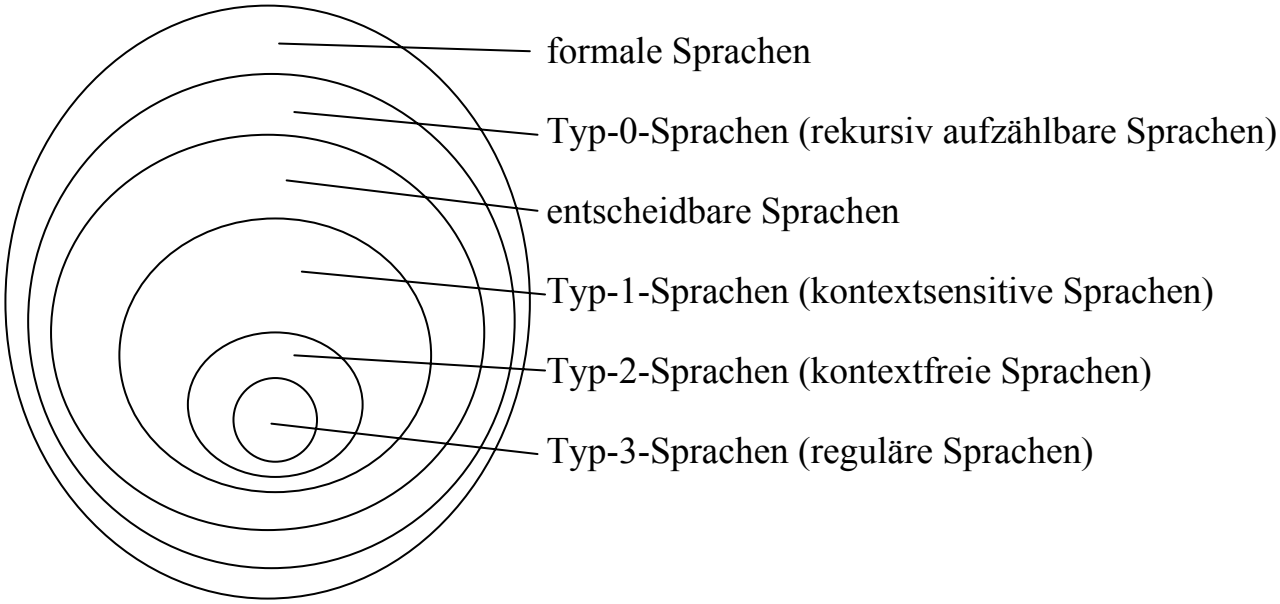
[10] Das Beispiel stammt aus Klenk (1995: 1588).

[11] Einzelheiten zu den klassischen Beispielen z.B. in Partee u.a. (1993: 520–523), Klenk (1995: 1581f.), Wegener (1996: 21–23).

(1974 [Bd. 2]: 39–41) an.[12] Die entscheidbaren Sprachen lassen sich allerdings durch kein formales Kriterium – d.h. durch keinen formal beschreibbaren Grammatiktyp – charakterisieren (Klenk 2003: 79). Während Levelt (1974 [Bd. 2]: 39) bezweifelt, dass die Klasse der natürlichen Sprachen allgemein noch weiter eingeschränkt werden kann (auf die Klasse der sog. ***kontextsensitiven*** Sprachen), gehen andere davon aus, dass natürliche Sprachen nicht nur als kontextsensitiv, sondern weitgehend[13] sogar als ***kontextfrei*** beschrieben werden können (vgl. z.B. Klenk 1995: 1599; Partee u.a. 1993: 501–503, 533).

Die Frage, welchem Typ formaler Sprachen die natürlichen Sprachen zuzuordnen sind, hat in der Geschichte der Generativen Grammatik eine zentrale Rolle gespielt: Als sich herausstellte, dass die durch Transformationsgrammatiken erzeugbaren Sprachen gerade die allgemeinen Regelsprachen sind, wurden Transformationen als formales Beschreibungsmittel von einer Reihe von Forschern aufgegeben (vgl. Klenk 2003: 79f., ausführlich: Partee u.a. 1993: 554–557).[14]

Die verschiedenen Typen formaler Sprachen bilden eine Hierarchie (Folge) von (echten) Teilmengen, die sog. Chomsky-Hierarchie:[15]



[12] Grundsätzliche Zweifel an der algorithmischen Beschreibbarkeit von natürlichen Sprachen – mit Blick auf Sprachwandel und andere Arten sprachinterner Variabilität – hegt hingegen Hockett (1967: 8–10).

[13] Den ersten allgemein akzeptierten Nachweis, dass es nicht-kontextfrei beschreibbare natürliche Sprachen gibt, hat Shieber (1985) geführt.

[14] Bemerkenswert ist, dass Transformationen in verallgemeinerten Form dann in der von Chomsky geprägten Richtung der Generativen Grammatik mit dem von ihm entworfenen Minimalistischen Programm erneut eingeführt wurden (dazu kritisch Johnson/Lappin 1997).

[15] Die Einordnung der entscheidbaren Sprachen geht nicht auf Chomsky zurück.

Bis auf die Klasse der entscheidbaren Sprachen ergeben sich alle Typen durch spezifische Bedingungen bei den jeweils zulässigen Ableitungsregeln in den erzeugenden Grammatiken (bei Typ-0-Grammatiken gibt es keine weiteren Einschränkungen (s.o.)):

(i) Eine Grammatik G ist vom ***Typ 1*** oder ***kontextsensitiv*** oder ***(längen)-monoton*** gdw wenn für alle Ableitungsregeln $w_1 \rightarrow w_2$ in G gilt: $|w_1| \leq |w_2|$, wobei S in w_2 nicht vorkommt, oder $w_1 = S$ und $w_2 = \varepsilon$.[16]

(ii) Eine Grammatik $G = \langle \Sigma, V, S, R \rangle$ ist vom ***Typ 2*** oder ***kontextfrei*** gdw alle Regeln die Form $A \rightarrow w$ haben mit $A \in V$ und $w \in (\Sigma \cup V)^*$.

(iii) Eine Grammatik G ist vom ***Typ 3*** oder ***regulär*** oder ***rechts-linear*** gdw alle Regeln die Form $A \rightarrow w$ haben mit $A \in V$ und $w = \varepsilon$ oder $w = aB$ mit $a \in \Sigma$ und $B \in V$.

Typ-1-Grammatiken heißen kontextsensitiv, weil Regeln der Form $uAv \rightarrow uwv$, d.h. die Ersetzung von A durch w in der Umgebung u_v, zugelassen sind. Anstelle von „$uAv \rightarrow uwv$“ schreibt man daher auch „$A \rightarrow w / u\_v$“. Typ-1-Grammatiken heißen (längen)monoton, weil die Länge eines resultierenden nicht-leeren Wortes gleich oder größer der Länge des Ausgangswortes ist.

Typ-2-Grammatiken heißen kontextfrei, weil immer nur genau eine Variable ersetzt wird und dabei der Kontext unberücksichtigt bleibt.

Eine Typ-3-Grammatik heißt rechts-linear, weil bei dem entsprechenden Ableitungsbaum immer nur der rechte Zweig expandiert wird. Reguläre Sprachen sind in gewisser Hinsicht die ‘einfachsten’ Sprachen: zu ihrer Erzeugung genügen schon sehr einfache Automaten. Reguläre Grammatiken sind allerdings offensichtlich ungeeignet, um die Struktur von natürlichsprachlichen Ausdrücken zu beschreiben.

Die Grammatiktypen werden in den verschiedenen Darstellungen z.T. etwas unterschiedlich definiert, was (i) den Einschluss bzw. Ausschluss von ε in der erzeugten Sprache betrifft sowie (ii) weitere Beschränkungen bei den Typ-1-Grammatiken (auf Regeln mit w_1, die nur Nicht-Terminale enthalten oder auf Regeln, bei denen genau ein Nicht-Terminal-Symbol von w_1 ersetzt wird) und (iii) engere oder weitere Fassungen der Beschränkung bei den regulären Sprachen (Zulassen von $A \rightarrow wB$ mit $w \in \Sigma^*$). Entscheidend ist, dass in allen Fällen einander entsprechende Typen von Sprachen erzeugt werden.[17]

---

[16] Kommt $S \rightarrow \varepsilon$ in G nicht vor, dann ist S in einem w_2 unschädlich. – „$|w|$“ := die Länge von w.

[17] Wenn ε bei einem Sprachtyp nicht als Element der Sprache zugelassen wird, führt das u.U. – wie z.B. bei Partee u.a. (1993: 450) – zu einer unnötigen Verkomplizierung der Chomsky-Hierarchie. Wie man Grammatiken, die ε nicht erzeugen, systematisch in Grammatiken umformen kann, die ε erzeugen, wird sehr übersichtlich in Schöning (2001: 18f.) gezeigt.

Typische (und klassische) Beispiele für die einzelnen Sprachtypen:

regulär / Typ-3: $\{a^n \mid a \in \Sigma\}$

kontextfrei / Typ-2 (aber nicht regulär): $\{a^n b^n \mid a, b \in \Sigma\}$

kontextsensitiv / Typ-1 (aber nicht kontextfrei): $\{a^n b^n c^n \mid a, b, c \in \Sigma\}$

Von besonderer Bedeutung ist jeweils, welche Operationen den Sprachtyp nicht verändern (sog. ***Abschlusseigenschaften*** der einzelnen Sprachtypen):[18]

	Vereinigung	Produkt	Stern	Schnitt	Komplement	Differenz	Umkehrung (L^R)
Typ-3	+	+	+	+	+	+	+
Typ-2	+	+	+	–	–	–	
Typ-1	+	+	+	+	+		
entscheidbare Sprachen	+			+	+	+	
Typ-0	+	+	+	+	–	–	

Da Vereinigung, Konkatenation und Stern die Regularität einer Sprache erhalten, heißen diese Operationen auch reguläre Operationen.

Die Klasse der regulären Sprachen lässt sich mit Bezug auf diese Abschlusseigenschaften charakterisieren:[19]

L ist eine reguläre Sprache über einem Alphabet Σ gdw (i) oder (ii) oder (iii) oder (iv) oder (v) gilt:

(i) $\varnothing$ ist eine reguläre Sprache.

(ii) Ist $a \in \Sigma$, so ist $\{a\}$ eine reguläre Sprache.

(iii) Sind L_1 und L_2 reguläre Sprachen, so auch $L_1 \cup L_2$.

(iv) Sind L_1 und L_2 reguläre Sprachen, so auch $L_1 L_2$.

(v) Ist L_1 eine reguläre Sprache, so auch L_1^*.

Aus (i) und (v) folgt z.B., dass auch $\{\varepsilon\} = \varnothing^*$ eine reguläre Sprache ist, und in Verbindung mit (ii) und (iii) folgt hieraus, dass $\{w\}$ für jedes $w \in \Sigma^*$ eine reguläre Sprache ist. Aus dieser Charakterisierung der regulären Sprachen ergibt sich, dass sie durch sog. ***reguläre Ausdrücke*** wie a*b oder a?b beschrieben werden können, wobei „*“ für ein beliebig langes Wort über dem Ausgangsalphabet und „?“ für genau ein Symbol aus dem Ausgangsalphabet steht. Solche Ausdrücke werden regelmäßig für die bequeme Suche in Datenbanken – z.B. den Bibliothekskatalogen – benötigt.

---

[18] Vgl. Klabunde (1998: 142), Schöning (2001: 89); für freigelassene Felder finden sich in der herangezogenen Literatur keine Angaben.

[19] Vgl. z.B. Partee u.a. (1993: 463) oder Klabunde (1998: 41) (Klabunde verwischt allerdings den Unterschied zwischen regulären Sprachen und regulären Ausdrücken).

4 Ausblick

In der Generativen Grammatik sind weitere Formalismen, die die hier besprochenen Regeltypen ergänzen, entwickelt worden, vgl. z.B. Klenk (1995: 1599) zu Grammatiken mit Metaregeln, 2-Schichten-Grammatiken und Transformationsgrammatiken, Partee u.a. (1993: 533ff.) zu indizierten Grammatiken, Baum-adjungierenden (‘tree adjoining’) Grammatiken, Kategorial- und Transformationsgrammatiken sowie Levelt (1974 [Bd.1]: 35ff.) zu probabilistischen Grammatiken.

5 Lösungshinweise zu ausgewählten Übungen

S. 3:
(1) wb = abb
(2) ww' = abc
(3) w'^3w^2 = cccabab
(4) $|ww''|$ = 5
(5) Σ^2 = {aa, bb, cc, ab, ac, ba, bc, ca, cb}
(6) $\varnothing^*$ = {ε}
(7) (a) w, (b) f, (c) w, (d) w, (e) w, (f) f
(8) (a) w, (b) w, (c) w, (d) f

S. 7f.:
(1) … = {000, 010, 001, 011}
(2) $L_1 \neq L_2$, denn z.B. 012 $\in L_1$, aber nicht 012 $\in L_2$.
für $\mathbb{N}$ = {1, …} gilt $L_3 \neq L_4$; für $\mathbb{N}$ = {0, 1, …} gilt: $L_3 = L_4$ (nachrechnen!)
(3) ja: durch nicht-syntaktische Eigenschaften
(4) –
(5) P' = {⟨S, 0⟩, ⟨S, 1⟩, ⟨S, ε⟩, ⟨S, 0S0⟩, ⟨S, 1S1⟩} erzeugt auch das leere Palindrom (wie werden jetzt 00 und 11 erzeugt?).
(6) 33 (wenn ich mich nicht verzählt habe)
(7) nein
(8) ja
(9) –; ja: in fiktionalen Texten wird keine andere Sprache gebraucht, sondern eine andersartige Welt entworfen

S. 9:
(1) Bei Bäumen dürfen getrennte Äste nicht wieder zusammengeführt werden.
(2) Die Reihenfolge, in der die Regeln angewendet werden. Die zugrundeliegenden Ableitungen sind also aus den Bäumen nicht rekonstruierbar.

(3) Das Beispiel zeigt, daß es auch umgekehrt zu einer Ableitung mehrere Bäume geben kann, da hier dieselbe Regel auf verschiedene Symbolvorkommen angewendet werden kann (2. Ableitungsschritt). Auf welches Symbol eine Regel angewendet wird, ist in den Bäumen, aber nicht in den Ableitungen repräsentierbar.

6 Literatur

Arbeiten aus der Theoretischen Informatik:

Hinweis: Jede Einführung in die Theoretische Informatik enthält eine Einführung in die Theorie der formalen Sprachen, oft auch als Vorlesungsskript im Netz.

Schöning, Uwe (2001): *Theoretische Informatik – kurzgefasst.* 4. Aufl. Heidelberg / Berlin: Spektrum Akademischer Verlag.

Verbeek, Rutger (2000): *Einführung die Theoretische Informatik B*. Skript der Fernuniversität Hagen.

Wegener, Ingo (1996): *Kompendium Theoretische Informatik – eine Ideensammlung.* Stuttgart: Teubner.

Wegener, Ingo (1999): *Theoretische Informatik.* Eine algorithmenorientierte Einführung. Stuttgart / Leipzig: Teubner.

Sprachwissenschaftliche Arbeiten:

Hockett, Charles F. (1967): *Language, Mathematics and Linguistics*. The Hague / Paris: Mouton.

Johnson, David / Shalom Lappin (1997): „A critique of the Minimalist Program“ in: *Linguistics and Philosophy* 20, S. 273–333.

Klabunde, Ralf (1998): *Formale Grundlagen der Linguistik*. Ein Arbeitsbuch. Tübingen: Narr. [Umfassende, gut lesbare Darstellung mit Aufgaben zu formalen Sprachen und den korrespondierenden Berechnungsmaschinen (Turingmaschinen).]

Klenk, Ursula (1995): „Formale Sprachen“ in: S. 1576–1605. [Mit einem kurzen geschichtlichen Überblick.]

Klenk, Ursula (2003): *Generative Syntax*. Tübingen: Narr. [Einordnung in die linguististische Theoriebildung, aber ohne systematische Entwicklung.]

Levelt, W[illem] J[ohannes] M[aria] (1974): *Formal Grammars in Linguistics and Psycholinguistics*. Bd. 1: An Introduction to the Theory of Formal Languages and Automata. – Bd. 2: Applications in Linguistic Theory. – Bd. 3: Psycho-linguistic Applications. [Gut lesbare Darstellung, die wichtige Aspekte beleuchtet, die in den übrigen Arbeiten nicht (mehr) angesprochen werden.]

Montague, Richard (1970): „English as a formal language“, wiederabgedruckt in ders. (1974): *Formal Philosophy*, hg. von R. H. Thomason.

Partee, Barbara H. / Alice ter Meulen / Robert E. Wall (1993): *Mathematical Methods in Linguistics*. 2., corr. printing of the first ed. Dordrecht / Boston / London: Kluwer. [Teil E]

Shieber, Stuart M. (1985): „Evidence against the context-freeness of natural languages“ in: *Linguistics and Philosophy* 8, S. 333–343.