Theorie der formalen Sprachen

Sonderegger Katja

5. Juni 2013

Kurzfassung

Computer verwenden zur Verarbeitung von Daten und Informationen künstliche, formale Sprachen (Assemblersprachen, Programmiersprachen, Wissensrepräsentationssprachen, usw). Der Vorteil formaler Sprachen gegenüber natürlicher Sprachen liegt vor allem in der exakten Definition der zulässigen Ausdrücke und deren Bedeutung.[1]

1 Einleitung

Dieses Dokument dient dazu, einen groben Überblick über formale Sprachen, deren Aufbau (Grammatiken) und Nutzen zu erhalten. Zuerst werden in Kapitel 2 bestimmte Begriffe genauer erläutert, um eine Grundlage zu schaffen auf der im Kapitel 3 Grammatiken und in Folge formale Sprachen definiert werden. Anschließend wird noch genauer auf die verschiedenen Formen formaler Sprachen eingegangen. Für detailliertere Ausführungen und Erklärungen wird auf [2] und [3] verwiesen.

2 Alphabete, Wörter und Sprachen

Um formale Sprachen genauer zu definieren, müssen zunächst einige Begriffe geklärt werden:

- ein *Alphabet* Σ ist eine endliche, nicht leere Menge von Zeichen.
- eine *Zeichenreihe* (auch Wort genannt) ist eine endliche Folge von Zeichen über einem Alphabet Σ. Es gibt auch das sogenannte *Leerwort* [1], welches das kleinste Wort darstellt, da es keine Zeichen enthält.

In natürlichen Sprachen hat ein Wort eine Länge - dies ist auch in den formalen Sprachen so - wobei die Länge eines Wortes wie folgt definiert ist:

- die *Länge* eines Wortes w ist die Anzahl Positionen in w.

  Beispiel Das Wort w besteht aus den Zeichen $w = a_1a_2a_3a_4a_5...a_i$.
  Die Anzahl Positionen des Wortes w ist damit i.

  Die Länge des Wortes w wird auch geschrieben als: |w|. Das Leerwort ϵ hat die Wortlänge 0.

- Die Schreibweise Σ^i ($i \in \mathbb{N}$) bezeichnet die Menge der Zeichenreihen mit der Wortlänge = i, während die Schreibweise Σ^* die Menge aller Wörter über dem Alphabet Σ bezeichnet.
- Wenn wir zwei Wörter v und w zu einem neuen Wort zusammenfügen wollen, dann verwenden wir hierfür die *Konkatenation* (formal: v·w [2]). Die Wortlänge von zwei miteinander konkatenierten Wörtern v und w ist definiert durch:

[1] wird dargestellt mit Hilfe von: ϵ

[2] · wird normalerweise nicht mit angeschrieben

$$|\mathrm{v} \cdot \mathrm{w}| = |\mathrm{vw}| = |\mathrm{v}| + |\mathrm{w}|.$$

▷ Dabei ist zu beachten, dass die Konkatenation mit dem Leerwort ϵ das ursprüngliche Wort nicht verändert. Also $\epsilon w = w\epsilon = w$.

Nun, da wir grundlegende Begriffe kennen gelernt haben, können wir den Begriff der Sprache definieren:

▷ Eine *formale Sprache L* über einem Alphabet Σ ist eine Menge von Wörtern über Σ und damit eine Teilmenge von Σ^* [3].

▷ Für formale Sprachen gelten einige der Mengenoperationen, wie die Vereinigung, das Komplement, der Durchschnitt und das Produkt. Das Produkt ist für formale Sprachen assoziativ.

▷ Auch bei den formalen Sprachen gibt es die Potenznotation. L^{k} ist die k-te Potenz von L.

▷ Man definiert den *Abschluss von L* (auch Kleene-Stern * genannt) wie folgt: [4]

$$L^* := \bigcup_{k \geq 0} L^{\mathrm{k}} = \{x_1 ... x_k | x_1, ..., x_k \in L \, und \, k \in \mathbb{N}, k \geq 0\}$$

3 Grammatiken und formale Sprachen

Bevor hier genauer auf Grammatiken eingegangen wird, noch kurz ein Hinweis darauf, in welchem Bereich formale Sprachen und Grammatiken eigentlich verwendet werden: Grammatiken werden meistens im Zusammenhang mit Parsern verwendet. Parser sind Teil eines Compilers und arbeiten mit einer rekursiven Struktur. Sie legen fest wie die Sprache aufgebaut ist und welche Kombinationen erlaubt sind. Für genauere Informationen zu Compilern und deren Phasen wird auf [2] verwiesen.

3.1 Aufbau einer Grammatik

Eine *Grammatik* G besteht aus:

▷ einer endlichen Menge von Variablen V [5],

▷ einem Alphabet Σ [6],

▷ einer endlichen Menge von Regeln R,

▷ dem Startsymbol S der Grammatik G (wobei S $\in$ V).

Somit ist eine Grammatik G ein Quadrupel, das wie folgt definiert ist:

$$G = (V, \Sigma, R, S)$$

Hier ist eine *Regel R* definiert als ein Paar P (*Prämisse*) $\rightarrow$ Q (*Konklusion*), wobei P und Q sowohl Teil der Variablen als auch Teil des Alphabets sind und P mindestens eine Variable enthält.

Bevor wir nun zu den verschiedenen Sprachen und Grammatiken kommen, müssen wir noch klären, wann ein Wort w aus $(V \cup \Sigma)^*$ *ableitbar* von einem anderen Wort v über derselben Menge ist. Dies ist genau dann der Fall, wenn eine Zahl n $\in \mathbb{N}$ existiert, sodass gilt:

$$v = u_0 \implies u_1 \implies ... \implies u_n = w \quad [7]$$

Das heißt wir kommen in n Schritten vom Wort v mit Hilfe der Grammatik G zum Wort w. Wenn unsere natürliche Zahl n=1 ist, dann bedeutet das für unser Beispiel, dass v = w ist.

[3] sowohl Σ^* als auch die leere Menge {} können Sprachen sein
[4] L^+ ist äquivalent dazu, nur dass k nicht bei 0 sondern bei 1 beginnt
[5] auch Nichtterminale genannt
[6] auch Terminale genannt - V $\cap \, \Sigma = \emptyset$
[7] wobei $u_0 ... u_n$ Wörter aus $(V \cup \Sigma)^*$ sind

Ein *Satz* ist ein Terminalwort, das vom Startsymbol S ableitbar und Element von Σ^* ist.

Die Sprache einer Grammatik G ist die Menge aller Sätze

$$L(G) := \{w \in \Sigma^* | \text{w aus S in G ableitbar}\}$$

- wobei hier L(G) auch *die von G erzeugte Sprache* genannt wird.

3.2 Verschiedene Arten von Grammatiken

Man unterscheidet Grammatiken normalerweise durch die - in der Grammatik G zugelassenen - Regeln. Somit entstehen folgende Klassen:

rechtslinear ist eine Grammatik G genau dann, wenn für alle Regeln P → Q gilt, dass P ∈ der Nichtterminale V von G ist und Q ∈ $\Sigma^* \cup \Sigma^+ V$ [8]

kontextfrei ist eine Grammatik G genau dann, wenn für alle Regeln P → Q gilt, dass P ∈ der Nichtterminale V von G ist und Q ∈ $(\Sigma \cup V)^*$

kontextsensitiv ist eine Grammatik G genau dann, wenn für alle Regeln P → Q gilt, dass:

- ▷ es Wörter u, v, w aus $(V \cup \Sigma)^*$ gibt, sodass P = uAv (wobei A ∈ V) und $Q = uwv$ (wobei $|w| \geq 1$)

  oder

- ▷ P = Startsymbol S und Q = ϵ [9]

beschränkt ist eine Grammatik G genau dann, wenn für alle Regeln P → Q gilt, dass

- ▷ $|P| \leq |Q|$

  oder

- ▷ P = Startsymbol S und Q = ϵ [10]

3.3 Formale Sprachen

Wie Grammatiken, können auch formale Sprachen in Klassen unterteilt werden:

Typ 3 - regulär ist eine formale Sprache L genau dann, wenn es eine rechtslineare Grammatik G gibt, sodass L = L(G) gilt.

Typ 2 - kontextfrei ist eine formale Sprache L genau dann, wenn es eine kontextfreie Grammatik G gibt, sodass L = L(G) gilt.

Typ 1 - kontextsensitiv ist eine formale Sprache L genau dann, wenn es eine kontextsensitive Grammatik G gibt, sodass L = L(G) gilt.

beschränkt ist eine formale Sprache L genau dann, wenn es eine beschränkte Grammatik G gibt, sodass L = L(G) gilt.

Typ 0 - rekursiv aufzählbar ist eine formale Sprache L genau dann, wenn es eine Grammatik G gibt, sodass L = L(G) gilt.

[8] $\Sigma^+ = \Sigma \cdot (\Sigma^*)$ (Σ kommt mindestens einmal vor)
[9] Grammatik G enthält die Regel S → ϵ und damit kommt S in keiner Konklusion vor
[10] Grammatik G enthält die Regel S → ϵ und damit kommt S in keiner Konklusion vor

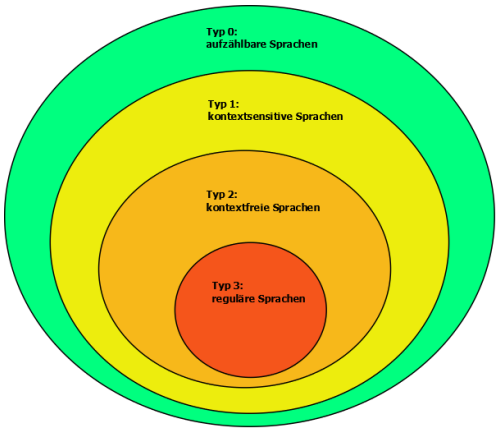

Abbildung 1: Chomsky-Hierarchie
`http://www.schulen.regensburg.de/wvsgym/images/Faecher/Informatik/Informatik_12/Bilder/1_3_Endliche_Automaten/chomsky.png`

In Abbildung 1 sieht man die Unterteilung der einzelnen Sprachen. Diese Unterteilung wird auch Chomsky-Hierarchie genannt (wobei beschränkte Sprachen nicht erwähnt werden, da sie in den rekursiv aufzählbaren Sprachen enthalten sind). Die Chomsky-Hierarchie zeigt auf, dass zum Beispiel jede reguläre Sprache eine kontextfreie Sprache ist. Gleichzeitig sieht man, dass nicht jede kontextfreie Sprache auch eine reguläre Sprache ist. [11]

4 Schlussfolgerung

Formale Sprachen sind in vielen Bereichen der Informatik von großer Bedeutung. Vor allem reguläre und kontextfreie Sprachen sind weit verbreitet.

Reguläre Sprachen werden mit Hilfe von endlichen Automaten und regulären Ausdrücken betrachtet. Die Anwendungsbereiche von regulären Sprachen sind sehr vielfältig und reichen vom Entwurf und Testen eines digitalen Schaltkreises bis hin zur Logik der Kontrolle von Spielfiguren in Computerspielen.

Kontextfreie Sprachen sind wichtige Bestandteile im Compilerbereich und werden unter anderem auch bei Parsergeneratoren verwendet. Das typischste Beispiel einer kontextfreien Sprache, ist die Sprache der Palindrome. Diese ist eine kontextfreie Sprache, aber keine reguläre Sprache.

Literatur

[1] Prof. Dr. Ralf Der. Kapitel 7 - Formale Sprachen und Grammatiken, 2002. Folien zur Vorlesung Digitale Informationsverarbeitung - Universität Leipzig.

[2] Helmut Herold, Bruno Lurz und Jürgen Wohlrab. *Grundlagen der Informatik : praktisch - technisch - theoretisch.* München ; Boston [u.a.] : Pearson Studium, 2007.

[3] Georg Moser. Einführung in die Theoretische Informatik, 2011. Skriptum zur Vorlesung, Universität Innsbruck, WS 2011.

[11] Die Chomsky-Hierarchie geht auf den Linguist Noam Chomsky zurück.