Reguläre Sprachen

Grundbegriffe

Alphabete Die Symbole, aus denen Eingabewörter bestehen können, fassen wir in einer Menge, dem Eingabealphabet, zusammen. Dabei wollen wir nur endliche Eingabealphabete zulassen. Ihre Elemente heißen Symbole oder Buchstaben. Zur Bezeichnung von Eingabealphabeten wird zumeist das Symbol Σ (Sigma) benutzt.[1]

Wörter Die endlich langen Zeichenfolgen, die über einem Alphabet Σ gebildet werden können, heißen Wörter über Σ. Wörter entstehen, indem Symbole oder bereits erzeugte Wörter aneinandergereiht (miteinander verkettet, konkateniert) werden.

Σ^*, die Menge aller Wörter, die über dem Alphabet Σ gebildet werden kann, ist wie folgt definiert:

(a) Jeder Buchstabe $a \in \Sigma$ ist auch ein Wort über Σ, d. h. $a \in \Sigma^*$.

(b) Werden bereits konstruierte Wörter hintereinandergeschrieben, entstehen neue Wörter, d. h. sind $v, w \in \Sigma^*$, dann ist auch ihre Verkettung (Konkatenation) $v, w \in \Sigma^*$.

(c) ε (epsilon), das leere Wort, ist ein Wort über (jedem Alphabet) Σ, d. h. es gilt immer $\varepsilon \in \Sigma^*$. ε ist ein definiertes Wort ohne „Ausdehnung“. Es hat die Eigenschaft: $\varepsilon w = w\varepsilon = w$ für alle $w \in \Sigma^*$. Manchmal wird für das leere Wort auch das Symbol λ (lambda) verwendet.

Mit Σ^+ bezeichnen wir die Menge aller Wörter über Σ ohne das leere Wort, d. h. $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.[2]

Sei Σ ein Alphabet. Eine formale Sprache L ist eine Teilmenge aller Wörter über Σ:

$$L \subseteq \Sigma^*$$

Grammatik

Eine Grammatik ist ein 4-Tupel mit $G = (V, \Sigma, P, S)$ und besteht aus:

- Einer endlichen Menge V von *Variablen* (Nonterminale) — Variablen
- Dem endlichen *Terminalalphabet* Σ mit $\Sigma \cap V = \emptyset$ — Terminalalphabet
- Der endlichen Menge an *Produktionen* — Produktionen
- Und einer *Startvariablen* S mit $S \in V$[3] — Startvariablen

[1] Vossen und Witt, *Grundkurs Theoretische Informatik*, Seite 15.
[2] Vossen und Witt, *Grundkurs Theoretische Informatik*, Seite 16.
[3] *Theoretische Informatik – Reguläre Sprachen*, Seite 17.

Beispiel

$G = (V, \Sigma, P, S)$ mit

- $V = \{Z, A\}$
- $\Sigma = \{0, 1\}$
- $P = \Big\{$

$$\begin{aligned} Z &\to 1A \,|\, 1 \\ A &\to 1A \,|\, 0A \,|\, 1 \end{aligned}$$

$\Big\}$

oder

$G = (\{Z, A\}, \{0, 1\}, \{Z \to 1A \,|\, 1, A \to 1A \,|\, 0A \,|\, 1\}, S)$

Eine reguläre Sprache wird durch eine reguläre Grammatik erzeugt, d. h. eine Grammatik mit Produktionsregeln der Form:
$\{A \to \varepsilon\}$ oder $\{A \to 1\}$ oder $\{A \to 0A\}$

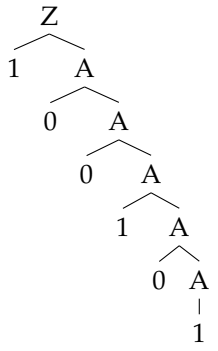
linke Seite: ein Nonterminal

rechte Seite: ε oder ein Terminal oder ein Terminal gefolgt von einem Nonterminal

Man spricht hierbei auch von rechtslinearer Grammatik (es gibt auch linkslineare Grammatiken)[4]

Syntax-Baum

Gegebene Grammatik: $G = (\{Z, A\}, \{0, 1\}, \{Z \to 1A \,|\, 1, A \to 1A \,|\, 0A \,|\, 1\}, S)$
Syntaxbaum zum Wort „100101“:



[4] *Theoretische Informatik – Reguläre Sprachen*, Seite 18.

Reguläre Grammatik $\rightarrow$ NEA

Wir können aus einer regulären Grammatik einen NEA machen: Jedes Non-Terminal (Variable) wird zu einem Zustand konvertiert. Dazu kommt der Zustand q_F, welcher der Endzustand ist. Existiert in der Grammatik eine Regel $S \rightarrow \varepsilon$, wobei S Startzustand ist, so ist S auch ein Endzustand. Dadurch wird erreicht, dass auch das leere Wort akzeptiert wird. Wenn es eine Regel $A \rightarrow a$ gibt, d. h. eine Regel auf deren rechten Seite nur ein Terminal zu finden ist, dann erzeugen wir eine Übergangsfunktion δ (zeichnen einen Pfeil) von dem Zustand A nach q_F mit der Inschrift a. Wenn es eine Regel $A \rightarrow aB$ gibt, dann erzeugen wir eine Übergangsfunktion (zeichnen einen Pfeil) von dem Zustand A in den Zustand B mit der Inschrift a. [5]

Eindeutige Grammatik

Eine Grammatik G ist dann eindeutig, wenn für jedes Wort aus $L(G)$ genau eine mögliche Ableitung aus dem Startsymbol existiert. [6]

Eine Sprache L heißt eindeutig, wenn es für L eine eindeutige Grammatik gibt. Ansonsten heißt L mehrdeutig. [7]

Äquivalenzklassen (nach Myhill, Nerode)[8]

Zwei Wörter $x, y \in \Sigma^*$ sind äquivalent bzgl. einer Sprache L, wenn für alle $z \in \Sigma^*$ gilt:

$$xz \in L \Leftrightarrow yz \in L$$

[9]

Eine Sprache L ist genau dann regulär, wenn sie *endlich viele Äquivalenzklassen* erzeugt.[10]

endlich viele Äquivalenzklassen

Die *Anzahl der Äquivalenzklassen* einer Sprache entspricht der *minimalen Anzahl von Zustanden* eines deterministen-endlichen Automaten, der diese Sprache erkennt. Darauf fußt der Minimalisierungsalgorithmus.

Anzahl der Äquivalenzklassen

minimalen Anzahl von Zustanden

Beispiel zu Äquivalenzklassen

$(1|11|111)0^*$ sind alle Binärzahlen, die mit einer, zwei oder drei Einsen beginnen, gefolgt von beliebig vielen Nullen.

(a) Sind $x = 1$ und $y = 10$ in der selben Äquivalenzklasse?

Nein, denn anfügen von $z = 1$:

[5] `http://www.informatikseite.de/theorie/node50.php`
[6] `http://u-helmich.de/inf/kursQ21/27/folge-27-2.html`
[7] `http://wwwmayr.in.tum.de/lehre/2008WS/ads-ei/2008-11-24.pdf`
[8] *Theoretische Informatik – Reguläre Sprachen*, Seite 60.
[9] *Wikipedia-Artikel „Nerode-Relation“*.
[10] *Wikipedia-Artikel „Satz von Myhill-Nerode“*.

- $xz = 11$ erlaubtes Wort
- $yz = 101$ unerlaubtes Wort

(b) Sind $x = 1$ und $y = 11$ in der selben Äquivalenzklasse?
Nein, denn anfügen von $z = 11$:

- $xz = 111$ erlaubtes Wort
- $yz = 1111$ unerlaubtes Wort

(c) Sind $x = 10$ und $y = 110$ in der selben Äquivalenzklasse?
Ja, fügt man nur 0en an, sind die Wörter in L, fügt man auch 1en an, sind sie nicht in L.

Äquivalente Zustände[11]

beiden Fallen
Endzustand
Nicht-Endzustand
zu einem Zustand verschmolzen

Zwei Zustände z, z' eines deterministischen Automaten sind äquivalent, wenn für alle möglichen Eingaben der Automat entweder in *beiden Fallen* in einen (nicht notwendig gleichen) *Endzustand* oder in beiden Fallen in einen (nicht notwendig gleichen) *Nicht-Endzustand* übergeht. Äquivalente Zustände können *zu einem Zustand verschmolzen* werden.

Abschlusseigenschaften reguläre Sprachen

Vereinigung: Die Vereinigung $L = L_1 \cup L_2$ zweier regulärer Sprachen L_1 und L_2 ist regulär.

Schnitt: Der Schnitt $L = L_1 \cup L_2$ zweier regulärer Sprachen L_1 und L_2 ist regulär.

Komplement: Das Komplement $\bar{L} = \Sigma^* \setminus L$ einer regulären Sprache L ist regulär.

Sei A ein deterministischer endlicher Automat, der L erkennt. Der Automat A erreicht für jedes Wort ω Element L einen Endzustand und für jedes Wort ω nicht Element L einen Nicht-Endzustand. Indem in A alle Endzustände zu Nicht-Endzuständen gemacht werden und umgekehrt, entsteht ein deterministischer endlicher Automat A, der L erkennt. Also ist L regulär. [12]

Komplement

Um zu zeigen, dass für jede reguläre Sprache L auch das Komplement $L = \Sigma^* \setminus L$ regulär ist, gehen wir in zwei Schritten vor: Im ersten Schritt konstruieren wir einen endlichen Automaten A mit $L(A) = L$. Dass ein solcher Automat für jede reguläre Sprache existieren muss, haben wir im

[11] *Theoretische Informatik – Reguläre Sprachen*, Seite 62.
[12] `https://www.inf.hs-flensburg.de/lang/theor/regulaer-abgeschlossen.htm`

vorherigen Abschnitt herausgearbeitet. Im zweiten Schritt konstruieren wir aus A den Komplementärautomaten A, der die Sprache $L(A)$ erzeugt. Dass der Komplementärautomat die Sprache $L(A)$ erzeugt, ist leicht einzusehen. Da wir die Menge der Endzustände invertiert haben, wird ein Wort ω genau dann von A akzeptiert, wenn es von A zurückgewiesen wird. Mit L wird damit immer auch L von einem Automaten akzeptiert, so dass die Menge der regulären Sprachen in Bezug auf das Komplement abgeschlossen ist.[13]

Produkt: Das Produkt $\{ uv \mid u \in L_1 \vee v \in L_2 \}$ zweier regulärer Sprachen L_1 und L_2 ist regulär.

Kleene-Stern: Der Kleene-Stern L^* einer regulären Sprache L, d. h. die beliebig häufige Konkatenation von Wörter aus der Sprache L vereinigt mit dem leeren Wort, ist regulär.[14]

Entscheidungsprobleme

Wortproblem: $\Rightarrow$ für reguläre Sprachen entscheidbar!

Leerheitsproblem: $\Rightarrow$ für reguläre Sprachen entscheidbar!

Endlichkeitsproblem: $\Rightarrow$ für reguläre Sprachen entscheidbar!

Äquivalenzproblem: $\Rightarrow$ für reguläre Sprachen entscheidbar![15]

Literatur

[1] Dirk W. Hoffmann. *Theoretische Informatik*. 2018.

[2] *Theoretische Informatik – Reguläre Sprachen*.

[3] Gottfried Vossen und Kurt-Ulrich Witt. *Grundkurs Theoretische Informatik. Eine anwendungsbezogene Einführung – Für Studierende in allen Informatik-Studiengängen*. 2016.

[4] *Wikipedia-Artikel „Nerode-Relation“*. `https://de.wikipedia.org/wiki/Nerode-Relation`.

[5] *Wikipedia-Artikel „Satz von Myhill-Nerode“*. `https://de.wikipedia.org/wiki/Satz_von_Myhill-Nerode`.

[13] Hoffmann, *Theoretische Informatik*, Seite 218-219.
[14] *Theoretische Informatik – Reguläre Sprachen*, Seite 68.
[15] *Theoretische Informatik – Reguläre Sprachen*, Seite 70-71.