Reguläre Häufigkeitsberechnungen

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

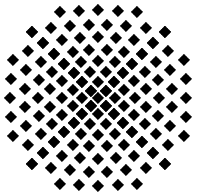
Vorgelegt von

Holger Austinat

aus Stuttgart

Hauptberichter:	Prof. Dr. rer. nat. Volker Diekert
Mitberichter:	Prof. Dr. rer. nat. Ulrich Hertrampf
	Prof. Dr. rer. nat. Heribert Vollmer

Tag der mündlichen Prüfung: 4. Juli 2005



Universität Stuttgart
Institut für Formale Methoden der Informatik (FMI)
2005

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit an den Instituten für Informatik (IfI) und für Formale Methoden der Informatik (FMI) an der Universität Stuttgart.

Mein besonderer Dank gilt Prof. Dr. Volker Diekert, der mich unablässig motivierte, mich mit dem vorliegenden Thema intensiv zu beschäftigen und stets neue Ideen für weitere Untersuchungen einbrachte. Ebenso bin ich Prof. Dr. Ulrich Hertrampf zu Dank verpflichtet, der mir das Thema „Häufigkeitsberechnungen“ näher brachte. Beide hatten immer ein offenes Ohr für Probleme und standen geduldig für Fragen und Diskussionen zur Verfügung. Mein Dank geht auch an Prof. Dr. Heribert Vollmer für die Anfertigung des Zweitgutachtens.

Zu guter Letzt möchte ich der gesamten Abteilung „Theoretische Informatik“ für die äußerst angenehme und freundschaftliche Atmosphäre danken, die häufig über den normalen Arbeitsalltag hinausreichte.

Inhaltsverzeichnis

English Abstract

This thesis deals with *frequency computations*, a recursion theoretic notion that was introduced by Rose in 1960. A function f is said to be computable with frequency m/n, if there is an algorithm that, for every n distinct input values, computes n output values, and at least m of these values coincide with the corresponding values of the function f.

A first question arose naturally: are there non-computable functions that are computable with a frequency close to 1? Trakhtenbrot answered this questions negatively in 1963 by showing that a function computable with frequency strictly greater than 1/2 is already computable in the usual sense. On the other hand, there are uncountably many functions computable with frequency 1/2, thus this result is optimal.

Research in this area intensified after Trakhtenbrot's result, with persons like Dëgtev, Kinber and Trakhtenbrot himself during the 70s and early 80s, and Beigel, Gasarch, Hinrichs, Kummer, Stephan, Tantau, Wechsung and others from the 90s onwards, working on different aspects of frequency computations and related notions.

In this work we mostly deal with *regular frequency computations*, which are frequency computations performed by finite automata. Kinber was the first to investigate this direction in 1976, and claimed that Trakhtenbrot's result carries over to finite automata. He proved this as a corollary of a more general result concerning the separation of languages via finite automata, which turned out to be wrong (a counterexample was given by Tantau in 2002). (Two disjoint languages A and B are seperable if an algorithm accepts all words from A and rejects all words from B.) Thus, the question whether the analogon of Trakhtenbrot's result holds for finite automata posed itself once more.

In this thesis, we prove the following results. In Chapter 2 we give two proofs of Trakhtenbrot's result. First, we present a self-contained version of his original proof—for two reasons: it was published only in Russian and it uses results from mathematical topology, which makes it hard to access. Second, we give a new combinatorial proof which has one major advantage over Trakhtenbrot's: it allows the transfer of this result to finite automata. Two minor results end this chapter: the existence of uncountably many non-frequency computable languages (which cannot be shown by cardinality arguments, since there are uncountably many frequency computable languages), and a connection between frequency computations and so-called *multi-selective* languages, a generalization of Jockusch's notion of *semirecursive* sets, introduced by Hemaspaandra et al. in 2000.

In Chapter 3 we work out the error in Kinber's proof about seperable languages, and give a concrete counterexample. Afterwards, we investigate the separation of so-called *path* and *anti-path languages*, which are defined as follows: let α be an infinite word over the alphabet $\{0,1\}$; then A_α is the set of finite prefixes of α, and B_α is the set of words from A_α with the last bit toggled. We show that A_α and B_α are seperable if and only if the positions of α containing a 1 are computable. On the other hand, there are uncountably many α for which A_α and B_α are seperable with frequency $1/2$.

More surprisingly, if it is possible to seperate A_α and B_α with one error (given at least three inputs), then these languages are already recursive. This comes quite unexpected, since an example of Tantau shows that there is an α such that A_α and B_α are seperable with frequency $3/5$, but not seperable (without errors) by Turing machines. Additionally, the result carries over to finite automata: path- and anti-path languages are regular if they can be seperated by finite automata with only one error (on three or more inputs).

We end this chapter by showing that Kinber's claim, a Trakhtenbrot-like result for the seperation of languages by finite automata, fails badly: for any constant $q < 1$ there are values $1 \leq m < n$ with $m/n > q$ and an α such that A_α and B_α can be seperated by finite automata with frequency m/n, but are not recursively seperable.

In Chapter 4 we investigate different aspects of regular frequency computations. First we show that Trakhtenbrot's result for general frequency computations carries over to finite automata, by a close inspection of the new proof given in Chapter 2. We then show that *aperiodic* automata can—in the frequency computation sense—only compute aperiodic regular languages (the class of aperiodic languages is equivalent to the class of star-free languages).

Next we prove a so-called *iteration criterion*, which is comparable to the well-known pumping lemma for regular languages, and allows us to show for many concrete example languages that they are not frequency computable. Then, in the last part, we investigate closure properties of the class of languages frequency computable by finite automata: we show that they form a Boolean algebra, but are not closed under the reversal operation. Furthermore, the union of two languages which are computable with frequency $1/n$ is, in general, not computable with frequency $1/n$. We give a lower bound which is very close to the best-known upper bound.

Kapitel 1

Einführung

Die vorliegende Arbeit beschäftigt sich mit sog. *Häufigkeitsberechnungen* (engl. *frequency computation*), einem Teilgebiet der Rekursionstheorie mit starken Einflüssen aus der Komplexitätstheorie und der Kombinatorik. Im folgenden geben wir eine kurze Einführung in dieses Gebiet.

In den nächsten beiden Abschnitten 1.1 und 1.2 definieren wir die in dieser Arbeit häufig verwendeten Begriffe und Bezeichnungen, um dann in Abschnitt 1.3 einen Überblick über die bislang erzielten Ergebnisse zu geben und diese in Abschnitt 1.4 in Beziehung zu verwandten Berechenbarkeitsbegriffen zu setzen. Schließlich geben wir in Abschnitt 1.5 einen kurzen Ausblick auf die in dieser Arbeit dargestellten Ergebnisse.

Der Begriff *Häufigkeitsberechnung* wurde von Rose 1960 auf einer Konferenz vorgeschlagen [Ros60] und bezeichnet einen approximativen Berechenbarkeitsbegriff, bei dem mehrere Werte einer Funktion gleichzeitig berechnet werden sollen, dabei aber ein gewisser Anteil der berechneten Funktionswerte falsch sein darf.

Etwas präziser formuliert realisiert ein Algorithmus eine (m, n)-*Berechnung* einer Funktion f, wenn auf jede Eingabe von n verschiedenen Werten n Ausgabewerte berechnet werden, von denen mindestens m mit den Funktionswerten von f übereinstimmen. Man sagt in diesem Fall auch, dass sich die Funktion f mit *Häufigkeit* oder *Frequenz* m/n berechnen bzw. approximieren lässt.

Bereits Rose war bekannt, dass es überabzählbar viele Funktionen gibt, die sich für keine Werte $1 \leq m \leq n$ mit Frequenz m/n berechnen lassen (siehe

Abschnitt 2.3). Andererseits stellte er sich die Frage, ob es nicht-berechenbare Funktionen gibt, die sich für geeignete m und n zumindest (m, n)-berechnen lassen. McNaughton wiederholte diese Frage ein Jahr später und warf, nach einer Diskussion mit Myhill, die Frage auf, ob es solche Funktionen sogar dann gibt, wenn die Frequenz m/n gegen 1 strebt [McN61, S. 393].

In einem Artikel aus dem Jahre 1963 erwähnen Rose und Ullian, dass Scott ihnen eine positive Antwort auf die erste Frage gegeben hat [RU63, S. 693, Fussnote 2]. Die zweite Frage hingegen wurde im gleichen Jahr von Trakhtenbrot negativ beantwortet: sein für das Gebiet der Häufigkeitsberechnungen zentrales Resultat besagt, dass die Klasse der (m, n)-berechenbaren Funktionen genau dann mit der Klasse der (im herkömmlichen Sinne) berechenbaren Funktionen übereinstimmt, wenn $m > n/2$ ist [Tra63].

Trakhtenbrots Ergebnis war die Initialzündung für eine fruchtbare Forschung auf dem Gebiet der Häufigkeitsberechnungen, wie viele Arbeiten aus den 70er und frühen 80er Jahren von Dëgtev, Kinber und Trakhtenbrot und ab Anfang der 90er Jahre von Beigel, Gasarch, Hinrichs, Kummer, Stephan, Tantau, Wechsung und anderen belegen. Eine ausführliche Darstellung der Ergebnisse, die seit Trakhtenbrots Resultat erzielt worden sind, und die Beziehung zu verwandten Berechenbarkeitsbegriffen folgen in den Abschnitten 1.3 und 1.4.

Es gibt mehrere Gründe, sich mit Häufigkeitsberechnungen zu beschäftigen. Der erste ist rekursionstheoretischer Natur: hier besteht stets das Bestreben, die Klassen der nicht-berechenbaren Funktionen bzw. der nicht-entscheidbaren Sprachen genauer zu untersuchen, um ein besseres Verständnis für diese Klassen zu erhalten.

Eine weitere Motivation für Häufigkeitsberechnungen entstammt dem Gebiet der Künstlichen Intelligenz. So verglich Rabin 1974 die approximative Berechnung von Funktionen mittels Häufigkeitsberechnungen mit dem menschlichen Denken: er vermutete, dass Menschen nur deshalb in der Lage seien, komplexe Probleme zu lösen, weil sie dabei Fehler machen — eine exakte Bestimmung der optimalen Lösung wäre durch die beschränkte „Rechenleistung“ des menschlichen Gehirns in den allermeisten Fällen nicht möglich [Rab74]. Trakhtenbrot führte diese Gedanken in [Tra77] fort.

Unterstützung erhält diese These aus dem Bereich der ressourcenbeschränkten Häufigkeitsberechnungen. Hier konnte gezeigt werden, dass es zu jeder rekursiven Funktion t und zu jedem $n \geq 2$ Funktionen gibt, die nicht in Zeit t berechenbar sind, sich aber in linearer Zeit $(n-1, n)$-berechnen lassen

[Kin75b]. Das Zulassen weniger falscher Antworten kann also eine drastische Reduktion der Komplexität einer Funktion bewirken. Ein allgemeineres Resultat von Hinrichs und Wechsung untermauert die These „Fehler sparen Zeit“ [HW97].[1]

Für endliche Automaten können Häufigkeitsberechnungen eine dramatische Verringerung der Zustandszahl bewirken. Ausgehend von einem endlichen Automaten kann dessen Größe (gemessen in der Anzahl der Zustände) jede vorgegebene Schranke überschreiten, wenn nur ein einziges (langes) Wort zur Sprache hinzugenommen oder aus der Sprache herausgenommen wird. Durch Häufigkeitsberechnungen kann also ein endlicher Automat für eine gegebene Sprache unter Umständen viel kompakter dargestellt werden. Dies kann sich im Bereich des *computergestützten Lernens* (engl. *computational learning*) als hilfreich erweisen, um beim „Lernen“ einer regulären Sprache den konstruierten Automaten möglichst klein zu halten. Typischerweise kann man beim Lernen eine geringe Anzahl von Fehlern tolerieren. Wir sehen hier für reguläre Häufigkeitsberechnungen durchaus ein noch nicht ausgeschöpftes Anwendungspotential. In der vorliegenden Arbeit beschäftigen wir uns jedoch ausschließlich mit den mathematischen Grundlagen.

1.1 Allgemeine Begriffe und Bezeichnungen

In dieser Arbeit beschäftigen wir uns mit *Sprachen* und *Funktionen* über einem endlichen *Grundalphabet*, das wir meist mit Σ bezeichnen. In der Regel hat die Größe des Alphabets keine Bedeutung für die dargestellten Ergebnisse. Ist dies doch der Fall, so wird ausdrücklich darauf hingewiesen.

Das *freie Monoid* Σ^* bezeichnet die Menge aller Wörter endlicher Länge über Σ. Die *Länge* eines Wortes $w \in \Sigma^*$ bezeichnen wir mit $|w|$, das *leere Wort* mit ε. Die Anzahl der Vorkommen des Buchstabens $a \in \Sigma$ im Wort w bezeichnen wir mit $|w|_a$. Für Wörter aus Σ^* verwenden wir Bezeichner wie u, v, x, y.

Die Menge Σ^ω bezeichnet die Menge aller Zeichenketten unendlicher Länge über Σ. Wörter aus Σ^ω werden mit α, β bezeichnet.

Sprachen sind Teilmengen von Σ^*. *Funktionen* sind (wenn nicht ausdrücklich anders erwähnt) totale Abbildungen von Σ^* nach Σ^*.

Mit $\mathbb{N} = \{0, 1, 2, \ldots\}$ bezeichnen wir die Menge der *natürlichen Zahlen*.

[1] Bereits Trakhtenbrot war dieser Aspekt bekannt, siehe z. B. [Tra73, §5].

Eine besondere Bedeutung wird die Menge $\mathbb{B} = \{0, 1\}$ haben: wir verwenden sie sowohl als Alphabet als auch als Menge der Zahlen Null und Eins. Ist $b \in \mathbb{B}$, so ist $\overline{b} := 1 - b$. Wir erweitern diese Notation auf Wörter und auf n-Tupel über $\mathbb{B}$: $\overline{b_1 \cdots b_n} := \overline{b_1} \cdots \overline{b_n}$ und $\overline{(b_1, \ldots, b_n)} := (\overline{b_1}, \ldots, \overline{b_n})$.

Die *Spiegelung* eines Wortes $w = a_1 a_2 \cdots a_n$, $a_i \in \Sigma$, schreiben wir als $w^{rev} := a_n a_{n-1} \cdots a_1$. Diese Notation erweitern wir auf Sprachen $L \subseteq \Sigma^*$: die *Spiegelsprache* ist definiert als $L^{rev} := \{w^{rev} \mid w \in L\}$.

Für ein endliches oder unendliches Wort w über Σ ist $w[i]$, $i \in \mathbb{N}$, das i-te Zeichen von w. Die Zählung beginnt bei 1, d. h. für ein endliches Wort w gilt $w = w[1]w[2] \cdots w[|w|]$. Diese Notation verallgemeinern wir auf zusammenhängende und unzusammenhängende Bereiche: $w[i \ldots j] := w[i]w[i+1] \cdots w[j]$, und für $1 \leq i_1 < i_2 < \ldots < i_n$ ist $w[i_1, i_2, \ldots, i_n] := w[i_1]w[i_2] \cdots w[i_n]$.

Mit $\mathrm{pref}(w)$, $w \in \Sigma^* \cup \Sigma^\omega$, bezeichnen wir die Menge der *Präfixe* endlicher Länge von w. Analog bezeichnet $\mathrm{suff}(w)$ für ein $w \in \Sigma^*$ die Menge der *Suffixe* von w. Wir erweitern die Präfix-Notation in kanonischer Weise auf Sprachen $L \subseteq \Sigma^*$: $\mathrm{pref}(L) := \bigcup_{w \in L} \mathrm{pref}(w)$. Die Präfixrelation selbst bezeichnen wir mit $\preceq$, d. h. $w \preceq w'$ genau dann, wenn w ein Präfix von w' ist ($w \in \Sigma^*$, $w' \in \Sigma^* \cup \Sigma^\omega$).

Wir verwenden zwei totale Ordnungen auf Σ^*. Sei dazu $\Sigma = \{a_1, \ldots, a_k\}$; dann definieren wir zunächst eine totale Ordnung auf Σ durch $a_i < a_j$ genau dann, wenn $i < j$ gilt. Für zwei verschiedene Wörter $u, v \in \Sigma^*$ ist bei der *lexikographischen Ordnung* $u < v$, falls u ein Präfix von v ist oder das erste Zeichen, bei dem sich u und v unterscheiden, bei u ein kleineres ist. Bei der *längenlexikographischen Ordnung* ist $u < v$, falls $|u| < |v|$ oder beide Wörter gleiche Länge haben, das erste sich unterscheidende Zeichen bei u aber kleiner ist.

Die *charakteristische Funktion* einer Sprache $L \subseteq \Sigma^*$ ist die totale Abbildung $\chi_L : \Sigma^* \to \mathbb{B}$ mit $\chi_L(x) = 1$ genau dann, wenn $x \in L$ gilt.

Die *Kardinalität* einer Menge M bezeichnen wir mit $|M|$ oder mit $\#M$, die *Potenzmenge* von M mit $\mathcal{P}(M)$.

Die *markierte Vereinigung* zweier Sprache $L, K \subseteq \mathbb{B}^*$ (im Englischen auch *Join* genannt) ist definiert als $L \oplus K := \{0\} \cdot L \cup \{1\} \cdot K$.

Wir werden in dieser Arbeit zwei Maschinenmodelle verwenden: *Turingmaschinen* und *endliche Automaten*. Wenn nichts anderes gesagt wird, arbeiten beide Maschinentypen deterministisch. Berechnet eine Turingmaschine

M bzw. ein endlicher Automat A auf Eingabe x einen Funktionswert y, so schreiben wir $M(x) = y$ bzw. $A(x) = y$. Mit M_n, $n \in \mathbb{N}$, bezeichnen wir die n-te Turingmaschine in einer kanonischen Aufzählung aller Turingmaschinen, und mit $\langle M \rangle$ die Codierung der Turingmaschine M.

Formal ist ein deterministischer endlicher Automat (ohne Ausgabe) ein Tupel $(Q, \Sigma, \delta, q_0, F)$ mit endlicher Zustandsmenge Q, Startzustand $q_0 \in Q$, Endzustandsmenge $F \subseteq Q$, Alphabet Σ und Übergangsfunktion $\delta : Q \times \Sigma \to Q$. Einen Übergang $\delta(q, a) = q'$ schreiben wir häufig als $q \cdot a = q'$. Diese Notation erweitern wir in kanonischer Weise auf Wörter $w \in \Sigma^*$, d. h. $q \cdot w = q'$ bedeutet, dass der Automat durch Lesen von w vom Zustand q in den Zustand q' übergeht. Der Automat akzeptiert (wie üblich) genau die Wörter $w \in \Sigma^*$, für die $q_0 \cdot w \in F$ gilt.

Ein weiteres Akzeptanzmodell, das wir in Kapitel 3 ausführlich behandeln werden, ist die *Separierung* von Sprachpaaren. Im Gegensatz zur Erkennung einer Sprache, bei der ein Wort akzeptiert werden muss, falls es zur Sprache gehört, und andernfalls abgelehnt werden muss, fasst man diese Bedingung bei der Separierung weniger stringent: man legt zwei disjunkte Mengen A und B fest und verlangt lediglich, dass die Wörter aus A akzeptiert und die aus B abgelehnt werden. Wir sprechen jetzt nicht mehr vom *Akzeptieren der Menge A*, sondern von der *Separierung der Mengen A und B*.

Definition 1.1 (separierbar, fa-separierbar) *Zwei Mengen $A, B \subseteq \Sigma^*$ heißen*

- separierbar*, falls eine entscheidbare Menge $C \subseteq \Sigma^*$ existiert mit $A \subseteq C$ und $C \cap B = \emptyset$;*
- fa-separierbar*, falls eine reguläre Menge $C \subseteq \Sigma^*$ existiert mit $A \subseteq C$ und $C \cap B = \emptyset$.*

Aus der Definition folgt, dass A und B disjunkt sein müssen. Offensichtlich ist eine Sprache $L \subseteq \Sigma^*$ genau dann entscheidbar (bzw. regulär), wenn die Mengen L und $\Sigma^* \setminus L$ separierbar (bzw. fa-separierbar) sind.

Ein immer wiederkehrendes Beispiel werden Pfad- und Anti-Pfadsprachen sein, die wir deshalb bereits an dieser Stelle einführen.

Definition 1.2 (Pfad-/Anti-Pfadsprache) *Sei $\alpha \in \mathbb{B}^\omega$. Wir definieren:*

- *Die* Pfadsprache A_α *ist die Menge aller endlichen, nicht-leeren Präfixe von α:*
$$A_\alpha = \{w \in \mathbb{B}^+ \mid w \preceq \alpha\}.$$
- *Die* Anti-Pfadsprache B_α *ist die Menge aller endlichen, nicht-leeren Präfixe von α, bei denen das jeweils letzte Bit negiert wurde:*
$$B_\alpha = \{w \in \mathbb{B}^+ \mid w = w'b, w' \in \mathbb{B}^*, b \in \mathbb{B} \text{ und } w'\overline{b} \preceq \alpha\}.$$
Man kann sich die Menge B_α auch als die minimalen Elemente (bzgl. der Präfixordnung) von $\mathbb{B}^* \setminus A_\alpha$ vorstellen.
- *Das* Positionsprädikat P_α *ist ein einstelliges Prädikat über den natürlichen Zahlen, das an der Stelle $i \in \mathbb{N}$ genau dann 1 ist, wenn $\alpha[i] = 1$ gilt.*

1.2 Häufigkeitsberechnungen

Bei Häufigkeitsberechnungen werden wir es mit n-Tupeln von Ein- und Ausgaben zu tun haben. Hierfür benötigen wir einige weitere Definitionen.

Sei $w = (w_1, \ldots, w_n) \in (\Sigma^*)^n$ ein n-Tupel von Wörtern über Σ. Die *Länge* von w ist die Länge des längsten Eintrags im Tupel: $|w| := \max\{|w_i| \mid 1 \leq i \leq n\}$. Die Projektion des Tupels auf einen einzelnen Wert bzw. einen Bereich schreiben wir als $(w_1, \ldots, w_n)[k] := w_k$ und $(w_1, \ldots, w_n)[i \ldots j] := (w_i, w_{i+1}, \ldots, w_j)$.

Wir erweitern die charakteristische Funktion χ_L einer Sprache $L \subseteq \Sigma^*$ auf n-Tupel von Wörtern über Σ^*: $\chi_L^{(n)} : (\Sigma^*)^n \to \mathbb{B}^n$ mit

$$\chi_L^{(n)}(x_1, \ldots, x_n) := (\chi_L(x_1), \ldots, \chi_L(x_n)).$$

Diese Funktion nennen wir *n-fache charakteristische Funktion* von L.

Sei M eine Turingmaschine, die eine Funktion von $(\Sigma^*)^n$ nach $(\Sigma^*)^n$ berechnet. Dann bezeichnet $M(x_1, \ldots, x_n)$ die Ausgabe von M auf Eingabe $(x_1, \ldots, x_n)$.

Auch die endlichen Automaten (DFA) werden häufig auf n-Tupeln von Wörtern operieren — diese nennen wir dann n-DFA. Ein n-DFA über dem Alphabet Σ ist ein Tupel $A = (Q, \Sigma, \delta, q_0, \tau, n)$ mit endlicher Zustandsmenge Q, Startzustand $q_0 \in Q$, Übergangsfunktion $\delta : Q \times (\Sigma \cup \{\$\})^n \to Q$ und Typfunktion $\tau : Q \to \mathbb{B}^n$ bzw. $\tau : Q \to (\Sigma^*)^n$ (je nachdem, ob wir Sprachen erkennen oder Funktionen berechnen wollen). Dabei ist \$ ein neues Symbol, das nicht in Σ vorkommt und als Füllzeichen verwendet wird. Statt $\delta(q, (a_1, \ldots, a_n)) = q'$, $a_i \in (\Sigma \cup \{\$\})$, schreiben wir wie bei herkömmlichen endlichen Automaten $q \cdot (a_1, \ldots, a_n) = q'$.

Als nächstes beschreiben wir die Arbeitsweise eines n-DFA A auf Eingabe $x = (x_1, \ldots, x_n) \in (\Sigma^*)^n$. Wir definieren ein n-Tupel $x' = (x'_1, \ldots, x'_n)$ durch $x'_i = x_i \cdot \$^{|x|-|x_i|}$, d. h. alle Eingabewörter werden durch Anhängen des neuen Symbols $\$ \notin \Sigma$ auf Länge $|x|$ aufgefüllt. Der endliche Automat liest diese n Eingabewörter synchron. Die Ausgabe von A auf Eingabe x ist jetzt definiert als der Typ des Zustands, der nach Lesen von x' erreicht wird, also als $\tau(q_0 \cdot x')$. Wie bei Turingmaschinen schreiben wir dies als $A(x)$ bzw. $A(x_1, \ldots, x_n)$.

Damit können wir den Begriff der Häufigkeitsberechnung durch Turingmaschinen bzw. durch endliche Automaten definieren.

Definition 1.3 ((m, n)-entscheidbar, (m, n)-erkennbar)
Sei $1 \leq m \leq n$. Eine Sprache $L \subseteq \Sigma^*$ heißt (m, n)-entscheidbar*, falls eine Turingmaschine M existiert, die für alle Eingaben von n paarweise verschiedenen Wörtern $x_1, \ldots, x_n \in \Sigma^*$ eine Ausgabe $M(x_1, \ldots, x_n) \in \mathbb{B}^n$ so berechnet, dass die Vektoren $M(x_1, \ldots, x_n)$ und $\chi_L^{(n)}(x_1, \ldots, x_n)$ an mindestens m Stellen übereinstimmen.*

Die Sprache L heißt (m, n)-erkennbar*, falls in obiger Definition die Turingmaschine M durch einen endlichen Automaten A ersetzt werden kann.*

Analog können wir die (m, n)-Berechnung einer *Funktion* $f : \Sigma^* \to \Sigma^*$ definieren: der Ausgabevektor der Turingmaschine M bzw. des endlichen Automaten A ist nun ein n-Tupel mit Einträgen aus Σ^*, und wiederum müssen mindestens m der n Werte mit den tatsächlichen Funktionswerten übereinstimmen. Wir werden uns nur in Kapitel 2 und in Abschnitt 4.1 mit (m, n)-berechenbaren *Funktionen* beschäftigen, in Kapitel 3 und den restlichen Abschnitten von Kapitel 4 hingegen ausschließlich mit (m, n)-entscheidbaren und (m, n)-erkennbaren *Sprachen*.

Definition 1.4 ($\Omega(m,n)$, Ω, REG(m,n), FREQ-REG) *Für $1 \leq m \leq n$ definieren wir folgende Sprachklassen:*

- $\Omega(m,n)$ *ist die Klasse aller (m,n)-entscheidbaren Sprachen.*
- Ω *ist die Vereinigung aller Klassen $\Omega(m,n)$, $1 \leq m \leq n$.*
- REG(m,n) *ist die Klasse aller (m,n)-erkennbaren Sprachen.*
- FREQ-REG *ist die Vereinigung aller Klassen* REG(m,n), $1 \leq m \leq n$.

Der Begriff der Häufigkeitsberechnung lässt sich auch auf die Separierung von Sprachen (mittels Turingmaschinen oder endlicher Automaten) übertragen. Um zwei Sprachen A, B zu separieren, sollte eine Turingmaschine oder ein endlicher Automat Wörter aus A mit 1 und Wörter aus B mit 0 beantworten; ist dies nicht der Fall, wird das als Fehler gewertet. Auf Eingaben aus $\Sigma^* \setminus (A \cup B)$ kann per Definition kein Fehler gemacht werden:

Definition 1.5 ((m,n)-separierbar, (m,n)-fa-separierbar)
Seien $A, B \subseteq \Sigma^*$ mit $A \cap B = \emptyset$. Die Mengen A und B heißen (m,n)-separierbar*, falls eine Turingmaschine M existiert, die für alle Eingaben von n paarweise verschiedenen Wörtern $x_1, \ldots, x_n \in \Sigma^*$ eine Ausgabe $M(x_1, \ldots, x_n) =: (b_1, \ldots, b_n) \in \mathbb{B}^n$ so berechnet, dass gilt:*

$$\#\{1 \leq i \leq n \mid (x_i \in A \wedge b_i = 0) \vee (x_i \in B \wedge b_i = 1)\} \leq n - m.$$

Die Sprachen A und B heißen (m,n)-fa-separierbar*, falls in obiger Definition die Turingmaschine M durch einen endlichen Automaten ersetzt werden kann.*

1.3 Eine Übersicht bekannter Ergebnisse auf dem Gebiet der Häufigkeitsberechnungen

Das folgende fundamentale Resultat wurde 1963 von Trakhtenbrot veröffentlicht:

Satz 1.6 (Trakhtenbrot [Tra63]) *Sei $f : \Sigma^* \to \Sigma^*$ eine totale, (m,n)-berechenbare Funktion mit $m > n/2$. Dann ist f berechenbar.*

Im nächsten Kapitel werden wir zwei verschiedene Beweise für diesen Satz geben.

Trakhtenbrot konnte auch zeigen, dass es bereits überabzählbar viele $(1,2)$-berechenbare Funktionen gibt, die nicht Turing-berechenbar sind — die Grenze $1/2$ ist also scharf.

Trakhtenbrots Arbeit gab Kinber Mitte der 70er Jahre den Anstoß, sich intensiv mit Häufigkeitsberechnungen zu beschäftigen [Kin72, Kin74, Kin75b, Kin75a, Kin76]. Er untersuchte beispielsweise die Frage, ob sich für $m > n/2$ die Fehlerzahl verringern oder die Frequenz erhöhen lässt, und gelangte zu folgenden Ergebnissen:

Satz 1.7 (Kinber [Kin72]) *Sei M eine Turingmaschine, $m > n/2$ und $L \subseteq \Sigma^*$ eine Sprache, die von M (m,n)-entschieden wird. Dann gilt:*

(1) Es ist im allgemeinen nicht effektiv möglich, eine Turingmaschine M' zu konstruieren, die die Sprache L (m',n')-entscheidet mit $n' - m' < n - m$.

(2) Für jedes $k \geq 1$ ist es effektiv möglich, eine Turingmaschine M'' zu konstruieren, die die Sprache L $(2m+k, 2n+k)$-entscheidet.

Teil (2) dieses Satzes zeigt also, dass man für $m > n/2$ die Frequenz (m,n)-entscheidbarer Sprachen effektiv erhöhen und beliebig nahe an 1 bringen kann.

Außerdem findet sich in dieser Arbeit das erste Ergebnis bzgl. ressourcenbeschränkter Häufigkeitsberechnungen. Kinber zeigt, dass es (bzgl. Zeit oder Platz) beliebig komplexe entscheidbare Sprachen gibt, die sich mit Häufigkeitsberechnungen effizienter berechnen lassen (d. h. deren (m,n)-Entscheidung in einer echt kleineren Zeit- oder Platzklasse liegt). Dieses Ergebnis konnte noch deutlich verschärft werden:

Satz 1.8 (Kinber [Kin75b]) *Zu jeder berechenbaren Funktion $t : \mathbb{N} \to \mathbb{N}$ und zu jedem $n \geq 2$ gibt es eine Sprache $L \subseteq \Sigma^*$, die sich nicht in Zeit t berechnen lässt, die aber in Linearzeit $(n-1,n)$-entscheidbar ist.*

Im Jahre 1976 übertrug Kinber das Modell der Häufigkeitsberechnungen auf endliche Automaten [Kin76]. Er untersuchte in dieser Arbeit (m,n)-erkennbare und (m,n)-fa-separierbare Sprachen, gelangte dabei jedoch zu teilweise fehlerhaften Resultaten. Wir beschäftigen uns mit diesen Begriffen in den Kapiteln 3 und 4 und werden die erzielten Ergebnisse dort aufführen.

Anfang der 80er Jahre begann Dëgtev mit der Untersuchung der Äquivalenz- und Inklusionsprobleme: für gegebene m, n, m', n' ist die Frage, ob $\Omega(m,n) = \Omega(m',n')$ oder $\Omega(m,n) \subseteq \Omega(m',n')$ gilt. Er kam zu folgendem Teilergebnis:

Satz 1.9 (Dëgtev [Dëg81]) *Seien $2m \leq n$, $2m' \leq n'$ und $n'-m' < n-m$. Dann gilt $\Omega(m,n) \setminus \Omega(m',n') \neq \emptyset$.*

Damit blieben zwei Fragen weiterhin offen:

(1) Gilt sogar $\Omega(m',n') \subsetneqq \Omega(m,n)$?

(2) Wie verhält sich die Inklusionsbeziehung für $n'-m' = n-m$?

Dëgtev führte seinen Beweis über den kombinatorischen Begriff (m,n)-*zulässiger Mengen* (engl. (m,n)-*admissible*), der leider nur eine hinreichende, aber keine notwendige Bedingung für eine Enthaltenseinsbeziehung darstellt. Lediglich für den Spezialfall „$\Omega(1,n') \subseteq \Omega(m,n)$?“ ermöglicht dieser Begriff eine exakte kombinatorische Charakterisierung.

Kummer und Stephan konnten das Äquivalenzproblem 1991 abschließend klären:

Satz 1.10 (Kummer, Stephan [KS91, KS95a, KS95b])
Seien $0 \leq n-2m < n'-2m'$. Dann gilt: $\Omega(m',n') \setminus \Omega(m,n) \neq \emptyset$.

Korollar 1.11 (Trakhtenbrot, Dëgtev, Kummer, Stephan)
Für $m \leq n$ und $m' \leq n'$ gilt:

$$\Omega(m,n) = \Omega(m',n') \;\Leftrightarrow\; (m > n/2 \wedge m' > n'/2) \vee (m = m' \wedge n = n').$$

Die Entscheidbarkeit des Inklusionsproblems konnte 1995 von McNicholl bewiesen werden:

Satz 1.12 (McNicholl [McN95]) *Die Menge*

$$\{(m, n, m', n') \mid m, n, m', n' \in \mathbb{N},\ \Omega(m, n) \subseteq \Omega(m', n')\}$$

ist entscheidbar.

Kummer und Stephan konnten McNicholls Beweis vereinfachen und erhielten durch Anpassung des Begriffes „(m, n)-zulässig“ eine komplette kombinatorische Charakterisierung der Inklusionsbeziehungen. Diese Charakterisierung beinhaltet jedoch die Untersuchung aller Teilmengen von $\mathbb{B}^n$, was mit heutigen Mitteln nur bis etwa $n = 10$ möglich ist.

Ab den 90er Jahren beschäftigten sich Kummer, Stephan, Hinrichs und Wechsung mit *polynomiell zeitbeschränkten Häufigkeitsberechnungen*, siehe z. B. [KS91, KS95b, HW97, Hin02]. Obwohl in der vorliegenden Arbeit ressourcenbeschränkte Häufigkeitsberechnungen nicht weiter betrachtet werden, wollen wir doch die wichtigsten Ergebnisse dieser Variante aufführen.

Nach Kinbers Satz 1.8 ist bereits klar, dass sich Trakhtenbrots Resultat *nicht* auf ressourcenbeschränkte Häufigkeitsberechnungen übertragen lässt. Kummer und Stephan konnten jedoch zeigen, dass der von Dëgtev eingeführte Begriff der (m, n)-zulässigen Mengen für eine vollständige kombinatorische Charakterisierung polynomiell zeitbeschränkter Häufigkeitsberechnungen ausreicht [KS91]. Auch hier verhindert die kombinatorische Explosion die praktische Lösung des theoretisch entscheidbaren Problems.

Hinrichs und Wechsung konnte jedoch für konstante Fehlerzahl folgende geschlossene Charakterisierung für die Inklusionsbeziehung der Polynomialzeit-Klassen (m, n)P zeigen:

Satz 1.13 (Hinrichs, Wechsung [HW97]) *Für alle $d \geq 0$ gilt:*

(1) $(m + 1, m + 1 + d)\mathrm{P} \subsetneqq (m, m + d)\mathrm{P}$ *für alle $m < 2^d$.*

(2) $\exists$ *eine Konstante $c(d) \in \mathbb{N}$:* $(m, m + d)\mathrm{P} = (m + 1, m + 1 + d)\mathrm{P}$ *für alle $m \geq c(d)$.*

Die Konstante $c(d)$ wurde mit Hilfe des Ramsey-Theorems ermittelt, d. h. die Lücke zwischen 2^d und $c(d)$ ist extrem groß. Es wird jedoch vermutet, dass $c(d) = 2^d$ für alle $d \geq 1$ gewählt werden kann. Die bereits verifizierten

Spezialfälle $d = 1, 2$ durch Hinrichs [Hin94] und $d = 3$ durch Hertrampf und Minnameier [Min04, HM05] sprechen für diese Vermutung.

Nachdem sich Kinber 1976 mit Häufigkeitsberechnungen durch endliche Automaten beschäftigt hatte, wurden diese ab dem Jahre 2000 erneut untersucht [ADHP00, Tan02a, Tan02b, ADH03, ADHP05]. Dieser Aspekt der Häufigkeitsberechnungen ist zentraler Bestandteil der vorliegenden Dissertation. Die erzielten Ergebnisse werden in den Kapiteln 3 und 4 ausführlich dargestellt.

1.4 Verwandte Berechenbarkeitsmodelle

Seit den 80er Jahren wurden viele Berechenbarkeitsbegriffe untersucht, die direkte Beziehungen zu Häufigkeitsberechnungen haben oder diesen recht ähnlich sind. Wir werden sie in dieser Arbeit nicht genauer untersuchen, führen sie der Vollständigkeit halber aber trotzdem kurz auf. Zwei Übersichtsartikel von Gasarch und Stephan erläutern diese Begriffe, deren Beziehungen untereinander und die erzielten Ergebnisse ausführlich [Gas91, GS99].

(Multi-)Selektive Mengen. Eine Sprache $L \subseteq \Sigma^*$ heißt *selektiv*, falls eine berechenbare Funktion $f : \Sigma^* \times \Sigma^* \to \Sigma^*$ existiert mit

$$f(x, y) \in \{x, y\} \quad \text{und} \quad \{x, y\} \cap L \neq \emptyset \Rightarrow f(x, y) \in L.$$

Ein Selektionsalgorithmus wählt also aus zwei Wörtern x und y eines aus, das „mit größerer Wahrscheinlichkeit“ zur Sprache L gehört. Der Begriff selektiver Mengen wurde 1966 von Jockusch eingeführt [Joc66, Joc68]; er selbst nannte diese Mengen *semirekursiv*.

Selman übertrug diesen Begriff 1979 auf Polynomialzeitberechnungen: eine Menge heißt P-*selektiv*, falls ein polynomieller Selektionsalgorithmus dafür existiert [Sel79]. In diesem komplexitätstheoretischen Kontext erfuhren selektive Mengen viel Beachtung (siehe z. B. [HT03, Hem04]) und schließlich eine weitere Verallgemeinerung zu *multi-selektiven* Mengen durch Hemaspaandra, Jiang, Rothe und Watanabe [HJRW97]: eine Sprache L heißt k-*selektiv*, falls eine berechenbare Funktion f so existiert, dass für jedes $n \geq 1$ und für paarweise verschiedene $x_1, \ldots, x_n \in \Sigma^*$ gilt:

1. $f(x_1, \ldots, x_n) \in \{x_1, \ldots, x_n\}$ und
2. $|\{x_1, \ldots, x_n\} \cap L| \geq k \Rightarrow f(x_1, \ldots, x_n) \in L$.

Wie man sich leicht klarmachen kann, entsprechen die selektiven Mengen genau den 1-selektiven Mengen. Im Polynomialzeitfall bilden die multi-selektiven Mengen eine echte Hierarchie [HJRW97].

In Abschnitt 2.4 werden wir zeigen, dass jede k-selektive Menge $(1, k+1)$-entscheidbar ist, die Umkehrung (jede Menge in $\Omega(1, k+1)$ ist k-selektiv) allerdings nicht gilt.

Der Zusammenhang P-selektiver Mengen zu Häufigkeitsberechnungen (in Polynomialzeit) wurde von Hinrichs untersucht [Hin02]. Die Frage ist deshalb interessant, weil sowohl die P-Selektivität von SAT als auch die (m, n)-Endscheidbarkeit von SAT implizieren würden, dass P=NP gilt.

Es folgen zwei Berechenbarkeitsmodelle, *Verboseness* und *Bounded Queries*, die von Beigel 1987 in seiner Dissertation eingeführt wurden [Bei87].

(Strong) Verboseness. Für $1 \leq m \leq 2^n$ nennen wir eine Sprache L *strongly* (m, n)*-verbose*[2] ($L \in V_S(m, n)$), falls eine Turingmaschine existiert, die auf Eingabe von n paarweise verschiedenen Wörtern $x_1, \ldots, x_n \in \Sigma^*$ eine Menge von höchstens m Vektoren aus $\mathbb{B}^n$ berechnet, die $\chi_L^{(n)}(x_1, \ldots, x_n)$ enthält. Es besteht folgender Zusammenhang zu Häufigkeitsberechnungen:

$$L \in \Omega(1, n) \quad \Leftrightarrow \quad L \in V_S(2^n - 1, n).$$

Wenn $L \in \Omega(1, n)$ durch eine Turingmaschine M ist, dann schließt die Antwort $M(x_1, \ldots, x_n) =: b \in \mathbb{B}^n$ den Vektor $\overline{b}$ als charakteristischen Vektor von $(x_1, \ldots, x_n)$ aus. Wir können also die Menge $\mathbb{B}^n \setminus \{\overline{b}\}$ ausgeben, d. h. $L \in V_S(2^n - 1, n)$.

Ist andererseits $L \in V_S(2^n - 1, n)$ durch eine Turingmaschine M, so gibt es auf Eingabe $(x_1, \ldots, x_n)$ mindestens einen Vektor $b \in \mathbb{B}^n$, der von M als charakteristischer Vektor von $(x_1, \ldots, x_n)$ ausgeschlossen wird. Dann stimmt der Vektor $\overline{b}$ aber an mindestens einer Stelle mit $\chi_L^{(n)}(x_1, \ldots, x_n)$ überein, denn sonst wäre der n-fache charakteristische Vektor gleich b. Also gilt $L \in \Omega(1, n)$.

[2] Den Begriff *verbose* kann man am ehesten mit „geschwätzig“ übersetzen; die Bedeutung für das Beiwort *strong* ist an dieser Stelle nicht wichtig.

Beschränkte Anzahl von Orakelfragen. Dieses Berechenbarkeitsmodell ist über Orakel definiert, die bei jeder Frage ein Bit an Information preisgeben (d. h. eine Ja-Nein-Antwort geben). Für eine nicht berechenbare Funktion f und ein Orakel X fragt man sich, ob sich die Funktion f durch k-maliges Befragen von X doch berechnen lässt. Die Beschränkung auf k Orakelfragen gibt diesem Modell seinen Namen (im englischen spricht man von *bounded queries*).[3] Wie bereits erwähnt führte Beigel diesen Begriff 1987 ein. Den Zusammenhang zu Häufigkeitsberechnungen untersuchte er 1996 in einer gemeinsamen Arbeit mit Gasarch und Kinber [BGK96].

Die Funktionenklasse $\mathrm{FQ}(k, X)$ ist definiert als die Klasse aller Funktionen, die sich mit k Anfragen an das Orakel X berechnen lassen. Die Beziehung zu den V_S-Klassen ist:

$$L \in V_S(2^k, n) \quad \Leftrightarrow \quad \exists \text{ Orakel } X : \chi_L^{(n)} \in \mathrm{FQ}(k, X).$$

Sei $L \in V_S(2^k, n)$, und auf Eingabe $(x_1, \ldots, x_n)$ sei die Menge der potentiellen charakteristischen Vektoren $\{v_1, \ldots, v_{2^k}\}$. Wir müssen den Index i ermitteln, für den $v_i = \chi_L^{(n)}(x_1, \ldots, x_n)$ gilt. Dies können wir mit k Anfragen an ein Orakel X bewerkstelligen, das die Fragen „1. Bit von i?“ bis „k. Bit von i?“ korrekt beantwortet. Also gilt $\chi_L^{(n)} \in \mathrm{FQ}(k, X)$ für ein geeignetes Orakel X.

Sei anderseits $\chi_L^{(n)} \in \mathrm{FQ}(k, X)$ für ein Orakel X durch eine Turingmaschine $M^?$.[4] Wir führen nun 2^k viele Simulationen von $M^?$ durch und ersetzen die k Antworten des Orakels durch alle 2^k möglichen Kombinationen. Jede dieser Simulationen von $M^?$ liefert uns genau einen Antwortvektor aus $\mathbb{B}^n$. Da die Antworten des richtigen Orakels in der Menge der möglichen Kombinationen enthalten waren, ist auch $\chi_L^{(n)}(x_1, \ldots, x_n)$ in der Menge der Antworten enthalten. Also ist $L \in V_S(2^k, n)$.

Kardinalität. Ein weiterer Forschungsgegenstand sind Kardinalitätsberechnungen, bei denen man für eine Sprache $L \subseteq \Sigma^*$ und eine Eingabe $x_1, \ldots, x_n \in \Sigma^*$ nicht an $\chi_L^{(n)}(x_1, \ldots, x_n)$ selbst interessiert ist, sondern lediglich an der Anzahl der zu L gehörenden Wörter:

$$\#_L^{(n)}(x_1, \ldots, x_n) := \#\{1 \leq i \leq n \mid x_i \in L\}.$$

[3] Berechenbare Funktionen werden nicht per se ausgeschlossen, sind aber uninteressant, da sie stets mit 0 Anfragen an X berechnet werden können.

[4] Mit $M^?$ bezeichnen wir wie in der Komplexitätstheorie üblich eine Orakel-Turingmaschine ohne festgelegtes Orakel.

Beigel stellte 1987 folgende Vermutung auf [Bei87]:

$$L \text{ ist entscheidbar} \quad \Leftrightarrow \quad \exists \text{ Orakel } X : \#_L^{(2^n)} \in \mathrm{FQ}(n, X).$$

Eine äquivalente Formulierung ist, dass eine Turingmaschine existiert, die auf Eingabe von je n verschiedenen Wörtern $x_1, \ldots, x_n \in \Sigma^*$ eine echte Teilmenge von $\{0, \ldots, n\}$ ausgibt, die den Wert $\#_L^{(n)}$ enthält [Kum92]. Owings konnte diese Vermutung für den Spezialfall $\#_L^{(2)}$ zeigen [Owi89]. Erst Kummer gelang es 1992, das sog. *Kardinalitätstheorem* allgemein zu beweisen [Kum92]. Eine ausführliche Darstellung des Zusammenhangs zwischen Häufigkeits- und Kardinalitätsberechnungen findet sich in [HKO92].

Tantau untersuchte die Begriffe Verboseness und Kardinalitätsberechnung auch im Zusammenhang mit endlichen Automaten [Tan02a, Tan02b, Tan03]. Es wird vermutet, dass sich das Kardinalitätstheorem auf endliche Automaten überträgt, doch bislang konnten lediglich Spezialfälle bewiesen werden.

1.5 Gliederung

Die Ergebnisse dieser Arbeit sind in drei Kapitel unterteilt.

In *Kapitel 2, Häufigkeitsberechnungen durch Turingmaschinen*, werden wir zwei verschiedene Beweise von Trakhtenbrots zentralem Resultat angeben. Der erste ist eine elementare Darstellung von Trakhtenbrots Originalbeweis, der nur in russischer Sprache veröffentlicht wurde und Kenntnisse der Topologie voraussetzt und daher schwer zugänglich ist. Der zweite ist ein neuer kombinatorischer Beweis, der uns ermöglichen wird, das Ergebnis auf Häufigkeitsberechnungen durch endliche Automaten zu übertragen. Außerdem wird in einem kurzen Abschnitt die Beziehung der selektiven Mengen zu Häufigkeitsberechnungen untersucht.

In *Kapitel 3, Separierbarkeit von Mengen* zeigen wir, dass ein Analogon zu Trakhtenbrots Resultat für „allgemeine“ Häufigkeitsberechnungen für Sprachen, die durch endliche Automaten separierbar sind, nicht gelten kann. Zunächst arbeiten wir die fehlerhafte Stelle in Kinbers Beweis heraus, der dieses Analogon ja vermutete, und geben ein Gegenbeispiel an. Anschließend zeigen wir einige Ergebnisse bzgl. der Separierbarkeit von Pfad- und Anti-Pfadsprachen und beweisen schließlich, dass es rekursiv nicht separierbare Sprachen gibt, die mittels Häufigkeitsberechnungen mit lediglich logarithmisch vielen Fehlern separiert werden können (sogar durch endliche Automaten).

In *Kapitel 4, Reguläre Häufigkeitsberechnungen*, zeigen wir, dass sich Trakhtenbrots Resultat auf reguläre Häufigkeitsberechnungen überträgt. Dies bewerkstelligen wir, indem wir unseren neuen Beweis aus Kapitel 2 erneut inspizieren und uns vergewissern, dass das Vorgehen auch durch einen endlichen Automaten realisiert werden kann. Danach beschäftigen wir uns mit der Frage, welche Sprachen überhaupt (m, n)-erkennbar sind. Dazu beweisen wir ein Iterationskriterium und ziehen einige Folgerungen daraus, mit deren Hilfe wir für viele Sprachen ausschließen können, dass sie durch endliche Automaten häufigkeitsberechenbar sind. Wir zeigen außerdem, dass die Klasse der (m, n)-erkennbaren Sprachen eine boolesche Algebra ist, dass es bei der Vereinigung zweier $(1, n)$-erkennbarer Sprachen aber nicht immer möglich ist, diese wiederum mit Frequenz $1/n$ zu erkennen.

Kapitel 2

Häufigkeitsberechnungen durch Turingmaschinen

In diesem Kapitel werden wir uns mit dem ursprünglichen Begriff der Häufigkeitsberechnungen beschäftigen, d. h. mit Häufigkeitsberechnungen, die von Turingmaschinen ohne Ressourcenbeschränkung durchgeführt werden.

Ein grundlegendes Resultat stammt von Trakhtenbrot: die häufigkeitsberechenbaren Funktionen sind bereits (Turing-)berechenbar, wenn stets mehr als die Hälfte der Antworten richtig sind:

Satz 2.1 (Trakhtenbrot [Tra63]) *Sei $f : \Sigma^* \to \Sigma^*$ eine totale, (m, n)-berechenbare Funktion mit $m > n/2$. Dann ist f berechenbar.*

Wir werden für dieses Resultat zwei Beweise geben.

Der erste ist eine elementare Darstellung von Trakhtenbrots Originalbeweis, der zum einen Ergebnisse aus der Topologie verwendet und zum anderen nur auf russisch veröffentlicht wurde und damit in seiner ursprünglichen Form schwer zugänglich ist [Tra63]. Trakhtenbrot geht dabei in zwei Schritten vor: zunächst zeigt er das Ergebnis für Prädikate (also für 2-wertige Funktionen), um anschließend den allgemeinen Fall auf diesen Spezialfall zu reduzieren.

Danach geben wir einen neuen kombinatorischen Beweis an, der ohne die Reduktion vom Funktionen- auf den Prädikatenfall auskommt und eine weitere Schlussfolgerung zulässt: das Analogon zu Trakhtenbrots Resultat gilt auch für endliche Automaten (siehe Abschnitt 4.1). Letzteres Ergebnis wurde schon 1976 von Kinber postuliert, sein Beweis über separierbare Mengen erwies sich jedoch als fehlerhaft ([Kin76]; mehr dazu in Abschnitt 3.1).

2.1 Trakhtenbrots topologischer Beweis

Trakhtenbrots Beweis ist topologischer Art. Deswegen führen wir zunächst die dafür benötigten topologischen Begriffe ein. Bei der verwendeten Notation halten wir uns dabei an das Lehrbuch von Armstrong [Arm97].

Definition 2.2 (Topologie, offene Menge) *Sei X eine Menge und $\mathcal{S}$ ein System von Teilmengen von X. Dann definiert $\mathcal{S}$ eine* Topologie *auf X, wenn folgende Bedingungen erfüllt sind:*

(1) Die Vereinigung einer beliebigen Familie von Mengen aus $\mathcal{S}$ gehört zu $\mathcal{S}$.

(2) Der Durchschnitt endlich vieler Mengen aus $\mathcal{S}$ gehört zu $\mathcal{S}$.

Die Mengen in $\mathcal{S}$ heißen offene Mengen.

Da wir für (1) die leere Vereinigung und für (2) den leeren Durchschnitt zulassen, sind $\emptyset$ und X stets offene Mengen.

Den *topologischen Raum* $(X, \mathcal{S})$ bezeichnen wir häufig einfach mit X. Es wird stets aus dem Zusammenhang heraus klar sein, welche Topologie auf X gemeint ist.

Definition 2.3 (abgeschlossene Menge) *Eine Teilmenge A einer Topologie X heißt* abgeschlossen*, wenn ihr Komplement $X \setminus A$ offen ist.*

Definition 2.4 (Umgebung) *Sei $x \in X$ Punkt eines topologischen Raumes X. Eine Menge $U \subseteq X$ heißt* Umgebung *von x, falls eine offene Menge V mit $x \in V \subseteq U$ existiert.*

Ist die Menge U außerdem offen/abgeschlossen/etc., so sprechen wir von einer offenen/abgeschlossenen/etc. Umgebung.

Für den Nachweis, dass eine Menge A abgeschlossen ist, genügt es offensichtlich zu zeigen, dass jeder Punkt $x \in X \setminus A$ in einer Umgebung liegt, die disjunkt zu A ist.

Definition 2.5 (Überdeckung) *Sei X ein topologischer Raum. Eine Überdeckung einer Menge $A \subseteq X$ ist eine Familie von Mengen, deren Vereinigung A enthält. Besteht die Familie ausschließlich aus offenen Mengen, so nennen wir die Überdeckung* offen.

Definition 2.6 (quasikompakt, kompakt) *Ein toplogischer Raum X heißt* quasikompakt, *falls jede offene Überdeckung von X bereits eine endliche Teilüberdeckung enthält:*

$$X = \bigcup \{U_i \mid i \in I, U_i \text{ offen}\} \quad \Rightarrow \quad \exists J \subseteq I, J \text{ endlich} : X = \bigcup_{j \in J} U_j$$

Der Raum X heißt kompakt, *wenn er quasikompakt ist und wenn je zwei verschiedene Punkte aus X disjunkte Umgebungen besitzen (diese Trennungsbedingung wird auch* Hausdorffsch *oder* T2 *genannt).*

Definition 2.7 (isolierter Punkt) *Sei X ein topologischer Raum und $A \subseteq X$. Ein Punkt $x \in A$ heißt* isolierter Punkt *von A, falls eine offene Umgebung U von x existiert mit $U \cap A = \{x\}$.*

In dieser Arbeit führen wir noch die folgende Trennungsbedingung ein:

Definition 2.8 *Die Trennungsbedingung (T) für einen topologischen Raum X ist definiert als:*

(T) Für alle Punkte $x, y \in X$, $x \neq y$, existiert eine offene und abgeschlossene Menge $A \subseteq X$ mit $x \in A$ und $y \notin A$.[1]

Lemma 2.9 *Sei X ein topologischer Raum, der kompakt ist und Bedingung (T) erfüllt. Dann existiert in jeder abzählbaren, abgeschlossenen Menge $A \subseteq X$ ein isolierter Punkt.*

[1] Für kompakte Räume X ist die Bedingung (T) gleichbedeutend damit, dass X *total unzusammenhängend* ist. Dies folgt aus [Bou66, Chapter II, §4, Proposition 6]. Die später benutzte Topologie $\mathbb{B}^\omega$ ist kompakt und total unzusammenhängend.

Beweis: Sei $A = \{a_1, a_2, \ldots\} \subseteq X$ eine abzählbare, abgeschlossene Menge. Da X die Bedingung (T) erfüllt, gibt es für je zwei Punkte $x, y \in A$ eine offene und abgeschlossene Umgebung von x, die y nicht enthält. Für $k = 1, 2, \ldots$ definieren wir Mengen A_k so lange, bis $A \setminus (\bigcup_{k\geq 1} A_k)$ nur noch einen Punkt enthält:

> Seien $i, j \in \mathbb{N}$, $i < j$, die beiden kleinsten Indizes, für die a_i und a_j Punkte aus A sind, die nicht von $\bigcup_{\ell=1}^{k-1} A_\ell$ überdeckt sind. Dann definieren wir A_k als eine offene und abgeschlossene Umgebung von a_i, die a_j nicht enthält.

Angenommen, das Verfahren bricht nach k Schritten ab und x ist der verbleibende Punkt in $A \setminus (\bigcup_{\ell=1}^{k} A_\ell)$. Dann ist x ein isolierter Punkt, denn $U := X \setminus (\bigcup_{\ell=1}^{k} A_\ell)$ ist eine offene Umgebung von x mit $U \cap A = \{x\}$.

Sollte das Verfahren nicht abbrechen, dann können wir die dabei konstruierte abzählbar unendliche Überdeckung $\bigcup_{k\geq 1} A_k$ von A zu einer offenen Überdeckung von X ergänzen, indem wir die offene Menge $X \setminus A$ hinzunehmen.

Aus der Kompaktheit von X folgt, dass für die Überdeckung von X bereits endlich viele offene Mengen ausreichen. Entfernen wir aus dieser endlichen Überdeckung die Menge $X \setminus A$, erhalten wir eine endliche Überdeckung von A. Obiges Verfahren hätte also nach endlich vielen Schritten abbrechen müssen. □

Der topologische Raum, der Trakhtenbrots Beweis zugrunde liegt, ist $\mathbb{B}^\omega$ mit den offenen Mengen $\bigcup_{i\in I} u_i\mathbb{B}^\omega$ für Indexmengen I und $u_i \in \mathbb{B}^*$. Wir werden die Tatsache benötigen, dass $\mathbb{B}^\omega$ die Trennungsbedingung (T) erfüllt und kompakt ist.

Behauptung: $\mathbb{B}^\omega$ erfüllt Bedingung (T).

Beweis: Seien $x, y \in \mathbb{B}^\omega$ zwei verschiedene Punkte. Sei $w \in \mathbb{B}^*$ der maximale gemeinsame Präfix von x und y, d. h. ohne Einschränkung $x = w0x'$ und $y = w1y'$ für Zeichenketten $x', y' \in \mathbb{B}^\omega$. Dann ist x in der offenen Menge $A = w0\mathbb{B}^\omega$ enthalten, y hingegen nicht. Die Menge A ist abgeschlossen, denn ihr Komplement lässt sich als offene Menge darstellen:

$$\{w'\overline{b}\mathbb{B}^\omega \mid w'b \preceq w0, w' \in \mathbb{B}^*, b \in \mathbb{B}\}.$$

□

Im folgenden werden wir eine präfixabgeschlossene Menge $T \subseteq \mathbb{B}^*$ häufig als Binärbaum mit Knotenmenge T und Kantenmenge $\{(t, tb) \mid t, tb \in T, b \in \mathbb{B}\}$ interpretieren.

Für den folgenden Beweis werden wir außerdem das wohlbekannte Lemma von König benötigen.

Lemma 2.10 (König 1936, [Kön36]) *Sei $T = (V, E)$ ein Baum mit Knotenmenge V, Kantenmenge E und endlichen Knotengraden. Wenn V unendlich ist, dann enthält T einen unendlich langen Pfad.*

Behauptung: $\mathbb{B}^\omega$ ist kompakt.

Beweis: Dies ist wohlbekannt und folgt aus dem Satz von Tychonoff. Wir geben einen elementaren Beweis, der keine topologischen Vorkenntnisse benutzt.

Wir müssen lediglich zeigen, dass $\mathbb{B}^\omega$ quasikompakt ist, denn $\mathbb{B}^\omega$ erfüllt Trennungsbedingung (T) und ist damit erst recht Hausdorffsch.

Sei $X = \bigcup_{i \in I} u_i \mathbb{B}^\omega$ für eine Indexmenge I und $u_i \in \mathbb{B}^*$. Für endliche I ist nichts zu zeigen; also nehmen wir an, I ist unendlich.

In einem ersten Schritt eliminieren wir alle Indizes $i \in I$, für die ein $j \in I$ existiert mit $w_j \preceq w_i$ (denn dann überdeckt $w_j \mathbb{B}^\omega$ die Menge $w_i \mathbb{B}^\omega$ vollständig). Die neue Indexmenge sei J, und wir nehmen an, dass auch J unendlich ist.

Diese Annahme führen wir zu einen Widerspruch, indem wir einen Punkt angeben, der nicht in $\bigcup_{j \in J} u_j \mathbb{B}^\omega$ enthalten ist. Wir betrachten den unendlichen Binärbaum, dessen Knoten genau den u_j (und deren Präfixen) entsprechen. Nach Lemma von König (Lemma 2.10) gibt es in diesem Baum einen unendlichen Pfad α. Angenommen, es gäbe ein $j \in J$ mit $u_j \preceq \alpha$. Dann würde nach unserem ersten Schritt $u_j \not\preceq u_{j'}$ für alle $j' \in J \setminus \{j\}$ gelten, d. h. u_j wäre ein Blatt — Widerspruch! Also kann es den unendlichen Pfad nicht geben und J muss endlich gewesen sein. □

Nun können wir uns dem Beweis von Satz 2.1 widmen. Dieser wird in zwei Schritte aufgeteilt: zunächst zeigen wir das Ergebnis für 2-wertige Funktionen (dies entspricht Sprachen, da deren charakteristische Funktionen 2-wertig sind); anschließend wird die Aussage für allgemeine Funktionen auf diesen Spezialfall zurückgeführt.

Lemma 2.11 *Sei $L \subseteq \Sigma^*$ eine (m, n)-entscheidbare Sprache mit $m > n/2$. Dann ist L entscheidbar.*

Beweis: Wir assoziieren eine Sprache $L \subseteq \Sigma^*$ mit einem unendlichen Wort $\alpha \in \mathbb{B}^\omega$ wie folgt: sei $\{w_1, w_2, w_3, \ldots\}$ eine Aufzählung von Σ^* (z. B. die längenlexikographische Aufzählung); dann gilt:

$$\alpha = \chi_L(w_1)\chi_L(w_2)\chi_L(w_3)\cdots$$

Sei M eine Turingmaschine, die eine Sprache $L \subseteq \Sigma^*$ (m,n)-entscheidet, d. h. auf je n verschiedenen Eingaben $x_1, \ldots, x_n \in \Sigma^*$ wird eine Ausgabe $(b_1, \ldots, b_n) \in \mathbb{B}^n$ produziert, die (m,n)-konsistent mit der Sprache L ist.

Wir können die Ergebnisse von M auf allen n-Tupeln von Eingaben aufzählen und damit die Menge der mit M konsistenten Sprachen definieren:

$$\mathcal{M}_M = \left\{ \alpha_1\alpha_2\ldots \in \mathbb{B}^\omega \;\middle|\; \begin{array}{l} \text{die Sprache } K \subseteq \Sigma^* \text{ mit } \chi_K(w_i) = \alpha_i \\ \forall i \geq 1 \text{ ist } (m,n)\text{-konsistent mit } M \end{array} \right\}$$

Für $\Sigma = \{a, b\}$, $m = 2$ und $n = 3$ ergibt beispielsweise die Antwort $(1, 0, 0)$ auf die Eingabe (ε, a, b) die möglichen Anfangsstücke $\alpha_1\alpha_2\alpha_3 = 100$, 000, 110 und 101.

Behauptung: $\mathcal{M}_M$ ist abzählbar.

Beweis: Seien $L, K \in \Sigma^*$ zwei Sprachen, die mit M konsistent sind. Dann behaupten wir, dass sich χ_L und χ_K an maximal $2(n-m)$ Stellen unterscheiden, woraus unmittelbar die Abzählbarkeit der Menge $\mathcal{M}_M$ folgt.

Angenommen, χ_L und χ_K differieren an mehr als $2(n-m)$ Stellen. Seien $x_1, \ldots, x_{2(n-m)+1}$ die ersten $2(n-m)+1$ sich unterscheidenden Stellen. Wegen $m > n/2$ gilt $2(n-m)+1 < 2(n-n/2)+1 = n+1$; wir können also die Ausgabe von M auf Eingabe $(x_1, \ldots, x_{2(n-m)+1}, x_{2(n-m)+2}, \ldots, x_n)$ berechnen, wobei die letzten Eingaben beliebig gewählt werden können, um auf n Eingaben aufzufüllen.

Nun gilt für alle $1 \leq i \leq 2(n-m)+1$, dass entweder $\chi_L(x_i)$ oder $\chi_K(x_i)$ von der Antwort M's auf x_i abweicht. Daraus folgt aber, dass entweder χ_L oder χ_K mehr als $n-m$ Abweichungen von M hat — Widerspruch! □

Behauptung: $\mathcal{M}_M$ ist eine abgeschlossene Menge.

Beweis: Sei $\beta_1\beta_2\ldots \in \Sigma^\omega \setminus \mathcal{M}_M$, also eine Sprache, die nicht konsistent mit den Antworten von M ist. Sei $(x_1, \ldots, x_n)$, $x_i \in \Sigma^*$, eine Eingabe für M, die an den entsprechenden Stellen von $\beta_1\beta_2\ldots$ weniger als m Übereinstimmungen hat, und ℓ der größte Index der x_i unter der längenlexikographischen

Aufzählung von Σ^*. Dann ist $\beta_1\beta_2\ldots\beta_\ell\Sigma^\omega$ eine offene Menge, die disjunkt zu $\mathcal{M}_M$ ist. □

Da der topologische Raum $\mathbb{B}^\omega$ kompakt ist und die Bedingung (T) erfüllt, existiert nach Lemma 2.9 ein isolierter Punkt in $\mathcal{M}_M$. Wir müssen nur noch zeigen, dass dieser einer entscheidbaren Sprache entspricht.

Behauptung: Jeder isolierte Punkt in $\mathcal{M}_M$ entspricht einer entscheidbaren Sprache.

Beweis: Sei $\alpha_1\alpha_2\ldots \in \mathcal{M}_M$ ein isolierter Punkt, d. h. es existiert ein $i_0 \geq 0$ mit $\alpha_1\alpha_2\ldots\alpha_{i_0}\Sigma^\omega \cap \mathcal{M}_M = \{\alpha_1\alpha_2\ldots\}$. Anders ausgedrückt heißt das, dass jeder andere unendliche Pfad mit Präfix $\alpha_1\alpha_2\ldots\alpha_{i_0}$ inkonsistent bzgl. der Antworten von M ist.

Ist ein Pfad $\beta_1\beta_2\ldots$ inkonsistent zu M, dann gibt es bereits einen endlichen Präfix davon, der diese Inkonsistenz bezeugt. Mit dieser Beobachtung können wir einen Entscheidungsalgorithmus für die Sprache $\alpha_1\alpha_2\ldots$ angeben.

Für Eingaben $w_i \in \Sigma^*$ mit Index $i \leq i_0$ schlagen wir die Antwort in einer Tabelle nach. Sei also $w_i \in \Sigma^*$ eine Eingabe mit $i > i_0$. Wir berechnen $\alpha_{i_0+1}, \ldots, \alpha_i$ Bit für Bit und geben am Ende α_i aus. Angenommen, $\alpha_1\cdots\alpha_j$ sei bereits berechnet. Mit einer Breitensuche durchlaufen wir parallel die Bäume $T_{j,0} = \alpha_1\cdots\alpha_j 0\mathbb{B}^\omega$ und $T_{j,1} = \alpha_1\cdots\alpha_j 1\mathbb{B}^\omega$ und überprüfen für alle Eingabetupel (bestehend aus jeweils n Wörtern der bereits besuchten Teile von $T_{j,0}$ bzw. $T_{j,1}$) die Konsistenz bzgl. M. Inkonsistente Pfade werden nicht weiterverfolgt. Wir brechen ab, falls sich alle Pfade in $T_{j,b}$ ($b \in \{0,1\}$) als inkonsistent erwiesen haben, und setzen dann $\alpha_{j+1} = \overline{b}$.

Seien $S_{j,0}$ und $S_{j,1}$ die tatsächlich besuchten, konsistenten Teile der Bäume $T_{j,0}$ und $T_{j,1}$. Angenommen, die parallele Breitensuche würde nicht abbrechen. Dann wären sowohl $S_{j,0}$ als auch $S_{j,1}$ unendlich groß. Nach Lemma von König (Lemma 2.10) existieren dann in beiden Bäumen $S_{j,0}$ und $S_{j,1}$ unendliche Pfade $\alpha_1\alpha_2\ldots\alpha_j 0\alpha_{0,j+2}\alpha_{0,j+3}\ldots$ bzw. $\alpha_1\alpha_2\ldots\alpha_j 1\alpha_{1,j+2}\alpha_{1,j+3}\ldots$. Beide Pfade würden Sprachen entsprechen, die mit M konsistent sind — ein Widerspruch zu der Voraussetzung, dass $\alpha_1\alpha_2\ldots$ ein isolierter Punkt in der Menge aller konsistenten Pfade $\mathcal{M}_M$ ist! □ □

Beweis (von Satz 2.1): Sei M eine Turingmaschine, die eine Funktion $f : \Sigma^* \to \Sigma^*$ (m,n)-berechnet. Wir werden zeigen, dass der Graph von f durch eine Turingmaschine M' mit einer Häufigkeit von m'/n' berechnet werden kann. Die Parameter m' und n' werden sich von m und n unterscheiden, aber

es wird weiterhin $m'/n' > 1/2$ gelten. Nach Lemma 2.11 ist der Graph von f dann entscheidbar und die Funktion f somit berechenbar.

Der Parameter n' wird ein Vielfaches von n sein — sei also $n' = h \cdot n$ für ein h, das wir erst am Ende des Beweises bestimmen werden. Desweiteren sei $(\langle x_1, y_1\rangle, \ldots, \langle x_{n'}, y_{n'}\rangle)$ eine Eingabe für M'. Wir teilen die Paare $\langle x_i, y_i\rangle$ in drei Klassen ein:

1. Das Paar heißt *einfach*, falls für alle $j \neq i$ gilt: $x_i \neq x_j$.
2. Das Paar heißt *doppelt*, falls genau ein $j \neq i$ existiert mit $x_i = x_j$.
3. Andernfalls heißt das Paar *mehrfach*.

Die Anzahl der einfachen, doppelten und mehrfachen Paare bezeichnen wir mit ν_1, ν_2 und ν_3 (es gilt $\nu_1 + \nu_2 + \nu_3 \leq n' = h \cdot n$). Wir geben die Ausgabe auf den drei Typen von Paaren an und schätzen die maximale Fehlerzahl ab.

1. Die ν_1 x-Werte der einfachen Paare bilden $\lfloor \frac{\nu_1}{n} \rfloor$ Blöcke der Größe n; die restlichen x-Werte füllen wir auf ein ganzzahliges Vielfaches von n auf und erhalten so einen weiteren Block der Größe n. Für jeden Block berechnen wir den zugehörigen Ausgabevektor von M. Ist die Ausgabe von M auf ein x_i gleich y, so setzen wir das Ausgabebit von M' auf $\langle x_i, y_i\rangle$ genau dann auf 1, wenn $y_i = y$ gilt.

   Die Fehlerzahl π_1 kann in diesem Fall abgeschätzt werden durch

$$\pi_1 \leq \underbrace{\left\lfloor \frac{\nu_1}{n} \right\rfloor (n-m)}_{\text{vollständige Blöcke}} + \underbrace{(n-m)}_{\text{aufgefüllter Block}} \leq (n-m)\left(\frac{\nu_1}{n} + 1\right)$$

2. Aus den doppelten Paaren stehen nur $\nu_2/2$ x_i zur Verfügung, die wir wie bei den einfachen Paaren auf ein Vielfaches von n auffüllen und zu Blöcken der Größe n zusammenfassen, um die Turingmaschine M darauf anwenden zu können. Die Antwort definiert sich wie im ersten Fall, mit dem Unterschied, dass jede Antwort von M auf ein x_i zu zwei Antworten für M' führt.

   Jede Antwort von M kann somit auch zu zwei Fehlern auf Paaren $\langle x_i, y_1\rangle$ und $\langle x_i, y_2\rangle$ führen. Die Fehlerzahl π_2 lässt sich also abschätzen durch

$$\pi_2 \leq 2\Big(\underbrace{\left\lfloor \frac{\nu_2}{2n} \right\rfloor (n-m)}_{\text{vollständige Blöcke}} + \underbrace{(n-m)}_{\text{aufgefüllter Block}} \Big) \leq (n-m)\left(\frac{\nu_2}{n} + 2\right)$$

3. Auf den mehrfachen Paaren verfahren wir genauso, allerdings stellen wir hier höchstens $\nu_3/3$ Fragen (aufgerundet auf ein Vielfaches von n). Für jedes mehrfache Paar können wir wie für doppelte Paare höchstens 2 Fehler machen.

   In diesem Fall lässt sich die Fehlerzahl also abschätzen durch

$$\pi_3 \leq 2\Big(\underbrace{\left\lfloor \frac{\nu_3}{3n} \right\rfloor (n-m)}_{\text{vollständige Blöcke}} + \underbrace{(n-m)}_{\text{aufgefüllter Block}} \Big) \leq (n-m)\left(\frac{\nu_3}{n} + 2\right)$$

Als obere Schranke für die Gesamtfehlerzahl π erhalten wir damit:

$$\pi = \pi_1 + \pi_2 + \pi_3 \leq (n-m)\left(\frac{\nu_1 + \nu_2 + \nu_3}{n} + 5\right) \leq (n-m)\,(h+5)\,.$$

Da weniger als die Hälfte der Antworten falsch sein sollen, muss gelten:

$$\frac{(n-m)(h+5)}{h \cdot n} < \frac{1}{2},$$

woraus sich $h > 10(n-m)/(2m-n)$ ergibt (ein nicht-negativer Wert, da $n \geq m$ und $m > n/2$). □

2.2 Ein neuer kombinatorischer Beweis

In diesem Abschnitt wollen wir einen kombinatorischen Beweis für Trakhtenbrots Resultat (Satz 2.1) geben. Wir benötigen dafür folgendes Lemma:

Lemma 2.12 *Seien $m > n/2$ und $f : \Sigma^* \to \Sigma^*$ eine Funktion, die durch eine Turingmaschine M (m,n)-berechnet bzw. durch einen n-DFA A (m,n)-fa-berechnet wird. Mit $X \in \{M, A\}$ bezeichnen wir im folgenden entweder die Turingmaschine M oder den endlichen Automaten A.*

(1) Seien $x_1, \ldots, x_{n-1}, y_1, \ldots, y_n \in \Sigma^*$ paarweise verschiedene Wörter, und für $1 \leq i \leq n$ sei $f_i = X(x_1, \ldots, x_{n-1}, y_i)[n]$. Wenn sich die Vektoren $(f_1, \ldots, f_n)$ und $X(y_1, \ldots, y_n)$ an mindestens m Stellen unterscheiden, dann gilt:

$$\exists 1 \leq i \leq n : f_i \neq f(y_i).$$

(Siehe Abbildung 2.1.)

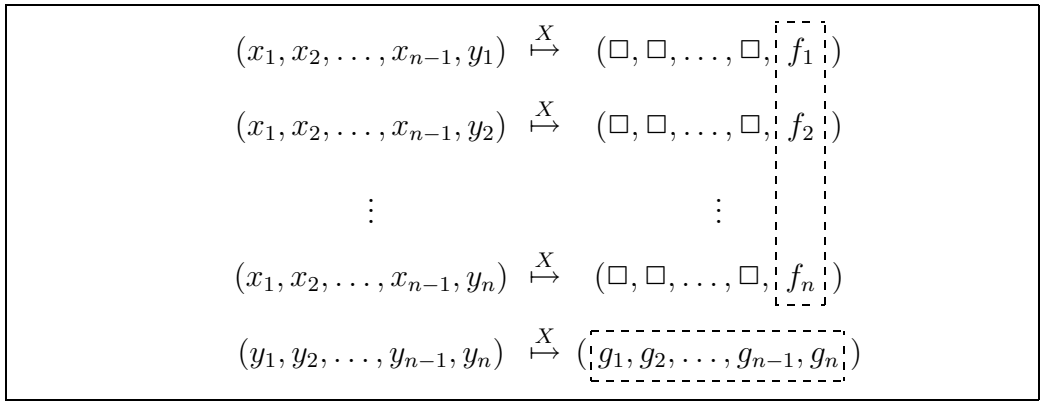


Abbildung 2.1: Veranschaulichung von Lemma 2.12 (1). Die beiden zu vergleichenden Vektoren sind durch gestrichelte Rahmen gekennzeichnet. $\square$ bezeichnet Ausgabewerte, die für die Aussage des Lemmas ohne Bedeutung sind.

(2) Wenn für alle paarweise verschiedenen $x_1, \ldots, x_{n-1} \in \Sigma^*$ unendlich viele $y \in \Sigma^*$ existieren mit $X(x_1, \ldots, x_{n-1}, y)[n] \neq f(y)$, dann existieren für jede Konstante $k \geq n$ Wörter $y_1, \ldots, y_k \in \Sigma^*$ mit:

> *$\forall 1 \leq i_1 < \ldots < i_n \leq k$: die Wörter $x_1, \ldots, x_{n-1}, y_{i_1}, \ldots, y_{i_n}$ erfüllen die Prämisse aus Teil (1) des Lemmas.*

Beweis:

(1) Angenommen, ein solches i existiert nicht. Da der Vektor $X(y_1, \ldots, y_n)$ mindestens m Werte enthält, die mit den $f(y_i)$ übereinstimmen, können sich also höchstens $n-m$ Stellen von $(f_1, \ldots, f_n)$ unterscheiden. Damit gilt als obere Schranke für die Anzahl der Unterschiede:

$$n - m < n - n/2 = n/2 < m.$$

(Die beiden Ungleichungen gelten jeweils wegen $m > n/2$.) Also gäbe es weniger als m sich unterscheidende Positionen, Widerspruch!

(2) Seien $x_1, \ldots, x_{n-1} \in \Sigma^*$ fest gewählt. Seien $y_1, \ldots, y_k \in \Sigma^*$ Wörter mit $X(x_1, \ldots, x_{n-1}, y_i)[n] \neq f(y_i)$ für alle $1 \leq i \leq k$. Dann erfüllt jede Auswahl $y_{i_1}, \ldots, y_{i_n}$ von n der k Wörter y_i die Prämisse aus Teil (1) des Lemmas: die n-ten Antworten auf $(x_1, \ldots, x_{n-1}, y_{i_\ell})$ sind alle falsch, der Ergebnisvektor auf Eingabe $(y_{i_1}, \ldots, y_{i_n})$ enthält aber mindestens m richtige Antworten — es gibt also mindestens m Unterschiede.

$\square$

Beweis (von Satz 2.1, kombinatorische Version): Sei $m > n/2$, $f : \Sigma^* \to \Sigma^*$ eine (m, n)-berechenbare Funktion und M eine Turingmaschine für f. Für fest gewählte $x_1, \ldots, x_{n-1} \in \Sigma^*$ sei $M(y)$ mit $y \in \Sigma^*$ und $y \neq x_i$ die Ausgabe von M auf Eingabe $(x_1, \ldots, x_{n-1}, y)$. Auf welche Wörter $x_1, \ldots, x_{n-1}$ sich $M(y)$ bezieht, wird jeweils aus dem Kontext heraus klar sein.

Wenn $x_1, \ldots, x_{n-1} \in \Sigma^*$ existieren, für die $M(y)[n] = f(y)$ für alle bis auf endlich viele $y \in \Sigma^*$ gilt, dann kann der Funktionswert $f(y)$ bis auf endlich viele Ausnahmen durch die Projektion von $M(x_1, \ldots, x_{n-1}, y)$ auf die n-te Komponente berechnet werden. In diesem Fall ist f also berechenbar.

Andernfalls existieren für alle $x_1, \ldots, x_{n-1} \in \Sigma^*$ unendlich viele $y \in \Sigma^*$ mit $M(y)[n] \neq f(y)$. Wir werden einen Algorithmus angeben, der auf $n-1$ Eingaben $x_1, \ldots, x_{n-1}$ mindestens m richtige Antworten liefert. Die Aussage ist damit per Induktion bewiesen.

Sei $Y = \{y_1, \ldots, y_k\} \subseteq \Sigma^*$, $k \geq 0$. Wir sagen, dass die Menge Y die Bedingung (B1) bzw. $(\mathrm{B2})_j$ erfüllt, falls gilt:

(B1) Für alle $1 \leq i_1 < \ldots < i_n \leq k$ gilt: die Vektoren $M(y_{i_1}, \ldots, y_{i_n})$ und $(M(y_{i_1})[n], \ldots, M(y_{i_n})[n])$ unterscheiden sich an mindestens m Stellen.

$(\mathrm{B2})_j$ $(1 \leq j \leq n-1)$

$$\#\{M(y_i)[j] \mid 1 \leq i \leq k\} \in \{1, k\}$$

Bedingung $(\mathrm{B2})_j$ besagt, dass die j-te Komponente der $M(y_i)$ nur einen einzigen Wert oder ausschließlich verschiedene Werte enthält.

Wir geben jetzt den $(m, n-1)$-Algorithmus für die Funktion f an, wofür wir noch zwei weitere Definitionen benötigen. Wir definieren eine Folge von Konstanten $(K_i)_{i=1,\ldots,n}$ durch $K_1 = 2n-1$ und $K_{i+1} = (K_i - 1)^2 + 1$ für $1 \leq i < n$. Dies ergibt in geschlossener Form

$$K_i := (2n-2)^{2^{i-1}} + 1 \qquad (1 \leq i \leq n).$$

Mit W_i bezeichnen wir die Menge der ersten i Wörter von $\Sigma^* \setminus \{x_1, \ldots, x_{n-1}\}$ in längenlexikographischer Aufzählung.

Eingabe: $(x_1, \ldots, x_{n-1}) \in (\Sigma^*)^{n-1}$

$i := K_n$
while $\nexists Y \subseteq W_i : |Y| = K_n$ und Y erfüllt Bed. (B1) **do**
  $i := i + 1$
end

Sei $Y_n \subseteq W_i$ mit $|Y_n| = K_n$ und Y_n erfüllt Bed. (B1)

for $j := n - 1$ **downto** 1 **do**
  Wähle eine Teilmenge $Y_j \subseteq Y_{j+1}$ mit $|Y_j| = K_j$ und Y_j erfüllt Bed. $(\mathrm{B2})_j$
end

Ausgabe: $(f_1, \ldots, f_{n-1})$, wobei $(f_1, \ldots, f_n) = M(y)$ für ein $y \in Y_1$

Wir müssen begründen, dass der Algorithmus terminiert und dass er stets mindestens m richtige Antworten liefert.

Die erste while-Schleife terminiert, denn es gibt unendlich viele $y \in \Sigma^*$ mit $M(y)[n] \neq f(y)$. Nach Lemma 2.12 (2) gibt es stets K_n Wörter y_i, die die Prämisse aus Teil (1) des Lemmas erfüllen (und damit Bedingung (B1)).

Als nächstes müssen wir zeigen, dass es in Y_{j+1}, $1 \leq j \leq n-1$, stets eine K_j-elementige Teilmenge Y_j gibt, die die Bedingung $(\mathrm{B2})_j$ erfüllt. Dazu überlegen wir uns zunächst, dass unter d^2+1 Zahlen stets $d+1$ Zahlen existieren, die alle gleich oder alle verschieden sind (Schubfachschluss). In unserem Fall finden wir also unter den

$$K_{j+1} = (2n-2)^{2^j} + 1 = \left((2n-2)^{2^{j-1}}\right)^2 + 1$$

Elementen von Y_{j+1} stets eine Teilmenge der Größe

$$(2n-2)^{2^{j-1}} + 1 = K_j,$$

in der alle Werte der j-ten Komponente gleich oder alle Werte der j-ten Komponente verschieden sind. Also existiert die Folge $Y_n \supseteq Y_{n-1} \supseteq \ldots \supseteq Y_1$.

Außerdem gilt: erfüllt eine Menge $Y \subseteq \Sigma^*$ die Bedingung (B1) oder die Bedingung $(\mathrm{B2})_j$, dann auch jede ihrer Teilmengen. Also erfüllt die Menge Y_1 alle Bedingungen (B1) und $(\mathrm{B2})_1$ bis $(\mathrm{B2})_{n-1}$. Letzteres heißt, dass für alle $1 \leq j < n$ die Werte $M(y_i)[j]$ alle gleich oder alle verschieden sind.

Sei jetzt $y \in Y_1$ und $M(y) = (f_1, \ldots, f_n)$. Wir behaupten, dass $(f_1, \ldots, f_{n-1})$ bereits m richtige Antworten auf die Eingabe $(x_1, \ldots, x_{n-1})$ enthält. Die Menge Y_1 enthält $K_1 = 2n - 1$ Elemente. Es kann nicht sein, dass n dieser Vektoren eine richtige Antwort in der letzten Komponente enthalten, denn Y_1

erfüllt Bedingung (B1) und damit die Prämisse aus Lemma 2.12 (1), d. h. jede Auswahl von n Werten aus Y_1 ergibt mindestens eine falsche Antwort in der letzten Komponente.

Also gibt es n Vektoren $z_1, \ldots, z_n \in Y_1$, bei denen die letzte Komponente den falschen Wert liefert. Sei $Z = \{z_1, \ldots, z_n\}$. Für $j = 1, \ldots, n-1$ streichen wir jeweils maximal einen Vektor aus Z:

- Ist j eine Komponente mit gleichen Werten, streichen wir nichts.
- Ist j eine Komponente mit unterschiedlichen Werten, dann streichen wir den Vektor, an dem die j-te Antwort korrekt ist (evtl. gibt es einen solchen Vektor nicht).

Die derart reduzierte Menge Z nennen wir Z'. Es gilt $|Z'| \geq 1$ und für alle Komponenten $1 \leq j \leq n-1$, in denen in $M(y)$, $y \in Y_1$, unterschiedliche Werte stehen, sind die Antworten in Z' falsch. Da auch die Antworten $M(z_i)[n]$ falsch sind, müssen sich die m richtigen Antworten von $M(z)$, $z \in Z'$, also unter denjenigen Komponenten befinden, die nur gleiche Werte enthalten. Dann enthält aber jedes $M(y)$, $y \in Y_1$, bereits m richtige Antworten innerhalb der ersten $n-1$ Komponenten.

Nachdem wir uns von der Korrektheit des Algorithmus überzeugt haben, können wir diesen deutlich kompakter formulieren:

Eingabe: $(x_1, \ldots, x_{n-1}) \in (\Sigma^*)^{n-1}$

$i := 2n-1$
while $\nexists Y \subseteq W_i : |Y| = 2n-1$ und Y erfüllt Bed. (B1), $(B2)_1$ bis $(B2)_{n-1}$ **do**
  $i := i+1$
end
sei $Y \subseteq W_i$ mit $|Y| = 2n-1$ und Y erfüllt Bed. (B1), $(B2)_1$ bis $(B2)_{n-1}$

Ausgabe: $(f_1, \ldots, f_{n-1})$, wobei $(f_1, \ldots, f_n) = M(y)$ für ein $y \in Y$

□

Einer der Vorteile dieses Beweises ist die Übertragbarkeit des Ergebnisses auf endliche Automaten (siehe Satz 4.3). In Kapitel 4 werden wir uns ausführlicher mit regulären Häufigkeitsberechnungen beschäftigen.

2.3 Überabzählbar viele nicht häufigkeitsberechenbare Sprachen

Rose wusste bereits 1960, dass es überabzählbar viele *Funktionen* gibt, die für alle $n \geq 1$ nicht $(1, n)$-berechenbar sind. Sein Konferenzbeitrag ist leider nicht in den eigentlichen Proceedings erschienen, weshalb wir seinen Beweis nicht kennen und an dieser Stelle einen eigenen Beweis für sein Ergebnis geben wollen. Man beachte, dass ein reines Kardinalitätsargument für den Beweis dieser Aussage nicht ausreicht, denn die Klassen $\Omega(m, n)$ enthalten für alle $1 \leq m \leq n/2$ überabzählbar viele Sprachen (siehe Beispiel 4.1).

Wir verschärfen die Aussage von Rose allerdings in zweierlei Hinsicht: zum einen zeigen wir, dass sie bereits für häufigkeitsberechenbare *Sprachen* gilt; zum anderen lassen wir ein noch viel allgemeineres Turingmaschinenmodell zu, das wir in folgender Definition einführen.

Definition 2.13 ($(1, \infty)$-Entscheidung) *Sei M eine Turingmaschine, die als Eingabe eine* unendliche *Folge von paarweise verschiedenen Wörtern $w_1, w_2, \ldots \in \Sigma^*$ erhält und daraufhin eine* unendliche *Folge von Bits $b_1, b_2, \ldots \in \mathbb{B}$ ausgibt (M terminiert also nicht). Die Turingmaschine M realisiert eine* $(1, \infty)$-Entscheidung *einer Sprache $L \subseteq \Sigma^*$, wenn gilt:*

$$\forall w_1, w_2, \ldots \in \Sigma^* \; \exists i \geq 1 : M(w_1, w_2, \ldots)[i] = \chi_L(w_i).$$

Wir verlangen also lediglich *eine* richtige Antwort von M und geben keine Schranke für die Anzahl der Wörter vor, die M dabei untersucht.

Satz 2.14 (Rose [Ros60]) *Es gibt überabzählbar viele Sprachen $L \subseteq \Sigma^*$, die nicht $(1, \infty)$-entscheidbar sind.*

Beweis: Sei $f_1 : (\Sigma^*)^\omega \to \mathbb{B}^\omega, f_2 : (\Sigma^*)^\omega \to \mathbb{B}^\omega, \ldots$ eine Aufzählung aller totalen, berechenbaren Funktionen, die von Turingmaschinen nach dem Modell von Definition 2.13 realisiert werden. Sei $w_1, w_2, \ldots$ die längenlexikographische Aufzählung von Σ^*.

Zu gegebenem $\alpha \in \mathbb{B}^\omega$ konstruieren wir eine Sprache $L_\alpha \subseteq \Sigma^*$ derart, dass keine der Funktionen f_i eine $(1, \infty)$-Entscheidung von L_α ist. Die Wörter $w_1, w_2, \ldots$ verteilen wir dabei wie folgt als Eingaben für die Funktionen $f_1, f_2, \ldots$:

- w_1 ist die erste Eingabe für f_1;
- w_2 dient nicht als Eingabe für ein f_i;
- w_3 und w_4 sind die zweite Eingabe für f_1 und die erste Eingabe für f_2;
- w_5 dient nicht als Eingabe für ein f_i;
- w_6, w_7 und w_8 sind die dritte Eingabe für f_1, die zweite Eingabe für f_2 und die erste Eingabe für f_3;
- w_9 dient nicht als Eingabe für ein f_i;
- usw.

Formal definieren wir für $j \geq 1$ den Wert $N_j := \sum_{k=1}^{j-1}(k+1)$ und legen als Eingabe für die i-te Funktion ($i \geq 1$) die folgenden Wörter fest:

$$w_{N_i+i},\ w_{N_{i+1}+i},\ w_{N_{i+2}+i}, \ldots$$

Jedes w_i dient also höchstens einer Funktion f_j als Eingabe. Mit dieser Zuordnung können wir jetzt L_α definieren.

(1) Sei w_i eine Eingabe für die Funktion f_j, die mit $b \in B$ beantwortet wird; dann legen wir fest:

$$w_i \in L_\alpha \quad \Leftrightarrow \quad b = 0.$$

Das bedeutet, dass f_j die Eingabe w_i falsch beantwortet.

(2) Sei w_i das k-te Wort, das nicht als Eingabe für eine der Funktionen f_j dient; in diesem Fall legen wir fest:

$$w_i \in L_\alpha \quad \Leftrightarrow \quad \alpha[k] = 1.$$

Der Fall (1) garantiert, dass keine der Funktionen f_j eine $(1, \infty)$-Entscheidung der Sprache L_α realisieren kann. Der Fall (2) ermöglicht durch Variation von α, überabzählbar viele Sprachen zu definieren, die allesamt nicht häufigkeitsberechenbar (in unserem allgemeineren Sinne) sein können. □

2.4 Selektive Mengen

In Abschnitt 1.4 haben wir die sog. multi-selektiven Mengen eingeführt. Wir wollen jetzt zeigen, dass alle k-selektiven Mengen $(1, k{+}1)$-entscheidbar sind[2], die Umkehrung jedoch nicht gilt: es gibt sogar überabzählbar viele $(1, k+1)$-entscheidbare Sprachen, die nicht k-selektiv sind.

Satz 2.15 *Für jedes $k \geq 1$ gilt: $\{L \subseteq \Sigma^* \mid L$ ist k-selektiv $\} \subsetneqq \Omega(1, k+1)$.*

Wir beweisen den Satz über die folgenden beiden Lemmata.

Lemma 2.16 *Für jedes $k \geq 1$ und jede k-selektive Sprache $L \subseteq \Sigma^*$ gilt: $L \in \Omega(1, k+1)$.*

Beweis: Sei f eine Selektionsfunktion für L. Auf Eingabe $(x_1, \ldots, x_{k+1})$ berechnen wir $f(x_1, \ldots, x_{k+1}) = x_i$ und geben $b := (\underbrace{0, \ldots, 0}_{i-1 \text{ mal}}, 1, \underbrace{0, \ldots, 0}_{k+1-i \text{ mal}})$ aus. Angenommen, keine der Antworten wäre richtig, d. h. $\chi_L^{(n)}(x_1, \ldots, x_{k+1}) = \overline{b}$. Dann wäre $x_i \notin L$ und $x_j \in L$ für alle $j \in \{1, \ldots, k+1\} \setminus \{i\}$, d. h. es gehören k der $k+1$ Eingaben zu L, doch f hat die eine Eingabe ausgewählt, die nicht zu L gehört — Widerspruch! □

Als nächstes werden wir zeigen, dass nur abzählbar viele der in Definition 1.2 auf Seite 16 eingeführten Pfadsprachen multi-selektiv sein können[3]. Andererseits werden wir später in Beispiel 4.1 sehen, dass sich alle Pfadsprachen durch endliche Automaten $(1, 2)$-erkennen lassen.

Lemma 2.17 *Sei $\alpha \in \mathbb{B}^\omega$ und $k \geq 1$. Die Pfadsprache A_α ist genau dann k-selektiv, wenn das Positionsprädikat P_α entscheidbar ist.*

[2] Der Spezialfall $k = 1$ wird bereits von Kummer und Stephan erwähnt [KS95b].

[3] Selman zeigte bereits 1979, dass aus der P-Selektivität von A_α auch $A_\alpha \in \mathrm{P}$ folgt [Sel79].

Beweis: Die Richtung von rechts nach links ist offensichtlich. Für die Umkehrung sei f eine Funktion, die die k-Selektivität von A_α belegt, und $n \in \mathbb{N}$ eine Eingabe, für die $P_\alpha(n)$ entschieden werden soll. Angenommen, das Anfangsstück von α der Länge $i \geq 0$, $\alpha[1 \ldots i]$, sei bereits bekannt. Um $\alpha[i+1]$ zu bestimmen, definieren wir zunächst eine Menge M, die als Eingabe für f dienen wird:

$$M := \{\alpha[1 \ldots i]w \mid w \in \mathbb{B}^* \text{ und } 1 \leq |w| \leq k\}.$$

Diese Menge ist endlich, wir können also $f(M)$ berechnen. Die Menge M enthält alle Wörter $\alpha[1 \ldots i+1]$ bis $\alpha[1 \ldots i+k]$, es gilt also $|M \cap A_\alpha| = k$. Also muss $f(M)$ eines dieser k Wörter sein, und wir können $\alpha[i+1]$ (und evtl. sogar weitere nachfolgende Zeichen) aus der Antwort ablesen.

Nach spätestens n Schritten haben wir $\alpha[1 \ldots n]$ rekonstruiert und kennen den Wert von $P_\alpha(n)$. □

Es gibt also überabzählbar viele Sprachen, die nicht k-selektiv, aber $(1, k+1)$-entscheidbar sind (sogar durch endliche Automaten). Durch Diagonalisierung lässt sich leicht eine konkrete Sprache konstruieren:

Sei $f_1, f_2, \ldots$ eine Aufzählung aller Selektionsfunktionen für Sprachen über dem Alphabet Σ. Wir konstruieren ein unendliches Wort $\alpha \in \mathbb{B}^*$ so, dass die zugehörige Pfadsprache A_α von keiner der Funktionen f_i korrekt k-selektiert wird. Sei $\alpha[1 \ldots k \cdot i]$, $i \geq 0$, bereits konstruiert. Wir definieren eine Eingabemenge M durch

$$M := \{\alpha[1 \ldots k \cdot i]0^j \mid 1 \leq j \leq k\} \cup \{\alpha[1 \ldots k \cdot i]1^j \mid 1 \leq j \leq k\}.$$

und betrachten die Ausgabe der $(i+1)$-ten Selektionsfunktion f_{i+1} auf M. Ist $f_{i+1}(M) = \alpha[1 \ldots k{\cdot}i]0^j$ für ein $1 \leq j \leq k$, so legen wir $\alpha[k{\cdot}i{+}1 \ldots k{\cdot}(i{+}1)] := 1^k$ fest, andernfalls $\alpha[k \cdot i + 1 \ldots k \cdot (i+1)] := 0^k$. In beiden Fällen enthält die Menge M genau k Wörter aus A_α, doch f_{i+1} antwortet mit einem Wort, das nicht zu A_α gehört. Also ist f_{i+1} keine k-Selektionsfunktion für A_α. Die so konstruierte Zeichenkette α setzt sich aus 0- und 1-Blöcken der Länge k zusammen.

Kapitel 3

Separierbarkeit von Mengen

In diesem Kapitel beschäftigen wir uns mit der Separierbarkeit von Sprachpaaren. Zur Erinnerung: um zwei disjunkte Sprachen $A, B \subseteq \Sigma^*$ zu separieren, genügt es, die Wörter aus A zu akzeptieren und die aus B abzulehnen (vgl. Definition 1.1).

Offensichtlich gibt es Mengen A und B, die beide nicht entscheidbar, aber separierbar sind: als Beispiel können wir markierte Versionen des speziellen Halteproblems K betrachten. Mit

$$K := \{\langle M \rangle \mid M \text{ hält auf Eingabe } \langle M \rangle\}$$

sind die Mengen $A = \{0\} \cdot K$ und $B = \{1\} \cdot K$ beide unentscheidbar, können aber trivialerweise separiert werden.

Beispiel 3.1 (Quadrate von Pfad- und Anti-Pfadsprachen)
Ein weniger offensichtliches Beispiel stammt von Tantau [Tan02b]: Für jedes $\alpha \in \mathbb{B}^\omega$ sind die Sprachen $A := (A_\alpha)^2$ und $B := (B_\alpha)^2 \setminus \{\langle w, w \rangle \mid w \in \mathbb{B}^*\}$ separierbar (A_α und B_α wurden in Definition 1.2 eingeführt). Auf Eingabe (x, y), $x, y \in \mathbb{B}^*$, stellen wir zunächst fest, ob die beiden Wörter bzgl. der Präfixeigenschaft vergleichbar sind. Ist dies der Fall, geben wir 1 aus, andernfalls 0.

Dies separiert A und B, denn bei $x \preceq y$ oder $y \preceq x$ kann es nicht sein, dass sowohl x als auch y zu B_α gehören, da Wörter aus B_α bzgl. der Präfixeigenschaft unvergleichbar sind — die Ausgabe 1 kann also nicht falsch sein. Im anderen Fall ist die Ausgabe 0 richtig, denn für zwei Wörter aus A_α muss eines Präfix des anderen sein.

Später werden wir sehen, dass die Sprachen A_α und B_α in der Regel nicht separierbar sind (Satz 3.4). ◇

Kinber untersuchte 1976 als erster separierbare Mengen im Zusammenhang mit Häufigkeitsberechnungen und verglich Trakhtenbrots Resultat mit den entsprechenden separierbaren Pendants (m, n)-separierbar (durch Turingmaschinen) und (m, n)-fa-separierbar (durch endliche Automaten). Er konnte zeigen, dass es (m, n)-separierbare Sprachpaare $A, B \subseteq \Sigma^*$ mit $m > n/2$ gibt, die nicht rekursiv separierbar sind [Kin76]. Also überträgt sich Trakhtenbrots Resultat nicht auf separierbare Mengen. Im nächsten Abschnitt werden wir ausführlich auf die Frage eingehen, ob zumindest für endliche Automaten ein ähnliches Ergebnis gelten könnte.

3.1 Kinbers Vermutung

Im Jahre 1976 veröffentlichte Kinber die Behauptung, dass das Analogon zu Trakhtenbrots Resultat für (m, n)-fa-separierbare Sprachpaare gilt: die Sprachen seien bereits durch endlichen Automaten separierbar, wenn $m > n/2$ gilt [Kin76]. Diese Behauptung erwies sich als falsch: Tantau konnte ein Sprachpaar finden, das durch einen endlichen Automaten $(3, 5)$-separiert, jedoch noch nicht einmal durch Turingmaschinen separiert werden kann [Tan02b]. Dieses spezielle Beispiel werden wir nicht wiedergeben, jedoch in Abschnitt 3.2 eine Verallgemeinerung davon untersuchen; diese wird zeigen, dass es nicht Turing-separierbare Sprachpaare gibt, die von einem n-DFA mit höchstens logarithmisch vielen Fehlern separiert werden können.

Kinber folgerte aus seinem „Theorem“ ein Trakhtenbrot-ähnliches Resultat für (m, n)-erkennbare Sprachen — nach Tantaus Gegenbeispiel stellte sich die Frage nach der Korrektheit dieser Behauptung also von neuem. In Abschnitt 4.1 werden wir zeigen, dass diese Aussage tatsächlich stimmt.

Im folgenden werden wir Kinbers „Beweis“ wiederholen, dabei die fehlerhafte Stelle herausarbeiten und ein Gegenbeispiel für die betreffende Teilaussage angeben.

Fehlerhafte Behauptung 3.2 ([Kin76, Theorem 3]) *Seien $A, B \subseteq \mathbb{B}^*$ disjunkte Mengen. Wenn A und B (m, n)-fa-separierbar sind und $m > n/2$ ist, dann sind A und B fa-separierbar.*

Kinbers Ansatz: Seien $m > n/2$ und $A, B \subseteq \mathbb{B}^*$ disjunkte Mengen, die (m,n)-fa-separierbar sind. Sei $\sigma : \mathbb{B}^* \to \mathbb{B}$ eine partiell definierte Funktion mit

$$\sigma(x) = \begin{cases} 1 & ; x \in A \\ 0 & ; x \in B \\ \text{undef} & ; \text{sonst} \end{cases}$$

Für $\alpha_i, \beta_j \in \mathbb{B}$ definieren wir

$$\alpha_1 \dots \alpha_k / \beta_1 \dots \beta_k := \{x \in \mathbb{B}^* \mid \sigma(x\alpha_1 \dots \alpha_i) = \beta_i \ \forall 1 \leq i \leq k\}.$$

Für $k = 0$ erhalten wir als Spezialfall $\varepsilon/\varepsilon = \mathbb{B}^*$.

Wir konstruieren einen Baum S wie folgt: die Wurzel (Ebene 0) wird mit $\varepsilon/\varepsilon = \mathbb{B}^*$ beschriftet.

Sei v ein Knoten in S auf Ebene k, der mit $\alpha_1 \dots \alpha_k/\beta_1 \dots \beta_k$ beschriftet ist. Für $i, j \in \mathbb{B}$ sei $N_{ij} := \alpha_1 \dots \alpha_k i / \beta_1 \dots \beta_k j$. Dann erhält v vier, zwei oder keinen Nachfolger nach folgender Regel: für $i \in \mathbb{B}$ erhält v Nachfolger mit Beschriftung N_{i0} und N_{i1}, wenn beide Mengen N_{i0} und N_{i1} unendlich sind.

Behauptung: Der so konstruierte Baum S ist endlich.

Beweis: Angenommen, S sei unendlich, d. h. nach dem Lemma von König (Lemma 2.10) gibt es einen unendlichen Pfad α_1/β_1, $\alpha_1\alpha_2/\beta_1\beta_2, \dots$ in S. Die Mengen $\alpha_1 \dots \alpha_k/\beta_1 \dots \beta_{k-1}\overline{\beta_k}$ sind stets unendlich; wir können also jeweils n verschiedene Wörter $x_1^k, \dots, x_n^k$ aus $\alpha_1 \dots \alpha_k/\beta_1 \dots \beta_{k-1}\overline{\beta_k}$ wählen und erhalten damit eine Folge $(x_1^1, \dots, x_n^1)$, $(x_1^2, \dots, x_n^2)$, $\dots$ über $(\mathbb{B}^*)^n$.

Sei M ein n-DFA mit Startzustand q_0, der A und B (m,n)-fa-separiert. Wir werden zeigen, dass M für alle $1 \leq i < j$ bei den Eingaben $(x_1^i\alpha_1 \cdots \alpha_i, \dots, x_n^i\alpha_1 \cdots \alpha_i)$ und $(x_1^j\alpha_1 \cdots \alpha_i, \dots, x_n^j\alpha_1 \cdots \alpha_i)$ in unterschiedliche Zustände gelangen muss — ein offensichtlicher Widerspruch zur Endlichkeit der Zustandsmenge von M.

Seien also $1 \leq i < j$ und

$$\begin{aligned} (y_1, \dots, y_n) &:= M(x_1^i\alpha_1 \cdots \alpha_i, \dots, x_n^i\alpha_1 \cdots \alpha_i) \text{ und} \\ (z_1, \dots, z_n) &:= M(x_1^j\alpha_1 \cdots \alpha_i, \dots, x_n^j\alpha_1 \cdots \alpha_i) \end{aligned}$$

die Ausgaben von M auf den betrachteten Eingaben.

Nach Wahl der x_ℓ^i gilt $\sigma(x_\ell^i\alpha_1 \dots \alpha_i) = \overline{\beta_i}$ für alle $1 \leq \ell \leq n$. Da M eine (m,n)-fa-Separierung von A und B ist, müssen mindestens $m > n/2$ der y_ℓ

mit $\overline{\beta_i}$ übereinstimmen. Nach Wahl der x_ℓ^j gilt ebenso $\sigma(x_\ell^j \alpha_1 \dots \alpha_i) = \beta_i$ für alle $1 \leq \ell \leq n$, es stimmen also mindestens $m > n/2$ der z_ℓ mit β_i überein.

Dies ist ein Widerspruch, der Baum S muss also endlich sein. □

Sei $\hat{s}$ die maximale Länge eines Pfades in S. Wir vervollständigen S, indem wir folgende Regeln anwenden, bis sich keine Änderung mehr ergibt. Sei $k \leq \hat{s}$ und v ein Knoten, der mit $\alpha_1 \dots \alpha_k / \beta_1 \dots \beta_k$ beschriftet ist.

- Wenn $N = \alpha_1 \dots \alpha_{k+1} / \beta_1 \dots \beta_{k+1}$ unendlich ist, dann erhält v einen mit N beschrifteten Nachfolger.
- Wenn $N_j = \alpha_1 \dots \alpha_k \alpha_{k+1} / \beta_1 \dots \beta_k j$, $j = 0, 1$, beide endlich sind, dann erhält v einen mit N_0 beschrifteten Nachfolger.

Ist $\alpha_1 \dots \alpha_k / \beta_1 \dots \beta_k$ $(1 \leq k \leq \hat{s} + 1)$ nach vollständiger Konstruktion des Baumes S keinem Knoten als Beschriftung zugewiesen, dann muss diese Menge endlich sein. Sei D die Vereinigung dieser Mengen; da S endlich ist, ist offensichtlich auch D eine endliche Menge.

Jetzt haben alle Pfade in S die Länge $\hat{s} + 1$. *Kinber behauptet nun folgendes: ist eines der Blätter mit $\alpha_1 \dots \alpha_{\hat{s}+1} / \beta_1 \dots \beta_{\hat{s}} j$ beschriftet, dann ist $\alpha_1 \dots \alpha_{\hat{s}+1} / \beta_1 \dots \beta_{\hat{s}} \overline{j}$ keinem Knoten als Beschriftung zugewiesen.* Diese Behauptung stimmt nicht, wie wir an nachfolgendem Beispiel 3.3 sehen werden.

Auf die Wiedergabe des abschließenden Beweisschrittes verzichten wir, da er auf dieser falschen Aussage beruht. □

Beispiel 3.3 Wir geben ein Gegenbeispiel zu Kinbers Behauptung an, dass im Baum S nie gleichzeitig zwei mit $\alpha_1 \dots \alpha_{\hat{s}+1} / \beta_1 \dots \beta_{\hat{s}+1}$ und $\alpha_1 \dots \alpha_{\hat{s}+1} / \beta_1 \dots \beta_{\hat{s}} \overline{\beta_{\hat{s}+1}}$ beschriftete Blätter existieren können.

Seien dazu $A = \{1^{2k}0 \mid k \geq 0\}$ und $B = \{1^n \mid n \geq 1\} \cup \{1^{2k+1}0 \mid k \geq 0\}$ zwei disjunkte und offensichtlich fa-separierbare Mengen. Die ersten drei Ebenen des vollständigen Baumes $\alpha_1 \dots \alpha_k / \beta_1 \dots \beta_k$ $(k = 0, 1, 2)$ sind in Abbildung 3.1 dargestellt.

Der Baum S besteht damit nach der ersten Stufe aus lediglich drei Knoten:

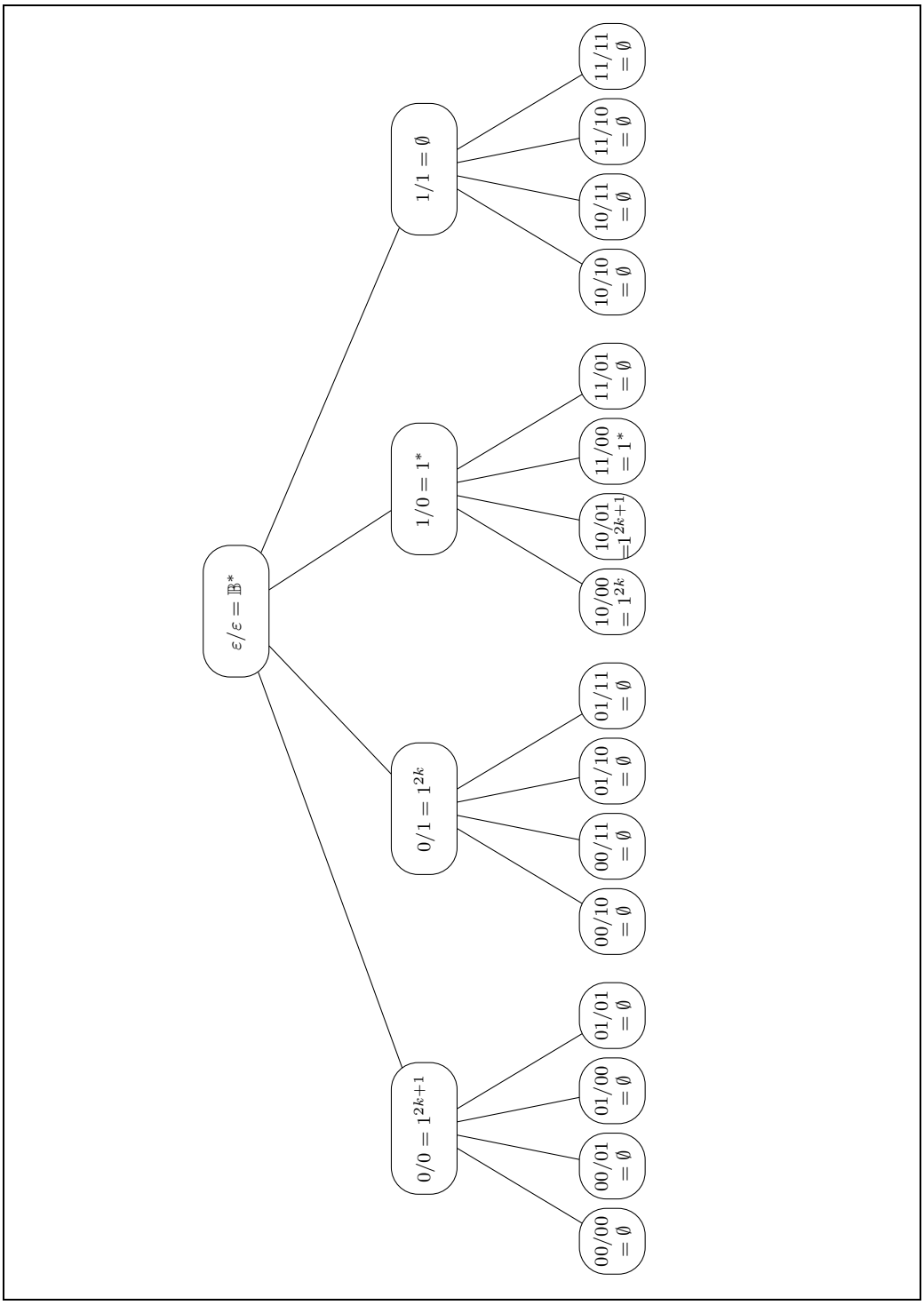
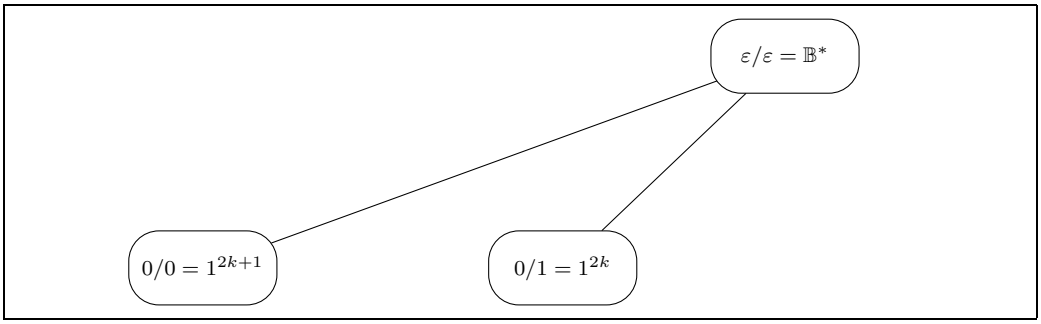
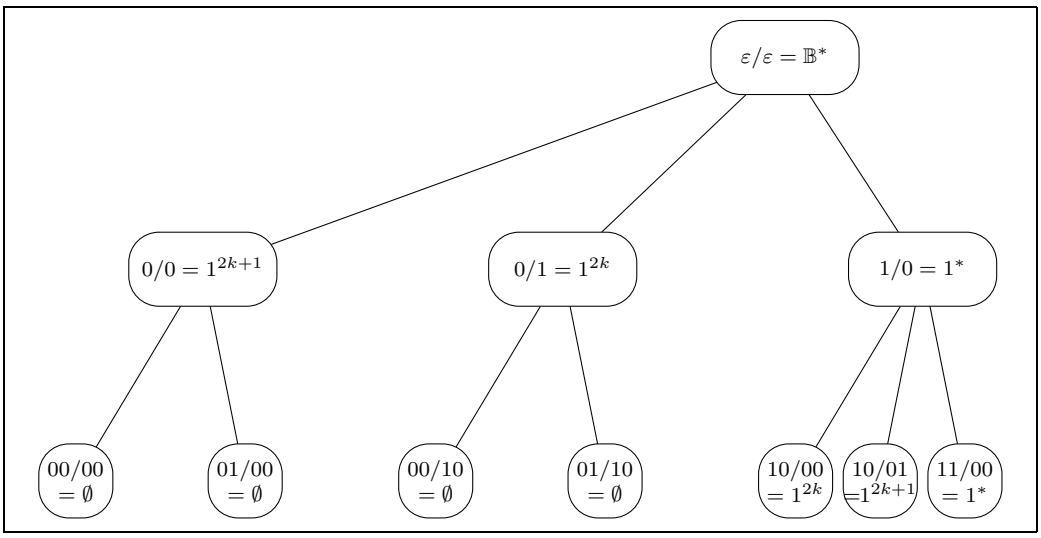


Abbildung 3.1: Die ersten drei Ebenen des vollständigen Baumes $\alpha_1 \dots \alpha_k / \beta_1 \dots \beta_k$ $(k = 0, 1, 2)$.



Die maximale Pfadlänge in S ist $\hat{s} = 1$. Werden jetzt in der zweiten Stufe wieder die Knoten ergänzt, die unendlichen Mengen entsprechen (und jeweils eine endliche Menge, wenn beide Nachfolgemengen endlich sind), erhalten wir:



Der endgültige Baum S enthält aber sowohl die Blätter 10/00 als auch 10/01, da beide unendlichen Mengen entsprechen. Dies widerspricht der Behauptung in Kinbers Beweis. ◇

3.2 Separierbarkeit von Pfadsprachen

In diesem Abschnitt betrachten wir die Separierbarkeit von Sprachpaaren, die aus einer Pfadsprache und der zugehörigen Anti-Pfadsprache bestehen. Zur Erinnerung: zu einem $\alpha \in \mathbb{B}^\omega$ ist die Pfadsprache A_α die Menge der

endlichen, nicht-leeren Präfixe von α und die Anti-Pfadsprache B_α die Menge der Wörter aus A_α, bei denen das jeweils letzte Bit negiert wurde (siehe Definition 1.2).

Wir können einen direkten Zusammenhang zwischen der Separierbarkeit der Mengen A_α und B_α und der Entscheidbarkeit des Positionsprädikats P_α herstellen.

Satz 3.4 *Sei $\alpha \in \mathbb{B}^\omega$. Dann sind A_α und B_α genau dann separierbar, wenn P_α entscheidbar ist.*

Beweis: Wenn P_α entscheidbar ist, kann ein endliches Anfangsstück von α Zeichen für Zeichen rekonstruiert werden. Somit sind A_α und B_α beide entscheidbar und damit separierbar.

Seien anderseits A_α und B_α separierbar durch eine Turing-Maschine M. Um festzustellen, ob $P_\alpha(n)$ gilt, werden wir den Präfix von α der Länge n rekonstruieren.

Angenommen, wir kennen das Anfangsstück w_i von α der Länge $i \geq 0$. Dann können wir das $(i+1)$-te Zeichen wie folgt ermitteln: Wir stellen der Turingmaschine M die Frage w_i0. Da genau eines der Wörter w_i0 und w_i1 zu A_α gehört, das jeweils andere zu B_α, ist das $(i+1)$-te Zeichen von α genau dann 0, wenn M das Wort w_i0 akzeptiert.

Nach der Rekonstruktion des n-ten Zeichens von α kennen wir somit den Wert von P_α an der Stelle n. □

Im nächsten Kapitel werden wir in Beispiel 4.1 sehen, dass die Sprachen A_α für alle $\alpha \in \mathbb{B}^\omega$ $(1,2)$-erkennbar sind. Mit dem folgenden Satz können wir schließen, dass A_α und B_α dann auch $(1,2)$-fa-separierbar sind.

Satz 3.5 *Sei $L \subseteq \Sigma^*$ eine $(1,2)$-erkennbare Sprache. Dann gilt für alle $A \subseteq L$ und $B \subseteq \Sigma^* \setminus L$, dass A und B $(1,2)$-fa-separierbar sind.*

Beweis: Aus $L \in \mathrm{REG}(1,2)$ folgt sofort, dass die Sprachen L und $\Sigma^* \setminus L$ $(1,2)$-fa-separierbar sind. Betrachtet man nun Teilmengen A von L und B von $\Sigma^* \setminus L$, so verkleinern wir lediglich die Menge der Wörter, auf denen wir eine definitive Antwort geben müssen. □

Korollar 3.6 *Es gibt überabzählbar viele Sprachpaare $A, B \subseteq \Sigma^*$, die $(1,2)$-separierbar, aber nicht rekursiv separierbar sind.*

Beweis: Aus Satz 3.4 folgt, dass es nur abzählbar viele $\alpha \in \mathbb{B}^\omega$ gibt, für die die Sprachen A_α und B_α rekursiv separierbar sind.

Satz 3.5 zeigt uns hingegen (zusammen mit Beispiel 4.1), dass die Sprachen A_α und B_α für alle $\alpha \in \mathbb{B}^\omega$ $(1,2)$-separierbar sind. □

Wenn wir bei drei oder mehr Eingaben einen Fehler erlauben, können wir nach wie vor nur die Pfad- und Anti-Pfadsprachen separieren, die wir im herkömmlichen Sinne separieren können. Diese Beobachtung trifft sowohl auf separierbare als auch auf fa-separierbare Sprachpaare zu. Diese Tatsache ist bemerkenswert, denn für fünf Eingaben erlaubt bereits das Zulassen von zwei Fehlern, dass *alle* A_α, B_α durch endliche Automaten separiert werden können [Tan02b].

Lemma 3.7 *Für alle $n \geq 3$ sind die Sprachen A_α und B_α genau dann $(n-1,n)$-separierbar, wenn sie entscheidbar sind.*

Beweis: Es genügt, die Aussage nur für $n = 3$ und nur die Richtung von links nach rechts zu zeigen.

Sei $x \in \mathbb{B}^*$ ein bereits bekanntes Anfangsstück von α. Wir ermitteln das nächste Bit durch die folgenden beiden Anfragen an den $(2,3)$-Algorithmus: $(x0, x1, x00)$ und $(x0, x1, x01)$. Die möglichen Antworten sind:

Frage 1

	x0	**x1**	**x00**
mögliche Antworten	0 ✓	1 ✓	∗ ?
	1 ✓	0 ✓	∗ ?
	0 ?	0 ?	∗ ✓
	1 ?	1 ?	∗ ✓

Frage 2

	x0	**x1**	**x01**
mögliche Antworten	0 ✓	1 ✓	∗ ?
	1 ✓	0 ✓	∗ ?
	0 ?	0 ?	∗ ✓
	1 ?	1 ?	∗ ✓

Dabei bedeutet $*$ eine beliebige (nicht näher spezifizierte) Antwort und ✓ bzw. ? hinter der Antwort besagt, ob die Antwort mit Sicherheit richtig ist oder nicht.

Ist die Antwort auf eine der beiden Fragen $(0, 1, *)$ oder $(1, 0, *)$, so sind die ersten beiden Antworten richtig, da der Algorithmus maximal einen Fehler

macht und genau eines der Wörter $x0$ und $x1$ zu A_α, das jeweils andere zu B_α gehört.

Ist die Antwort auf beide Fragen $(0, 0, *)$ oder $(1, 1, *)$, so ist jeweils eine der ersten beiden Antworten falsch und die jeweils dritte Antwort muss richtig sein. Sei b_0 die Antwort auf $x00$ und b_1 die Antwort auf $x01$. Gilt $(b_0, b_1) = (0, 0)$, dann gilt $x00 \not\preceq \alpha$ und $x01 \not\preceq \alpha$, d. h. es muss $x1 \preceq \alpha$ gelten; für $(b_0, b_1) = (1, 0)$ bzw. $(0, 1)$ gilt $x00 \preceq \alpha$ bzw. $x01 \preceq \alpha$, womit wir sogar zwei Bits auf einmal definieren können; $(b_0, b_1) = (1, 1)$ ist nicht möglich, da die beiden dritten Antworten richtig sind und dies bedeuten würde, dass sowohl $x00$ als auch $x01$ Präfixe von α sind. □

Lemma 3.8 *Für alle $n \geq 3$ sind die Sprachen A_α und B_α genau dann $(n-1, n)$-fa-separierbar, wenn sie regulär sind.*

Beweis: Wiederum beschränken wir uns auf den Fall $n = 3$ und die Richtung von links nach rechts.

Wir konstruieren einen endlichen Automaten M, der wie im vorigen Beweis versuchen wird, ein Anfangsstück von α zu rekonstruieren. Der Automat kann dieses Anfangsstück nicht komplett speichern, muss dies aber auch nicht, da er parallel zum Lesen seiner Eingabe das Wort α rekonstruieren und den 3-DFA auf Eingabe (α, α, α) simulieren kann.

Sei Q die Zustandsmenge des 3-DFA, der A_α und B_α $(2, 3)$-separiert, und $w = a_1 \cdots a_n$ eine Eingabe für M. Zustände des neuen Automaten sind Paare der Form

$$(s \in \{0, 1, 2\},\ q \in Q)$$

mit folgender Bedeutung: wenn der neue Automat $i \geq 0$ Zeichen der Eingabe gelesen hat, dann ist q der Zustand, den der 3-DFA nach Lesen von $(\alpha[1 \ldots i], \alpha[1 \ldots i], \alpha[1 \ldots i])$ erreicht und

$$s = \begin{cases} 0, & \text{falls } a_1 \cdots a_i \preceq \alpha \\ 1, & \text{falls } a_1 \cdots a_{i-1}\overline{a_i} \preceq \alpha \\ 2, & \text{sonst} \end{cases}$$

Startzustand ist $(0, q_0)$, wobei $q_0 \in Q$ der Startzustand des 3-DFA ist. Die im Zustandspaar gespeicherte Information kann beim Lesen des $(i+1)$-ten Zeichens a_{i+1} aktualisiert werden: der Nachfolgezustand ist (s', q') mit

- $s' = \begin{cases} 2, & \text{falls } s = 1 \text{ oder } s = 2 \\ 1, & \text{falls } s = 0 \text{ und } a_{i+1} = \overline{\alpha[i+1]} \\ 0, & \text{falls } s = 0 \text{ und } a_{i+1} = \alpha[i+1] \end{cases}$ und
- $q' = q \cdot (\alpha[i+1], \alpha[i+1], \alpha[i+1])$.

Dabei hängt $\alpha[i+1]$ lediglich vom Zustand q ab (und kann deshalb fest in die Übergangsfunktion des neuen Automaten M eingebaut werden): nach der Argumentation im vorangegangenen Beweis müssen dazu lediglich die Antworten in den Zuständen $q \cdot (0, 1, 00)$ und $q \cdot (0, 1, 01)$ betrachtet werden.

Nach Lesen der Eingabe gibt der Wert s an, ob diese zu A_α ($s = 0$), zu B_α ($s = 1$) oder zu keiner der beiden Mengen ($s = 2$) gehört. □

Tantau zeigte, dass es ein $\alpha \in \mathbb{B}^\omega$ gibt, für das das zugehörige Pfad-/Anti-Pfadsprachpaar A_α, B_α $(3, 5)$-fa-separierbar, aber nicht rekursiv separierbar ist [Tan02b]. Desweiteren fragte er sich, für welche m und n es (m, n)-fa-separierbare Sprachpaare gibt, die nicht fa-separierbar sind [Tan02b, Abschnitt 5]. Eine Teillösung liefert der folgende Satz, der zeigt, dass es für wachsendes n Sprachpaare gibt, bei denen die Anzahl der Fehler bei der parallelen Separierung von n Eingaben nur logarithmisch mit n wächst.

Satz 3.9 *Für alle $n \geq 2$ und alle $\alpha \in \mathbb{B}^\omega$ sind die Sprachen A_α und B_α (m, n)-fa-separierbar, wobei $m = \max\{n - \lfloor 2 \log(n+1) \rfloor, 1\}$ gilt.*

Beweis: Wir geben einen Algorithmus an, der A_α und B_α mit der geforderten Maximal-Fehlerzahl separiert. Über die Realisierbarkeit des Algorithmus durch einen endlichen Automaten machen wir uns erst am Ende Gedanken.

Sei $(x_1, \ldots, x_n)$, $x_i \in \mathbb{B}^+$, eine Eingabe für unseren Algorithmus. Wir definieren $y_i \in \mathbb{B}^*$ $(1 \leq i \leq n)$ durch $x_i = y_i b_i$, $b_i \in \mathbb{B}$. Damit gilt für $x_i, x_j \in A_\alpha \cup B_\alpha$ stets $y_i \preceq y_j$ oder $y_j \preceq y_i$.

Für $n \geq 6$ ist $n - \lfloor 2 \log(n+1) \rfloor \geq 1$. Unser Algorithmus wird für diese n maximal $\lfloor 2 \log(n+1) \rfloor$ viele Fehler machen.

Für $2 \leq n \leq 5$ hingegen benötigen wir eine Sonderbehandlung. Es muss lediglich eine richtige Antwort gegeben werden. Wir unterscheiden drei Fälle:

- $\exists 1 \leq i < j \leq n : y_i \not\preceq y_j$ und $y_j \not\preceq y_i$: Ausgabe $(\ldots, \underset{\uparrow}{0}_{i}, \ldots, \underset{\uparrow}{0}_{j}, \ldots)$.

  Die Antwort enthält maximal einen Fehler, da es nicht sein kann, dass sowohl x_i als auch x_j zu A_α gehören.

- $\exists 1 \leq i, j \leq n$, $i \neq j$: $y_i \preceq y_j$, aber $y_i \neq y_j$. Falls $x_i \not\preceq y_j$ ist, geben wir $(\ldots, \underset{\uparrow}{0}_{i}, \ldots, \underset{\uparrow}{0}_{j}, \ldots)$ aus. Andernfalls ($x_i \preceq y_j$) ist die Ausgabe $(\ldots, \underset{\uparrow}{1}_{i}, \ldots, \underset{\uparrow}{0}_{j}, \ldots)$.

  Die Antwort im ersten Fall enthält maximal einen Fehler, denn $x_i \not\preceq x_j$ und somit kann nicht gleichzeitig $x_i \in A_\alpha$ und $x_j \in A_\alpha$ sein. Die Antwort im zweiten Fall enthält ebenso maximal einen Fehler, denn für $x_i \preceq x_j$ kann es nicht sein, dass $x_i \notin A_\alpha$ und $x_j \in A_\alpha$ ist.

- Es verbleibt der Fall $y_i = y_j$ für alle $1 \leq i < j \leq n$, also $n = 2$ und $y_1 = y_2$. Wir geben $(0, 0)$ und damit offensichtlich mindestens eine korrekte Antwort aus.

Wir werden jetzt einen Algorithmus angeben, der für $n \geq 6$ höchstens $\lfloor 2 \log(n+1) \rfloor$ falsche Antworten gibt. Dazu benötigen wir noch eine weitere Definition: für alle $w \in \mathbb{B}^*$ sei $I_w := \{1 \leq i \leq n \mid w \preceq y_i\}$.

Jetzt können wir die Ausgabe auf x_i festlegen: sie ist genau dann 1, falls

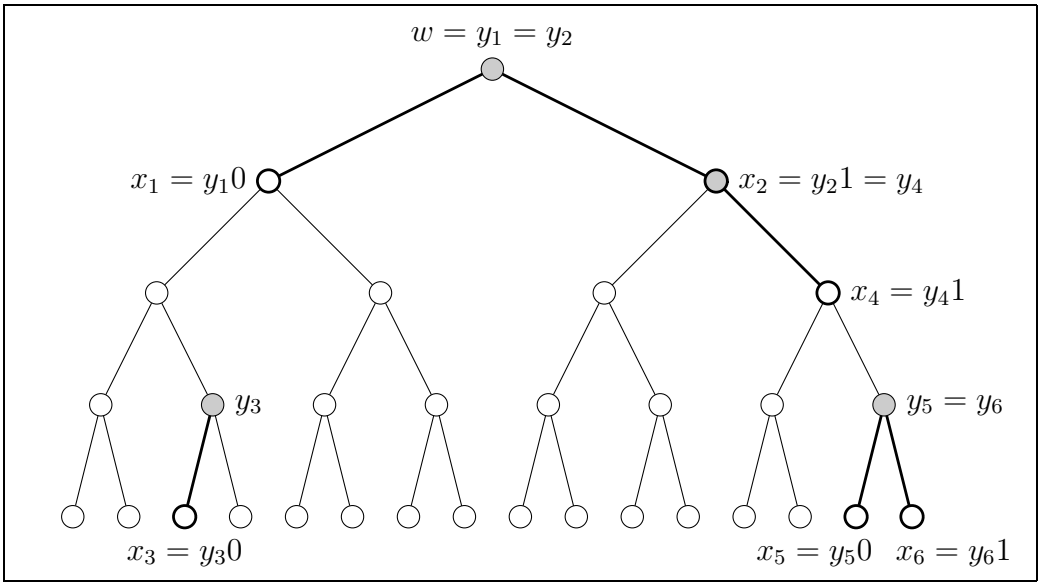
$$(x_i = y_i 0 \ \wedge \ |I_{y_i 0}| > |I_{y_i 1}|) \quad \vee \quad (x_i = y_i 1 \ \wedge \ |I_{y_i 0}| < |I_{y_i 1}|).$$

Dahinter steckt folgende Idee: Zu einer Eingabe x_i betrachten wir die Eingaben x_j, die sich im Binärbaum unterhalb von $y_i 0$ bzw. $y_i 1$ befinden. Wir vermuten, dass der Pfad α durch den Unterbaum gehen wird, der mehr x_j enthält, und definieren die Antwort auf x_i dementsprechend. Liegen wir mit der Vermutung falsch, beantworten wir zwar auch x_i falsch, können aber im größeren Unterbaum keinen einzigen Fehler mehr machen.

Behauptung: Sei $w = y_i$ für ein $1 \leq i \leq n$. Dann werden auf der Teileingabe $\{x_i \mid i \in I_w\}$ höchstens $\lfloor 2 \log(|I_w| + 1) \rfloor$ falsche Antworten gegeben.

Beweis: Sei $Y_w := \{y_i \mid i \in I_w\}$. Wir führen einen Induktionsbeweis über $|Y_w|$. Indizes i und j werden im folgenden *Zwillinge* genannt, falls $i \neq j$ und $y_i = y_j$ gilt. Es gilt stets $|Y_w| \leq |I_w|$ (echt kleiner gilt genau dann, wenn I_w Zwillinge enthält).

Das folgende Beispiel veranschaulicht die Definitionen. Hierbei ist $I_w = \{1, \ldots, 6\}$, $Y_w = \{y_1, y_3, y_4, y_5\}$, also $|I_w| = 6$ und $|Y_w| = 4$. Knoten, die Eingabewörtern entsprechen, sind dick umrandet; Knoten aus Y_w sind schattiert; die Verbindungskanten zwischen y_i und x_i sind fett gezeichnet. Die Indizes 1 und 2 sowie 5 und 6 sind Zwillinge.



Für $|Y_w| = 1$ gilt $|I_w| = 1$ oder $|I_w| = 2$, es können also höchstens $2 \leq \lfloor 2 \log(|I_w| + 1) \rfloor$ falsche Antworten gegeben werden.

Für $|Y_w| \geq 2$ setzt sich die Anzahl der Fehler aus drei Teilen zusammen:

- $f_0 :=$ Anzahl der Fehler im linken Teilbaum (ohne $x_i = w0$)
- $f_1 :=$ Anzahl der Fehler im rechten Teilbaum (ohne $x_j = w1$)
- $f_2 :=$ Anzahl der Fehler auf $x_i = w0$ und $x_j = w1$

Mit $f := f_0 + f_1 + f_2$ bezeichnen wir die Gesamtzahl der Fehler.

Da nicht gleichzeitig $w0 \preceq \alpha$ und $w1 \preceq \alpha$ gelten können, ist mindestens einer der Werte f_0 bzw. f_1 gleich 0. Gilt $f_0 = f_1 = 0$, so auch $f = f_2 \leq 2 \leq \lfloor 2 \log(|I_w| + 1) \rfloor$. Für $f_2 = 0$ ist die Aussage nach Induktionsvoraussetzung wahr (die Fehlerzahl hat sich nicht erhöht, trotz einer Erhöhung von $|Y_w|$).

Ohne Einschränkung bleibt der Fall $f_0 > 0$, $f_1 = 0$ und $f_2 > 0$ zu untersuchen. Wegen $f_0 > 0$ muss $w0 \preceq \alpha$ gelten. Folgende Fälle können auftreten:

(1) $w = y_i$ für ein $i \in I_w$ und i ist kein Zwilling ($\Rightarrow f_2 = 1$, denn $f_2 > 0$ und $f_2 = 2$ ist nicht möglich)

(1a) $x_i = w0 \;\Rightarrow\; |I_{w0}| \leq |I_{w1}|$

(1b) $x_i = w1 \;\Rightarrow\; |I_{w0}| < |I_{w1}|$

(2) $w = y_i = y_j$ für $i \neq j \in I_w$ (d. h. i und j sind Zwillinge); ohne Einschränkung sei $x_i = w0$ und $x_j = w1$

(2a) $f_2 = 1 \;\Rightarrow\; |I_{w0}| = |I_{w1}|$

(2b) $f_2 = 2 \;\Rightarrow\; |I_{w0}| < |I_{w1}|$

Wir können die vier Fälle zusammenfassen, indem wir $f_2 \leq 2$ abschätzen und die (Un-)Gleichungen zwischen $|I_{w0}|$ und $|I_{w1}|$ auf $|I_{w0}| \leq |I_{w1}|$ abschwächen.

$$\begin{aligned} f = f_0 + f_2 &\leq 2\log(|I_{w0}| + 1) + 2 && \text{(wegen Ind.-vorauss., } f_2 \leq 2\text{)} \\ &= 2\log(|I_{w0}| + 1) + 2\log 2 \\ &= 2\log(2|I_{w0}| + 2) \\ &\leq 2\log((|I_{w0}| + |I_{w1}| + 1) + 1) && \text{(wegen } |I_{w0}| \leq |I_{w1}|\text{)} \\ &\leq 2\log(|I_w| + 1) \end{aligned}$$

□

Damit ist die Behauptung bewiesen und wir können uns der Korrektheit des Satzes widmen. Sei dazu Y die Menge der minimalen Elemente von $\{y_1, \ldots, y_n\}$ bzgl. der Präfix-Ordnung. Wir partitionieren die Menge der Eingabewörter $\{x_1, \ldots, x_n\}$ in Mengen M_y, $y \in Y$, wie folgt: $x_i \in M_y$ genau dann, wenn $y \preceq x_i$. Jetzt gilt, dass höchstens in einer der Mengen M_y Antworten des Algorithmus auf darin enthaltene x_i falsch sein können. Sei $M_{\hat{y}}$ diese Menge. Dann ist die Anzahl der Fehler (nach obiger Behauptung) kleiner oder gleich $2\log(|M_{\hat{y}}| + 1) \leq 2\log(n + 1)$, die behauptete Fehlerzahl wird also eingehalten.

Schließlich müssen wir uns noch vergewissern, dass der beschriebene Algorithmus auch von einem endlichen Automaten durchgeführt werden kann. Um die Ausgabe für ein x_i festlegen zu können, müssen lediglich die Präfix-Beziehungen der x_j bzw. y_j bestimmt werden, was für einen endlichen Automaten kein Problem ist. Da es nur endlich viele dieser Beziehungen gibt, kann die anschließende Ausgabe fest verdrahtet werden. □

Satz 3.4 zeigt, dass es überabzählbar viele $\alpha \in \mathbb{B}^\omega$ gibt, für die das zugehörige Pfad-/Anti-Pfadsprachpaar nicht rekursiv separierbar ist. Daraus und aus dem eben bewiesenen Satz 3.9 folgt, dass es — entgegen Kinbers Vermutung — für separierbare Sprachen kein Analogon zu Trakhtenbrots Theorem für entscheidbare Sprachen geben kann.

Korollar 3.10 *Für alle Konstanten* $0 < q < 1$ *existieren* $m, n \in \mathbb{N}$ *mit* $m/n > q$ *und disjunkte Mengen* $A, B \subseteq \Sigma^*$*, die* (m,n)*-fa-separierbar, aber nicht rekursiv separierbar sind.*

Kapitel 4

Reguläre Häufigkeitsberechnungen

In diesem Kapitel beschäftigen wir uns mit der Häufigkeitsberechnung von Sprachen mittels endlicher Automaten. Im wesentlichen werden die folgenden drei Ergebnisse dargestellt:

1. Das Analogon der regulären Häufigkeitsberechnungen zu Trakhtenbrots zentralem Resultat für allgemeine Häufigkeitsberechnungen: die Klasse der (m, n)-erkennbaren Sprachen ist genau dann gleich der Klasse der regulären Sprachen, wenn $m > n/2$ gilt.

2. Ein Iterationskriterium, das uns eine strukturelle Eigenschaft regulärer Häufigkeitsberechnungen an die Hand gibt, mit der von vielen Sprachen nachgewiesen werden kann, dass sie nicht zur Klasse FREQ-REG gehören.

3. Abschlusseigenschaften (m, n)-erkennbarer Sprachen; u. a. ist die Klasse FREQ-REG eine boolesche Algebra. Die Vereinigung zweier $(1, n)$-erkennbarer Sprachen ist allerdings nicht in $\mathrm{REG}(1, n)$ enthalten.

Wir beginnen mit zwei generischen Beispielen $(1, 2)$-erkennbarer, nicht regulärer Sprachen.

Beispiel 4.1 (Pfadsprachen) Sei $\alpha \subseteq \mathbb{B}^\omega$ und A_α die zugehörige Pfadsprache. Dann ist A_α $(1, 2)$-erkennbar durch den in Abbildung 4.1 dargestellten

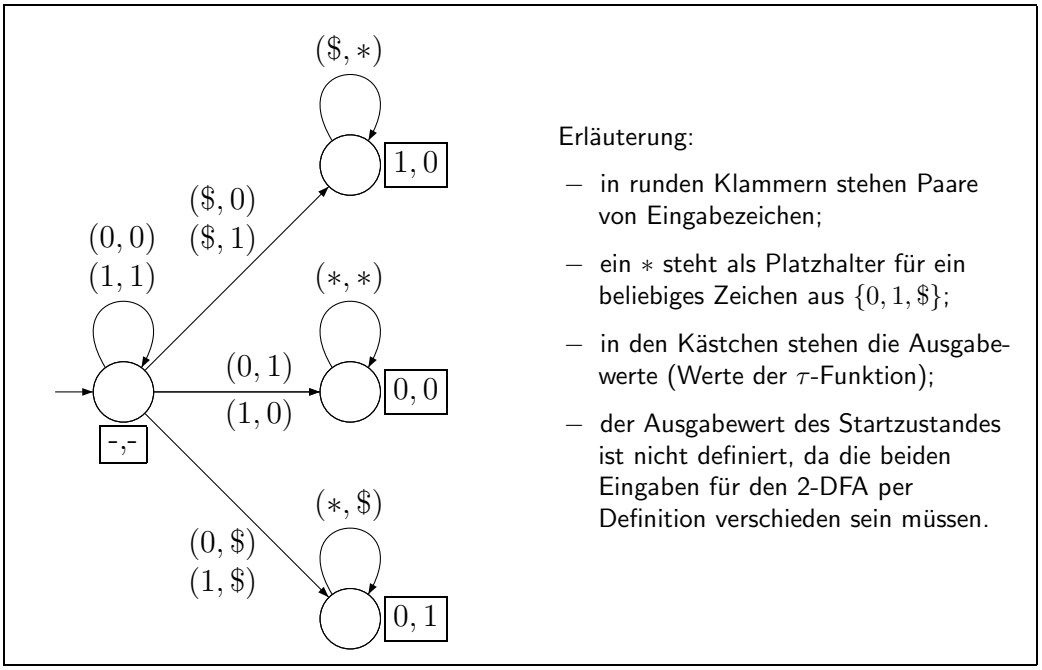


Abbildung 4.1: Der 2-DFA zur $(1,2)$-Erkennung von Pfadsprachen.

2-DFA A. Der Automat A muss auf Eingabe (u, v) lediglich feststellen, ob $u \preceq v$, $v \preceq u$ gilt oder ober beide Wörter bzgl. der Präfixeigenschaft unvergleichbar sind. Die Ausgabe lautet dann:

$$\tau(q_0 \cdot (u,v)) = \begin{cases} (1,0), & \text{falls } u \preceq v \\ (0,1), & \text{falls } v \preceq u \\ (0,0), & \text{sonst} \end{cases}$$

Die Korrektheit im dritten Fall ist klar. Angenommen, $u \preceq v$ und beide Antworten wären falsch, d. h. $u \notin A_\alpha$ und $v \in A_\alpha$. Das würde bedeuten, dass $u \preceq v \preceq \alpha$, aber $u \not\preceq \alpha$, ein Widerspruch! Die Argumentation für $v \preceq u$ ist analog.

Man beachte, dass ein einziger Automat genügt, um sämtliche Pfadsprachen mit maximal einem Fehler zu erkennen. ◇

Das vorangegangene Beispiel zeigt zwei Dinge: zum einen gibt es überabzählbar viele nicht reguläre Sprachen, die $(1,2)$-erkennbar sind;[1] zum anderen folgt aus $L \in \mathrm{REG}(1,2)$ auch $L \in \mathrm{REG}(m,n)$ für alle $1 \leq m \leq n/2$ —

[1] Unter den $(1,2)$-erkennbaren Sprachen sind sogar überabzählbar viele *nicht rekursiv aufzählbare* Sprachen, was für das vorliegende Kapitel jedoch keine weitere Rolle spielt.

es gibt also in jeder Klasse $\mathrm{REG}(m,n)$ überabzählbar viele nicht reguläre Sprachen.

Auch die Anti-Pfadsprachen B_α sind $(1,2)$-erkennbar: auf Eingabe (u,v) unterscheiden wir vier Fälle (im folgenden ist u' eindeutig definiert durch $u = u'b$, $b \in \mathbb{B}$, und ohne Einschränkung gelte $|u| \leq |v|$):

1. $u \preceq v$: Ausgabe $(0,0)$
2. $|u| = |v|$: Ausgabe $(0,0)$
3. $|u| < |v|$, $u \not\preceq v$ und $u' \preceq v$: Ausgabe $(1,0)$
4. $|u| < |v|$, $u \not\preceq v$ und $u' \not\preceq v$: Ausgabe $(0,0)$

Von der Korrektheit der Ausgaben kann man sich leicht überzeugen. Auf die Angabe des zugehörigen 2-DFA verzichten wir.

Beispiel 4.2 (Links-Schnitte) Für eine endliche oder unendliche Zeichenkette x über dem Alphabet $\mathbb{B}$ sei $\mathrm{bin}(x)$ der Wert der Zahl $0{,}x$ interpretiert als Binärzahl, also $\mathrm{bin}(x) = \sum_{i\geq 1} x[i]\cdot 2^{-i}$. Dann können wir zu jedem $\alpha \subseteq \mathbb{B}^\omega$ den sog. *Links-Schnitt* (engl. *Left-Cut*) von α definieren:

$$\mathrm{LC}_\alpha := \{x \in \mathbb{B}^* \mid \mathrm{bin}(x) < \mathrm{bin}(\alpha)\},$$

also die Menge aller endlichen Wörter x über $\mathbb{B}$, für die $0{,}x < 0{,}\alpha$ gilt (bei Interpretation als Binärzahlen). Diese Sprachen werden alle von dem in Abbildung 4.2 dargestellten Automaten B $(1,2)$-erkannt.

Der Automat B muss auf Eingabe (u,v) lediglich feststellen, welches der beiden Wörter einen kleineren Zahlenwert repräsentiert als das andere. Für das kleinere wird 1 ausgegeben, für das größere 0. Bei dieser Ausgabe machen wir höchstens einen Fehler, denn angenommen beide Antworten sind falsch, dann ist das „kleinere“ Wort größer als $0{,}\alpha$, das „größere“ hingegen kleiner als $0{,}\alpha$ — Widerspruch! (Sind beide Werte gleich, z. B. bei $(u,v) = (101, 10100)$, so geben wir $(1,0)$ aus.) ◇

Dieses Beispiel liefert uns erneut überabzählbar viele nicht reguläre Sprachen, die $(1,2)$-erkennbar sind. Eine minimale Variation des Beispieles erhält man, wenn man zu gegebenem $\alpha \in \mathbb{B}^\omega$ die Menge der Wörter betrachtet, die lexikographisch kleiner als α sind.

In Abschnitt 4.5 werden wir als weitere Beispiele boolesche Kombinationen von Pfadsprachen bzw. Links-Schnitten betrachten.

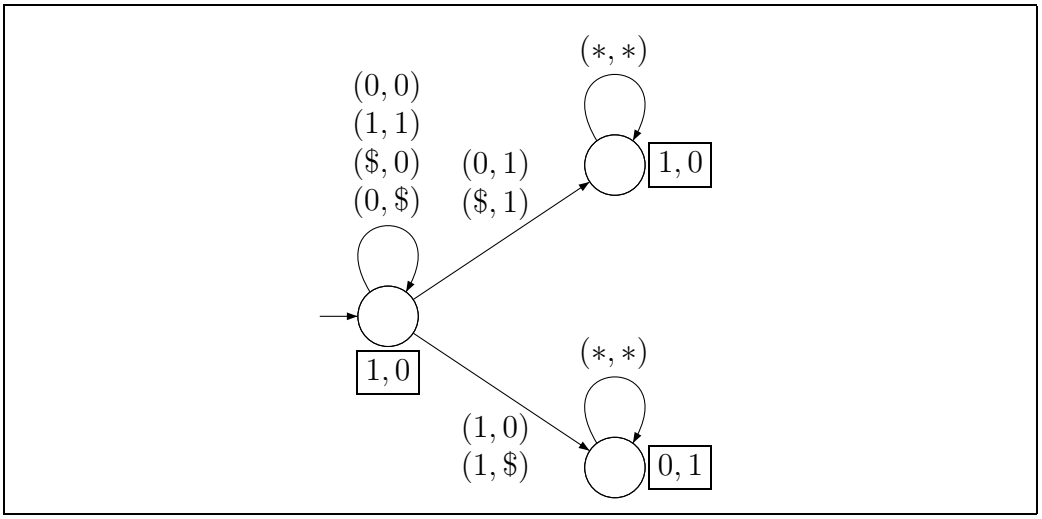


Abbildung 4.2: Der 2-DFA zur $(1, 2)$-Erkennung von Links-Schnitten (die Darstellung des DFA ist in Abbildung 4.1 erläutert).

4.1 Analogon zu Trakhtenbrots Resultat

In Abschnitt 2.2 haben wir einen neuen Beweis für Trakhtenbrots Aussage gegeben, dass (m, n)-berechenbare Funktionen für $m > n/2$ bereits berechenbar im herkömmlichen Sinne sind. Durch eine genauere Inspektion unseres Beweises wollen wir zeigen, dass sich diese Aussage auch auf reguläre Häufigkeitsberechnungen überträgt.

Satz 4.3 *Sei $m > n/2$ und $f : \Sigma^* \to \Sigma^*$ eine (m, n)-fa-berechenbare Funktion. Dann ist f durch einen endlichen Automaten berechenbar.*

Beweis: Wir verwenden den gleichen (neuen) Beweis, den wir auf Seite 37 ff. gegeben haben. Wir müssen uns lediglich vergewissern, dass der dort aufgeführte Algorithmus von einem endlichen Automaten realisiert werden kann. Wir wiederholen zunächst den Algorithmus von Seite 39:

Eingabe: $(x_1, \ldots, x_{n-1}) \in (\Sigma^*)^{n-1}$

$i := 2n - 1$
while $\nexists Y \subseteq W_i : |Y| = 2n - 1$ und Y erfüllt Bed. (B1), $(B2)_1$ bis $(B2)_{n-1}$ **do**
  $i := i + 1$
end
sei $Y \subseteq W_i$ mit $|Y| = 2n - 1$ und Y erfüllt Bed. (B1), $(B2)_1$ bis $(B2)_{n-1}$

Ausgabe: $(f_1, \ldots, f_{n-1})$, wobei $(f_1, \ldots, f_n) = A(y)$ für ein $y \in Y$

Dabei ist A jetzt ein n-DFA (mit Zustandsmenge Q_A und Übergangsfunktion δ_A), der die Funktion f (m,n)-fa-berechnet. Wir werden zeigen, dass obiger Algorithmus von einem $(n-1)$-DFA C durchgeführt werden kann.

Als Zwischenschritt konstruieren wir einen *nichtdeterministischen* $(n-1)$-DFA B, dessen Ausgabe eine von zwei Formen annimmt:

1. $(f_1, \ldots, f_{n-1})$, und mindestens m der f_i stimmen mit den $f(x_i)$ überein; oder
2. $(\perp, \ldots, \perp)$, falls die erste Ausgabe nicht möglich ist (dies entspricht dem Wert *undefiniert*).

Die Zustandsmenge Q_B von B enthält Zustände

$$(q_1, \ldots, q_{2n-1}, r_1, \ldots, r_N) \quad \text{mit } N = \binom{2n-1}{n} \text{ und } q_i, r_i \in Q_A$$

Startzustand ist $(q_{A,0}, \ldots, q_{A,0})$, wobei $q_{A,0}$ der Startzustand des DFA A ist.

Um die Übergangsrelation δ_B beschreiben zu können, benötigen wir eine Bijektion φ zwischen geordneten n-Tupeln $1 \leq i_1 < \ldots < i_n \leq 2n-1$ und Elementen der Menge $\{1, \ldots, N\}$. Es sei also $\varphi : \{1, \ldots, 2n-1\}^n \to \{1, \ldots, N\}$ eine solche Abbildung, die restringiert auf die geordneten n-Tupel bijektiv ist.

In jedem Schritt liest B Zeichen $a_1, \ldots, a_{n-1} \in (\Sigma \cup \{\$\})$. Zusätzlich „rät“ B in jedem Schritt $2n-1$ weitere Zeichen $b_1, \ldots, b_{2n-1} \in \Sigma$. Für diese Wahl der b_i enthält die Übergangsrelation δ_B folgenden Eintrag:

$$\big(\ (q_1, \ldots, q_{2n-1}, r_1, \ldots, r_N),\ (a_1, \ldots, a_{n-1}),\ (q'_1, \ldots, q'_{2n-1}, r'_1, \ldots, r'_N)\ \big)$$

mit $q'_i = \delta_A(q_i, (a_1, \ldots, a_{n-1}, b_i))$, $1 \leq i \leq 2n-1$ und $r'_i = \delta_A(r_i, (b_{i_1}, \ldots, b_{i_n}))$, wobei $1 \leq i_1 < \ldots < i_n \leq N$ und $i = \varphi(i_1, \ldots, i_n)$ ist.

Die Idee hinter dieser Konstruktion ist folgende: die geratenen b_i entsprechen der Menge Y im Algorithmus, und in den Zuständen r_i wird für jede Auswahl von n der $2n-1$ Wörter aus Y protokolliert, in welchen Zustand der DFA A bei Eingabe dieser n Wörter gelangen würde.

Zur vollständigen Definition von B müssen wir noch die Ausgabewerte festlegen. Da dies sehr technisch wird, wollen wir auch hier zunächst die Idee beschreiben: nach Lesen der Eingabe $(x_1, \ldots, x_{n-1})$ (und parallelem Lesen

der geratenen $2n-1$ Wörter aus Y) befinden wir uns in einem Zustand q. Wir wollen eine Menge Y finden, die die Bedingungen (B1) und $(\text{B2})_1$ bis $(\text{B2})_{n-1}$ erfüllt: also setzen wir in Gedanken die Wörter aus Y zu solchen fort, die dies bewerkstelligen. Evtl. gibt es für die bereits festgelegten Anfangsstücke der Wörter aus Y eine solche Fortsetzung nicht — dann ist der Ausgabewert undefiniert (dies entspricht $(\perp, \ldots, \perp)$). Andernfalls sei p der Zustand, den wir von q aus erreichen, wenn die Fortsetzung in Gedanken erfolgreich ist, und wir definieren die Ausgabe von B als die Projektion der Ausgabe von A im Zustand p auf die ersten $n-1$ Komponenten.

Formal definieren wir zu einem Zustand $q \in Q_B$ eine Menge von Zuständen $R_q \subseteq Q_B$ mit $p \in R_q$ genau dann, wenn folgendes gilt: es existieren $2n-1$ paarweise verschiedene Wörter $y_1, \ldots, y_{2n-1} \in \Sigma^*$, und $p \in Q_B$ ist der Zustand, den wir mit Übergängen wie oben beschrieben von q aus erreichen, wobei $a_i = \varepsilon$ für alle $1 \leq i \leq n-1$ gilt und die Zeichen für Zeichen gelesenen y_j die zu ratenden Zeichen $b_1, \ldots, b_{2n-1}$ ersetzen. Ist beispielsweise $n = 2$, $y_1 = a$, $y_2 = bb$ und $y_3 = abba$, so sind die vier geratenen Tupel (a, b, a), $(\$, b, b)$, $(\$, \$, b)$ und $(\$, \$, a)$.

Da jetzt auch $b_i = \$$ möglich ist und daher für die Definition von r'_i Übergänge in A mit $(\$, \ldots, \$)$ benötigt werden (die in A nicht vorhanden sind), definieren wir $\delta_A(q, (\$, \ldots, \$)) = q$ für alle $q \in Q_A$. Dies entspricht der Tatsache, dass der Automat A die Eingabe vollständig gelesen hat und im erreichten Zustand „verharrt“.

Mit Hilfe der Menge R_q können wir die Ausgabe von B im Zustand q definieren: angenommen, ein $p = (q_1, \ldots, q_{2n-1}, r_1, \ldots, r_N) \in R_q$ existiert, das folgende Eigenschaften erfüllt:

(B1′) $\forall 1 \leq i_1 < \ldots < i_n \leq 2n-1$: die Vektoren $(A(q_{i_1})[n], \ldots, A(q_{i_n})[n])$ und $A(r_{\varphi(i_1,\ldots,i_n)})$ unterscheiden sich an mindestens m Stellen;

$(\text{B2}')_j$ $(1 \leq j \leq n-1)$

$$\#\{A(q_i)[j] \mid 1 \leq i \leq 2n-1\} \in \{1, 2n-1\}$$

Dann definieren wir $B(p) = A(q_1)[1, \ldots, n-1]$ für einen solchen Zustand p (gibt es mehrere solcher Zustände, ist die Wahl willkürlich). Andernfalls ist $B(p) = (\perp, \ldots, \perp)$.

Schließlich können wir den deterministischen $(n-1)$-DFA C definieren, der aus dem nichtdeterministischen $(n-1)$-DFA B durch Potenzautomatenkonstruktion entsteht. Sei $Q_C = \mathcal{P}(Q_B)$ die Zustandsmenge von C und

$P = \{p_1, \ldots, p_\ell\} \in Q_C$ ein erreichbarer Zustand in C (um nicht erreichbare Zustände werden wir uns im folgenden nicht kümmern). Die Ausgabe $C(P)$ ist gleich $B(p_i)$, falls ein $1 \leq i \leq \ell$ existiert, für das $B(p_i)$ ungleich $(\bot, \ldots, \bot)$ ist.

Eine Ausgabe für den anderen Fall definieren wir erst gar nicht, denn wir werden argumentieren, dass ein entsprechendes i immer existiert. Da es zu gegebenen $x_1, \ldots, x_{n-1} \in \Sigma^*$ unendlich viele $y \in \Sigma^*$ gibt, für die $A(q_0 \cdot (x_1, \ldots, x_{n-1}, y))[n] \neq f(y)$ ist, können wir auch $2n-1$ solcher Wörter $y_1, \ldots, y_{2n-1} \in \Sigma^*$ mit $|x| \leq |y_i|$, $1 \leq i \leq 2n-1$, wählen.

Für $1 \leq i \leq 2n-1$ sei $y_i = y_i' y_i''$ mit $|y_i'| = |x|$. Dann hatte B die Möglichkeit, während der Verarbeitung der Eingabe $(x_1, \ldots, x_{n-1})$ genau die Wörter $y_1', \ldots, y_{2n-1}'$ zu raten. Ferner gibt es im so erreichten Zustand eine Fortsetzung, nämlich mittels $y_1'', \ldots, y_{2n-1}''$, die die Bedingungen (B1′) und $(\text{B2}')_1$ bis $(\text{B2}')_{n-1}$ erfüllt — was beweist, dass C in jedem erreichbaren Zustand P ein $p_i \in P$ findet mit $B(p_i) \neq (\bot, \ldots, \bot)$.

Die Bedingungen (B1′) und $(\text{B2}')_1$ bis $(\text{B2}')_{n-1}$ entsprechen den Bedingungen (B1) und $(\text{B2})_1$ bis $(\text{B2})_{n-1}$, die Argumentation zur Korrektheit kann also dem kombinatorischen Beweis von Satz 2.1 entnommen werden (vgl. Seite 37 ff.). Also realisiert C eine $(m, n-1)$-fa-Berechnung der Funktion f. □

4.2 Aperiodische Sprachen

Im vorigen Abschnitt haben wir uns mit (m, n)-fa-berechenbaren *Funktionen* beschäftigt; jetzt beschränken wir uns wieder auf (m, n)-erkennbare *Sprachen* (d. h. die Ausgaben der Automaten stammen aus der Menge $\mathbb{B}^n$). Wir werden das folgende Resultat zeigen: durch reguläre Häufigkeitsberechnungen mit einem aperiodischen Automaten werden entweder nicht-reguläre Sprachen oder nicht aperiodische Sprachen erkannt. Für einen guten Überblick über aperiodische Sprachen sei auf [Per90, Abschn. 6] verwiesen.

Definition 4.4 (aperiodische Sprache) *Eine reguläre Sprache $L \subseteq \Sigma^*$ heißt* aperiodisch*, falls eine Konstante $k \in \mathbb{N}$ so existiert, dass für alle $u, v, w \in \Sigma^*$ gilt:*

$$uv^k w \in L \quad \Leftrightarrow \quad uv^{k+1} w \in L.$$

Analog kann man *aperiodische Automaten* für „herkömmliche“ endliche Automaten definieren und diese Definition auf n-DFA ausweiten:

Definition 4.5 (aperiodischer Automat) *Ein endlicher Automat A mit Startzustand q_0 und Endzustandsmenge F heißt* aperiodisch*, falls eine Konstante $k \in \mathbb{N}$ so existiert, dass für alle $u, v, w \in \Sigma^*$ gilt:*

$$q_0 \cdot uv^k w \in F \quad \Leftrightarrow \quad q_0 \cdot uv^{k+1} w \in F. \tag{4.1}$$

Für einen n-DFA A mit Startzustand q_0 und Typ-Funktion τ lautet Gleichung 4.1 entsprechend:

$$\tau(q_0 \cdot uv^k w) = \tau(q_0 \cdot uv^{k+1} w). \tag{4.2}$$

Dabei sind u, v, w jetzt n-Tupel von Wörtern über Σ.

Eine alternative Definition, die zur gleichen Sprachklasse führt, wären sog. *zyklenfreie Automaten*: ein DFA heißt zyklenfrei, wenn es kein Wort $u \in \Sigma^+$ und keine paarweise verschiedenen Zustände $q_0 \ldots, q_{\ell-1} \in Q$, $\ell \geq 2$, gibt mit $q_i \cdot u = q_{i+1 \bmod \ell}$ $\forall 0 \leq i < \ell$. Ein zyklenfreier Automat darf also keine nicht-triviale Schleife enthalten (eine Schleife heißt trivial, wenn $\ell = 1$ gilt).

Die aperiodischen regulären Sprachen entsprechen genau den Sprachen, die durch aperiodische Automaten erkannt werden können. Es gibt viele weitere Charakterisierungen aperiodischer Sprachen: ein bekanntes Resultat von Schützenberger besagt, dass die aperiodischen Sprachen genau den *sternfreien* Sprachen entsprechen, also den Sprachen, die sich durch erweiterte reguläre Ausdrücke (inkl. Komplement) ohne Kleene-Stern darstellen lassen [Sch65]. Außerdem entsprechen die aperiodischen Sprachen genau den Sprachen, die sich durch prädikatenlogische Formeln der 1. Stufe (*first order*- bzw. *FO-Formeln*) ausdrücken lassen.

Satz 4.6 *Sei $1 \leq n$ und $L \subseteq \Sigma^*$ eine reguläre Sprache, die durch einen aperiodischen Automaten A $(1, n)$-erkannt wird. Dann ist L eine aperiodische Sprache.*

Beweis: Sei $L \subseteq \Sigma^*$ eine reguläre Sprache, die von einem aperiodischen Automaten A $(1, n)$-erkannt wird. Wir nehmen an, L sei nicht aperiodisch, d. h. kein DFA B mit $L(B) = L$ ist aperiodisch. Sei also B ein DFA mit $L(B) = L$, dann gibt es Wörter $u, v, w \in \Sigma^*$ und ein $\ell \geq 2$ derart, dass für alle $i \geq 0$ gilt:

$$uv^{i \cdot \ell} w \in L \quad \text{und} \quad uv^{i \cdot \ell + 1} w \notin L. \tag{4.3}$$

Dies kann man wie folgt einsehen: Der DFA B ist nicht aperiodisch, d. h. Gleichung 4.1 gilt nicht. Also existieren für jedes $k \in \mathbb{N}$ Wörter $u, v, w \in \Sigma^*$ mit

$$q_0 \cdot uv^k w \in F \quad \Leftrightarrow \quad q_0 \cdot uv^{k+1} w \notin F.$$

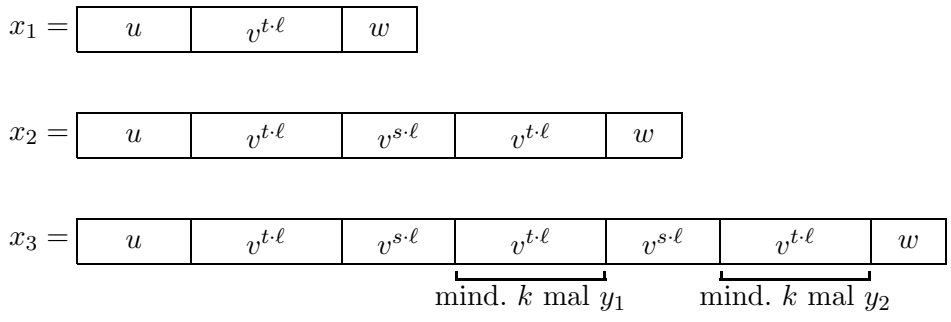
Wenn wir k größer als die Zustandszahl von B wählen, durchläuft dieser Automat beim Lesen von v^k eine Schleife, die mit v^ℓ, $\ell \geq 2$, beschriftet ist ($\ell = 1$ ist nicht möglich, da sonst $q_0 \cdot uv^k w = q_0 \cdot uv^{k+1} w$ wäre). Auf dieser Schleife gibt es „Austrittspunkte“ für w, von denen manche zu Endzuständen, andere zu Nicht-Endzuständen führen. Indem wir einige der Wörter v an u anhängen, können wir erzwingen, dass beim Eintritt in die Schleife (4.3) erfüllt ist.

Sei $k \in \mathbb{N}$ jetzt die Konstante, für die Gleichung 4.2 für den n-DFA A erfüllt ist. Seien $s, t \geq 0$ so gewählt, dass $|v^{s \cdot \ell}| \geq |w|$ und $t \cdot \ell \geq k$. Wir betrachten die Eingabe $x = (x_1, \ldots, x_n)$ für A mit

$$x_i = uv^{(i \cdot t + (i-1) \cdot s) \cdot \ell} w \qquad (1 \leq i \leq n).$$

Das Wort v ist im $(i+1)$-ten Eingabewort $(s+t) \cdot \ell$ mal häufiger enthalten als im i-ten Eingabewort. Die $s \cdot \ell$ vielen v überbrücken das w von x_i, da $s \cdot \ell \cdot |v| \geq |w|$ ist; die nächsten $t \cdot \ell$ vielen v garantieren eine mindestens k-malige Wiederholung von $(\underbrace{\$^{|v|}, \ldots, \$^{|v|}}_{i \text{ mal}}, \underbrace{v, \ldots, v}_{n-i \text{ mal}})$, da $t \cdot \ell \geq k$ ist. Dieses n-Tupel nennen wir y_i.

Folgende Abbildung veranschaulicht für $n = 3$ die Eingabe (x_1, x_2, x_3) und die Bereiche, in denen sich die Tupel y_1 und y_2 befinden:



Es gilt $\chi_L^{(n)}(x) = (1, \ldots, 1)$, d. h. die zugehörige Ausgabe $b = (b_1, \ldots, b_n)$ von A wird mindestens eine 1 enthalten.

Wir modifizieren die Eingabewörter der Reihe nach für $i = 1$ bis n: ist $b_i = 0$, so lassen wir das i-te Wort unverändert; ist $b_i = 1$, so wiederholen wir in dem Teil der $t \cdot \ell$ vielen Tupel y_i, der hinter dem w des $(i-1)$-ten Wortes liegt, das Tupel y_i ein mal. Um die Zugehörigkeit von x_{i+1} bis x_n zur Sprache nicht zu ändern, wiederholen wir in dem danach folgenden Überhang (zwischen dem i-ten und dem $(i+1)$-ten Wort) das Tupel y_{i+1} genau $\ell - 1$ mal.

Sei $x' = (x'_1, \ldots, x'_n)$ die derart modifizierte Eingabe. Da A aperiodisch ist, gilt $\tau(q_0 \cdot x) = \tau(q_0 \cdot x')$, denn die y_i wurden nur in solchen Bereichen zusätzlich eingefügt, in denen der Automat bereits k mal y_i gelesen hat und dies von $k+1$ mal y_i nicht mehr unterscheiden kann. Also wird auch auf x' die Ausgabe b gegeben. Nach Konstruktion gilt aber $\chi_L^{(n)}(x') = \overline{b}$, d. h. A gibt keine einzige richtige Antwort mehr. Dies ist ein Widerspruch, L muss also aperiodisch gewesen sein. □

Die Automaten der Beispiele 4.1 und 4.2 sind beide aperiodisch. Nach dem eben bewiesenen Satz 4.6 sind also alle regulären Pfadsprachen und alle regulären Links-Schnitte aperiodische Sprachen.

4.3 Iterationskriterium

In diesem Abschnitt beweisen wir ein „Iterationskriterium“ für reguläre Häufigkeitsberechnungen, das uns ähnlich dem Pumping-Lemma für reguläre Sprachen häufig den Nachweis erlauben wird, dass eine Sprache nicht zur Klasse FREQ-REG gehört.

Wir benötigen hierfür noch eine weitere Definition. Für ein $r \geq 1$ und eine Sprache $L \subseteq \Sigma^*$ sei

$$L_r := \{xyz \in L \mid xy^*z \subseteq L,\ |y| \geq 1 \wedge |yz| \leq r\}.$$

Man beachte, dass aus $xyz \in L_r$ (mit $xy^*z \subseteq L$, $|y| \geq 1 \wedge |yz| \leq r$) sofort $xy^*z \subseteq L_r$ folgt.

Diese Menge erinnert stark an das bekannte Pumping-Lemma für reguläre Sprachen: innerhalb der letzten r Zeichen eines Wortes der Sprache befindet sich ein nicht-leerer Teil, der wiederholt werden kann. Intuitiv besagt das

Iterationskriterium, dass nach Entfernen von L_r aus einer (m, n)-erkennbaren Sprache L keine unendlichen regulären Strukturen mehr enthalten sind (nicht einmal mehr im Präfixabschluss von $L \setminus L_r$). Die Konstante r wird hierbei der Anzahl der Zustände eines n-DFA für L entsprechen.

Satz 4.7 (Iterationskriterium für FREQ**-**REG**)** *Sei $L \subseteq \Sigma^*$ eine $(1, n)$-erkennbare Sprache. Dann gibt es ein $r \geq 1$ derart, dass die Menge* $\mathrm{pref}(L \setminus L_r)$ *keine unendliche reguläre Teilmenge enthält.*

Beweis: Sei $L \subseteq \Sigma^*$ eine $(1, n)$-erkennbare Sprache und n minimal, d. h. L ist nicht $(1, n-1)$-erkennbar. Sei $A = (Q, \Sigma, \delta, q_0, \tau, n)$ ein deterministischer endlicher Automat, der L $(1, n)$-erkennt. Wir werden zeigen, dass die Aussage für $r = |Q|$ gilt.

Für $n = 1$ ist L eine reguläre Sprache. Aus dem Beweis des Pumping-Lemmas für reguläre Sprachen folgt, dass sich jedes Wort $w \in L$ mit $|w| \geq r$ zerlegen lässt in $w = xyz$ mit $|y| \geq 1$, $|yz| \leq r$ und $xy^*z \subseteq L$. Folglich enthält $L \setminus L_r$ nur endliche viele Wörter (mit Länge $< r$), und $\mathrm{pref}(L \setminus L_r)$ ist ebenfalls endlich.

Von jetzt an können wir also $n \geq 2$ annehmen. Wir nehmen im Gegensatz zur Aussage des Satzes an, dass $\mathrm{pref}(L \setminus L_r)$ eine unendliche reguläre Teilmenge xy^*z mit $|y| \geq 1$ enthält. Dies würde bedeuten, dass $\mathrm{pref}(L \setminus L_r)$ auch die unendliche reguläre Teilmenge xy^* enthält.

Wir ersetzen y durch eine geeignete, große Potenz von y (und nennen diese Potenz von y wiederum y) so, dass für alle Zustände $q \in Q$ gilt:

$$q \cdot (\varepsilon, \ldots, \varepsilon, y) = q \cdot (\varepsilon, \ldots, \varepsilon, y^2). \tag{4.4}$$

Es genügt z. B., y durch $y^{r!}$ zu ersetzen.

Die Idee des restlichen Beweises ist nun folgende: wir konstruieren zunächst einen endlichen Automaten A' mit $n-1$ Komponenten, der auf Eingabe $u = (u_1, \ldots, u_{n-1})$ den ursprünglichen Automaten A auf der Eingabe $(u_1, \ldots, u_{n-1}, u_n)$ simuliert, wobei u_n der Präfix von xy^ω der Länge $|u|$ ist. Da n minimal gewählt war, kann A' die Sprache L nicht $(1, n-1)$-erkennen. Es gibt also eine Eingabe $(u_1, \ldots, u_{n-1})$, bei der sämtliche Antworten von A' falsch sind. Dies wird Rückschlüsse auf die n-te Antwort von A zulassen, die letztendlich zu einem Widerspruch führen werden.

Zunächst definieren wir formal den deterministischen endlichen Automaten $A' = (Q', \Sigma, \delta', q_0', \tau', n-1)$:

- $Q' = Q \times \text{suff}(xy)$.
- $q'_0 = \langle q_0, xy \rangle$.
- Für $c_1, \ldots, c_{n-1} \in (\Sigma \cup \{\$\})$, $q \in Q$ und $s \in \text{suff}(xy)$ ist

  $$\delta'(\langle q, s \rangle, (c_1, \ldots, c_{n-1})) = \langle \delta(q, (c_1, \ldots, c_{n-1}, a)), s' \rangle,$$

  wobei $a \in \Sigma$ und $s' \in \Sigma^*$ eindeutig bestimmt sind durch $as' = s$, falls $s \neq \varepsilon$, und $as' = y$ sonst.
- Die Definition der neuen Typen τ' geschieht in zwei Schritten.

  Für $i \geq 1$ wählen wir zunächst Wörter z_i mit $xy^i z_i \in L \setminus L_r$ und $|z_i| > r$. Diese existieren, da $xy^* \subseteq \text{pref}(L \setminus L_r)$ ist. Damit definieren wir für jedes $i \geq 1$ und jeden Zustand $\langle q, s \rangle \in Q'$ einen Typ $\tau_i(\langle q, s \rangle)$ durch

  $$\tau_i(\langle q, s \rangle) := \tau(q \cdot (\varepsilon, \ldots, \varepsilon, syz_i)).$$

  Mit Hilfe der τ_i können wir schließlich die Typen τ' definieren; $\tau'(\langle q, s \rangle)$ ist einer der Typen $\tau_i(\langle q, s \rangle)$, die unendlich oft vorkommen, restringiert auf die ersten $n-1$ Komponenten. Kommen mehrere Typen unendlich oft vor, wird willkürlich einer davon ausgewählt. Andererseits gibt es einen solchen Typ, da es maximal 2^n verschiedene Typen $\tau_i(\langle q, s \rangle)$ gibt.

Abbildung 4.3 veranschaulicht diese Definitionen und die nachfolgenden Argumentationsschritte.

Da n minimal gewählt wurde, kann A' die Sprache L nicht $(1, n-1)$-erkennen. Also gibt es eine Eingabe $(u_1, \ldots, u_{n-1})$, die A' vollständig falsch beantwortet, d. h. die Vektoren $\tau'(q'_0 \cdot (u_1, \ldots, u_{n-1}))$ und $\chi_L^{(n-1)}(u_1, \ldots, u_{n-1})$ sind komplementär zueinander.

Mit $q'_0 \cdot (u_1, \ldots, u_{n-1}) = \langle q, s \rangle$ ist $u_n s = xy^k$ für ein $k \geq 1$. Sei $i \geq k+1$ ein Wert, für den $\tau'(\langle q, s \rangle) = \tau_i(\langle q, s \rangle)[1, \ldots, n-1]$ ist. Dann gilt:

$$\begin{aligned}
\tau'(\langle q, s \rangle) &= \tau(q \cdot (\varepsilon, \ldots, \varepsilon, syz_i))[1, \ldots, n-1] \\
&= \tau(q_0 \cdot (u_1, \ldots, u_{n-1}, u_n syz_i))[1, \ldots, n-1] \\
&= \tau(q_0 \cdot (u_1, \ldots, u_{n-1}, xy^{k+1} z_i))[1, \ldots, n-1] \\
&= \tau(q_0 \cdot (u_1, \ldots, u_{n-1}, xy^i z_i))[1, \ldots, n-1] \qquad \text{(wegen Gl. 4.4)} \\
&= (b_1, \ldots, b_n)[1, \ldots, n-1]
\end{aligned}$$

mit $(b_1, \ldots, b_n) := \tau(q_0 \cdot (u_1, \ldots, u_{n-1}, xy^i z_i))$.

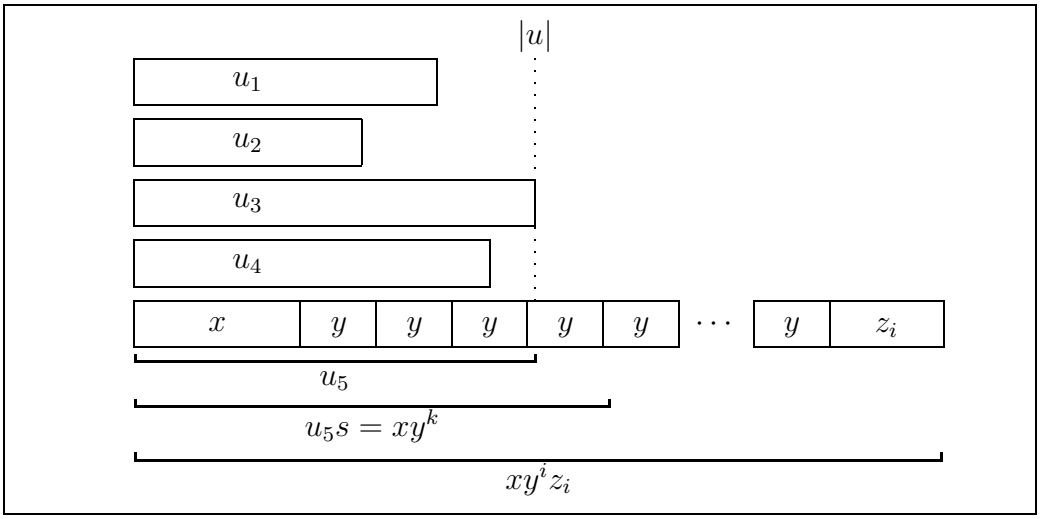


Abbildung 4.3: Veranschaulichung der Simulation des n-DFA A durch den $(n-1)$-DFA A' für $n = 5$; das Wort u_5 wird ist der Präfix von xy^ω der Länge $|u|$; nach Wahl der z_i gilt $xy^i z_i \in L \setminus L_r$.

Da die ersten $n-1$ Antworten falsch sind, muss $b_n = \chi_L(xy^i z_i)$ sein, denn der ursprüngliche Automat muss mindestens eine richtige Antwort geben. Nach unserer Wahl von z_i gehört $xy^i z_i$ zu L, also muss $b_n = 1$ gelten.

Wegen $|z_i| > r$ gibt es eine Zerlegung von $xy^i z_i$ in $x'y'z'$ mit $|y'| \geq 1$ und $|y'z'| \leq r$, und y' ist als Teilwort vollständig in z_i enthalten. Wegen $|x'| \geq |u|$ sind die ersten $n-1$ Eingaben bereits „verbraucht“ und wir können das Tupel $(\varepsilon, \ldots, \varepsilon, y')$ „pumpen“ (d. h. weglassen oder beliebig oft wiederholen). In A werden wir dabei immer in den gleichen Zustand gelangen.

Da wir die ersten $n-1$ Eingaben nicht verändert haben, sind die Antworten darauf immer noch falsch und die Antwort $b_n = 1$ auf die n-te Komponente muss richtig sein: es gilt also $x'(y')^* z' \subseteq L$ und wegen $|y'z'| \leq r$ sogar $x'(y')^* z' \subseteq L_r$. Insbesondere gilt $x'y'z' = xy^i z_i \in L_r$, ein Widerspruch zu unserer Wahl $xy^i z_i \in L \setminus L_r$. Also muss unsere Annahme, dass $\mathrm{pref}(L \setminus L_r)$ eine unendliche reguläre Teilmenge enthält, falsch gewesen sein. □

4.4 Anwendungen des Iterationskriteriums

Als erste Anwendung werden wir zeigen, dass alle $(1, n)$-erkennbaren Sprachen über einem unären Alphabet regulär sind. Dieses Resultat wurde bereits

1976 von Kinber gezeigt. In [ADHP00] haben wir ebenfalls einen direkten Beweis für diese Aussage gegeben. Mit Hilfe des Iterationskriteriums gestaltet sich der Beweis aber sehr elegant.

Satz 4.8 (Kinber [Kin76], Austinat et al. [ADHP00]) *Sei Σ ein einelementiges Alphabet und $L \subseteq \Sigma^*$ eine $(1,n)$-erkennbare Sprache. Dann ist L regulär.*

Beweis: Sie $\Sigma = \{a\}$. Das Iterationskriterium garantiert uns, dass für L eine Konstante $r \in \mathbb{N}$ derart existiert, dass $\mathrm{pref}(L \setminus L_r)$ keine unendliche reguläre Teilmenge enthält. Aus der Präfixabgeschlossenheit dieser Menge und der Tatsache $|\Sigma| = 1$ folgt die Endlichkeit von $\mathrm{pref}(L \setminus L_r)$ (andernfalls wäre $a^* \subseteq \mathrm{pref}(L \setminus L_r)$) und auch von $L \setminus L_r$.

Die Sprache L_r lässt sich als Vereinigung von Sprachen xy^*z mit $|yz| \leq r$ und $|y| \geq 1$ schreiben. Wegen dem einelementigen Alphabet ist $xy^*z = a^s(a^t)^*$ für $s = |x| + |z|$ und $t = |y|$, und es gilt außerdem $1 \leq t \leq r$. Damit lässt sich die Menge L_r schreiben als

$$L_r = \bigcup_{i \in I} a^{s_i}(a^{t_i})^* \qquad \text{mit } s_i \geq 0 \text{ und } 1 \leq t_i \leq r.$$

Jetzt streichen wir alle $i \in I$, für die ein $j \in I$ existiert mit $t_i = t_j$ und $s_i = s_j + k \cdot t_i$, $k \geq 1$. Dies ändert nichts an der Vereinigung, denn $a^{s_i}(a^{t_i})^* \subsetneqq a^{s_j}(a^{t_j})^*$. Wir erhalten

$$L_r = \bigcup_{i \in I} a^{s_i}(a^{t_i})^* \quad \text{mit } 1 \leq t_i \leq r \text{ und } \forall 1 \leq t \leq r : \#\{\langle s_i, t\rangle \mid i \in I\} \leq t.$$

Damit ist I endlich und L_r offensichtlich regulär. Es folgt die Regularität von L, denn $L = (L \setminus L_r) \cup L_r$. □

Wir wollen das Iterationskriterium für den Nachweis benutzen, dass gewisse Sprachen für kein $n \geq 1$ $(1,n)$-erkennbar sind. Insbesondere werden wir dies zeigen für $L' = \{a^i b^i \mid i \geq 0\}$, $L'' = \{ww \mid w \in \{a,b\}^*\}$ und $L''' = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$. Ohne das Iterationskriterium wäre zunächst völlig unklar, wie ein solcher Nachweis geführt werden sollte, da bereits für $n = 2$ überabzählbar viele $(1,2)$-erkennbare Sprachen existieren (vgl. Beispiele 4.1 und 4.2).

Lemma 4.9 *Für alle $n \geq 1$ ist die Sprache $L' = \{a^i b^i \mid i \geq 0\}$ nicht $(1,n)$-erkennbar.*

Beweis: Offensichtlich gilt $L'_r = \emptyset$ und damit $L' \setminus L'_r = L'$ für alle $r \geq 1$. Die Menge $\text{pref}(L')$ enthält jedoch die unendliche reguläre Teilmenge a^*, was nach Iterationskriterium für $(1, n)$-reguläre Sprache nicht sein kann. □

Ein noch einfacherer Beweis ergibt sich unter Verwendung des folgenden Korollars, das quasi eine Verallgemeinerung des eben geführten Beweises darstellt.

Korollar 4.10 (aus Satz 4.7) *Sei $L \subseteq \Sigma^*$ eine $(1, n)$-erkennbare Sprache. Wenn* $\text{pref}(L)$ *eine unendliche reguläre Teilmenge enthält, dann enthält bereits L selbst eine unendliche reguläre Teilmenge.*

Beweis: Angenommen, $\text{pref}(L)$ enthält eine unendliche reguläre Teilmenge, L jedoch nicht. Aus der zweiten Annahme folgt $L_r = \emptyset$ für alle $r \geq 1$. Also ist $\text{pref}(L) = \text{pref}(L \setminus L_r)$, und aus dem Iterationskriterium folgt, dass $\text{pref}(L \setminus L_r)$ (für geeignetes r) keine unendliche reguläre Teilmenge enthält. Dies widerspricht unserer Annahme bzgl. $\text{pref}(L)$. □

Beweis (von Lemma 4.9 mit Hilfe von Korollar 4.10): Es gilt $a^* \subseteq \text{pref}(L')$, doch L' enthält keine unendliche reguläre Teilmenge. Nach Korollar 4.10 gilt damit $L' \notin$ FREQ-REG. □

Weitere Beispiele von Sprachen, für die dieses Beweisprinzip funktioniert, sind $\{w\$w \mid w \in \{a, b\}^*\}$ und $\{w\$w^{rev} \mid w \in \{a, b\}^*\}$, da sie wiederum keine unendliche reguläre Teilmenge enthalten.

Die Sprachen L' und $\{w\$w^{rev} \mid w \in \{a, b\}^*\}$ gehören beide zur Klasse der *inhärent kontextfreien Sprachen*, d. h. zur Klasse der unendlichen kontextfreien Sprachen, die keine unendliche reguläre Teilmenge enthalten. Keine der Sprachen aus dieser Klasse gehört zu FREQ-REG.

Korollar 4.11 (aus Satz 4.7) *Sei $L \subseteq \Sigma^*$ eine inhärent kontextfreie Sprache. Dann gilt $L \notin$* FREQ-REG.

Beweis: Per Definition enthält L keine unendliche reguläre Teilmenge. Aus dem Pumping-Lemma für kontextfreie Sprachen folgt aber, dass $\text{pref}(L)$ eine unendliche reguläre Teilmenge enthält: man nehme ein Wort $z \in L$, dessen Länge die Pumping-Konstante übersteigt, und betrachtet den Präfix z' von z, der nach dem ersten nicht-leeren Pumpteil endet (wenn z in $uvwxy$ zerlegt wird: $z' = uv$, falls $v \neq \varepsilon$ bzw. $z' = uvwx$ sonst). Dann ist die unendliche reguläre Menge uv^* bzw. $uvwx^*$ in $\text{pref}(L)$ enthalten, und nach Korollar 4.10 kann L nicht zu FREQ-REG gehören. □

Auf die nicht kontextfreie Sprache $L'' = \{ww \mid w \in \{a,b\}^*\}$ können wir Korollar 4.11 nicht anwenden. Wir benötigen eine weitere Folgerung aus dem Iterationskriterium.

Korollar 4.12 (aus Satz 4.7) *Sei $L \in \mathrm{REG}(1,n)$. Dann gibt es ein $r \geq 1$ derart, dass für alle $u,v,w \in \Sigma^*$ gilt:*

$$|uv^*w \cap L| = \infty \quad \Rightarrow \quad |uv^*w \cap L_r| = \infty.$$

Beweis: Angenommen, für alle $r \geq 1$ existieren $u,v,w \in \Sigma^*$ mit $|v| \geq 1$ derart, dass $|uv^*w \cap L| = \infty$, aber $|uv^*w \cap L_r| < \infty$ gilt. Dann gibt es unendlich viele $i \geq 1$ mit $uv^iw \in L \setminus L_r$, und $uv^* \subseteq \mathrm{pref}(L \setminus L_r)$, ein Widerspruch zum Iterationskriterium. □

Damit können wir jetzt zeigen, dass $L'' \notin$ FREQ-REG ist.

Lemma 4.13 *Für alle $n \geq 1$ ist die Sprache $L'' = \{ww \mid w \in \{a,b\}^*\}$ nicht $(1,n)$-erkennbar.*

Beweis: Für $r \geq 1$ sei $u = \varepsilon$, $v = a^rba^rb$ und $w = \varepsilon$. Dann gilt $uv^*w = (a^rba^rb)^* \subseteq L''$, aber $(a^rba^rb)^* \cap L''_r = \emptyset$. Nach Korollar 4.12 gilt somit $L'' \notin$ FREQ-REG. □

Die Klasse FREQ-REG ist unter Komplement abgeschlossen (Lemma 4.18). Daher kann auch die kontextfreie Sprache $\overline{L''} = \{w \in \{a,b\}^* \mid \forall u \in \Sigma^* : w \neq uu\}$ nicht in FREQ-REG enthalten sein.

Einen ähnlichen Beweis wie für L'' können wir für die Sprache L''' führen:

Lemma 4.14 *Für alle $n \geq 1$ ist die Sprache $L''' = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$ nicht $(1,n)$-erkennbar.*

Beweis: Für $r \geq 1$ sei $u = a^r$, $v = ab$ und $w = b^r$. Dann gilt $uv^*w \subseteq L'''$, aber $uv^*w \cap L'''_r = \emptyset$. Nach Korollar 4.12 gilt somit $L''' \notin$ FREQ-REG. □

Eine weitere Anwendung des Iterationskriteriums sehen wir im nächsten Abschnitt: dort wird gezeigt, dass die Klasse der (m,n)-erkennbaren Sprachen nicht unter Spiegelung abgeschlossen ist.

Wir haben jetzt viele Beispiele kontextfreier und kontextsensitiver Sprachen gesehen, die nicht zu FREQ-REG gehören. Deshalb wollen wir zum Abschluss dieses Abschnitts noch zeigen, dass FREQ-REG auch kontextfreie, nicht reguläre Sprachen enthält (Stephan und Lange wiesen uns auf dieses Beispiel hin).

Beispiel 4.15 Sei $\alpha = 0101^201^301^4\ldots$ und A_α die zugehörige Pfadsprache. Die Sprache A_α ist *inhärent kontextsensitiv*, d. h. sie ist eine unendliche kontextsensitive Sprache und enthält keine unendliche kontextfreie Teilmenge. Man vergleiche dies mit der Tatsache, dass FREQ-REG keine inhärent kontextfreien Sprachen enthält (Korollar 4.11).

Wir wissen aus Beispiel 4.1, dass $A_\alpha \in \mathrm{REG}(1,2)$ gilt. Wegen dem Komplementabschluss der Klasse FREQ-REG (Lemma 4.18) gilt auch $B := \overline{A_\alpha} \in \mathrm{REG}(1,2)$. Wir behaupten, dass B kontextfrei ist (genauer gesagt ist B kontextfrei und inhärent mehrdeutig).

Um einzusehen, dass B kontextfrei ist, schreiben wir diese Sprache als Vereinigung fünf kontextfreier Sprachen:

$$\begin{aligned}
B_1 &= \{1v \mid v \in \{0,1\}^*\} \\
B_2 &= \{011v \mid v \in \{0,1\}^*\} \\
B_3 &= \{u00v \mid u,v \in \{0,1\}^*\} \\
B_4 &= \{u01^i01^j0v \mid u,v \in \{0,1\}^* \wedge j \neq i+1\} \\
B_5 &= \{u01^i01^j \mid u \in \{0,1\}^* \wedge j > i+1\}
\end{aligned}$$

Alle fünf Sprachen sind offensichtlich kontextfrei, und es gilt $B = \bigcup_{i=1}^{5} B_i$. $\diamond$

4.5 Abschlusseigenschaften

In diesem letzten Abschnitt wollen wir zeigen, dass die Klasse FREQ-REG einerseits nicht gegen Spiegelung abgeschlossen ist, andererseits jedoch eine boolesche Algebra bildet.

Satz 4.16 *Es gibt überabzählbar viele* $(1,2)$*-erkennbare Sprachen* L*, für die die zugehörigen Spiegelsprachen* L^{rev} *nicht in* FREQ-REG *liegen.*

Beweis: Wir betrachten ein Wort $\alpha = 0101^201^3\cdots \in \mathbb{B}^\omega$ und die zugehörige Pfadsprache A_α. Wie in Beispiel 4.1 gezeigt, ist A_α $(1,2)$-erkennbar. Wir werden zeigen, dass $(A_\alpha)^{rev}$ nicht in FREQ-REG liegt. Die Sprache A_α enthält offensichtlich keine unendliche reguläre Teilmenge, also kann auch die Spiegelsprache $(A_\alpha)^{rev}$ keine unendliche reguläre Teilmenge enthalten. Andererseits enthält $\mathrm{pref}((A_\alpha)^{rev})$ aber die unendliche reguläre Teilmenge 1^*. Nach Korollar 4.10 kann $(A_\alpha)^{rev}$ also nicht zur Klasse FREQ-REG gehören.

Damit kennen wir *eine* Sprache aus FREQ-REG, deren Spiegelung aus der Klasse herausführt. Daraus können wir aber durch leichte Modifikation überabzählbar viele Sprachen mit gleicher Eigenschaft ableiten: zu jedem $\gamma \in \mathbb{B}^\omega$ ersetzen wir die i-te 0 in α durch 00 genau dann, wenn das i-te Zeichen von γ eine 1 ist. Obige Beweisführung ändert sich dadurch nicht. □

Satz 4.17 *Die Klasse* FREQ-REG *ist eine boolesche Algebra.*

Dieses Ergebnis ergibt sich aus dem Abschluss von FREQ-REG unter Komplementbildung (Lemma 4.18) und Vereinigung (Lemma 4.19).

Der Abschluss von FREQ-REG unter Komplementbildung ist trivial:

Lemma 4.18 *Für alle* $1 \leq m \leq n$ *ist Klasse* $\mathrm{REG}(m,n)$ *unter Komplementbildung abgeschlossen (und damit auch die Klasse* FREQ-REG*).*

Beweis: Ist $L \in \mathrm{REG}(m,n)$ bzgl. eines n-DFA A, dann erhält man einen n-DFA, der die Sprache $\overline{L}$ (m,n)-erkennt, indem man die Ausgabe-Typen von A (d. h. die Funktionswerte von τ) invertiert. □

Interessanter ist die Frage, ob die Klasse FREQ-REG unter Vereinigung abgeschlossen ist. Dies wird im folgenden positiv beantwortet, doch anders als beim Komplementabschluss ist es nicht möglich, für die Vereinigungssprache mit den gleichen Parametern (m,n) auszukommen (siehe Satz 4.24).

Lemma 4.19 *Die Klasse* FREQ-REG *ist unter (endlicher) Vereinigung abgeschlossen.*

Beweis: Es genügt zu zeigen, dass für zwei Sprachen $L, K \in \mathrm{REG}(1,n)$ ein $N \geq 1$ existiert, für das $L \cup K \in \mathrm{REG}(1,N)$ gilt. Der Wert N wird eine geeignete Ramsey-Zahl und damit sehr groß sein und wird erst im Laufe des Beweises genauer definiert — für den Moment ist lediglich $N \geq n$ wichtig. A und B seien die n-DFA, die die Sprachen L und K jeweils $(1,n)$-erkennen. Sei C der zu konstruierende N-DFA, der die Sprache $L \cup K$ $(1,N)$-erkennen wird. Der Automat C erhält als Eingabe den Vektor $(u_1, \ldots, u_N)$.

Wir definieren zunächst einen vollständigen Hypergraphen $H = (V, E)$ der Dimension n mit Knotenmenge $V = \{1, \ldots, N\}$ und Kantenmenge $E = \binom{V}{n}$, wobei $\binom{V}{n}$ die Menge der n-elementigen Teilmengen von V ist. Jeder Hyperkante $e \in \binom{V}{n}$ wird eine Farbe $(c,d) \in \mathbb{B}^n \times \mathbb{B}^n$ zugewiesen, die sich wie folgt ergibt: ist $e = \{i_1, \ldots, i_n\}$ $(1 \leq i_1 < \ldots < i_n \leq N)$, dann sind c bzw. d die Ausgabevektoren von A bzw. B auf die Eingabe $(u_{i_1}, \ldots, u_{i_n})$. Die Anzahl der vergebenen Farben ist also kleiner oder gleich 4^n.

Nach Lesen der Eingabe $(u_1, \ldots, u_N)$ „kennt“ C den Hypergraphen H, denn C kann parallel für alle n-Tupel $(u_{i_1}, \ldots, u_{i_n})$ die Automaten A und B simulieren und deren Ausgaben in seiner Zustandsmenge speichern, da es nur endlich viele vollständige Hypergraphen der Dimension n mit N Knoten und 4^n Farben gibt.

Jetzt können wir das N definieren: N ist die minimale Anzahl an Knoten, die für einen Hypergraphen der Dimension n mit 4^n Farben nötig ist, um einen monochromatischen Untergraphen $G' = (V', E')$ der Größe $4n-2$ garantieren zu können. Dieses N existiert nach dem Ramsey-Theorem (siehe z. B. [GRS80]). Da der Automat C den kompletten Hypergraphen kennt, kennt er auch $4n-2$ Knoten, die einen monochromatischen Untergraphen bilden. Um die Darstellung im folgenden zu vereinfachen, sei $V' = \{1, \ldots, 4n-2\} \subseteq V$ die Knotenmenge des Untergraphen G' und (c,d) die Farbe aller seiner Kanten.

Die Einfarbigkeit des Untergraphen G' bedeutet also, dass die Automaten A und B auf sämtlichen Eingaben $(u_{i_1}, \ldots, u_{i_n})$, $1 \leq i_1 < \ldots < i_n \leq 4n-2$, den gleichen Antwortvektor c bzw. d geben.

Die Ausgabe von C definieren wir als

$$b := (\underbrace{0,1,\ldots,0,1}_{4n-2 \text{ Bits}}, \underbrace{0,\ldots,0}_{N-(4n-2) \text{ Bits}}),$$

also als alternierende Folge von 0en und 1en auf unserem monochromatischen Untergraphen G' und beliebigen Werten (hier: 0en) auf allen anderen

Eingaben. Wir nehmen an, dass keines der Ausgabebits korrekt ist, d. h. die $2n-1$ Wörter $u_1, u_3, \ldots, u_{4n-3}$ gehören zu $L \cup K$, die anderen $2n-1$ Wörter $u_2, u_4, \ldots, u_{4n-2}$ zu $\overline{L \cup K}$. Von den $2n-1$ Wörtern mit ungeradem Index gehören mindestens n zu L oder mindestens n zu K; zur weiteren Vereinfachung nehmen wir ohne Einschränkung an, dass die ersten n Wörter $u_1, \ldots, u_{2n-1} \in L$ sind.

Damit können wir für die ersten $2n$ Wörter $u_1, \ldots, u_{2n}$ garantieren, dass alle Wörter mit ungeradem Index zu L gehören und keines der Wörter mit geradem Index zu $L \cup K$. Im folgenden werden wir nur noch die ersten $2n$ Wörter u_i betrachten — bzgl. der Zugehörigkeit zu L bzw. $L \cup K$ gibt es auf diesen Wörtern keinen Unterschied mehr.

Die eindeutige Farbe von G' ist (c, d), wobei $c = (c_1, \ldots, c_n)$ sei. Da die u_i durch eine Folge aus $2n$ alternierenden 0en und 1en beantwortet werden, gibt es Indizes $1 \leq i_1 < \ldots < i_n \leq 2n$ mit $c_\ell = b[i_\ell]$ für alle $1 \leq \ell \leq n$. Sei i_ℓ der Index einer Eingabe u_{i_ℓ}, die der Automat A auf Eingabe $(u_{i_1}, \ldots, u_{i_n})$ richtig beantwortet (es gibt mindestens eine solche Eingabe, da A die Sprache L $(1, n)$-erkennt); es gilt also $c_\ell = \chi_L(u_{i_\ell})$. Nach Wahl der i_j gilt außerdem $c_\ell = b[i_\ell]$. Nach unserer Annahme, dass C keine richtige Antwort gibt, muss $b[i_\ell] \neq \chi_L(u_{i_\ell})$ sein, ein Widerspruch. Also ist mindestens eine Antwort von C korrekt. □

Der Wert N, für den die Vereinigungssprache $(1, N)$-erkennbar ist, ist wegen der Anwendung des Ramsey-Theorems extrem groß. Das folgende Beispiel suggeriert, dass er in vielen Fällen deutlich kleiner gewählt werden kann. Im Anschluss an das Beispiel werden wir uns mit der Frage beschäftigen, wie der optimale Wert für N aussieht.

Beispiel 4.20 (Vereinigung von Pfadsprachen)
Seien $\ell \geq 2$, $\alpha_1, \ldots, \alpha_\ell \in \mathbb{B}^\omega$, $A_{\alpha_1}, \ldots, A_{\alpha_\ell}$ die zugehörigen Pfadsprachen und $B := \bigcup_{i=1}^{\ell} A_{\alpha_i}$ die Vereinigung dieser Sprachen. Dann gilt $B \in \mathrm{REG}(1, \ell+1)$.

Wir konstruieren einen $(\ell+1)$-DFA D, der die Sprache B $(1, \ell+1)$-erkennt. Sei $(x_1, \ldots, x_{\ell+1})$ eine Eingabe für D. Der Automat bestimmt beim Lesen der Eingabe die Präfixbeziehung aller x_i untereinander. Wir unterscheiden zwei Fälle:

1. Die x_i sind paarweise unvergleichbar: dann gibt D den Vektor $(0, \ldots, 0)$ aus und gibt somit mindestens eine richtige Antwort, da jeweils nur ein x_i Präfix eines α_j sein kann.

2. Es gibt Indizes $i \neq j$ mit $x_i \preceq x_j$: Dann gibt D auf diesen beiden Wörten 1 bzw. 0 aus, wovon mindestens eine Antwort richtig sein muss (Argumentation analog zu Beispiel 4.1).

$\diamond$

Man kann sich leicht davon überzeugen, dass der Beweis zu Lemma 4.19 auch für Häufigkeitsberechnungen durch Turingmaschinen funktioniert. Dass Ω eine boolesche Algebra ist, wurde bereits von Kummer und Stephan bewiesen [KS95b, Theorem 2.3]. Deren Beweis lässt sich mit einigen Anpassungen auf reguläre Häufigkeitsberechnungen übertragen.[2] Sie kommen zu folgendem Ergebnis:

Lemma 4.21 *Seien $k \geq 1$ und $L, K \in \mathrm{REG}(1,k)$. Dann ist $L \cup K \in \mathrm{REG}(1,N)$ für ein $N \geq 1$.*

Für die Bestimmung des N benötigen wir ein kombinatorisches Lemma von Beigel.

Lemma 4.22 (Beigel [Bei87, Lemma 4.8.9]) *Sei $X \subseteq \mathbb{B}^n$ und $1 \leq k \leq n$. Wenn alle Projektionen von X auf k Komponenten ungleich $\mathbb{B}^k$ sind, dann ist $|X| \leq S(n,k)$, wobei $S(n,k)$ definiert ist als*

$$S(n,k) := \sum_{i=0}^{k-1} \binom{n}{i}.$$

Der Wert von N in Lemma 4.21 ist nun der kleinstmögliche Wert, für den $S(2N, 2k-1) < 2^N$ gilt (dieser existiert, da $S(2N, 2k-1)$ ein Polynom in N vom Grad $2k-1$ ist). Diese Abschätzung liefert in der Regel ein deutlich kleineres N als das durch das Ramsey-Theorem gewonnene in unserem Beweis.

Eine deutlich bessere Schranke stammt von Tantau:

[2] Kummer und Stephan führen ihren Beweis über die Tatsache, dass Ω unter btt-Reduktionen (*bounded truth-table-Reduktionen*, siehe z. B. [HO02]) abgeschlossen ist; dieser Reduktionsbegriff lässt sich aber nicht ohne weiteres auf reguläre Häufigkeitsberechnungen übertragen.

Lemma 4.23 (Tantau, pers. Kommunikation)
Seien $L, K \subseteq \Sigma^*$ zwei Sprachen aus $\mathrm{REG}(1,k)$. Dann gilt:

(1) $L \cup K \in \mathrm{REG}(1,(2k)^2)$ für alle $k \geq 1$.

(2) Für jedes $\varepsilon > 0$ existiert eine Konstante $k_0 \geq 1$ derart, dass $L \cup K \in \mathrm{REG}(1, \lceil (2k)^{1+\varepsilon} \rceil)$ für alle $k \geq k_0$.

Der Beweis beruht auf einer feineren Abschätzung mit Hilfe sog. *Pools* (das sind Mengen potentieller charakteristischer Vektoren, siehe z. B. [Nic99]) und verwendet ebenfalls Beigels kombinatorisches Lemma 4.22.

Die Frage nach dem minimalen N, für das die Vereinigung zweier $(1,n)$-erkennbarer Sprachen $(1,N)$-erkennbar ist, ist nach wie vor offen. Wir können jedoch eine untere Schranke beweisen, die sehr nahe an die obere Schranke aus Lemma 4.23 (2) herankommt.

Satz 4.24 *Für alle $k \geq 2$ gibt es $(1,k)$-erkennbare Sprachen, deren Vereinigung nicht $(1,2k-2)$-erkennbar ist. Insbesondere gibt es also Sprachen $L, K \in \mathrm{REG}(1,2)$ mit $L \cup K \notin \mathrm{REG}(1,2)$.*

Beweis: Wir werden zeigen, dass es für jedes $\ell \geq 2$ Wörter $\alpha_1, \ldots, \alpha_\ell \in \mathbb{B}^\omega$ so gibt, dass für die Vereinigung der zugehörigen Pfadsprachen gilt:

$$V := \bigcup_{i=i}^{\ell} A_{\alpha_i} \notin \mathrm{REG}(1,\ell).$$

Andererseits wissen wir nach den Ausführungen in Beispiel 4.20, dass die Vereinigung von $\ell \geq 2$ Pfadsprachen stets in $\mathrm{REG}(1,\ell+1)$ liegt.

Daraus folgt die Aussage des Satzes, denn für $k \geq 2$ können wir schließen, dass $L = A_{\alpha_1} \cup \ldots \cup A_{\alpha_{k-1}}$ und $K = A_{\alpha_k} \cup \ldots \cup A_{\alpha_{2k-2}}$ jeweils $(1,k)$-erkennbar sind, $L \cup K$ jedoch im allgemeinen nicht $(1,2k-2)$-erkennbar sein kann.

Sei also $\ell \geq 2$. Wir benötigen zunächst einige Definitionen ($1 \leq i \leq \ell$, $1 \leq j$):

- Die $\alpha_i \in \mathbb{B}^\omega$ werden sich aus Blöcken $B_{i,j}$ zusammensetzen:

  $$B_{i,j} := (0^{j \cdot 2^i} 1^{j \cdot 2^i})^{2^{\ell-i}}$$

  Man beachte, dass für alle $1 \leq i \leq \ell$ die Größe des j-ten Blockes

  $$2 \cdot j \cdot 2^i \cdot 2^{\ell-i} = j \cdot 2^{\ell+1}$$

  beträgt, also nicht von i abhängt. Diese Größe nennen wir $s_j := j \cdot 2^{\ell+1}$.

- Damit definieren wir α_i, $1 \leq i \leq \ell$, als

$$\alpha_i := B_{i,1}B_{i,2}B_{i,3}\cdots$$

Wie oben ist V die Vereinigung der A_{α_i}, $1 \leq i \leq \ell$. Im Gegensatz zur Aussage des Satzes nehmen wir an, dass es einen ℓ-DFA A gibt, der die Sprache V $(1,\ell)$-erkennt. Sei q_0 der Startzustand und r die Anzahl der Zustände von A. Wir definieren weiter:

- Die Eingabe $(u_1, \ldots, u_{2k-2})$ als

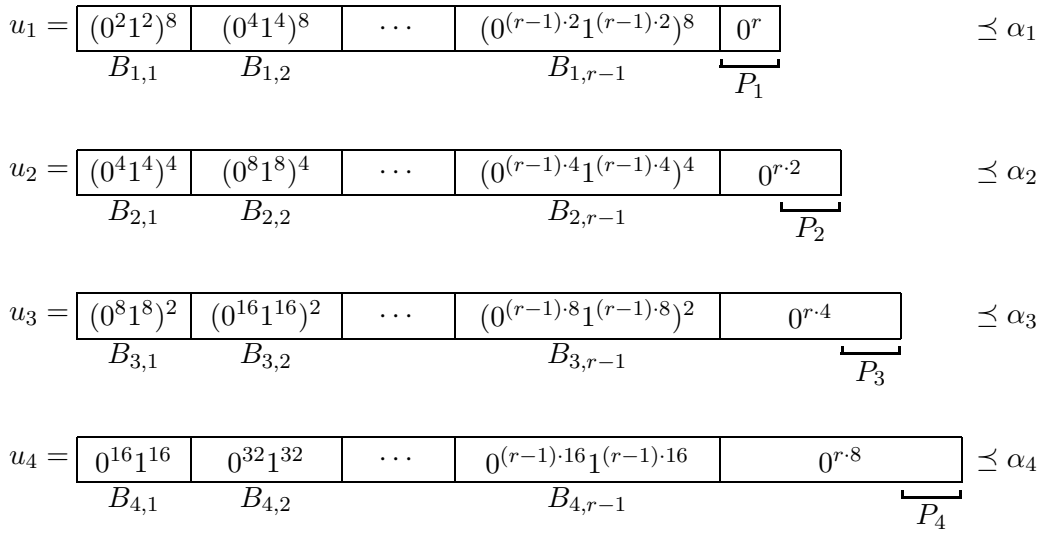
$$u_i := B_{i,1}B_{i,2}B_{i,3}\cdots B_{i,r-1}0^{r\cdot 2^{i-1}} \preceq \alpha_i$$

- Pumpbereiche P_i als Intervalle (mit $S = \sum_{j=1}^{r-1} s_j$):

$$\begin{array}{lcl} P_1 & = & [S+1, S+r] \\ P_i & = & [S + r\cdot 2^{i-2} + 1, S + r\cdot 2^{i-1}] \quad \text{für } i \geq 2 \end{array}$$

Unser Ziel wird sein, innerhalb der Bereiche P_i einen Teil von u_i „aufzupumpen“, um die Zugehörigkeit zu V zu verändern, den dabei erreichten Zustand aber invariant zu lassen. Da in allen ℓ Komponenten gleichzeitig gepumpt wird, verschieben sich beim Pumpen innerhalb eines Bereiches P_i die Intervallgrenzen aller P_j mit $j > i$ entsprechend nach rechts — diese Anpassung der Intervallgrenzen werden wir nur in Gedanken vornehmen (für $j < i$ verändern sich die Grenzen nicht, da der Pumpbereich P_i hinter dem Ende der Wörter $u_1, \ldots, u_{i-1}$ liegt).

Die folgende Abbildung veranschaulicht obige Definitionen (für $\ell = 4$):



Da $u_i \preceq \alpha_i$ für alle $1 \leq i \leq \ell$ gilt, ist $\chi_L^{(\ell)}(u_1, \ldots, u_\ell) = 1^\ell$. Sei $(b_1, \ldots, b_\ell)$ die Ausgabe des Automaten A. Diese enthält mindestens eine richtige Antwort, also mindestens eines der b_i ist 1.

Wir modifizieren der Reihe nach die Eingabewörter u_1 bis u_ℓ.

Im Falle $b_i = 0$ verändern wir die Eingabe nicht. Wir definieren $v_i := u_i$, und es gilt $v_i \preceq \alpha_i$.

Für $b_i = 1$ betrachten wir den Pumpbereich P_i. Dieser hat mindestens die Länge r, es gibt also ein Teilstück

$$(\underbrace{\$, \ldots, \$}_{i-1 \text{ mal}}, \underbrace{0, \ldots, 0}_{\ell-i+1 \text{ mal}})^t$$

mit $1 \leq t \leq r$, das auf einer Schleife in A liegt. (Mehrmaliges) Wiederholen dieses Teilstücks ändert am erreichten Zustand also nichts. Wir wiederholen dieses Teilstück nun gerade so oft, bis das i-te Wort auf mehr als $r \cdot 2^i$ Nullen endet. Das in der i-ten Zeile neu entstandene Wort nennen wir v_i, und es gilt $v_i \not\preceq \alpha_i$.

Wörter u_j mit $j < i$ werden durch das Pumpen nicht beeinträchtigt, da der Pumpbereich P_i hinter dem Ende dieser Wörter liegt.

Wörter u_j mit $j > i$ werden dadurch allerdings verändert. Wir behaupten aber, dass sie nach wie vor alle Präfixe der zugehörigen α_j sind. Dies sieht man leicht, denn u_i wird maximal so stark aufgepumpt, bis die Anzahl der Nullen (zusammen mit allen bisher eingefügten Nullen) zwischen $r \cdot 2^i + 1$ und $r \cdot 2^i + r$ liegt, d. h. es wurden maximal $r \cdot 2^{i-1} + r \leq r \cdot 2^i$ Nullen eingefügt. Für u_j müssten aber mehr als $r \cdot 2^{j-1} \geq r \cdot 2^i$ Nullen angehängt werden, damit es nicht mehr Präfix von α_j ist.

Wir haben jetzt eine neue Eingabe $(v_1, \ldots, v_\ell)$ konstruiert, die folgende Eigenschaften erfüllt:

(1) $q_0 \cdot (v_1, \ldots, v_\ell) = q_0 \cdot (u_1, \ldots, u_\ell)$, und

(2) für alle $1 \leq i \leq \ell$ gilt: $v_i \not\preceq \alpha_i \Leftrightarrow b_i = 1$.

Wegen (1) ist die Ausgabe von A auf $(v_1, \ldots, v_\ell)$ nach wie vor $(b_1, \ldots, b_\ell)$, und wegen (2) ist keine der gegebenen Antworten richtig. Dies ist ein Widerspruch zu unserer Annahme, dass A die Sprache V $(1, \ell)$-erkennen kann. Also gilt $V \notin \mathrm{REG}(1, \ell)$. □

Ein weiteres interessantes Beispiel zum booleschen Abschluss der Klasse FREQ-REG sind die sog. Intervallsprachen.

Beispiel 4.25 (Intervallsprachen) *Intervallsprachen* sind Sprachen, die aus den Links-Schnitten aus Beispiel 4.2 durch Vereinigung und Komplementbildung hervorgehen.

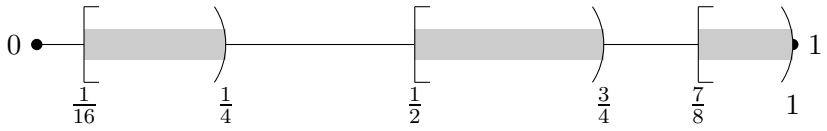
Seien $x_1, \ldots, x_k \in \mathbb{B}^\omega$ mit $\text{bin}(x_1) < \ldots < \text{bin}(x_k)$ und $n_i := \text{bin}(x_i)$, $1 \leq i \leq k$. Außerdem sei $n_{k+1} := 1$. Dann ist die Intervallsprache $I_{x_1,\ldots,x_k}$ definiert als

$$I_{x_1,\ldots,x_k} = \bigcup_{i=1}^{\lceil \frac{k}{2} \rceil} \{w \in \mathbb{B}^* \mid n_{2i-1} \leq \text{bin}(w) < n_{2i}\}.$$

Wir veranschaulichen diese Definition an einem konkreten Beispiel mit $k = 5$:

$$\begin{array}{ll} x_1 = 00010^\omega & n_1 = 1/16 \\ x_2 = 010^\omega & n_2 = 1/4 \\ x_3 = 10^\omega & n_3 = 1/2 \\ x_4 = 110^\omega & n_4 = 3/4 \\ x_5 = 1110^\omega & n_5 = 7/8 \end{array}$$

Dann besteht $I_{x_1,\ldots,x_5}$ aus allen Wörtern $w \in \mathbb{B}^*$, für die $\text{bin}(w)$ in einem der Intervalle $[1/16, 1/4)$, $[1/2, 3/4)$ oder $[7/8, 1)$ liegt:



Wir zeigen, dass eine Intervallsprache $I_{x_1,\ldots,x_k}$ stets in $\text{REG}(1, k+1)$ liegt.

Sei $(u_1, \ldots, u_{k+1})$ die Eingabe für den zu definierenden $(k+1)$-DFA A. Der Automat A bestimmt die Größenbeziehungen der u_i zueinander, was er während des Lesens der Eingabe bewerkstelligen kann. Wir nehmen jetzt ohne Einschränkung an, dass $\text{bin}(u_1) < \ldots < \text{bin}(u_{k+1})$ gilt. Dann gibt A den Vektor $(0, 1, 0, 1, \ldots, 0, 1\,(, 0))$ aus. Wir unterscheiden zwei Fälle:

(1) Alle Eingaben liegen in unterschiedlichen Intervallen (was nur möglich ist, falls $n_1 > 0$ ist), d. h. mit $n_0 := 0$ gilt für alle $1 \leq i \leq k+1$: $u_i \in [n_{i-1}, n_i)$. Dann gibt A auf allen Eingaben die richtige Antwort.

(2) Andernfalls liegen zwei Eingaben im gleichen Intervall, d. h. es existieren zwei Eingaben u_i, u_{i+1} $(1 \leq i \leq k)$ und ein Intervall $[n_{j-1}, n_j)$ $(1 \leq j \leq k+1)$ mit $u_i, u_{i+1} \in [n_{j-1}, n_j)$. Da A auf diesen beiden Eingaben je eine 0 und eine 1 vergibt, ist genau eine der beiden Antworten richtig.

$\diamond$

Wir beschließen dieses Kapitel mit dem Abschluss von FREQ-REG unter markierter Vereinigung (zum Abschluss von Ω unter markierter Vereinigung siehe [KS95b]).

Satz 4.26 *Seien $L \in \mathrm{REG}(1, \ell)$ und $K \in \mathrm{REG}(1, k)$. Dann ist die markierte Vereinigung $L \oplus K \in \mathrm{REG}(1, \ell + k - 1)$.*

Beweis: Für jede Eingabe $(x_1, \ldots, x_{\ell+k-1})$ gilt, dass mindestens ℓ der Wörter mit 0 oder mindestens k der Wörter mit 1 beginnen. Durch den ℓ-DFA für L bzw. den k-DFA für K werden wir in beiden Fällen garantieren können, dass eine von ℓ bzw. eine von k Eingaben richtig beantwortet wird.

Ein endlicher Automat A für $L \oplus K$ stellt nun nach Lesen des ersten Bits aller x_i fest, welcher der beiden Fälle vorliegt. Angenommen, es gibt mindestens ℓ mit 0 beginnender Eingabewörter. Dann simuliert A den zugehörigen ℓ-DFA auf den ersten ℓ dieser Eingaben. Als Ausgabe wird auf diesen Wörtern die Ausgabe des ℓ-DFA übernommen, auf allen anderen Eingaben ist die Ausgabe beliebig. $\square$

Kapitel 5

Zusammenfassung und Ausblick

5.1 Zusammenfassung

In dieser Arbeit haben wir verschiedene Aspekte von Häufigkeitsberechnungen untersucht. Der Schwerpunkt lag auf solchen Berechnungen, die durch endliche Automaten durchgeführt werden können. In diesem Bereich konnten wir zahlreiche Ergebnisse präsentieren, die ein besseres Verständnis (m, n)-erkennbar Sprachen ermöglichen.

Zunächst zeigten wir, dass sich Trakhtenbrots Resultat für Häufigkeitsberechnungen durch Turingmaschinen auf endliche Automaten überträgt (wenn mehr als die Hälfte der Antworten korrekt sind, sind die erkannten Sprachen bereits regulär). Die Gültigkeit dieses Resultats war offen, nachdem sich Kinbers Beweis aus den 70er Jahren als fehlerhaft herausgestellt hatte.

Anschließend konnten wir zeigen, dass es für separierbare Sprachen kein vergleichbares Ergebnis geben kann: zu jedem vorgegebenen $q < 1$ gibt es Sprachen, die nicht rekursiv separierbar sind, sich aber (m, n)-separieren lassen mit Frequenz $m/n > q$.

Schließlich haben wir die Klasse FREQ-REG der (m, n)-erkennbaren Sprachen genauer untersucht. Zum einen konnten wir eine erste strukturelle Eigenschaft dieser Sprachen nachweisen und diese im sog. *Iterationskriterium* festhalten. Dieses ermöglicht — ähnlich dem bekannten Pumping-Lemma — für viele konkrete Sprachen den Nachweis, dass diese nicht zu FREQ-REG gehören.

Desweiteren haben wir Abschlusseigensschaften von FREQ-REG untersucht. Wir konnten zeigen, dass diese Klasse eine boolesche Algebra ist, dass sich bei der Vereinigung zweier $(1, n)$-erkennbarer Sprachen aber in der Regel das n fast auf den doppelten Wert vergrößern muss.

5.2 Offene Probleme

Einige der Ergebnisse, die in dieser Arbeit dargestellt wurden, sind nicht optimal gelöst. So ist konnte zwar gezeigt werden, dass es zu jeder vorgegebenen Frequenz $q < 1$ nicht rekursiv separierbare Sprachen gibt, die (m, n)-separierbar sind mit $m/n > q$. Trotzdem ist unklar, wie nah man für festes n an die Frequenz 1 herankommen kann. Interessant wäre die Frage, ob aus der $(n-1, n)$-Separierung zweier Sprachen für $n \geq 3$ bereits folgt, dass die Sprachen rekursiv separierbar sind. Es wäre auch denkbar, dass diese Folgerung allgemein nicht gilt, für Separierungen durch endliche Automaten jedoch zutrifft.

Für zwei $(1, n)$-erkennbare Sprachen L und K ist ebenfalls nicht bekannt, wie der optimale Wert N aussieht, für den $L \cup K \in \mathrm{REG}(1, N)$ garantiert werden kann. Wir wissen zwar, dass für jedes $\varepsilon > 0$ ein Schwellwert n_0 derart existiert, dass der Wert von N für alle $n \geq n_0$ im Bereich $2n - 2 \leq N \leq \lceil (2n)^{1+\varepsilon} \rceil$ liegt. Es ist nicht klar, ob sich die obere, die untere oder gar beide Schranken weiter verbessern lassen.

Das *Inklusions-* und das *Äquivalenzproblem* für die Klassen $\mathrm{REG}(m, n)$ haben wir in dieser Arbeit überhaupt nicht untersucht. Hierzu sind keinerlei Ergebnisse bekannt. Es wäre interessant zu wissen, ob sich die Situation ähnlich wie bei den allgemeinen Häufigkeitsberechnungen darstellt (das alle Klassen für $1 \leq m \leq n/2$ verschieden sind). Ebenso gibt es unseres Wissens nach keinerlei Untersuchungen dieser Probleme für separierbare Sprachen.

Für die allgemeinen Häufigkeitsberechnungen wurde in Abschnitt 1.3 ja erwähnt, dass das Inklusionsproblem vollständig gelöst ist. Leider lässt sich dieses Problem bislang nur mittels kombinatorischer Methoden lösen, womit man in der Praxis schnell an die Grenzen heutiger (und auch zukünftiger) Rechenkapazitäten stößt. Deshalb wäre es durchaus wünschenswert, ein anderes, leichter zu überprüfendes Kriterium für die Inklusionsbeziehungen herzuleiten.

Im Bereich ressourcenbeschränkter Häufigkeitsberechnungen ist die momentan wohl spannendste offene Frage die *Hinrichs-Wechsung-Vermutung*: für Häufigkeitsberechnungen in Polynomialzeit mit konstanter Fehlerzahl $d \geq 1$ ist bekannt, dass sich die Klassen $(1, 1+d)\mathrm{P}$ bis $(2^d, 2^d+d)\mathrm{P}$ alle unterscheiden, es wird jedoch vermutet, dass

$$(2^d, 2^d + d)\mathrm{P} = (2^d + 1, 2^d + 1 + d)\mathrm{P} = \ldots$$

gilt. Die Vermutung konnte bislang nur für $d \leq 3$ bewiesen werden [Hin94, Min04, HM05].

Schließlich gibt es auch auf verwandten Gebieten offene Fragen, von denen eine im Zusammenhang mit endlichen Automaten hier gestellt werden soll: gilt das *Kardinalitätstheorem* auch für endliche Automaten? Die Vermutung lautet konkret: Wenn eine Sprache L durch einen endlichen Automaten erkannt werden kann, der auf jeder Eingabe von n paarweise verschiedenen Wörtern $x_1, \ldots, x_n$ einen der $n+1$ möglichen Werte für $\#_L^{(n)}(x_1, \ldots, x_n)$ ausschließen kann, dann ist L bereist regulär. Auch hierfür konnten bislang nur zwei Spezialfälle bewiesen werden: der Fall $n = 2$ und das sog. *eingeschränkte Kardinalitätsproblem*, bei dem der endliche Automat bei jeder Eingabe einen der Werte 0 und n ausschließt [Tan02b].

Literaturverzeichnis

[ADH03] Holger Austinat, Volker Diekert, and Ulrich Hertrampf, *A structural property of regular frequency computations*, Theoretical Computer Science **292** (2003), no. 1, 33–43.

[ADHP00] Holger Austinat, Volker Diekert, Ulrich Hertrampf, and Holger Petersen, *Regular frequency computations*, Proceedings of the RIMS Symposium on Algebraic Systems, Formal Languages and Computation (Kyoto, Japan), 2000, pp. 35–42.

[ADHP05] ———, *Regular frequency computations*, Theoretical Computer Science **330** (2005), no. 1, 15–21.

[Arm97] M. A. Armstrong, *Basic toplogy*, Undergraduate Texts in Mathematics, Springer-Verlag, Berlin, 1997.

[Bei87] Richard Beigel, *Query-limited reducibilities*, Ph.D. thesis, Stanford University, 1987.

[BGK96] Richard Beigel, William I. Gasarch, and Efim B. Kinber, *Frequency computation and bounded queries*, Theoretical Computer Science **163** (1996), no. 1–2, 177–192.

[Bou66] Nicolas Bourbaki, *General Topology (Part 1)*, Elements of Mathematics, Hermann, Paris, 1966.

[Dëg81] Alexandr N. Dëgtev, *On (m, n)–computable sets*, Algebraic Systems (D. I. Moldavanskij, ed.), Ivanovo Gos. University, Ivanovo, 1981, (Russian; MR 86b:03049), pp. 88–99.

[Gas91] William I. Gasarch, *Bounded queries in recursion theory: a survey*, Proceedings of the Sixth Annual Structure in Complexity Theory Conference, University of Chicago, Illinois, 1991, pp. 62–78.

[GRS80] Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer, *Ramsey Theory*, Wiley Interscience Series in Discrete Mathematics, Wiley, New York, 1980.

[GS99] William I. Gasarch and Frank Stephan, *A techniques oriented survey of bounded queries*, Models and Computability: Invited Papers from Logic Colloquium '97, European Meeting of the Association for Symbolic Logic, Leeds, July 1997 (S. Barry Cooper and John K. Truss, eds.), Cambridge University Press, 1999.

[Hem04] Lane A. Hemaspaandra, *Advice for Semifeasible Sets and the Complexity-Theoretic Cost(lessness) of Algebraic Properties*, Tech. Report TR-2004-843, University of Rochester, 2004, (based on a talk given at the 6th Workshop on Descriptional Complexity of Formal Systems, London, Ontario, Canada).

[Hin94] Maren Hinrichs, *Häufigkeitsberechnungen*, Master's thesis, Friedrich-Schiller-Universität Jena, 1994.

[Hin02] ———, *Frequency computations under certain resource bounds*, Ph.D. thesis, Fakultät für Mathematik und Informatik der Friedrich-Schiller-Universität Jena, 2002.

[HJRW97] Lane A. Hemaspaandra, Zhigen Jiang, Jörg Rothe, and Osamu Watanabe, *Polynomial-Time Multi-Selectivity*, Journal of Universal Computer Science **3** (1997), no. 3, 197–229, `http://www.jucs.org/jucs_3_3/polynomial_time`.

[HKO92] Valentina S. Harizanov, Martin Kummer, and James C. Owings, *Frequency computations and the cardinality theorem*, Journal of Symbolic Logic **57** (1992), no. 2, 682–687.

[HM05] Ulrich Hertrampf and Christoph Minnameier, *Ressource Bounded Frequency Computations with Three Errors*, Tech. Report 2005/3, University of Stuttgart, 2005.

[HO02] Lane A. Hemaspaandra and Mitsunori Ogihara, *The Complexity Theory Companion*, Texts in Theoretical Computer Science, Springer-Verlag, Berlin, 2002.

[HT03] Lane A. Hemaspaandra and Leen Torenvliet, *Theory of Semi-Feasible Algorithms*, Monographs in Theoretical Computer Science, Springer-Verlag, Berlin, 2003.

[HW97] Maren Hinrichs and Gerd Wechsung, *Time bounded frequency computations*, Information and Computation **139** (1997), no. 2, 234–257.

[Joc66] Carl G. Jockusch Jr., *Reducibilities in recursive function theory*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1966.

[Joc68] ———, *Semirecursive sets and positive reducibility*, Transactions of the AMS **131** (1968), no. 2, 420–436.

[Kin72] Efim B. Kinber, *Frequency calculations of general recursive predicates and frequency enumeration of sets*, Soviet Mathematics Doklady **13** (1972), 873–876.

[Kin74] ———, *On frequency-enumerable sets*, Algebra i Logika **13** (1974), 398–419, (Russian; English translation in *Algebra and Logic*, 13:226–237, 1974).

[Kin75a] ———, *Frequency-computable functions and frequency-enumerable sets*, Ph.D. thesis, Riga, 1975, (Russian).

[Kin75b] ———, *On frequency real-time computations*, Latviĭ. gosudarst. Univ., učenye Zapiski **233** (1975), 174–182, (Russian; MR 58: 3679, Zbl 335:02023).

[Kin76] ———, *Frequency computations in finite automata*, Kibernetika **2** (1976), 7–15, (Russian; English translation in *Cybernetics*, 12:179–187, 1976).

[Kön36] Dénes König, *Theorie der endlichen und unendlichen Graphen: Kombinatorische Topologie der Streckenkomplexe*, Akademische Verlagsanstalt Leipzig, 1936.

[KS91] Martin Kummer and Frank Stephan, *Some aspects of frequency computation*, Tech. Report 21/91, Fakultät für Informatik, Universität Karlsruhe, October 1991.

[KS95a] ———, *The power of frequency computation*, Proceedings of the Tenth International Congress on Fundamentals of Computation Theory (FCT 1995) (Horst Reichel, ed.), LNCS 969, 1995, pp. 323–332.

[KS95b] ———, *Recursion theoretic properties of frequency computation and bounded queries*, Information and Computation **120** (1995), 59–77.

[Kum92] Martin Kummer, *A proof of Beigel's cardinality conjecture*, Journal of Symbolic Logic **57** (1992), no. 2, 677–681.

[McN61] Robert McNaughton, *The theory of automata, a survey*, Advances in Computers **2** (1961), 379–421.

[McN95] Timothy McNicholl, *The inclusion problem for generalized frequency classes*, Ph.D. thesis, George Washington University, Washington D.C., 1995.

[Min04] Christoph F. Minnameier, *Polynomialzeitbeschränkte Häufigkeitsberechnungen mit 3 Fehlern*, Master's thesis, Universität Stuttgart, 2004, Diplomarbeit Nr. 2114.

[Nic99] Arfst Nickelsen, *Polynomial time partial information classes*, Ph.D. thesis, Technische Universität Berlin, 1999.

[Owi89] James C. Owings, *A cardinality version of Beigel's nonspeedup theorem*, Journal of Symbolic Logic **54** (1989), no. 3, 761–767.

[Per90] Dominique Perrin, *Finite automata*, Handbook of Theoretical Computer Science (Jan van Leeuwen, ed.), vol. B, Elsevier, 1990, pp. 1–57.

[Rab74] Michael O. Rabin, *Theoretical impediments to artifical intelligence*, Information Processing 74 — Proceedings of the IFIP Congress 74, Stockholm (Jack L. Rosenfeld, ed.), 1974, pp. 615–619.

[Ros60] Gene F. Rose, *An extended notion of computability*, Abstracts of the International Congress for Logic, Methodology, and Philosophy of Science (Stanford, California), 1960, p. 14.

[RU63] Gene F. Rose and Joseph S. Ullian, *Approximation of functions on the integers*, Pacific Journal of Mathematics **13** (1963), 693–701.

[Sch65] Marcel Paul Schützenberger, *On finite monoids having only trivial subgroups*, Information and Control **8** (1965), no. 2, 190–194.

[Sel79] Alan L. Selman, *P-selective Sets, Tally Languages, and the Behavior of Polynomial Time Reducibilities on NP*, Mathematical Systems Theory **13** (1979), 55–65.

[Tan02a] Till Tantau, *Comparing verboseness for finite automata and Turing machines*, 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS'02), Antibes - Juan les Pins, France (Helmut Alt and Afonso Ferreira, eds.), LNCS 2285, 2002, pp. 465–476.

[Tan02b] ———, *Towards a cardinality theorem for finite automata*, Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS 2002), Warsaw, Poland (Krzysztof Diks and Wojciech Rytter, eds.), LNCS 2420, 2002, pp. 625–636.

[Tan03] ———, *On Structural Similarities of Finite Automata and Turing Machine Enumerability Classes*, Ph.D. thesis, Technische Universität Berlin, 2003.

[Tra63] Boris A. Trakhtenbrot, *On the frequency computation of functions*, Algebra i Logika **2** (1963), 25–32, (Russian).

[Tra73] ———, *Frequency computations*, Proceedings of the Steklov Institute of Mathematics, vol. 133, 1973, pp. 223–234.

[Tra77] ———, *Frequency algorithms and computations*, Proceedings of the 6th Symposium on Mathematical Foundations of Computer Science (MFCS 1977), Tatranská Lomnica, Czechoslovakia (Jozef Gruska, ed.), LNCS 53, 1977, pp. 148–161.

Index