Das Pumpinglemma für reguläre Sprachen

zuletzt aktualisiert: 17. September 2020

Dieser Skriptauszug orientiert sich an Ingo Wegeners Buch *Theoretische Informatik* [Weg99] sowie einem Skript von Heiko Röglin [Rö17].

1 Was ist eine formale Sprache?

In diesem Kapitel führen wir einen grundlegenden Begriff ein, den der *formalen Sprache*. Dafür benötigen wir zunächst einmal den Begriff *Alphabet*. Dabei handelt es sich um eine endliche Menge von *Symbolen* oder *Zeichen*. Wir verwenden die Variable Σ für das Alphabet. Ein *Wort* ist eine endliche Zeichenkette, also eine Folge von Symbolen aus Σ. Ein beliebiges Wort bezeichnen wir häufig mit der Variablen ω. Die Menge aller Wörter bezeichnen wir mit Σ^*. Diese enthält auch das *leere Wort*, d.h. die Zeichenkette der Länge Null. Das leere Wort bezeichnen wir mit ε. Wenn wir das leere Wort ausschließen wollen, sprechen wir von Σ^+.

Definition 1. *Sei Σ eine endliche Menge von Symbolen. Wir definieren $\Sigma^* = \{\omega | \omega = w_1 \dots w_n, n \in \mathbb{N}^{\geq 0}, w_i \in \Sigma \ \forall i \in \{1, \dots, n\}\}$ und $\Sigma^+ := \Sigma^* \backslash \{\varepsilon\}$.*

Eine (formale) Sprache über dem Eingabealphabet Σ ist eine Menge $L \subseteq \Sigma^*$.

Die gerade eingeführten Begriffe sind offenbar an die natürliche Sprache angelehnt. Sie unterscheiden sich aber von dem, was wir intuitiv unter einem *Wort* oder einer *Sprache* verstehen. Verwenden wir zum Beispiel das deutsche Alphabet als Σ, so wäre 'hjkl' genauso ein Wort wie 'Theorie', obwohl wir nur eine dieser Zeichenketten als deutsches Wort bezeichnen würden. Und wenn auch Leer- und Satzzeichen ein Teil von Σ sind, ist auch 'Theoretische Informatik ist großartig!' ein (einziges) Wort. So erklärt sich auch, warum eine Sprache als eine Menge von Worten definiert ist: Wir können die deutsche Sprache als Menge aller gültigen deutschen Sätze auffassen, die alle 'Wörter' über dem erweiterten deutschen Alphabet sind. So können wir die deutsche Sprache tatsächlich im obigen Sinne als formale Sprache modellieren.

Natürliche Sprachen sind allerdings sehr komplex. Wir werden uns später mit einer Hierarchie von Sprachbegriffen verschiedener Mächtigkeit beschäftigen. Dann werden wir auch sehen, dass es möglich ist, Sprachen zu definieren, für die gar nicht entschieden werden kann, ob ein Wort in der Sprache ist oder nicht.

2 Reguläre Sprachen

In diesem Kapitel beschäftigen wir uns mit einer Klasse von sehr eingeschränkten Sprachen, den *regulären Sprachen*. Wir werden mehrere Möglichkeiten kennenlernen, reguläre Sprachen zu definieren. Die Alternativen veranschaulichen wir anhand einer Beispielsprache,

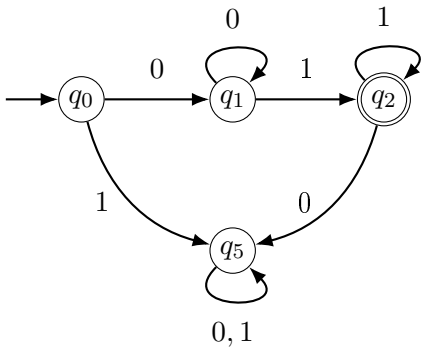
die (wie sich zeigen wird) regulär ist. Diese Beispielsprache ist über dem in der Informatik sehr beliebten Alphabet $\Sigma = \{0, 1\}$ definiert und lautet:

$$L_1 := \{0^m 1^n | m, n \in \mathbb{N}^{\geq 1}\}$$

Dabei verwenden wir für ein Zeichen $a \in \Sigma$ und ein $i \in \mathbb{N}^{\geq 0}$ die Schreibweise a^i für die i-fache Wiederholung des Zeichens a. Die Beispielsprache L_1 besteht also aus allen Zeichenketten, die mit einer oder mehreren Nullen beginnen, worauf dann eine oder mehrere Einsen folgen. Ein zulässiges Wort aus dieser Sprache ist 01111, wohingegen 0101 nicht zu L_1 gehört.

2.1 Endliche Automaten

Eine mögliche Definition für reguläre Sprachen ist die folgende: Eine Sprache L ist genau dann regulär, wenn sich mit Hilfe eines *endlichen Automaten* entscheiden lässt, ob ein Wort ω in L ist oder nicht. Die folgende Abbildung zeigt einen möglichen endlichen Automaten für L_1:



Wir nennen die Kreise *Zustände* und die Pfeile *Zustandsübergänge*. Wenn wir entscheiden wollen, ob das Wort 0101 in unserer Beispielsprache ist, so folgen wir den Zustandsübergängen. Der Zustandsübergang 'aus dem nichts' gibt an, in welchem Zustand wir starten, dem Zustand q_0. Nun lesen wir unser Wort und folgen für jedes Zeichen dem Pfeil, der mit dem nächsten Symbol beschriftet ist. Bei der Verarbeitung des Worts 0101 gelangen wir also erst in den Zustand q_1, dann in Zustand q_2, dann in Zustand q_5, und dort enden wir dann, nachdem wir das letzte Zeichen gelesen haben. Nun stellen wir fest, dass wir nicht in dem Zustand geendet sind, der mit einem doppelten Kreis als *akzeptierender* Zustand markiert ist. Das bedeutet, dass unser Wort nicht zu L_1 gehört, was ja auch stimmt. Folgen wir jedoch den Zustandsübergängen für das Wort 00011, so gelangen wir nach q_2, da $00011 \in L_1$ ist. Es sei nun dem Leser überlassen, sich zu überlegen, dass der obige Automat für alle Wörter über dem Alphabet $\{0, 1\}^n$ die richtige Entscheidung trifft.

Jetzt liefern wir eine formale Definition von endlichen Automaten nach. Der endliche Automat ist ein einfaches Rechenmodell, mit dem ein Rechner mit begrenztem Speicher modelliert werden kann. Obwohl diese Aussage auch auf moderne Rechner zutrifft (da der Speicher unserer Rechner immer durch eine endliche Zahl beschränkt ist), gibt es für die Analyse von Algorithmen geeignetere Modelle. Unter einem endlichen Automaten sollten wir uns eher eine Maschine vorstellen, die von der Mächtigkeit einem Snackautomaten ähnelt: Dieser muss mit einem sehr kleinen Eingabe'alphabet' (alle Münzwerte) entscheiden,

ob das eingegebene Wort (die Geldmenge und der ausgewählte Snack) gültig ist (und welches Restgeld ausgegeben werden soll; mit Ausgaben beschäftigen wir uns aber in dieser Vorlesung nicht).

Definition 2. *Ein* (deterministischer) endlicher Automat M *besteht aus*

- *einer endlichen* Zustandsmenge Q *mit ausgezeichnetem* Startzustand $q_0 \in Q$,
- *einem endlichen* Eingabealphabet Σ,
- *einer* Zustandsübergangsfunktion $\delta : Q \times \Sigma \to Q$, *sowie*
- *einer Menge* $F \subseteq Q$ *von* akzeptierenden Zuständen.

Nun müssen wir noch definieren, was es bedeutet, wenn ein endlicher Automat eine Sprache *entscheidet*. Man spricht hierbei auch von dem *Wortproblem*. Es besteht darin, für ein Wort $\omega \in \Sigma^*$ zu entscheiden, ob $\omega \in L$ ist. Ein endlicher Automat löst das Wortproblem, indem er das gegebene Wort ω zeichenweise 'liest' und genau dann mit Ja antwortet, wenn er nach dem letzten gelesenen Zeichen in einem akzeptierenden Zustand endet. Wir leiten daher nun von der Zustandsübergangsfunktion δ noch eine Funktion δ^* ab, die eine Erweiterung für Wörter darstellt.

Definition 3. *Sei ein endlicher Automat M durch Q, q_0, Σ, δ und F gegeben. Wir definieren $\delta^* : Q \times \Sigma^*$ rekursiv durch*

- $\delta^*(q, \epsilon) := q$ *für alle* $q \in Q$
- $\delta^*(q, a) := \delta(q, a)$ *für alle* $q \in Q, a \in \Sigma$
- $\delta^*(q, w_1 \dots w_n) := \delta^*(\delta(q, w_1), w_2 \dots w_n)$ *für alle* $q \in Q$ *und alle* $\omega \in \Sigma^*$, *wobei* $\omega = w_1 \dots w_n$ *mit* $n \in \mathbb{N}^{\geq 1}$ *und* $w_i \in \Sigma$ *für alle* $i \in \{1, \dots, n\}$.

Nun können wir definieren, welche Sprache ein endlicher Automat akzeptiert, und was der Ausdruck *reguläre Sprache* bedeutet.

Definition 4. *Sei ein endlicher Automat M durch Q, q_0, Σ, δ und F gegeben. Die von M akzeptierte Sprache ist*

$$L(M) = \{w \in \Sigma^* | \delta^*(q_0, \omega) \in F\}.$$

Wir nennen eine Sprache $L \subset \Sigma^*$ regulär*, wenn es einen endlichen Automaten M gibt, so dass $L = L(M)$ ist.*

2.2 Reguläre Grammatiken

In diesem Kapitel beschäftigen wir uns mit *Grammatiken*, insbesondere mit solchen, durch die reguläre Sprachen erzeugt werden. Dafür definieren wir als erstes, was der Begriff *Grammatik* bedeutet.

Definition 5. *Eine* Grammatik *besteht aus*

- *einem endlichen Alphabet Σ, dessen Elemente wir* Terminalsymbole *nennen,*

- *einer endlichen nicht-leeren Menge V, deren Elemente wir als* Nichtterminale *oder* Variablen *bezeichnen,*
- *einem ausgezeichneten* Startsymbol $S \in V$ *und*
- *einer endlichen Menge $P \subset V^+ \times (V \cup \Sigma)^*$ von* Ableitungsregeln. *Dabei bedeutet eine Regel $(l, r) \in P$, dass wir Vorkommen von l als Teilwort durch r ersetzen dürfen. Daher schreiben wir eine Regel $(l, r) \in P$ mit $l \in V^+$ und $r \in (V \cup \Sigma)^*$ auch als $l \to r$, und verwenden die Abkürzung $l \to r_1, \ldots, r_k$, wenn P die Regeln $(l, r_1), (l, r_2), \ldots, (l, r_k)$ mit $k \in \mathbb{N}^{\geq 2}$, $l \in V^+$ und $r_1, \ldots, r_k \in (V \cup \Sigma)^*$ enthält.*

Wir erinnern uns an die Beispielsprache $L_1 = \{0^m 1^n | m, n \in \mathbb{N}^{\geq 1}\}$ aus dem letzten Abschnitt. Eine Grammatik für diese Sprache ist zum Beispiel gegeben durch $\Sigma = \{0, 1\}$, $V = \{S, N, A\}$ und

$$\begin{aligned} P = \{ \quad & S \to 0N, \\ & N \to 0N, 1E \\ & E \to 1E, \varepsilon \\ \}. & \end{aligned}$$

Hier folgen wir den Ableitungsregeln und behaupten, dass wir dabei genau die Wörter der Beispielsprache L_1 erzeugen können. Für das Wort 01111 folgen wir den Ableitungsregeln folgendermaßen:

$$S \to 0N \to 01E \to 011E \to 0111E \to 01111E \to 01111\varepsilon = 01111.$$

Wir können aber zum Beispiel kein Wort erzeugen, dass mit 1 beginnt, da S nur nach $0N$ abgeleitet werden kann; ebenso können wir keine 0 mehr hinzufügen, sobald wir das erste mal die Ableitungsregel $N \to 1E$ verwendet haben.

In obigem Beispiel haben wir bereits die Relation $\to$ auf Wörter aus $(\Sigma \cup V)^+$ angewendet, obwohl sie in dieser Allgemeinheit bisher nicht definiert ist. Diese Erweiterung liefern wir nach und definieren auch, wann ein Wort durch mehrfaches Ableiten aus einem anderen erzeugt werden kann.

Definition 6. *Sei eine Grammatik G durch Σ, V, S und P gegeben. Wir definieren für alle $w_1, w_2 \in (\Sigma \cup V)^+$:*

- $w_1 \to w_2 \Leftrightarrow \exists x, y \in (\Sigma \cup V)^*, l \in V^+, r \in (\Sigma \cup V)^+$ *mit* $(l, r) \in P$, *so dass* $w_1 = xly, w_2 = xry$,
- $w_1 \to^1 w_2 \Leftrightarrow w_1 \to w_2$,
- $w_1 \to^i w_2 \Leftrightarrow \exists w' \in (\Sigma \cup V)^+$ *mit* $w_1 \to^{i-1} w'$ *und* $w' \to^{i-1} w_2$ *und*
- $w_1 \to^* w_2 \Leftrightarrow \exists i \in \mathbb{N}^{\geq 0}$ *mit* $w_1 \to^i w_2$.

Die von G erzeugte Sprache ist nun

$$L(G) = \{\omega \in \Sigma^* \mid S \to^* \omega\}.$$

Definition 5 und Definition 6 erlauben weitaus komplexere Grammatiken als die von uns betrachtete Beispielgrammatik. Grammatiken werden von der *Chomsky-Hierarchie* in vier Klassen eingeteilt. Wir besprechen hier aber nur die eingeschränkteste Klasse.

Definition 7. *Sei eine Grammatik G durch Σ, V, S und P gegeben. G heißt* regulär *oder* Chomsky-3-Grammatik, *wenn alle Ableitungsregeln in P die Form $A \to v$ mit $A \in V$ und $v \in \{\varepsilon\} \cup \{aB \mid a \in \Sigma, B \in V\}$ haben.*

Reguläre Grammatiken erlauben also zum Beispiel keine zwei Nichtterminale auf der 'linken' Seite der Ableitungsregel, und beschränken die 'rechte' Seite auf ein Terminalsymbol gefolgt von einem Nichtterminal. Wir stellen fest, dass unsere Grammatik für die Beispielsprache L_1 regulär ist. Den folgenden Satz zitieren wir aus [Weg99] (dort Satz 5.3.1). Wir werden den Beweis in diesem Kurzskript auslassen.

Theorem 8. *Die Klasse der von endlichen Automaten akzeptierten Sprachen und die Klasse der von Chomsky-3-Grammatiken erzeugten Sprachen stimmen überein.*

2.3 Reguläre Ausdrücke

Als dritte Charakterisierung von regulären Sprachen führen wir die Beschreibung durch reguläre Ausdrücke ein. Die Beispielsprache L_1 lässt sich zum Beispiel durch den regulären Ausdruck $((0) + (0)^*) + ((1) + (1)^*)$ beschreiben: Ein Wort aus L_1 beginnt mit einer 0, auf die erst beliebig viele (eventuell keine) Nullen folgen, dann eine 1, und dann beliebig viele Einsen. Als nächstes definieren wir die Menge aller gültigen regulären Ausdrücke.

Definition 9. *Sei Σ ein endliches Alphabet. Jeder Ausdruck, der durch endliche Anwendung der folgenden Regeln erzeugt werden kann, ist ein regulärer Ausdruck über Σ.*

- *Die Zeichen $\emptyset$, ε und a für $a \in \Sigma$ sind reguläre Ausdrücke.*
- *Sind A_1 und A_2 reguläre Ausdrücke, dann sind auch $(A_1) + (A_2)$, $(A_1) \cdot (A_2)$ und $(A_1)^*$ reguläre Ausdrücke.*

Für die Interpretation regulärer Ausdrücke benötigen wir noch die Definition der Produktoperation für Sprachen. Für zwei Sprachen L_A, L_B, die beide über einem gemeinsamen Alphabet Σ definiert sind, ist das Produkt $L_A L_B$ definiert durch:

$$L_A L_B := \{\omega_1 \omega_2 \mid \omega_1 \in L_A, \omega_2 \in L_B\}.$$

Wir verwenden außerdem die Abkürzung $L^2 = LL$ und L^i für das Produkt $LL \ldots L$ mit i Wiederholungen von L.

Definition 10. *Die von einem regulären Ausdruck A über Σ beschriebene Sprache $L(A)$ ist*

$$L(A) = \begin{cases} \emptyset & A = \emptyset \\ \{\varepsilon\} & A = \varepsilon \\ \{a\} & A = a \text{ für } a \in \Sigma \\ L(A_1) \cup L(A_2) & A = (A_1) + (A_2) \text{ für reguläre Ausdrücke } A_1, A_2 \\ L(A_1)L(A_2) & A = (A_1) \cdot (A_2) \text{ für reguläre Ausdrücke } A_1, A_2 \\ \cup_{i \geq 0}(L(A_1))^i & A = (A_1)^* \text{ für einen regulären Ausdruck } A_1. \end{cases}$$

Die folgende Aussage ist Satz 5.3.3 aus [Weg99] und wird dort auch bewiesen.

Theorem 11. *Genau die regulären Sprachen lassen sich durch reguläre Ausdrücke beschreiben.*

Damit haben wir nun drei Charakterisierungen regulärer Sprachen kennengelernt, die auf endlichen Automaten, regulären Grammatiken und regulären Ausdrücken beruhen.

3 Das Pumping-Lemma

Wir besprechen nun das Kernthema dieses Kurzskripts: das Pumping-Lemma. Dieses liefert eine *notwendige* Bedingung dafür, dass eine Sprache regulär ist. Es besteht also aus einer Aussage der Form: Wenn L regulär ist, dann gilt auch Eigenschaft X für L. Solche notwendigen Bedingungen kann man häufig einsetzen, um zu zeigen, dass etwas *nicht* gilt: Zeigt man, dass X für L nicht wahr ist, so kann L auch nicht regulär sein.

Das Pumping-Lemma gilt jedoch nur in eine Richtung. Ist Eigenschaft X für L erfüllt, können wir daraus nichts über die Regularität von L ableiten: Es kann trotzdem sein, dass die Sprache L nicht regulär ist. Eine vollständige Charakterisierung liefert der *Satz von Nerode*, den wir aber im Rahmen dieser Kurzvorlesung nicht besprechen werden.

Auch für das Pumping-Lemma wollen wir eine Beispielsprache betrachten. Sie ist auch über $\Sigma = \{0, 1\}$ definiert und ähnelt unserer Beispielsprache L_1:

$$L_2 := \{0^n 1^n \mid n \in \mathbb{N}^{\geq 1}\}$$

Der Unterschied zu L_1 ist also nur, dass die Anzahl der Nullen mit der Anzahl der Einsen übereinstimmen muss. Es gilt $L_2 \subsetneq L_1$: Jede Zeichenkette der Form $0^n 1^n$ ist auch in L_1, aber das Wort 001 ist in L_1, jedoch nicht in L_2.

Wir wollen nun zeigen, dass L' keine reguläre Sprache ist. Versucht man, einen endlichen Automaten zu entwerfen, der das Wortproblem für L' löst, hat man intuitiv folgendes Problem: Ein endlicher Automat hat nur eine endliche Anzahl an Zuständen. Möchte man nun Wörter beliebiger Länge akzeptieren, muss der Automat einen Kreis enthalten, d.h. eine Folge von Zustandsübergängen, die wieder im Ausgangszustand endet. So können wir zum Beispiel 0^n für beliebige $n \in N^{\geq 0}$ akzeptieren, indem wir einen (trivialen) Kreis erzeugen, der immer weitere Nullen akzeptiert und dabei immer in demselben Zustand verharrt. Dann können wir jedoch nicht kontrollieren, wie oft dieser Kreis durchlaufen wird und so auch nicht sicherstellen, dass ein weitere Kreis für das Erzeugen der Einsen genau gleich oft durchlaufen wird.

Das Pumping-Lemma formalisiert diese Beobachtung. Genauer gesagt bemerkt es, dass es für jede reguläre Sprache eine Konstante n gibt, so dass Wörter, die aus mindestens n Symbolen bestehen, 'aufgepumpt' werden können. Das bedeutet, dass es bei Wörtern dieser Mindestlänge immer ein Teilwort v geben muss für das durch Weglassen von v oder Einfügen von mehreren Kopien von v wieder ein Wort aus der Sprache entsteht. Für die Sprache L_1 geht das: Wir können in jedem Wort $0^n 1^m$ einen Teil der Nullen auswählen und vervielfachen, ohne dass wir die Sprache L_1 verlassen. Für L_2 funktioniert das aber nicht, da wir durch 'Aufpumpen'der Nullen ein Wort erzeugen, für das die Anzahl Nullen und Einsen nicht mehr übereinstimmt.

Wir formulieren das Pumping-Lemma nun im Detail. Die Aussage wirkt zunächst kompliziert. Der Beweis macht jedoch intuitiv verständlich, warum die einzelnen Teilaussagen gelten.

Lemma 12 (Pumping-Lemma)**.** *Sei L eine reguläre Sprache. Dann gibt es eine Konstante n, so dass sich jedes Wort $\omega \in L$, welches $|\omega| \geq n$ erfüllt, in drei Teile $u, v, w \in \Sigma^*$ zerlegen lässt, so dass gilt:*

- $\omega = uvw$,
- $|uv| \leq n$,
- $|v| \geq 1$, *d.h.* $v \in \Sigma^+$, *und*
- *für alle* $i \in \mathbb{N}^{\geq 0}$ *ist* $uv^iw \in L$.

Beweis. Da L regulär ist, gibt es einen endlichen Automaten, der L akzeptiert. Sei A ein solcher Automat, gegeben durch die Zustandsmenge Q, den Startzustand q_0, das Alphabet Σ, die Zustandsübergangsfunktion δ und die Menge F der akzeptierenden Zustände.

Wir dürfen die Konstante n wählen und setzen sie auf $|Q|$. Diese Wahl treffen wir, weil das Verarbeiten eines Wortes ω immer $|\omega|$ Zustandsübergänge verwendet, und damit $|\omega| + 1$ Zustände betritt (die nicht alle verschieden sein müssen). Für Wörter mit $n = |Q|$ Symbolen ist es also nicht möglich, bei jedem Zustandsübergang zu einem bisher nicht betretenen Zustand überzugehen.

Wir betrachten also nun ein beliebiges Wort $\omega \in L$ mit $|\omega| \geq n$. Wie bereits beobachtet, wird bei der Verarbeitung von ω mindestens ein Zustand mehrfach betreten (wobei wir auch den Start in q_0 als 'betreten' von q_0 auffassen). Wir wählen nun den Zustand q' aus, der als erster Zustand zum zweiten Mal betreten wird.

Nun zerlegen wir ω: Das Teilwort u besteht aus allen Symbolen, die verarbeitet wurden bis zum ersten Betreten von q'. Dieses Teilwort kann leer sein, da q' auch der Startzustand sein könnte. Mit v bezeichnen wir dann das Teilwort, was direkt nach u beginnt und endet, wenn q' zum zweiten Mal betreten wurde. Dieses Teilwort kann nicht leer sein, da eine erneutes Betreten von q' die Verarbeitung von mindestens einem Symbol voraussetzt. Außerdem können u und v zusammen nach unserer obigen Beobachtung nicht mehr als n Zeichen besitzen, da die erste Wiederholung eines Zustands bereits während der ersten n Zeichen auftritt. Alle Symbole, die nach dem zweiten Betreten von q' noch verarbeitet werden, fassen wir als w zusammen. Genauso wie v kann auch w leer sein.

Nach Definition unserer Teilworte gilt nun: $\delta^*(q_0, u) = q'$, $\delta^*(q', v) = q'$ und $\delta^*(q', w) = q''$ für einen Zustand $q'' \in F$, also einen akzeptierenden Zustand. Dementsprechend gilt auch $\delta^*(q_0, uw) = \delta^*(q', w) = q''$, d.h. das Wort, das durch Weglassen von v entsteht, wird ebenfalls akzeptiert, und für alle $i \in \mathbb{N}^{\geq 2}$ gilt

$$\delta^*(q_0, uv^iw) = \delta^*(q', v^iw) = \delta^*(q', v^{i-1}w) = \delta^*(q', v^{i-2}w) = \ldots = \delta^*(q', vw) = \delta^*(q', w) = q'',$$

d.h. auch uv^iw wird für jedes $i \in \mathbb{N}^{\geq 2}$ akzeptiert. Damit haben wir das Pumping-Lemma bewiesen. □

Wir schließen diese Kurzskript nun ab, indem wir das Pumping-Lemma auf L_2 anwenden. Wir nehmen an, L_2 sei regulär. Sei dann n die Konstante, die laut Lemma 12 existieren muss. Wir betrachten das Wort $\omega = 0^n1^n$, das aus $2n$ Symbole besteht und damit die Voraussetzung $|\omega| \geq n$ erfüllt. Jede Zerlegung $\omega = uvw$ gemäß der Regeln in Lemma 12 muss die Bedingung $|uv| \leq n$ einhalten. Damit besteht v für unser Wort ausschließlich aus Nullen. Da v aus mindestens einem Symbol besteht, verletzen wir durch Weglassen von v die Bedingung, dass die Anzahl der Nullen mit der Anzahl der Einsen übereinstimmt. Auch

das Einfügen mehrerer Kopien von v hat diesen Effekt. Wir erhalten einen Widerspruch zu Lemma 12 und schlussfolgern, dass L_2 nicht regulär ist.

Literatur

[Rö17] Röglin, Heiko: *Logik und diskrete Strukturen.* 2017. – Vorlesungsskript

[Weg99] Wegener, Ingo: *Theoretische Informatik - eine algorithmenorientierte Einführung (2. Auflage).* Teubner, 1999. – ISBN 3–519–12123–9