15 REGULÄRE AUSDRÜCKE UND RECHTSLINEARE GRAMMATIKEN

Am Ende von Einheit 14 über endliche Automaten haben wir gesehen, dass manche formale Sprachen zwar von kontextfreien Grammatiken erzeugt, aber nicht von endlichen Akzeptoren erkannt werden können. Damit stellt sich für Sie vielleicht zum ersten Mal die Frage nach einer *Charakterisierung*, nämlich der der mit endlichen Akzeptoren erkennbaren Sprachen. Damit ist eine präzise Beschreibung dieser formalen Sprachen gemeint, die nicht (oder jedenfalls nicht offensichtlich) Automaten benutzt.

In den beiden Abschnitten dieser Einheit werden Sie zwei solche Charakterisierungen kennenlernen, die über reguläre Ausdrücke und die über rechtslineare Grammatiken.

15.1 REGULÄRE AUSDRÜCKE

Der Begriff *regulärer Ausdruck* geht ursprünglich auf Stephen Kleene (1956) zurück und wird heute in unterschiedlichen Bedeutungen genutzt. In dieser Einheit führen wir kurz regulären Ausdrücken nach der „klassischen" Definition ein.

Etwas anderes (nämlich allgemeineres) sind die Varianten der *Regular Expressions*, von denen Sie möglicherweise schon im Zusammenhang mit dem ein oder anderen Programm (emacs, grep, sed, ...) oder der ein oder anderen Programmiersprache (Java, Python, Perl) gelesen haben. Für Java gibt es das Paket java.util.regex. Regular expressions sind eine deutliche Verallgemeinerung regulärer Ausdrücke, auf die wir in dieser Vorlesung nicht eingehen werden. Alles was wir im folgenden über reguläre Ausdrücke sagen, ist aber auch bei regular expressions anwendbar.

Wir kommen direkt zur Definition regulärer Ausdrücke. Sie wird sie hoffentlich an das ein oder andere aus der Einheit 6 über formale Sprachen und die zugehörigen Übungsaufgaben erinnern.

regulärer Ausdruck

Es sei A ein Alphabet, das keines der fünf Zeichen aus $Z = \{|, (,), *, \emptyset\}$ enthält. Ein *regulärer Ausdruck* über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt. Die Menge der regulären Ausdrücke ist wie folgt festgelegt:

- $\emptyset$ ist ein regulärer Ausdruck.
- Für jedes $a \in A$ ist a ein regulärer Ausdruck.
- Wenn R_1 und R_2 reguläre Ausdrücke sind, dann sind auch $(R_1|R_2)$ und (R_1R_2) reguläre Ausdrücke.
- Wenn R ein regulärer Ausdruck ist, dann auch $(R*)$.
- Nichts anderes sind reguläre Ausdrücke.

Um sich das Schreiben zu vereinfachen, darf man Klammern auch weglassen. Im Zweifelsfall gilt „Stern- vor Punkt- und Punkt- vor Strichrechnung", d. h. $R_1|R_2R_3*$ ist z. B. als $(R_1|(R_2(R_3*)))$ zu verstehen. Bei mehreren gleichen binären Operatoren gilt das

als links geklammert; zum Beispiel ist $R_1|R_2|R_3$ als $((R_1|R_2)|R_3)$ zu verstehen.

Man kann die korrekte Syntax regulärer Ausdrücke auch mit Hilfe einer kontextfreien Grammatik beschreiben: Zu gegebenem Alphabet A sind die legalen regulären Ausdrücke gerade die Wörter, die von der Grammatik

$$\begin{aligned} G &= (\{R\},\ \{|, (,), *, \emptyset\} \cup A,\ R,\ P) \\ \text{und } P &= \{R \to \emptyset,\ \ R \to (R|R),\ \ R \to (RR),\ \ R \to (R*)\} \\ &\quad \cup \{R \to a \mid a \in A\} \end{aligned}$$

erzeugt werden.

Die folgenden Zeichenketten sind alle reguläre Ausdrücke über dem Alphabet $\{0, 1\}$:

(a) $\emptyset$	(b) 0	(c) 1
(d) (01)	(e) $((01)0)$	(f) $(((01)0)0)$
(g) $((01)(00))$	(h) $(\emptyset\|1)$	(i) $(0\|1)$
(j) $((0(0\|1))\|1)$	(k) $(0\|(1\|(0\|0)))$	(l) $(\emptyset*)$
(m) $(0*)$	(n) $((10)(1*))$	(o) $(((10)1)*)$
(p) $((0*)*)$	(q) $(((((01)1)*)*)\|(\emptyset*))$	

Wendet man die Klammereinsparungsregeln an, so ergibt sich aus den Beispielen mit Klammern:

(d) 01	(e) 010	(f) 0100
(g) $01(00)$	(h) $\emptyset\|1$	(i) $0\|1$
(j) $0(0\|1)\|1$	(k) $(0\|(1\|(0\|0)))$	(l) $\emptyset*$
(m) $0*$	(n) $101*$	(o) $(101)*$
(p) $0**$	(q) $(011)**\|\emptyset*$	

Die folgenden Zeichenketten sind dagegen auch bei Berücksichtigung der Klammereinsparungsregeln *keine* regulären Ausdrücke über $\{0, 1\}$:

- $(|1)$ falsch: vor $|$ fehlt ein regulärer Ausdruck;
- $|\emptyset|$ falsch: vor und hinter $|$ fehlt je ein regulärer Ausdruck;
- $()01$ falsch: zwischen $($ und $)$ fehlt ein regulärer Ausdruck;
- $((01)$ falsch: Klammern müssen „in der üblichen Weise gepaart“ auftreten;
- $*(01)$ falsch: vor $*$ fehlt ein regulärer Ausdruck;
- $2*$ falsch: 2 ist nicht Zeichen des Alphabetes;

Reguläre Ausdrücke werden benutzt, um formale Sprachen zu spezifizieren. Auch dafür bedient man sich wieder einer induktiven Vorgehensweise; man spricht auch von einer induktiven Definition:

durch R *beschriebene Sprache* $\langle R \rangle$

Die *von einem regulären Ausdruck* R *beschriebene formale Sprache* $\langle R \rangle$ ist wie folgt definiert:

- $\langle \emptyset \rangle = \{\}$ (d. h. die leere Menge).
- Für $a \in A$ ist $\langle a \rangle = \{a\}$.

- Sind R_1 und R_2 reguläre Ausdrücke, so ist $\langle R_1 \texttt{|} R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$.
- Sind R_1 und R_2 reguläre Ausdrücke, so ist $\langle R_1 R_2 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$.
- Ist R ein regulärer Ausdruck, so ist $\langle R\texttt{*} \rangle = \langle R \rangle^*$.

Betrachten wir drei einfache Beispiele:

- $R = \texttt{a|b}$: Dann ist $\langle R \rangle = \langle \texttt{a|b} \rangle = \langle \texttt{a} \rangle \cup \langle \texttt{b} \rangle = \{\texttt{a}\} \cup \{\texttt{b}\} = \{\texttt{a}, \texttt{b}\}$.
- $R = \texttt{(a|b)*}$: Dann ist $\langle R \rangle = \langle \texttt{(a|b)*} \rangle = \langle \texttt{a|b} \rangle^* = \{\texttt{a}, \texttt{b}\}^*$.
- $R = \texttt{(a*b*)*}$: Dann ist $\langle R \rangle = \langle \texttt{(a*b*)*} \rangle = \langle \texttt{a*b*} \rangle^* = (\langle \texttt{a*} \rangle \langle \texttt{b*} \rangle)^* = (\langle \texttt{a} \rangle^* \langle \texttt{b} \rangle^*)^* = (\{\texttt{a}\}^* \{\texttt{b}\}^*)^*$.

Mehr oder weniger kurzes Überlegen zeigt übrigens, dass für die Sprachen des zweiten und dritten Beispiels gilt: $(\{\texttt{a}\}^*\{\texttt{b}\}^*)^* = \{\texttt{a}, \texttt{b}\}^*$. Man kann also die gleiche formale Sprache durch verschiedene reguläre Ausdrücke beschreiben — wenn sie denn überhaupt so beschreibbar ist.

Damit klingen (mindestens) die beiden folgenden Fragen an:

1. Kann man allgemein algorithmisch von zwei beliebigen regulären Ausdrücken R_1, R_2 feststellen, ob sie die gleiche formale Sprache beschreiben, d. h. ob $\langle R_1 \rangle = \langle R_2 \rangle$ ist?
2. Welche formalen Sprachen sind denn durch reguläre Ausdrücke beschreibbar?

Die Antwort auf die erste Frage ist *ja*. Allerdings hat das Problem, die Äquivalenz zweier regulärer Ausdrücke zu überprüfen, die Eigenschaft PSPACE-vollständig zu sein wie man in der Komplexitätstheorie sagt. Was das ist, werden wir in Einheit 16 über Turingmaschinen kurz anreißen. Es bedeutet unter anderem, dass alle *bisher bekannten* Algorithmen im allgemeinen *sehr sehr langsam* sind: die Rechenzeit wächst „stark exponentiell“ mit der Länge der regulären Ausdrücke (z. B. wie 2^{n^2} o.ä.). Es sei noch einmal betont, dass dies für alle bisher bekannten Algorithmen gilt. Man weiß nicht, ob es vielleicht doch signifikant schnellere Algorithmen für das Problem gibt, aber man sie „nur noch nicht gefunden“ hat.

Nun zur Antwort auf die zweite Frage. (Was rechtslineare Grammatiken sind, werden wir in nachfolgenden Abschnitt 15.2 gleich noch beschreiben. Es handelt sich um einen Spezialfall kontextfreier Grammatiken.)

15.1 *Satz.* *Für jede formale Sprache* L *sind die folgenden drei Aussagen äquivalent:*

1. L *kann von einem endlichen Akzeptor erkannt werden.*
2. L *kann durch einen regulären Ausdruck beschrieben werden.*
3. L *kann von einer rechtslinearen Grammatik erzeugt werden.*

Eine formale Sprache, die die Eigenschaften aus Satz 15.1 hat, heißt *reguläre Sprache*. Da jede rechtslineare Grammatik eine kontextfreie Grammatik ist, ist jede reguläre Sprache eine kontextfreie Sprache.

reguläre Sprache

Zwar werden wir Satz 15.1 nicht im Detail beweisen, aber wir wollen zumindest einige Dinge andeuten, insbesondere auch eine grundlegende Vorgehensweise.

Satz 15.1 hat folgende prinzipielle Struktur:

- Es werden drei Aussagen $\mathcal{A}$, $\mathcal{B}$ und $\mathcal{C}$ formuliert.
- Es wird behauptet:
  - $\mathcal{A} \Longleftrightarrow \mathcal{B}$
  - $\mathcal{B} \Longleftrightarrow \mathcal{C}$
  - $\mathcal{C} \Longleftrightarrow \mathcal{A}$

Man kann nun natürlich einfach alle sechs Implikationen einzeln beweisen. Aber das muss man gar nicht! Dann wenn man zum Beispiel schon gezeigt hat, dass $\mathcal{A} \Longrightarrow \mathcal{B}$ gilt und dass $\mathcal{B} \Longrightarrow \mathcal{C}$, dann folgt $\mathcal{A} \Longrightarrow \mathcal{C}$ automatisch. Das sieht man anhand der folgenden Tabelle:

	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{A} \Longrightarrow \mathcal{B}$	$\mathcal{B} \Longrightarrow \mathcal{C}$	$\mathcal{A} \Longrightarrow \mathcal{C}$
1				W	W	W
2			W	W	W	W
3		W		W		W
4		W	W	W	W	W
5	W				W	
6	W		W		W	W
7	W	W		W		
8	W	W	W	W	W	W

In allen Zeilen 1, 2, 4 und 8, in denen sowohl für $\mathcal{A} \Longrightarrow \mathcal{B}$ als auch für $\mathcal{B} \Longrightarrow \mathcal{C}$ ein W (für *wahr*) eingetragen ist, ist das auch für $\mathcal{A} \Longrightarrow \mathcal{C}$ der Fall. Statt *falsch* haben wir der besseren Übersicht wegen die entsprechenden Felder freigelassen.

Wenn man $\mathcal{A} \Longrightarrow \mathcal{B}$ und $\mathcal{B} \Longrightarrow \mathcal{C}$ schon bewiesen hat, dann muss man also $\mathcal{A} \Longrightarrow \mathcal{C}$ gar nicht mehr beweisen. Und beweist man nun zusätzlich noch $\mathcal{C} \Longrightarrow \mathcal{A}$, dann

- folgt mit $\mathcal{A} \Longrightarrow \mathcal{B}$ sofort $\mathcal{C} \Longrightarrow \mathcal{B}$ und
- mit $\mathcal{B} \Longrightarrow \mathcal{C}$ folgt sofort $\mathcal{B} \Longrightarrow \mathcal{A}$,

und man ist fertig.

Statt sechs Implikationen zu beweisen zu müssen, reichen also drei. Für einen Beweis von Satz 15.1 genügen daher folgende Konstruktionen:

- zu gegebenem endlichen Akzeptor A ein regulärer Ausdruck R mit $\langle R \rangle = L(A)$:
  Diese Konstruktion ist „mittel schwer“. Man kann z. B. einen Algorithmus benutzen, dessen Struktur und Idee denen des Algorithmus von Warshall ähneln.
- zu gegebenem regulären Ausdruck R eine rechtslineare Grammatik G mit $L(G) = \langle R \rangle$:
  Diese Konstruktion ist „relativ leicht“.

- zu gegebener rechtslinearer Grammatik G ein endlicher Akzeptor A mit $L(A) = L(G)$:
  Diese Konstruktion ist die schwierigste.

Wie wertvoll Charakterisierungen wie Satz 15.1 sein können, sieht man an folgendem Beispiel: Es sei L eine reguläre Sprache, z. B. die Sprache aller Wörter, in denen irgendwo das Teilwort `abbab` vorkommt. Aufgabe: Man zeige, dass auch das Komplement $L' = \{a, b\}^* \setminus L$, also die Menge aller Wörter, in denen nirgends das Teilwort `abbab` vorkommt, regulär ist.

Wüssten wir nur, dass reguläre Sprachen die durch reguläre Ausdrücke beschreibbaren sind, und hätten wir nur einen solchen für L, dann stünden wir vor einem Problem. Damit Sie das auch merken, sollten Sie einmal versuchen, einen regulären Ausdruck für L' hinzuschreiben.

Aber wir wissen, dass wir uns auch endlicher Akzeptoren bedienen dürfen. Und dann ist alles *ganz* einfach: Denn wenn A ein endlicher Akzeptor ist, der L erkennt, dann bekommt man daraus den für L', indem man einfach akzeptierende und ablehnende Zustände vertauscht.

15.2 Rechtslineare Grammatiken

Mit beliebigen kontextfreien Grammatiken kann man jedenfalls zum Teil andere formale Sprachen erzeugen, als man mit endlichen Akzepteren erkennen kann. Denn die Grammatik $G = (\{X\}, \{a, b\}, X, \{X \to aXb \mid \varepsilon\})$ erzeugt $\{a^k b^k \mid k \in \mathbb{N}_0\}$ und diese Sprache ist nicht regulär.

Aber die folgende einfache Einschränkung tut „das Gewünschte“. Eine *rechtslineare Grammatik* ist eine kontextfreie Grammatik $G = (N, T, S, P)$, die der folgenden Einschränkung genügt: Jede Produktion ist entweder von der Form $X \to w$ oder von der Form $X \to wY$ mit $w \in T^*$ und $X, Y \in N$. Auf der rechten Seite einer Produktion darf also höchstens ein Nichterminalsymbol vorkommen, und wenn dann nur als letztes Symbol.

rechtslineare Grammatik

Die oben erwähnte Grammatik $G = (\{X\}, \{a, b\}, X, \{X \to aXb \mid \varepsilon\})$ ist also *nicht* rechtslinear, denn in der Produktion $X \to aXb$ steht das Nichtterminalsymbol X nicht am rechten Ende.

Und da wir uns übelegt hatten, dass die erzeugte formale Sprache nicht regulär ist, kann es auch gar keine rechtslineare Grammatik geben, die $\{a^k b^k \mid k \in \mathbb{N}_0\}$ erzeugt.

Zum Abschluss sei noch die folgende Sprechweise eingeführt: Rechtslineare Grammatiken heißen auch *Typ-3-Grammatiken* und die schon eingeführten kontextfreien Grammatiken nennt man auch *Typ-2-Grammatiken*. Hier ahnt man schon, dass es noch weiter geht. Es gibt auch noch *Typ-1-Grammatiken* und *Typ-0-Grammatiken*.

Typ-3-Grammatiken

Typ-2-Grammatiken

Wenn für ein $i \in \{0, 1, 2, 3\}$ eine formale Sprache L von einer Typ-i-Grammatik erzeugt wird, dann sagt man auch, L sei eine *Typ-i-Sprache* oder kurz *vom Typ* i.

Beweise für die Behauptungen aus Satz 15.1 werden Sie vielleicht in der Vorlesung „Theoretische Informatik“ oder in „Formale Systeme“ sehen. Insbesondere ist es dafür nützlich, sich mit nichtdeterministischen endlichen Automaten zu beschäftigen, auf die wir am Ende von Einheit 14 schon hingewiesen haben.

Sie werden sehen, dass reguläre Ausdrücke bei der Verarbeitung von Textdateien des öfteren nützlich sind. Dabei kommen zu dem, was wir in Abschnitt 15.1 definiert haben, zum einen noch bequeme Abkürzungen hinzu, denn wer will schon z. B.

```
a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
```

schreiben müssen als regulären Ausdruck für einen einzelnen Kleinbuchstaben. Zum anderen gibt es aber auch noch Erweiterungen, die dazu führen, dass die resultierenden *regular expressions* mächtiger sind als reguläre Ausdrücke. Wer sich dafür (jetzt schon) genauer interessiert, dem sei das Buch von Friedl (2006) empfohlen.

LITERATUR

Friedl, Jeffrey (2006). *Mastering Regular Expressions*. 3rd edition. O'Reilly Media, Inc.

Kleene, Stephen C. (1956). “Representation of Events in Nerve Nets and Finite Automata”. In: *Automata Studies*. Hg. von Claude E. Shannon und John McCarthy. Princeton University Press. Kap. 1, S. 3–40.
Eine Vorversion ist online verfügbar; siehe http://www.rand.org/pubs/research_memoranda/2008/RM704.pdf (8.12.08).

