Prof. Dr. Jürgen Dassow
und
Dr. Bianca Truthe

Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik

KOMPLEXITÄT VON BESCHREIBUNGEN

Vorlesungsmanuskript

Magdeburg, April – Juli 2008

Vorwort

Die Komplexitätstheorie als eines der wesentlichen Teilgebiete der Theoretischen Informatik beschäftigt sich mit der Frage, welche Ressourcen zum Lösen gewisser Aufgaben erforderlich sind. In der Grundvorlesung zur Theoretischen Informatik wurden bereits die Zeit- und Raumkomplexität behandelt, d. h. die Frage nach der Zeit und dem Speicherplatz, die erforderlich sind, um eine Aufgabe/ein Problem zu lösen. Ein weiteres Komplexitätsmaß kann darin gesehen werden, wie groß die Beschreibung des zum Lösen verwendeten Algorithmus ist. Ist die Beschreibung durch ein Programm gegeben, so kann als Größe des Programms die Anzahl der Befehle im Programm (Anzahl der „lines of code“) angesehen werden; wird der Algorithmus durch eine TURING-Maschine beschrieben, so kann z. B. die Anzahl der Zustände der Maschine als Größe interpretiert werden. Die Beschreibungskomplexität beschäftigt sich allgemein mit der Größe von Beschreibungen von Objekten. So können z. B. Boolesche Funktionen durch Schaltkreise, reguläre Sprachen durch endliche Automaten und kontextfreie Sprachen durch kontextfreie Grammatiken beschrieben werden; die zugehörige Größe der Beschreibung kann dann z. B. die Anzahl der Gatter des Schaltkreises, die Anzahl der Zustände des Automaten oder die Anzahl der Regeln in der Grammatik sein. In allen Fällen stellen sich die Fragen nach
– der Größe der kleinsten Beschreibung eines gegebenen Objektes,
– der Minimierung von Beschreibungen für ein Objekt,
– einem Algorithmus zum Auffinden einer minimalen Beschreibung
usw. Wir behandeln diese Probleme für Boolesche Funktionen, formale Sprachen und einige verwandte Strukturen. Als Größen werden die oben genannten Parameter sowie einige Variationen davon betrachtet.

Außerdem wird eine kurze Einführung in die Kolmogorov-Komplexität gegeben, bei der kürzeste Beschreibungen von Wörtern durch Algorithmen Gegenstand der Untersuchung sind.

Zum Verständnis der Vorlesung sind die üblicherweise in den Grundvorlesungen zur *Mathematik* (für Informatiker), *Algorithmen und Datenstrukturen*, *Logik* und *Theoretischen Informatik* vermittelten Kenntnisse erforderlich. Dies betrifft insbesondere die Grundbegriffe der Analysis (Grenzwerte, Landau-Symbolik etc.), der Graphentheorie, der regulären und kontextfreien Grammatiken, der endlichen Automaten, Entscheidbarkeit und NP-Vollständigkeit von Berechnungsproblemen und die Ermittlung der Berechnungskomplexität von Algorithmen. Wir verweisen hier auf die Lehrbücher [2], [18], [20], [22], [3], [16], [19].

Jeder Abschnitt der Vorlesung endet mit einigen Übungsaufgaben. Sie sollen dem Leser die Möglichkeit geben, sein Wissen zu überprüfen.

Wir danken Dr. Bernd Reichel, Dr. Claudia Krull sowie Herrn Ivo Rössling, Herrn Ronny Harbich und Herrn Andreas Zöllner für die sorgfältige Durchsicht älterer Versionen des Skriptes, wodurch sich die Qualität deutlich verbessert hat.

Jürgen Dassow / Bianca Truthe April – Juli 2008

Inhaltsverzeichnis

Kapitel 1

Komplexität Boolescher Funktionen

1.1. Boolesche Funktionen

Wir geben zuerst unsere Notation für Boolesche Funktionen und einige Fakten über Boolesche Funktionen an, die – bis auf die Notation – bereits aus der Vorlesung zur Logik bekannt sind.

Unter einer *Booleschen Funktion* verstehen wir eine Funktion, deren Definitionsbereich eine kartesische Potenz von $\{0,1\}$ und deren Wertevorrat eine Teilmenge einer kartesischen Potenz von $\{0,1\}$ sind. Daher gibt es zu jeder Booleschen Funktion f natürliche Zahlen $n \geq 0$ und $m \geq 0$ so, dass

$$f : \{0,1\}^n \longrightarrow \{0,1\}^m$$

gilt. Wir schreiben dann auch

$$f(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m).$$

Eine einfache Darstellungsform von Booleschen Funktionen sind Tabellen. Dabei geben wir in der linken Spalte die n-Tupel des Definitionsbereichs und in der rechten Spalte das zugehörigen m-Tupel von Werten an. Da jedes n-Tupel $(x_1, x_2, \ldots, x_{n-1}, x_n)$ eineindeutig der Zahl

$$x_1 \cdot 2^{n-1} + x_2 \cdot 2^{n-2} + \cdots + x_{n-1} \cdot 2^1 + x_n$$

entspricht, können wir die Tupel der linken Spalte so ordnen, dass die ihnen zugeordneten Zahlen der Reihe nach $0, 1, \ldots, 2^n - 1$ sind. Tabelle 1.1 veranschaulicht diese Art der Darstellung.

Wir setzen

$$B_{n,m} = \{\, f \mid f \text{ bildet } \{0,1\}^n \text{ in } \{0,1\}^m \text{ ab} \,\},$$

und da für uns vor allem Funktionen mit einem Wertebereich in $\{0,1\}$ von Interesse sein werden, führen wir noch

$$B_n = B_{n,1} \quad \text{für } n \geq 1,$$
$$B = \bigcup_{n \geq 1} B_n$$

als abkürzende Bezeichnungen ein.

x_1	x_2	x_3	$\ldots$	x_{n-2}	x_{n-1}	x_n	$f(x_1, x_2, x_3, \ldots, x_{n-2}, x_{n-1}, x_n)$
0	0	0	$\ldots$	0	0	0	$f(0,0,0,\ldots,0,0,0)$
0	0	0	$\ldots$	0	0	1	$f(0,0,0,\ldots,0,0,1)$
0	0	0	$\ldots$	0	1	0	$f(0,0,0,\ldots,0,1,0)$
0	0	0	$\ldots$	0	1	1	$f(0,0,0,\ldots,0,1,1)$
0	0	0	$\ldots$	1	0	0	$f(0,0,0,\ldots,1,0,0)$
			$\vdots$				$\vdots$
0	1	1	$\ldots$	1	1	1	$f(0,1,1,\ldots,1,1,1)$
1	0	0	$\ldots$	0	0	0	$f(1,0,0,\ldots,0,0,0)$
			$\vdots$				$\vdots$
1	1	1	$\ldots$	1	1	0	$f(1,1,1,\ldots,1,1,0)$
1	1	1	$\ldots$	1	1	1	$f(1,1,1,\ldots,1,1,1)$

Tabelle 1.1: Tabellarische Darstellung einer Booleschen Funktion

Lemma 1.1 *Es gibt genau* 2^{2^n} *Funktionen in* B_n.

Beweis: Wie wir bereits oben festgestellt haben, gibt es (wegen der Entsprechung zu den Zahlen $0, 1, \ldots, 2^n - 1$) genau 2^n mögliche Tupel in $\{0,1\}^n$. Jedem dieser Tupel kann genau ein Wert aus $\{0,1\}$ zugeordnet werden. Es handelt sich also um Variationen von 2 Elementen mit Wiederholung zu je 2^n Werten. Die aus der Kombinatorik bekannte Formel für die Anzahl solcher Variationen (mit Wiederholung) liefert 2^{2^n}. □

Tabelle 1.2 gibt alle Booleschen Funktionen aus B_1 mit ihren Bezeichnungen an.

	Identität	Negation	Konstante 0	Konstante 1
x	x	$\overline{x}$	k_0	k_1
0	0	1	0	1
1	1	0	0	1

Tabelle 1.2: Funktionen aus B_1

Für $x \in \{0,1\}$ setzen wir

$$x^0 = \overline{x} \quad \text{und} \quad x^1 = x,$$

womit sich

$$0^0 = 1^1 = 1 \quad \text{und} \quad 0^1 = 1^0 = 0$$

ergeben.

Tabelle 1.3 gibt einige Funktionen aus B_2 an.

Interpretieren wir x_1 und x_2 als Dualziffern, so gibt die Parity-Funktion als Ergebnis die letzte Ziffer der Dualdarstellung von $x_1 + x_2$. Bezüglich der Multiplikation ergibt sich stets wieder eine Ziffer; diese wird gerade durch den Wert von $x_1 \wedge x_2$ angegeben. Daher verwenden wir für die Konjunktion auch die Bezeichnung $x_1 \cdot x_2$.

		Konjunktion AND-Funktion	Disjunktion OR-Funktion	Parity-Funktion XOR-Funktion
x_1	x_2	$x_1 \wedge x_2$	$x_1 \vee x_2$	$x_1 \oplus x_2$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Tabelle 1.3: Einige Funktionen aus B_2
Die Bezeichnungen AND, OR, XOR kommen aus der englischen Sprache, da diese Funktionen umgangssprachlich durch *und*, das einschließende *oder* und das ausschließende *oder* (engl.: exclusive or) widergespiegelt werden.

Wir geben nun einige bekannte leicht zu verifizierende Beziehungen für die gegebenen Funktionen an; der formale Beweis bleibt dem Leser überlassen.

$$x_1 \wedge x_2 = \overline{\overline{x_1} \vee \overline{x_2}} \text{ und } x_1 \vee x_2 = \overline{\overline{x_1} \wedge \overline{x_2}}, \tag{1.1}$$

$$\overline{\overline{x}} = x, \quad \overline{x} = x \oplus 1, \quad x \oplus x = 0, \quad x \wedge x = x \vee x = x, \tag{1.2}$$

$$x_1 \circ x_2 = x_2 \circ x_1 \text{ und } (x_1 \circ x_2) \circ x_3 = x_1 \circ (x_2 \circ x_3) \text{ für } \circ \in \{\wedge, \vee, \oplus\}, \tag{1.3}$$

$$(x_1 \oplus x_2) \cdot x_3 = (x_1 \cdot x_3) \oplus (x_2 \cdot x_3). \tag{1.4}$$

Die Relationen aus (1.1) heißen DE MORGANsche Regeln, (1.3) enthält die üblichen Kommutativ- und Assoziativgesetze, und (1.4) ist das Distributivgesetz. Dabei haben wir die Relationen nur jeweils für zwei bzw. drei Variablen formuliert; mittels vollständiger Induktion können diese leicht auf beliebige Anzahlen von Variablen erweitert werden. (Man beachte, dass wir die Konjunktion in (1.1) und (1.3) mit dem Symbol $\wedge$, in (1.4) dagegen mit dem Symbol $\cdot$ bezeichnet haben.)

Für ein Tupel $\underline{a} = (a_1, a_2, \ldots, a_n) \in \{0,1\}^n$ definieren wir die Funktionen

$$m_{\underline{a}} : \{0,1\}^n \to \{0,1\} \quad \text{und} \quad s_{\underline{a}} : \{0,1\}^n \to \{0,1\}$$

vermöge

$$m_{\underline{a}}(x_1, x_2, \ldots, x_n) = x_1^{a_1} \wedge x_2^{a_2} \wedge \cdots \wedge x_n^{a_n}$$

und

$$s_{\underline{a}}(x_1, x_2, \ldots, x_n) = x_1^{\overline{a_1}} \vee x_2^{\overline{a_2}} \vee \cdots \vee x_n^{\overline{a_n}}.$$

Offenbar gelten für $m_{\underline{a}}$ und $s_{\underline{a}}$ die Beziehungen

$$\begin{aligned} m_{\underline{a}}(x_1, x_2, \ldots, x_n) = 1 &\text{ genau dann, wenn } x_i^{a_i} = 1 \text{ für } 1 \le i \le n \\ &\text{genau dann, wenn } x_i = a_i \text{ für } 1 \le i \le n \\ &\text{genau dann, wenn } (x_1, x_2, \ldots, x_n) = (a_1, a_2, \ldots, a_n) \end{aligned}$$

und

$$\begin{aligned} s_{\underline{a}}(x_1, x_2, \ldots, x_n) = 0 &\text{ genau dann, wenn } x_i^{\overline{a_i}} = 0 \text{ für } 1 \le i \le n \\ &\text{genau dann, wenn } x_i \ne \overline{a_i} \text{ für } 1 \le i \le n \\ &\text{genau dann, wenn } (x_1, x_2, \ldots, x_n) = (a_1, a_2, \ldots, a_n). \end{aligned}$$

Satz 1.2 *Für jede Boolesche Funktion $f \in B_n$ gelten*

$$a)\quad f(x_1, x_2, \ldots, x_n) = \bigvee_{\underline{a} \in f^{-1}(1)} m_{\underline{a}}(x_1, x_2, \ldots, x_n),$$

$$b)\quad f(x_1, x_2, \ldots, x_n) = \bigwedge_{\underline{a} \in f^{-1}(0)} s_{\underline{a}}(x_1, x_2, \ldots, x_n),$$

$$c)\quad f(x_1, x_2, \ldots, x_n) = \bigoplus_{\{i_1,\ldots,i_m\} \subseteq \{1,\ldots,n\}} a_{i_1 i_2 \ldots i_m} x_{i_1} x_{i_2} \ldots x_{i_m}$$

für gewisse $a_{i_1 i_2 \ldots i_m} \in \{0, 1\}$.

Beweis: a) Für eine Teilmenge S von $\{0,1\}^n$ betrachten wir die Funktion

$$f_S(x_1, x_2, \ldots, x_n) = \bigvee_{\underline{a} \in S} m_{\underline{a}}(x_1, x_2, \ldots, x_n).$$

Dann gilt unter Beachtung obiger Relationen

$$\begin{aligned} f_S(x_1, x_2, \ldots, x_n) = 1 &\text{ genau dann, wenn } m_{\underline{a}}(x_1, x_2, \ldots, x_n) = 1 \text{ für ein } \underline{a} \in S \\ &\text{genau dann, wenn } (x_1, x_2, \ldots, x_n) = \underline{a} \text{ für ein } \underline{a} \in S. \end{aligned}$$

Nun folgt die Aussage von a) aus der Definition von $f^{-1}(1)$.

b) ergibt sich analog.

c) Aufgrund der DE MORGANschen Regeln ergibt sich

$$\begin{aligned} s_{\underline{a}}(x_1, x_2, \ldots, x_n) &= x_1^{\overline{a_1}} \vee x_2^{\overline{a_2}} \vee \cdots \vee x_n^{\overline{a_n}} \\ &= \overline{x_1^{a_1} \wedge x_2^{a_2} \wedge \cdots \wedge x_n^{a_n}} \\ &= (x_1^{a_1} \wedge x_2^{a_2} \wedge \cdots \wedge x_n^{a_n}) \oplus 1 \\ &= (x_1^{a_1} \cdot x_2^{a_2} \cdot \cdots \cdot x_n^{a_n}) \oplus 1 \\ &= \big((x_1 \oplus b_1) \cdot (x_2 \oplus b_2) \cdot \cdots \cdot (x_n \oplus b_n)\big) \oplus 1, \end{aligned}$$

wobei

$$b_i = \begin{cases} 0 & \text{für } x_i = a_i \\ 1 & \text{für } x_i \neq a_i \end{cases} \qquad \text{für } 1 \leq i \leq n$$

gesetzt wurde. Durch Ausmultiplizieren der so geänderten Bestandteile $s_{\underline{a}}(x_1, x_2, \ldots, x_n)$ aus Teil b) entsprechend dem obigen Distributivgesetz erhalten wir die zu beweisende Aussage. □

Die Darstellungen aus Satz 1.2 a) und b) sind spezielle *disjunktive* bzw. *konjunktive Normalformen* von f. Die Darstellung in c) wird als *Polynomdarstellung* von f bezeichnet. Die Polynomdarstellung einer Funktion f ist eindeutig bestimmt. Dies sieht man wie folgt: Es gibt genau 2^n Teilmengen $\{i_1, i_2, \ldots, i_m\}$ von $\{1, 2, \ldots, n\}$. In jeder Darstellung verwenden wir eine Auswahl dieser Mengen. Folglich gibt es 2^{2^n} Polynome (mit n Variablen). Aus der Übereinstimmung der Anzahl von Booleschen Funktionen (siehe Lemma 1.1) und der Polynome folgt die Eindeutigkeit der Darstellung.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Tabelle 1.4: Funktion f aus Beispiel 1.3

Beispiel 1.3 *Wir betrachten die durch die Tabelle 1.4 gegebene dreistellige Boolesche Funktion f.*

Nach Konstruktion erhalten wir bei Verwendung von $\cdot$ anstelle von $\wedge$ die disjunktive Normalform

$$\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 \vee \overline{x_1} \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot x_2 \cdot x_3$$

sowie die konjunktive Normalform

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \cdot (\overline{x_1} \vee x_2 \vee x_3) \cdot (\overline{x_1} \vee x_2 \vee \overline{x_3}).$$

Entsprechend obiger Konstruktion erhalten wir für die Polynomdarstellung

$$\begin{aligned}
&f(x_1, x_2, x_3) = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \cdot (\overline{x_1} \vee x_2 \vee x_3) \cdot (\overline{x_1} \vee x_2 \vee \overline{x_3}) \\
&= \overline{(\overline{x_1} \cdot x_2 \cdot x_3)} \cdot \overline{(x_1 \cdot \overline{x_2} \cdot \overline{x_3})} \cdot \overline{(x_1 \cdot \overline{x_2} \cdot x_3)} \\
&= (\overline{x_1} \cdot x_2 \cdot x_3 \oplus 1) \cdot (x_1 \cdot \overline{x_2} \cdot \overline{x_3} \oplus 1) \cdot (x_1 \cdot \overline{x_2} \cdot x_3 \oplus 1) \\
&= ((x_1 \oplus 1) \cdot x_2 \cdot x_3 \oplus 1) \cdot (x_1 \cdot (x_2 \oplus 1) \cdot (x_3 \oplus 1) \oplus 1) \cdot (x_1 \cdot (x_2 \oplus 1) \cdot x_3 \oplus 1) \\
&= (x_1x_2x_3 \oplus x_2x_3 \oplus 1) \cdot (x_1x_2x_3 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1 \oplus 1) \cdot (x_1x_2x_3 \oplus x_1x_3 \oplus 1) \\
&= x_1x_2x_3 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1 \oplus 1.
\end{aligned}$$

Übungsaufgaben

1. Beweisen Sie die in (1.1), (1.2), (1.3) und (1.4) angegebenen Beziehungen.

2. Beweisen Sie, dass $(x_1 \wedge x_2) \vee (x_1 \oplus x_2) = (x_1 \wedge x_2) \oplus (x_1 \oplus x_2)$ gilt.

3. Ermitteln Sie eine konjunktive Normalform, eine disjunktive Normalform und die Polynomdarstellung der dreistelligen Booleschen Funktion f, die genau dann den Wert 1 annimmt, wenn genau zwei der Argumentwerte mit 1 belegt sind.

4. Ermitteln Sie eine konjunktive Normalform, eine disjunktive Normalform und die Polynomdarstellung der dreistelligen Booleschen Funktion f, die genau dann den Wert 1 annimmt, wenn genau zwei der Argumentwerte mit 0 belegt sind.

1.2. Schaltkreise und Komplexitätsmaße

Wir definieren nun Schaltkreise und geben für diese einige Komplexitätsmaße an. Dann ordnen wir jedem Schaltkreis eine Boolesche Funktion zu und übertragen die Komplexitätsmaße auf Boolesche Funktionen.

Definition 1.4 *Es sei $\mathcal{S}$ eine endliche Menge von Funktionen aus B. Ein (n,m)-Schaltkreis über $\mathcal{S}$ ist ein (knoten-)markierter, gerichteter und azyklischer Graph S mit folgenden Eigenschaften:*

- *n paarweise verschiedene Knoten von S sind mit $x_1, x_2, \ldots, x_n$ markiert,*
- *die mit einem x_i, $1 \leq i \leq n$, markierten Knoten haben keinen Vorgänger,*
- *die restlichen Knoten von S sind mit Elementen aus $\mathcal{S}$ markiert,*
- *die mit einem $f \in \mathcal{S} \cap B_k$ markierten Knoten haben genau k Vorgänger,*
- *m Knoten von S sind zusätzlich noch mit $y_1, y_2, \ldots, y_m$ markiert.*

Die mit x_i, $1 \leq i \leq n$, bzw. y_j, $1 \leq j \leq m$, markierten Knoten heißen Eingangs- bzw. Ausgangsknoten. Die mit einem Element aus $\mathcal{S}$ markierten Knoten werden auch Gatter genannt.

Geht man davon aus, dass zur praktischen Realisierung eines Schaltkreises ein gewisser Platz erforderlich ist, so liegen folgende Maße für Schaltkreise nahe: der gesamte Platzbedarf, der etwa der Anzahl der Knoten entspricht, und die lineare Ausdehnung in einer Richtung. Dies wird durch die beiden Maße der folgenden Definition widergespiegelt.

Definition 1.5 *Es sei S ein (n,m)-Schaltkreis über $\mathcal{S}$.*

i) *Wir definieren die Größe (oder Komplexität) $C(S)$ von S als die Anzahl der mit Elementen aus $\mathcal{S}$ markierten Knoten von S.*

ii) *Für einen Knoten g von S definieren wir die Tiefe $D(g)$ als die maximale Länge eines Weges von einem mit x_i, $1 \leq i \leq n$, markierten Knoten nach g.*

iii) *Unter der Tiefe $D(S)$ des Schaltkreises S verstehen wir die maximale Tiefe der Knoten von S.*

Der letzte Teil der Definition lässt sich formal als

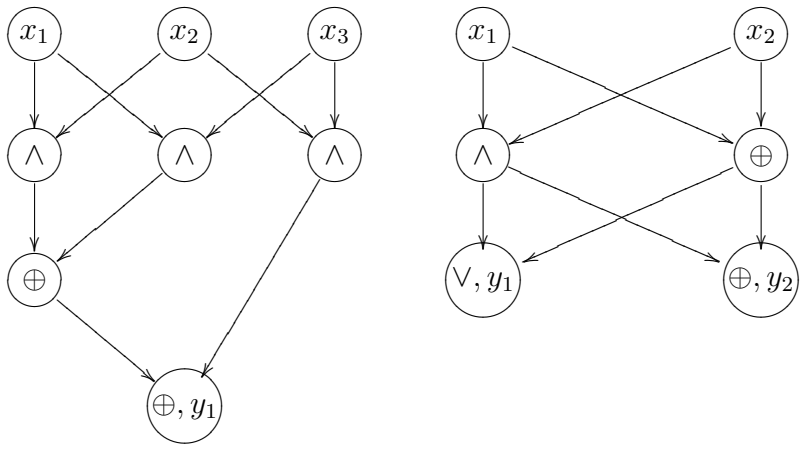
$$D(S) = \max\{D(g) \mid g \text{ ist Knoten von } S\}$$

schreiben. Entsprechend der Definition ist die Tiefe $D(S)$ die maximale Länge eines Weges in S, da aufgrund der Definition alle Wege in S, die nicht in einem mit x_i, $1 \leq i \leq n$, markierten Knoten beginnen, verlängert werden können.

Beispiel 1.6 *Wir betrachten die Menge*

$$\mathcal{S} = \{\wedge, \vee, \oplus\}$$

und die Graphen S_1 und S_2 aus Abbildung 1.1.



S_1 S_2

Abbildung 1.1: Beispiele für Schaltkreise

Dann sind S_1 ein $(3,1)$-Schaltkreis über $\mathcal{S}$ und S_2 ein $(2,2)$-Schaltkreis über $\mathcal{S}$. Jedoch kann S_1 auch als Schaltkreis über $\{\wedge, \oplus\}$ aufgefasst werden. Für die Größe und Tiefe der Schaltkreise ergeben sich die Werte

$$C(S_1) = 5 \quad \textit{und} \quad C(S_2) = 4$$

und

$$D(S_1) = 3 \quad \textit{und} \quad D(S_2) = 2.$$

Schaltkreise sollen uns als Beschreibungen Boolescher Funktionen dienen. Daher ist es notwendig, eine Beziehung zwischen Schaltkreisen und Booleschen Funktionen herzustellen. In der für uns wichtigen Richtung wird dies durch die folgende Definition geleistet.

Definition 1.7 *Es sei S ein (n,m)-Schaltkreis über $\mathcal{S} \subseteq B$.*

i) *Dann definieren wir die Boolesche Funktion f_g, die in einem Knoten g von S induziert wird induktiv über die Tiefe des Knotens wie folgt:*

- *Es sei $D(g) = 0$. Dann ist g mit einer Variablen x_i, $1 \leq i \leq n$, markiert, und wir setzen*

$$f_g(x_1, x_2, \ldots, x_n) = x_i.$$

- *Es sei $D(g) > 0$. Dann ist g mit einer Funktion $f \in \mathcal{S}$ markiert. Ist $f \in B_k$, sind $g_1, g_2, \ldots, g_k$ die Vorgängerknoten von g und sind $f_{g_1}, f_{g_2}, \ldots, f_{g_k}$ die in den Vorgängerknoten induzierten Funktionen, so setzen wir*

$$f_g(x_1, x_2, \ldots, x_n) = f(f_{g_1}(x_1, \ldots, x_n), f_{g_2}(x_1, \ldots, x_n), \ldots, f_{g_k}(x_1, \ldots, x_n)).$$

ii) *Sind $h_1, h_2, \ldots, h_m$ die Knoten von S, die zusätzlich mit $y_1, y_2, \ldots, y_m$ markiert sind, so berechnet S die Funktion $f : \{0,1\}^n \to \{0,1\}^m$, die durch*

$$f(x_1, x_2, \ldots, x_n) = (f_{h_1}(x_1, \ldots, x_n), f_{h_2}(x_1, \ldots, x_n), \ldots, f_{h_m}(x_1, \ldots, x_n))$$

definiert ist.

Beispiel 1.8 *Wir setzen Beispiel 1.6 fort und berechnen der Reihe nach die von den Knoten induzierten Funktionen und damit auch die von S_1 und S_2 berechneten Booleschen Funktionen.*

Bei S_1 werden in den drei Knoten der Tiefe 1 (von links nach rechts betrachtet) die folgenden Funktionen induziert:

$$x_1 \wedge x_2, \quad x_1 \wedge x_3, \quad x_2 \wedge x_3.$$

Damit werden im Knoten der Tiefe 2 und in dem der Tiefe 3

$$(x_1 \wedge x_2) \oplus (x_1 \wedge x_3)$$

und

$$f_1(x_1, x_2, x_3) = (x_1 \wedge x_2) \oplus (x_1 \wedge x_3) \oplus (x_2 \wedge x_3) \tag{1.5}$$

induziert. Entsprechend unserer Definition wird die in (1.5) gegebene Funktion von S_1 berechnet.

Bei S_2 werden in den Knoten der Tiefe 1 die Funktionen

$$x_1 \wedge x_2 \quad \textit{und} \quad x_1 \oplus x_2$$

und in den Knoten der Tiefe 2 die Funktionen

$$(x_1 \wedge x_2) \vee (x_1 \oplus x_2) \quad \textit{und} \quad (x_1 \wedge x_2) \oplus (x_1 \oplus x_2)$$

induziert. Man rechnet leicht nach, dass die beiden letztgenannten mit $x_1 \vee x_2$ übereinstimmen (siehe Übungsaufgabe 2 zu Abschnitt 1.1). Folglich berechnet S_2 die Boolesche Funktion

$$f_2(x_1, x_2) = (x_1 \vee x_2, x_1 \vee x_2). \tag{1.6}$$

Beispiel 1.9 *Wir betrachten nun die umgekehrte Aufgabe. Wir geben eine Boolesche Funktion vor und bestimmen dazu einen Schaltkreis, der diese Funktion berechnet. Wir wollen einen Schaltkreis konstruieren, der im Wesentlichen die Addition von drei Dualziffern x_1, x_2, x_3 vornimmt. Offenbar gilt wegen $0 \leq x_i \leq 1$, $1 \leq i \leq 3$, auch $0 \leq \sum_{i=1}^{3} x_i \leq 3$. In Dualdarstellung hat das Ergebnis also (maximal) zwei Ziffern, und wenn wir führende Nullen erlauben sogar genau zwei Dualziffern. Daher werden wir einen Schaltkreis mit zwei markierten Knoten y_1 und y_2 konstruieren, bei dem in Dualschreibweise*

$$y_1 y_2 = \sum_{i=1}^{3} x_i$$

gilt. Damit ergeben sich folgende Aussagen für y_1 und y_2:

- *$y_1 = 1$ gilt genau dann, wenn mindestens zwei der Ziffern x_1, x_2, x_3 den Wert 1 annehmen,*
- *$y_2 = 1$ gilt genau dann, wenn alle drei oder genau eine der Ziffern x_1, x_2, x_3 den Wert 1 annehmen.*

Somit gelten

$$y_1 = (x_1 \wedge x_2) \oplus (x_1 \wedge x_3) \oplus (x_2 \wedge x_3) \quad \textit{und} \quad y_2 = x_1 \oplus x_2 \oplus x_3.$$

Damit kann y_1 wegen (1.5) durch den Schaltkreis S_1 realisiert werden, und wir haben eine Ergänzung durch die Gatter zur Realisierung von y_2 vorzunehmen. Hieraus folgt, dass der Schaltkreis S_3 aus Abbildung 1.2 der Größe 7 und der Tiefe 3 das Gewünschte leistet.

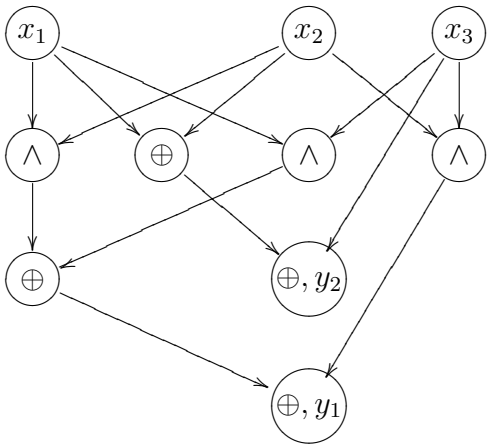


Abbildung 1.2: Schaltkreis S_3 zur Addition von drei Dualziffern

Wir übertragen nun unsere Komplexitätsmaße Größe und Tiefe auf Boolesche Funktionen, indem wir über alle die Funktion berechnenden Schaltkreise minimieren.

Definition 1.10 *Für eine Boolesche Funktion $f : \{0,1\}^n \to \{0,1\}^m$ und eine endliche Teilmenge $\mathcal{S}$ von B definieren wir die Größe $C_{\mathcal{S}}(f)$ und die Tiefe $D_{\mathcal{S}}(f)$ von f bezüglich $\mathcal{S}$ durch*

$$C_{\mathcal{S}}(f) = \min\{\, C(S) \mid S \textit{ berechnet } f \textit{ und ist Schaltkreis über } \mathcal{S} \,\}$$

und

$$D_{\mathcal{S}}(f) = \min\{\, D(S) \mid S \textit{ berechnet } f \textit{ und ist Schaltkreis über } \mathcal{S} \,\}.$$

Beispiel 1.11 *Wir betrachten die Funktionen f_1 und f_2, die durch die Schaltkreise aus Beispiel 1.6 über $\{\vee, \wedge, \oplus\}$ berechnet werden. Wegen der Minimierung bei der Komplexitätsbestimmung erhalten wir direkt*

$$C_{\mathcal{S}}(f_1) \leq C(S_1) = 5 \quad \textit{und} \quad D_{\mathcal{S}}(f_1) \leq D(S_1) = 3$$

sowie

$$C_{\mathcal{S}}(f_2) \leq C(S_2) = 4 \quad \textit{und} \quad D_{\mathcal{S}}(f_2) \leq D(S_2) = 2.$$

Aufgrund von (1.6) berechnet aber der Schaltkreis S_4 aus Abbildung 1.3 ebenfalls die Funktion f_2. Damit gilt

$$C_{\mathcal{S}}(f_2) = C(S_4) = 1 \quad \textit{und} \quad D_{\mathcal{S}}(f_2) = D(S_4) = 1,$$

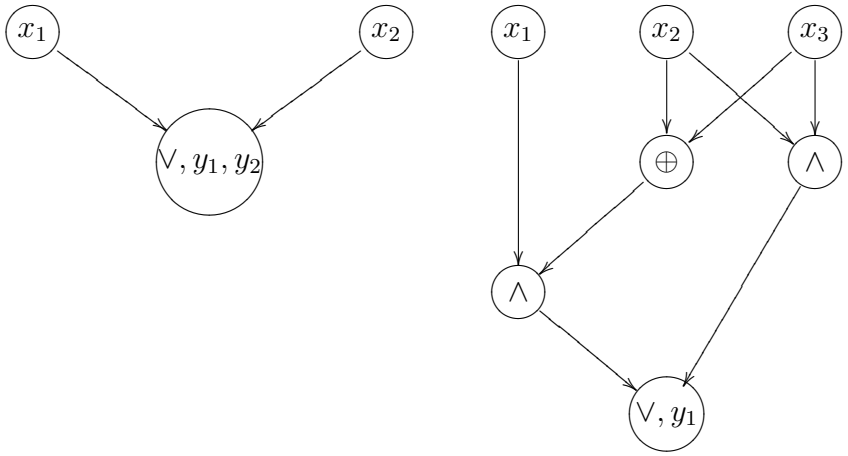


Abbildung 1.3: Schaltkreise S_4 und S_5

da S_4 offenbar der minimale Schaltkreis zur Berechnung von f_2 ist.

Der Schaltkreis S_5 berechnet wegen (1.5) und

$$f_1 = (x_1 \wedge x_2) \oplus (x_1 \wedge x_3) \oplus (x_2 \wedge x_3) = (x_1 \wedge (x_2 \oplus x_3)) \vee (x_2 \oplus x_3)$$

ebenfalls f_1. Damit gilt zumindest

$$C_{\mathcal{S}}(f_1) \leq C(S_5) = 4 \quad \textit{und} \quad D_{\mathcal{S}}(f_1) \leq D(S_5) = 3.$$

Man kann sich durch Durchmustern aller kleineren Schaltkreise über $\mathcal{S}$ leicht davon überzeugen, dass S_5 sogar ein sowohl bezüglich Größe als auch Tiefe minimaler Schaltkreis zur Berechnung von f_1 ist, womit sogar

$$C_{\mathcal{S}}(f_1) = 4 \quad \textit{und} \quad D_{\mathcal{S}}(f_1) = 3$$

gelten.

Wir wollen nun noch ein weiteres Komplexitätsmaß einführen, das durch Grenzen der Schaltkreisherstellung bedingt ist. Während der Eingangsgrad (hier auch *fan-in*[1] genannt) eines Knotens in einem Schaltkreis bei Markierung mit x_i, $1 \leq i \leq n$, Null und bei Markierung mit $h \in \mathcal{S}$ die Stelligkeit von h und damit beschränkt ist, haben wir hinsichtlich des Ausgangsgrads (auch *fan-out* genannt) bisher keine Beschränkung vorgenommen. Aus technischen Gründen muss aber bei der praktischen Realisierung eines Schaltkreises eine Beschränkung des Ausgangsgrads bei den Gattern vorgenommen werden.

Definition 1.12

i) *Wir sagen, dass ein Schaltkreis S fan-out-k-beschränkt ist, wenn der fan-out eines jeden Knotens von S höchstens k ist.*

[1]Diese Bezeichnung kommt vom englischen Wort *fan* für *Fächer*, da die eingehenden (und analog auch die ausgehenden) Kanten direkt beim Knoten anschaulich einen Fächer bilden.

ii) *Für eine Boolesche Funktion* $f : \{0,1\}^n \to \{0,1\}^m$ *und eine endliche Teilmenge* $\mathcal{S}$ *von* B *definieren wir* $C_{k,\mathcal{S}}(f)$ *als das Minimum der Größen* $C(S)$*, wobei das Minimum über alle Schaltkreise über* $\mathcal{S}$ *genommen wird, die* f *berechnen und fan-out-*k*-beschränkt sind.*

Für Boolesche Funktionen spielt noch die Länge als ein weiteres (von Schaltkreisen unabhängiges) Komplexitätsmaß eine wichtige Rolle. Dieses kann wie folgt definiert werden. Zuerst definieren wir in der üblichen Weise induktiv Formeln (auch Ausdrücke genannt) über $\mathcal{S}$ durch die drei folgenden Bedingungen:

1. Es sei f eine n-stellige Funktion aus $\mathcal{S}$, $n \geq 1$, so ist $f(x_1, x_2, \ldots, x_n)$ eine Formel über $\mathcal{S}$.

2. Ist f eine n-stellige Funktion aus $\mathcal{S}$ und sind $H_1, H_2, \ldots, H_n$ Formeln über $\mathcal{S}$ oder Variable, so ist auch $f(H_1, H_2, \ldots, H_n)$ eine Formel über $\mathcal{S}$.

3. Formeln entstehen nur durch endlich oft maliges Anwenden von 2. aus 1.

Als *Länge* $L_{\mathcal{S}}(F)$ einer Formel F definieren wir nun die Anzahl der in F vorkommenden Funktionssymbole.

Die Formel (1.5) kann als eine Darstellung von f_1 durch eine Formel über $\{\oplus, \wedge\}$ interpretiert werden. Deren Länge beträgt 5, da $\wedge$ bzw. $\oplus$ in der Formel dreimal bzw. zweimal auftreten.

Offensichtlich ist aber die Länge $L_{\mathcal{S}}$ mit dem Maß $C_{1,\mathcal{S}}$ identisch. Dies folgt daraus, dass jede Formel einfach in einen Schaltkreis umgesetzt werden kann, bei dem jeder innere Knoten den fan-out 1 hat, und umgekehrt kann jeder Schaltkreis mit fan-out 1 als Formel repräsiert werden, wobei dann die Anzahl der Funktionssymbole und die Anzahl der Gatter übereinstimmen.

Übungsaufgaben

1. Es sei $\mathcal{S}$ die aus den Funktionen Negation, Konjunktion und Disjunktion sowie der Parity-Funktion bestehende Menge. Ein Schaltkreis S über $\mathcal{S}$ sei durch Abbildung 1.4 gegeben.

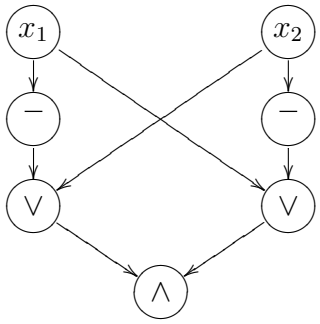


Abbildung 1.4: Schaltkreis für Aufgabe 1 (zu Abschnitt 1.2.)

a) Bestimmen Sie die Werte $C(S)$ und $D(S)$.

b) Beweisen Sie, dass die von S induzierte Funktion f_S genau dann den Wert 1 annimmt, wenn die Argumentwerte für x_1 und x_2 übereinstimmen.
c) Zeigen Sie, dass $C_{\mathcal{S}}(f_S) = D_{\mathcal{S}}(f_S) = 2$ gilt.

2. a) Beweisen Sie, dass für eine Funktion f und eine Menge $\mathcal{S}$ von Funktionen genau dann $D_{\mathcal{S}}(f) = 1$ gilt, wenn $f \in \mathcal{S}$ gilt.
b) Beweisen Sie, dass für eine Funktion f und eine Menge $\mathcal{S}$ von Funktionen genau dann $L_{\mathcal{S}}(f) = 1$ gilt, wenn $f \in \mathcal{S}$ gilt.

3. Ermitteln Sie die Werte $C_{\mathcal{S}}(x_1 \oplus x_2)$, $D_{\mathcal{S}}(x_1 \oplus x_2)$ und $L_{\mathcal{S}}(x_1 \oplus x_2)$, wobei die Menge $\mathcal{S}$ aus der Negation, der Konjunktion und der Disjunktion bestehe.

1.3. Das Vollständigkeitskriterium von Post

Jedem Schaltkreis ist nach dem vorhergehenden Abschnitt eine Boolesche Funktion zugeordnet. In diesem Abschnitt untersuchen wir die Umkehrung: Gibt es zu jeder Booleschen Funktion f auch einen Schaltkreis, der f berechnet?

Die in Satz 1.2 gegebenen Darstellungen von Booleschen Funktionen lassen sich offenbar direkt in Schaltkreise umsetzen. Daher ist jede Funktion durch einen Schaltkreis über

$$\{\overline{\ }, \vee, \wedge\} \quad \text{oder} \quad \{\wedge, \oplus, k_1\}$$

berechenbar.

Andererseits reicht die Konjunktion allein sicher nicht aus, um alle Booleschen Funktionen zu berechnen. Ein Schaltkreis, der nur Konjunktionen enthält, berechnet nämlich eine Funktion f, die $f(0, 0, \ldots, 0) = 0$ erfüllt, da bei einer Eingabe, die nur aus Nullen besteht, jedes AND-Gatter auch eine Null ausgibt.

Definition 1.13 *Eine Menge $\mathcal{S} \subseteq B$ heißt vollständig, falls jede Boolesche Funktion durch einen Schaltkreis über $\mathcal{S}$ berechnet werden kann.*

Entsprechend dieser Definition lässt sich die obige Frage wie folgt formulieren: Wann ist eine Menge $\mathcal{S}$ Boolescher Funktionen vollständig?

Das folgende Lemma gibt ein erstes Vollständigkeitskriterium. Es ist aber nur bedingt brauchbar, da es nur besagt, dass es reicht, wenn wir die Booleschen Funktionen mit einem Ausgabeknoten berechnen können.

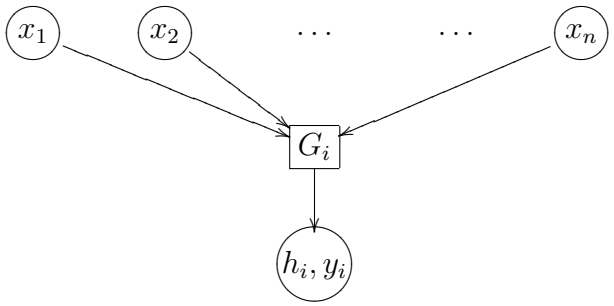
Lemma 1.14 *Eine Menge $\mathcal{S} \subseteq B$ ist genau dann vollständig, wenn jede Boolesche Funktion aus B durch einen Schaltkreis über $\mathcal{S}$ berechnet werden kann.*

Beweis: Wenn es eine Boolesche Funktion f aus B gibt, die nicht durch einen Schaltkreis über $\mathcal{S}$ berechnet werden kann, so ist $\mathcal{S}$ nach Definition nicht vollständig.

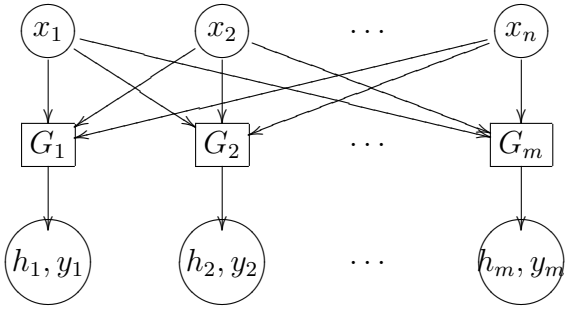
Es sei daher nun jede Funktion aus B berechenbar. Ferner sei f eine Boolesche Funktion aus $B_{n,m}$ für gewisse natürliche Zahlen $n \geq 1$ und $m \geq 1$. Dann gibt es offenbar m Funktionen $h_1, h_2, \ldots, h_m$ aus B derart, dass

$$f(x_1, x_2, \ldots, x_n) = (h_1(x_1, x_2, \ldots, x_n), h_2(x_1, x_2, \ldots, x_n), h_m(x_1, x_2, \ldots, x_n))$$

gilt. Für $1 \leq i \leq m$ sei nun



ein Schaltkreis über $\mathcal{S}$, der im Knoten y_i die Funktion h_i berechnet. Hierbei ist h_i die Funktion mit der der Ausgabeknoten markiert ist, und G_i ist der restliche Schaltkreis (man beachte, dass alle Knoten von G_i (nur) mit Elementen aus $\mathcal{S}$ markiert sind). Dann berechnet der Schaltkreis



offenbar die Funktion f. Damit kann jede Boolesche Funktion f durch einen Schaltkreis über $\mathcal{S}$ berechnet werden. Somit ist $\mathcal{S}$ vollständig. □

Lemma 1.15 *Ist $\mathcal{S}_1$ eine vollständige Menge und ist jede Funktion $f \in \mathcal{S}_1$ durch einen Schaltkreis über $\mathcal{S}_2$ berechenbar, so ist auch $\mathcal{S}_2$ vollständig.*

Beweis: Es sei g eine beliebige Boolesche Funktion. Da $\mathcal{S}_1$ vollständig ist, gibt es einen Schaltkreis S über $\mathcal{S}_1$, der g berechnet. Für eine Funktion $f \in \mathcal{S}_1$ sei S_f ein Schaltkreis über $\mathcal{S}_2$, der f berechnet. S'_f entstehe aus S_f durch Streichen der Eingangsknoten. In S ersetzen wir nun jedes Gatter, das mit $f \in \mathcal{S}_1$ markiert ist, durch S'_f. Der so entstehende Schaltkreis S' enthält (außer den Eingangsknoten) nur Knoten, die mit Elementen aus $\mathcal{S}_2$ markiert sind, und berechnet offenbar auch g. Folglich ist g durch einen Schaltkreis über $\mathcal{S}_2$ berechenbar und damit ist $\mathcal{S}_2$ vollständig. □

Wir wollen nun ein Vollständigkeitskriterium geben, das auf den amerikanischen Mathematiker Emil L. Post (1897–1954) zurückgeht. Zu seiner Formulierung benötigen wir fünf Mengen Boolescher Funktionen. Wir setzen

$$\begin{aligned}
T_0 &= \{f \mid f \in B_n, n \geq 1, f(0,0,\ldots,0) = 0\},\\
T_1 &= \{f \mid f \in B_n, n \geq 1, f(1,1,\ldots,1) = 1\},\\
Lin &= \{f \mid f \in B_n, n \geq 1, f(x_1,x_2,\ldots,x_n) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n\\
&\qquad\qquad \text{für gewisse } a_i \in \{0,1\}, 1 \leq i \leq n\},
\end{aligned}$$

$$Mon = \{f \mid f \in B_n, n \geq 1, f(a_1, a_2, \ldots, a_n) \leq f(b_1, b_2, \ldots, b_n) \text{ für } a_i \leq b_i, 1 \leq i \leq n\},$$
$$Sd = \{f \mid f \in B_n, n \geq 1, f(a_1, a_2, \ldots, a_n) = \overline{f(\overline{a_1}, \overline{a_2}, \ldots, \overline{a_n})}\}.$$

Die Funktionen aus T_0 heißen 0-*bewahrend* und die aus T_1 heißen 1-*bewahrend*. Die Funktionen aus *Lin* und *Mon* werden *linear* bzw. *monoton* genannt. Die zu einer Booleschen Funktion $f \in B_n$ *duale* Funktion f_d definieren wir durch

$$f_d(a_1, a_2, \ldots, a_n) = \overline{f(\overline{a_1}, \overline{a_2}, \ldots, \overline{a_n})}.$$

Entsprechend den DE MORGANschen Regeln ist die zur Konjunktion duale Funktion die Disjunktion und umgekehrt. Offenbar gilt für jede Boolesche Funktion $(f_d)_d = f$, d. h. durch zweifaches Dualisieren entsteht stets die Ausgangsfunktion. Die Funktionen aus *Sd* haben die Eigenschaft, dass sie ihre eigene duale Funktion sind, und heißen daher *selbstdual*.

Da die Negation nicht in T_0, T_1 und *Mon* liegt und die Disjunktion weder in *Sd* noch in *Lin* enthalten ist, sind alle diese fünf Mengen von B verschieden. Das folgende Kriterium kann vom algebraischen Standpunkt aus dahingehend interpretiert werden, dass es sich bei T_0, T_1, *Mon*, *Lin* und *Sd* um die maximalen Teilmengen von B handelt, die von B verschieden sind und bei denen die Komposition von Funktionen nicht aus der Menge führt (man vergleiche Teil i) des Beweises des folgenden Satzes).

Satz 1.16 *Eine Menge $\mathcal{S} \subseteq B$ ist genau dann vollständig, wenn sie in keiner der Mengen T_0, T_1, Lin, Mon und Sd enthalten ist.*

Beweis: Wegen Lemma 1.14 beschränken wir uns auf Funktionen aus B bzw. Schaltkreise mit genau einem als Ausgang markierten Knoten.

i) Wir beweisen zuerst die Notwendigkeit der angegebenen Vollständigkeitsbedingung. Hierfür reicht es zu zeigen, dass für jede Menge $Q \in \{T_0, T_1, Mon, Lin, Sd\}$ und $\mathcal{S} \subseteq Q$ jede durch einen Schaltkreis über $\mathcal{S}$ berechenbare Funktion in Q liegt. Wir zeigen dies durch vollständige Induktion über die Tiefe D der Schaltkreise.

Induktionsanfang $D = 0$. Ein Schaltkreis der Tiefe 0 ist offenbar ein Schaltkreis, bei dem ein Eingangsknoten als Ausgang markiert ist. Hat der Schaltkreis n Eingangsknoten, so wird eine Projektion

$$P_i^n(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) = x_i,$$

$1 \leq i \leq n$, berechnet. Es folgt sofort aus den Definitionen, dass

$$P_i^n \in T_0,\ P_i^n \in T_1,\ P_i^n \in Mon,\ P_i^n \in Lin,\ P_i^n \in Sd$$

gelten.

Induktionsschritt von Tiefe kleiner D auf Tiefe gleich D. Es sei S ein Schaltkreis der Tiefe D. Hat der mit y markierte Ausgangsknoten G die Markierung g, $g \in Q$, und k Vorgängerknoten, in denen (nach Induktionsvoraussetzung) die Funktionen $g_1, g_2, \ldots, g_n \in Q$ berechnet werden, so berechnet S die Funktion

$$f(x_1, x_2, \ldots, x_n) = g(g_1(x_1, \ldots, x_n), g_2(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)).$$

Wir diskutieren nun die fünf Fälle für Q.

Fall 1. $Q = T_0$. Wegen der Induktionsvoraussetzung und wegen $g \in Q$ gelten

$$\begin{aligned} g_i(0,0,\ldots,0) &= 0 \quad \text{für} \quad 1 \leq i \leq k, \\ g(0,0,\ldots,0) &= 0, \end{aligned}$$

und wir erhalten

$$\begin{aligned} f(0,0,\ldots,0) &= g(g_1(0,0,\ldots,0), g_2(0,0,\ldots,0), \ldots, g_k(0,0,\ldots,0)) \\ &= g(0,0,\ldots,0) \\ &= 0, \end{aligned}$$

womit gezeigt ist, dass auch $f \in T_0$ gilt.

Fall 2. $Q = T_1$. Der Beweis von $f \in T_1$ kann analog geführt werden.

Fall 3. $Q = Mon$. In diesem Fall gelten wegen der Induktionsvoraussetzung für beliebige Tupel $(a_1, a_2, \ldots, a_n)$ und $(b_1, b_2, \ldots, b_n)$ mit $a_j \leq b_j$, $1 \leq j \leq n$ und $1 \leq i \leq k$

$$c_i = g_i(a_1, a_2, \ldots, a_n) \leq g_i(b_1, b_2, \ldots, b_n) = d_i.$$

Ferner ergibt sich dann wegen $g \in Mon$ noch

$$g(c_1, c_2, \ldots, c_k) \leq g(d_1, d_2, \ldots, d_k),$$

woraus

$$\begin{aligned} f(a_1, a_2, \ldots, a_n) &= g(g_1(a_1, \ldots, a_n), g_2(a_1, \ldots, a_n), \ldots, g_k(a_1, \ldots, a_n)) \\ &= g(c_1, c_2, \ldots, c_k) \\ &\leq g(d_1, d_2, \ldots, d_k) \\ &= g(g_1(b_1, \ldots, b_n), g_2(b_1, \ldots, b_n), \ldots, g_k(b_1, \ldots, b_n)) \\ &= f(b_1, b_2, \ldots, b_n) \end{aligned}$$

folgt. Dies bedeutet aber gerade, dass $f \in Mon$ gilt.

Fall 4. $Q = Lin$. In diesem Fall gibt es wegen der Induktionsvoraussetzung $a_{i,j} \in \{0,1\}$, $1 \leq i \leq k$, $0 \leq j \leq n$, derart, dass

$$g_i(x_1, x_2, \ldots, x_n) = a_{i,0} \oplus a_{i,1}x_1 \oplus a_{i,2}x_2 \oplus \cdots \oplus a_{i,n}x_n$$

gelten. Ferner gibt es wegen $g \in Lin$ noch $a_i \in \{0,1\}$, $0 \leq i \leq k$, derart, dass

$$g(x_1, x_2, \ldots, x_k) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_kx_k$$

gilt, woraus

$$\begin{aligned} f(a_1, a_2, \ldots, a_n) &= a_0 \oplus \bigoplus_{i=1}^{k} a_i(a_{i,0} \oplus \bigoplus_{j=1}^{n} a_{i,j}x_j) \\ &= (a_0 \oplus \bigoplus_{i=1}^{k} a_i a_{i,0}) \oplus (\bigoplus_{i=1}^{k} a_i a_{i,1})x_1 \oplus (\bigoplus_{i=1}^{k} a_i a_{i,2})x_2 \oplus \cdots \oplus (\bigoplus_{i=1}^{k} a_i a_{i,n})x_n \\ &= b_0 \oplus b_1x_1 \oplus b_2x_2 \oplus \cdots \oplus b_nx_n \end{aligned}$$

mit

$$b_0 = a_0 \oplus (\bigoplus_{i=1}^{k} a_0 a_{i,0}),$$

$$b_j = \bigoplus_{i=1}^{k} a_i a_{i,j} \quad \text{für} \quad 1 \leq j \leq n$$

folgt. Dies bedeutet aber gerade, dass $f \in Lin$ gilt.

Fall 5. $Q = Sd$. Wir überlassen den Beweis von $f \in Sd$ dem Leser als Übungsaufgabe (siehe Übungsaufgabe 1).

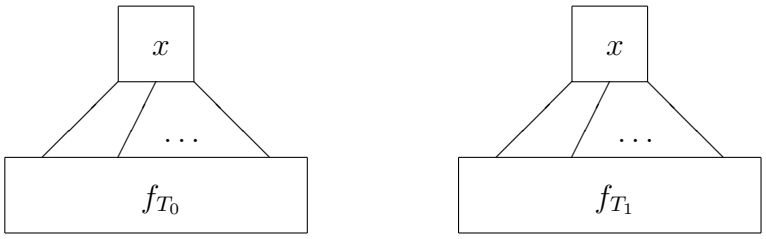
Damit haben wir bewiesen, dass Schaltkreise über einer Teilmenge $\mathcal{S}$ von Q nur Funktionen aus Q berechnen können. Da $Q \subset B$ gilt, kann $\mathcal{S}$ nicht vollständig sein.

ii) Wir beweisen nun die Hinlänglichkeit der Bedingung. Nach Voraussetzung enthält $\mathcal{S}$ Funktionen

$$f_{T_0} \notin T_0,\ f_{T_1} \notin T_1,\ f_{Mon} \notin Mon,\ f_{Lin} \notin Lin,\ f_{Sd} \notin Sd.$$

Wir zeigen, dass mittels dieser Funktionen der Reihe nach die konstanten Funktionen k_0 und k_1 (a), die Negation (b), die Konjunktion (c) und die Disjunktion (d) erzeugt werden können. Wegen der Vollständigkeit von $\{\vee, \wedge, \bar{}\}$ und Lemma 1.15 folgt dann die Vollständigkeit von $\mathcal{S}$.

(a) Für die Funktionen f_{T_0} und f_{T_1} seien f'_{T_0} und f'_{T_1} die Funktionen, die durch die beiden folgenden Schaltkreise berechnet werden, wobei die Ausgangsknoten durch f_{T_0} bzw. f_{T_1} markiert sind und wir für die Knoten ein Kästchen statt eines Kreises verwenden.



Aufgrund ihrer Definition gelten

$$f_{T_0}(0, 0, \ldots, 0) = 1 \quad \text{und} \quad f_{T_1}(1, 1, \ldots, 1) = 0.$$

Wir diskutieren nun die vier Fälle für die Werte von $f_{T_0}(1, 1, \ldots, 1)$ und $f_{T_1}(0, 0, \ldots, 0)$.

Fall 1. $f_{T_0}(1, 1, \ldots, 1) = 1$ und $f_{T_1}(0, 0, \ldots, 0) = 0$. Dann gelten offenbar $f'_{T_0} = k_1$ und $f'_{T_1} = k_0$. Folglich können die beiden konstanten Funktionen durch Schaltkreise über $\mathcal{S}$ berechnet werden.

Die restlichen drei Fälle werden wir zusammen analysieren, nachdem wir erst einmal feststellen, dass in allen diesen Fällen $\{f'_{T_0}, f'_{T_1}\}$ die Negation enthält.

Fall 2. $f_{T_0}(1, 1, \ldots, 1) = 0$ und $f_{T_1}(0, 0, \ldots, 0) = 0$. Dann erhalten wir $f'_{T_0}(x) = \overline{x}$ und erneut $f'_{T_1} = k_0$.

Fall 3. $f_{T_0}(1, 1, \ldots, 1) = 1$ und $f_{T_1}(0, 0, \ldots, 0) = 1$. Dann ergeben sich $f'_{T_1}(x) = \overline{x}$ und $f'_{T_0} = k_1$.

Fall 4. $f_{T_0}(1,1,\ldots,1) = 0$ und $f_{T_1}(0,0,\ldots,0) = 1$. Dann gilt offenbar, dass sowohl f'_{T_0} als auch f'_{T_1} die Negation ist.

Nun gelten

$$x^a = \begin{cases} x, & a = 1, \\ \overline{x}, & a = 0 \end{cases}$$

und wegen $f_{Sd} \notin Sd$

$$f_{Sd}(a_1, a_2, \ldots, a_n) = f_{Sd}(\overline{a_1}, \overline{a_2}, \ldots, \overline{a_n})$$

für gewisse $a_i \in \{0,1\}$, $1 \leq i \leq n$. Für die Funktion

$$g(x) = f_{Sd}(x^{a_1}, x^{a_2}, \ldots, x^{a_n})$$

ergibt sich damit

$$\begin{aligned} g(0) &= f_{Sd}(0^{a_1}, 0^{a_2}, \ldots, 0^{a_n}) \\ &= f_{Sd}(\overline{a_1}, \overline{a_2}, \ldots, \overline{a_n}) \\ &= f_{Sd}(a_1, a_2, \ldots, a_n) \\ &= f_{Sd}(1^{a_1}, 1^{a_2}, \ldots, 1^{a_n}) \\ &= g(1). \end{aligned}$$

Somit ist g eine konstante Funktion und diese wird durch den Schaltkreis aus Abbildung 1.5 berechnet.

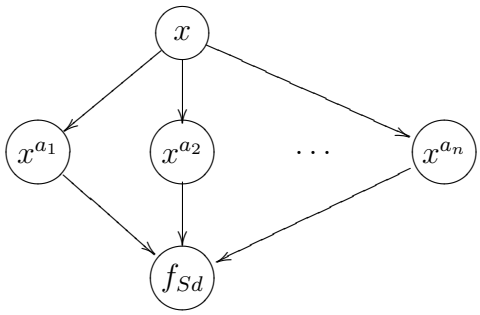


Abbildung 1.5: Schaltkreis, der im Knoten f_{Sd} eine Konstante berechnet (x^1 ist die Identität, und wir verbinden den Eingangsknoten direkt mit f_{Sd}; x^0 ist die Negation)

Folglich können wir eine Konstante und die Negation durch Schaltkreise über $\mathcal{S}$ berechnen. Hieraus ist einfach durch Hintereinandersetzen dieser Schaltkreise einer für die andere Konstante zu gewinnen. Daher sind beide Konstanten durch Schaltkreise über $\mathcal{S}$ berechenbar. (Man beachte, dass die aufwendige Konstruktion mit f_{Sd} eigentlich nur im Fall 4 erforderlich ist, da in den Fällen 2 und 3 bereits $\{f'_{T_0}, f'_{T_1}\}$ die Negation und eine Konstante enthält.)

(b) Wir betrachten die Funktion $f_{Mon} \notin Mon$. Nach Definition gibt es zwei Tupel $(a_1, a_2, \ldots, a_m)$ und $(b_1, b_2, \ldots, b_m)$ mit $a_i \leq b_i$, $1 \leq i \leq m$, und

$$f_{Mon}(a_1, a_2, \ldots, a_m) > f_{Mon}(b_1, b_2, \ldots, b_m). \tag{1.7}$$

Für die Tupel

$$\underline{c_i} = (b_1, b_2, \ldots, b_{i-1}, a_i, a_{i+1}, \ldots, a_m), \quad 1 \leq i \leq m+1,$$

ergibt sich bei komponentenweiser Erweiterung der Ordnung

$$(a_1, a_2, \ldots, a_m) = \underline{c_1} < \underline{c_2} < \cdots < \underline{c_{m-1}} < \underline{c_m} < \underline{c_{m+1}} = (b_1, b_2, \ldots, b_m).$$

Würde nun $f_{Mon}(\underline{c_i}) \leq f_{Mon}(\underline{c_{i+1}})$ für $1 \leq i \leq m$ gelten, so ergibt sich ein Widerspruch zu (1.7). Daher gibt es eine natürliche Zahl j, $1 \leq j \leq m$, mit $f_{Mon}(\underline{c_j}) > f_{Mon}(\underline{c_{j+1}})$. Wir erhalten also

$$f_{Mon}(b_1, b_2, \ldots, b_{j-1}, 0, a_{j+1}, a_{j+2}, \ldots, a_m) > f_{Mon}(b_1, b_2, \ldots, b_{j-1}, 1, a_{j+1}, a_{j+2}, \ldots, a_m)$$

($a_j = 0$ und $b_j = 1$). Daher berechnet der Schaltkreis aus Abbildung 1.6 die Negation. Ersetzen wir die Konstanten durch die Schaltkreise aus (a), so erhalten wir einen Schaltkreis über $\mathcal{S}$, der die Negation berechnet.

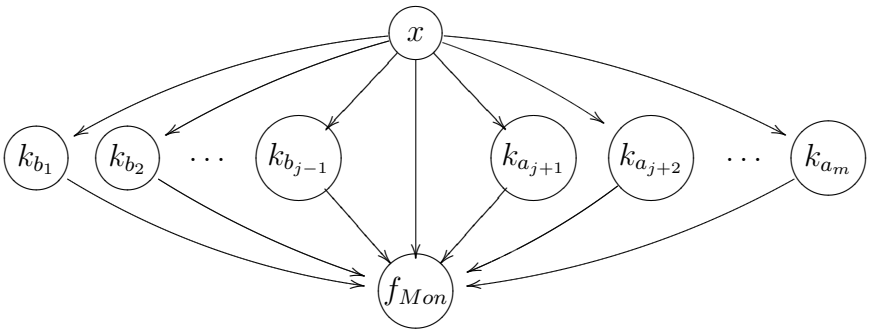


Abbildung 1.6: Schaltkreis zur Berechnung der Negation

(c) Für die nicht in *Lin* liegende Funktion f_{Lin} aus $\mathcal{S}$ gibt es die Darstellung

$$f_{Lin}(x_1, x_2, \ldots, x_k) = x_1x_2 \cdot f_1(x_3, x_4, \ldots, x_k) \oplus x_1 \cdot f_2(x_3, x_4, \ldots, x_k) \\ \oplus x_2 \cdot f_3(x_3, x_4, \ldots, x_k) \oplus f_4(x_3, x_4, \ldots, x_k)$$

mit gewissen Funktionen $f_1, f_2, f_3, f_4 \in B$ und $f_1 \neq k_0$ (ohne Beschränkung der Allgemeinheit nehmen wir dabei an, dass die Nichtlinearität bezüglich der Variablen x_1 und x_2 vorliegt). Wegen $f_1 \neq k_0$ gibt es Werte $d_3, d_4, \ldots, d_k \in \{0, 1\}$ mit $f_1(d_3, d_4, \ldots, d_k) = 1$. Wir konstruieren den Schaltkreis S aus Abbildung 1.7 a). Offenbar berechnet S die Funktion f'_{Lin} mit

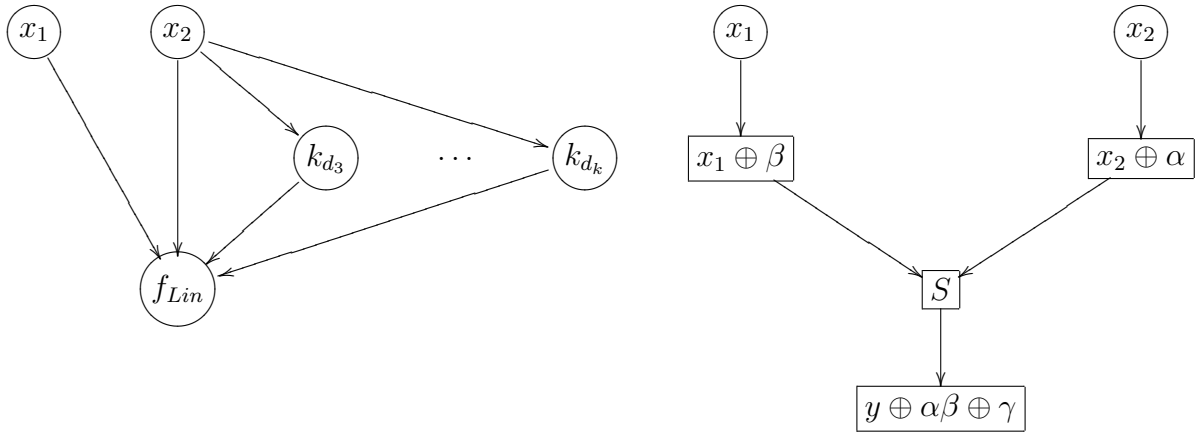
$$f'_{Lin}(x_1, x_2) = x_1x_2 \oplus \alpha x_1 \oplus \beta x_2 \oplus \gamma,$$

für gewisse $\alpha, \beta, \gamma \in \{0, 1\}$. Daher berechnet der Schaltkreis aus Abbildung 1.7 b) wegen

$$x_1x_2 = \big((x_1 \oplus \beta)(x_2 \oplus \alpha) + \alpha(x_1 \oplus \beta) \oplus \beta(x_2 \oplus \alpha) \oplus \gamma\big) \oplus (\alpha\beta \oplus \gamma)$$

die Konjunktion. Da in Abhängigkeit von α, β, γ die Funktionen $x_1 \oplus \beta$, $x_2 \oplus \alpha$ und $y \oplus \alpha\beta \oplus \gamma$ die Identität oder Negation sind, die Konstanten nach (a), die Negation nach (b) durch Schaltkreise über $\mathcal{S}$ berechnet werden können, können auch die Schaltkreise aus der Abbildung 1.7 über $\mathcal{S}$ realisiert werden. Somit ist die Konjunktion durch einen Schaltkreis über $\mathcal{S}$ berechenbar.

(d) Die Disjunktion kann nun entsprechend den DE MORGANschen Regeln durch einen Schaltkreis über $\mathcal{S}$ berechnet werden. □



a) b)

Abbildung 1.7: Schaltkreise zum Berechnen der Konjunktion

Übungsaufgaben

1. Beweisen Sie, dass aus $g \in Sd$ und $g_i \in Sd$ für $1 \leq i \leq k$ auch

$$f(x_1, x_2, \ldots, x_n) = g(g_1(x_1, \ldots, x_n), g_2(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)) \in Sd$$

folgt.

2. Untersuchen Sie, ob die folgenden Mengen von Funktionen vollständig sind:
   a) $\{\overline{\ }, \wedge\}$, b) $\{\oplus, \vee\}$, c) $\{f_0, f_1, \ldots, f_{2^n-1}\}$,
   wobei die Funktionen $f_i \in B_n$ für $0 \leq i \leq 2^n - 1$ durch

$$f_i(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & x_1 x_2 \ldots x_n \text{ ist die Dualdarstellung von } i, \\ 0, & \text{sonst} \end{cases}$$

   definiert sind.

3. Beweisen Sie, dass für eine Funktion $f \notin T_0$ (oder $f \notin T_1$) auch $f \notin Mon$ oder $f \notin Sd$ gilt.

4. a) Beweisen Sie, dass für jede vollständige Menge $\mathcal{S}$ eine vollständige Teilmenge $\mathcal{S}' \subseteq \mathcal{S}$ existiert, die aus höchstens 5 Funktionen besteht.
   b) Beweisen Sie, dass für jede vollständige Menge $\mathcal{S}$ eine vollständige Teilmenge $\mathcal{S}' \subseteq \mathcal{S}$ existiert, die aus höchstens 4 Funktionen besteht. (Hinweis: Verwenden Sie Aufgabe 3.)

1.4. Beziehungen zwischen den Maßen

Wir haben in Abschnitt 1.2. eine unendliche Menge von Komplexitätsmaßen für Schaltkreise eingeführt, denn die angegebenen Maße hängen zum einen von der Menge $\mathcal{S}$, über

der die Schaltkreise definiert sind, und zum anderen von der Beschränkung des fan-out ab. Beide Parameter gestatten unendlich viele verschiedene Belegungen. Ein Ziel dieses Abschnitts ist es zu zeigen, dass bis auf konstante Faktoren die Maße vielfach übereinstimmen, so dass im Wesentlichen nur drei verschiedene Maße existieren. Weiterhin werden wir Relationen zwischen diesen drei Maßen angeben.

Wir zeigen zuerst, dass sich die Maße bez. verschiedener vollständiger Mengen nur durch konstante Faktoren unterscheiden.

Satz 1.17 *Sind $\mathcal{S}_1$ und $\mathcal{S}_2$ zwei vollständige Mengen, so gibt es (von $\mathcal{S}_1$ und $\mathcal{S}_2$ abhängige) Konstanten $c_1, c_2, d_1, d_2 > 0$ und $c_{k,1}, c_{k,2} > 0$, $k \geq 1$ so, dass für jede Boolesche Funktion f*

$$c_1 \cdot C_{\mathcal{S}_2}(f) \leq C_{\mathcal{S}_1}(f) \leq c_2 \cdot C_{\mathcal{S}_2}(f),$$
$$d_1 \cdot D_{\mathcal{S}_2}(f) \leq D_{\mathcal{S}_1}(f) \leq d_2 \cdot D_{\mathcal{S}_2}(f),$$
$$c_{k,1} \cdot C_{k,\mathcal{S}_2}(f) \leq C_{k,\mathcal{S}_1}(f) \leq c_{k,2} \cdot C_{k,\mathcal{S}_2}(f)$$

gelten.

Beweis: Wir beweisen nur die Relation für die Größe der Schaltkreise. Die Beweise für die Tiefe bzw. die fan-out-beschränkte Größe können analog geführt werden.

Wir setzen

$$c_2 = \max\{\, C_{\mathcal{S}_1}(g) \mid g \in \mathcal{S}_2 \,\}.$$

Für eine Funktion f sei S ein Schaltkreis über $\mathcal{S}_2$ mit $C(S) = C_{\mathcal{S}_2}(f)$. Wir konstruieren nun den Schaltkreis S' über $\mathcal{S}_1$, indem wir jeden Knoten von S, der mit einer Funktion $h \in \mathcal{S}_2$ markiert ist, durch einen Schaltkreis über $\mathcal{S}_1$ ersetzen, der h berechnet. Damit wird jeder Knoten aus S mit Markierung in $\mathcal{S}_2$ durch höchstens c_2 Knoten mit Markierung in $\mathcal{S}_1$ ersetzt. Daher gilt $C(S') \leq c_2 C(S)$. Beachten wir noch, dass S' nicht der bez. der Größe minimale Schaltkreis über $\mathcal{S}_1$ sein muss, der f berechnet, so ergibt sich

$$C_{\mathcal{S}_1}(f) \leq C(S') \leq c_2 \cdot C(S) = c_2 \cdot C_{\mathcal{S}_2}(f).$$

Damit ist der zweite Teil der zu beweisenden Relation bewiesen.

Unter Verwendung von

$$c_1' = \max\{\, C_{\mathcal{S}_2}(g) \mid g \in \mathcal{S}_1 \,\}$$

können wir völlig analog

$$C_{\mathcal{S}_2}(f) \leq c_1' \cdot C_{\mathcal{S}_1}(f)$$

zeigen. Wegen $c_1' > 0$ erhalten wir daraus

$$\frac{1}{c_1'} \cdot C_{\mathcal{S}_2}(f) \leq C_{\mathcal{S}_1}(f)$$

woraus mittels der Setzung $c_1 = \frac{1}{c_1'}$ auch der erste Teil der Relation folgt. □

Damit ist gezeigt, dass durch den Parameter der (vollständigen) Mengen, über der die Schaltkreise definiert sind, keine Unendlichkeit vorliegt, wenn wir (wie bei Komplexitätsuntersuchungen meist üblich) Unterschiede durch konstante Faktoren nicht berücksichtigen.

Wir betrachten nun den zweiten Parameter, die fan-out-Beschränkung, durch den unendlich viele verschiedene Maße entstehen können. Aus der Definition folgt sofort

$$C_{1,\mathcal{S}}(f) \geq C_{2,\mathcal{S}}(f) \geq C_{3,\mathcal{S}}(f) \geq \cdots \geq C_{k,\mathcal{S}}(f) \geq \cdots \geq C_{\mathcal{S}}(f) \tag{1.8}$$

für jede Boolesche Funktion f, jede Menge $\mathcal{S}$ und jede natürliche Zahl $k \geq 3$.

Satz 1.18 *Für jede natürliche Zahl $k \geq 2$ und jede vollständige Menge $\mathcal{S}$ gibt es eine Konstante c derart, dass*

$$C_{k,\mathcal{S}}(f) \leq c \cdot C_{\mathcal{S}}(f)$$

für alle Funktionen $f \in B$ gilt.

Beweis: Es sei S ein bez. der Größe minimaler Schaltkreis über $\mathcal{S}$ für f, d. h. es gelte $C_{\mathcal{S}}(f) = C(S)$. Da $f \in B$ ist, hat S genau einen Ausgabeknoten und wegen der Minimalität von S ist dies das einzige Gatter, das den Ausgangsgrad 0 hat. Wir numerieren nun die Gatter mit einem positiven Ausgangsgrad mit $1, 2, \ldots, C(S) - 1$. Es sei r_i der Ausgangsgrad des Gatters i.

Wir konstruieren nun einen fan-out-k-beschränkten Schaltkreis S' aus S in der folgenden Weise:

- Wir verändern das Ausgangsgatter nicht.
- Gilt $r_i \leq k$ für den Ausgangsgrad r_i des Gatters i, so verändern wir das Gatter i nicht und setzen $p_i = 0$.
- Hat das mit $h \in \mathcal{S}$ markierte Gatter i den Ausgangsgrad $r_i > k$, so setzen wir

  $$p_i = \left\lfloor \frac{r_i - k}{k-1} \right\rfloor \quad \text{und} \quad t_i = r_i - p_i(k-1)$$

  und ersetzen das Gatter i durch den Schaltkreis aus Abbildung 1.8.

Nach dieser Konstruktion berechnet S' offenbar auch die Funktion f und ist fan-out-k-beschränkt. Folglich gilt

$$C_{k,\mathcal{S}}(f) \leq C(S') = C(S) + \sum_{i=1}^{C(S)-1} p_i \cdot C_{k,\mathcal{S}}(id). \tag{1.9}$$

Zur Analyse dieser Abschätzung berechnen wir jetzt $\sum_{i=1}^{C(S)-1} p_i$. Ist $r_i > k$ für ein Gatter i, so gilt entsprechend unserer Definition von p_i

$$p_i < \frac{r_i - k}{k-1} + 1 = \frac{r_i - 1}{k-1}.$$

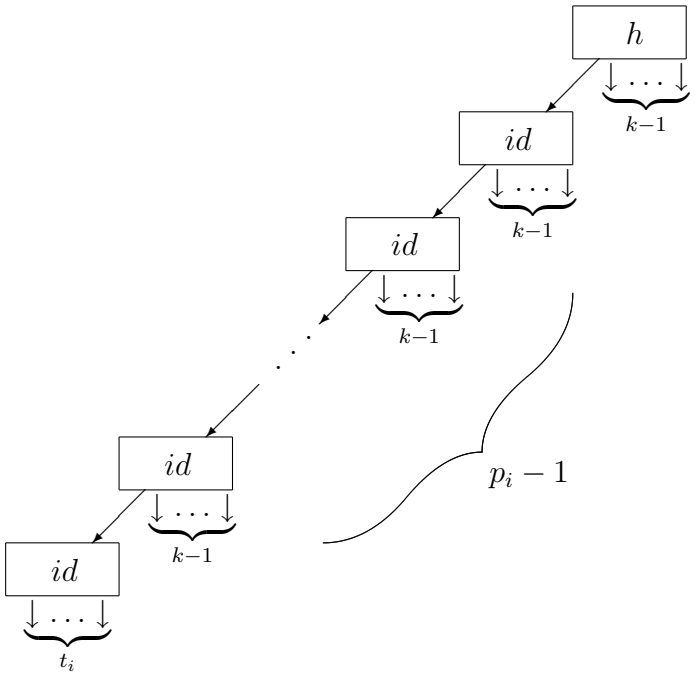


Abbildung 1.8: Reduktion des Ausgangsgrades auf k

Für ein Gatter i mit $r_i \leq k$ ergibt sich wegen $p_i = 0$ und $r_i \geq 1$ auch $p_i \leq \frac{r_i - 1}{k-1}$. Damit erhalten wir

$$\sum_{i=1}^{C(S)-1} p_i < \sum_{i=1}^{C(S)-1} \frac{r_i - 1}{k-1} = \frac{(\sum_{i=1}^{C(S)-1} r_i) - (C(S)-1)}{k-1}. \tag{1.10}$$

Dabei ist $\sum_{i=1}^{C(S)-1} r_i$ die Summe der Ausgangsgrade der Gatter. Außerdem hat mindestens ein Eingangsknoten den Ausgangsgrad 1. Wir setzen nun noch s als das Maximum der Stelligkeit von Funktionen aus $\mathcal{S}$ (und damit als das Maximum der Eingangsgrade von Gattern). Da die Eingangsknoten den Eingangsgrad 0 haben, folgt aus der Tatsache, dass jede Kante im Schaltkreis jeweils 1 zu den Ein- und Ausgangsgraden beiträgt,

$$1 + \sum_{i=1}^{C(S)-1} r_i \leq C(S) \cdot s.$$

Verwenden wir dies in (1.10), so ergibt sich

$$\sum_{i=1}^{C(S)-1} p_i \leq \frac{(C(S) \cdot s - 1) - (C(S) - 1)}{k-1} = C(S) \cdot \frac{s-1}{k-1}.$$

Kombinieren wir dies mit (1.9), so erhalten wir

$$C_{k,\mathcal{S}}(f) \leq C(S) + C(S) \cdot \frac{s-1}{k-1} \cdot C_{k,\mathcal{S}}(id) = C(S) \cdot \Big(1 + \frac{s-1}{k-1} \cdot C_{k,\mathcal{S}}(id)\Big).$$

Damit ist

$$c = 1 + \frac{s-1}{k-1} \cdot C_{k,\mathcal{S}}(id)$$

die gesuchte Konstante und der Satz bewiesen. □

Aus (1.8) und Satz 1.18 folgt

$$C_{\mathcal{S}}(f) \leq C_{k,\mathcal{S}}(f) \leq c \cdot C_{\mathcal{S}}(f)$$

für jede Boolesche Funktion f, jede vollständige Menge $\mathcal{S}$ und jedes $k \geq 2$. Folglich stimmen bis auf konstante Faktoren die Komplexitätsmaße $C_{k,\mathcal{S}}$ und $C_{\mathcal{S}}$ überein. Damit bleiben nur noch drei wesentlich verschiedene Maße übrig. Dies sind die Größe, die Tiefe und die Länge (letztere wegen $C_{1,\mathcal{S}} = L_{\mathcal{S}}$) bez. einer (festen oder dem Problem gerade angepassten) vollständigen Menge $\mathcal{S}$.

Als nächstes zeigen wir nun, dass bei gewissen vollständigen Mengen bis auf konstante Faktoren die Tiefe der Logarithmus der Länge ist.

Satz 1.19

i) *Es seien $\mathcal{S}$ eine vollständige Menge mit $\mathcal{S} \subseteq B_2$ und f eine beliebige Funktion aus B. Dann gilt*

$$\log(L_{\mathcal{S}}(f) + 1) \leq D_{\mathcal{S}}(f).$$

ii) *Es seien $\mathcal{S}$ eine vollständige Menge mit $\{k_0, k_1\} \subseteq \mathcal{S} \subseteq B_2$ und f eine beliebige Funktion aus B. Dann gilt*

$$D_{\mathcal{S}}(f) \leq k(\mathcal{S}) \cdot \log(L_{\mathcal{S}}(f) + 1)$$

mit

$$k(\mathcal{S}) = \frac{1 + D_{\mathcal{S}}(\overline{x}y \vee xz)}{\log(3) - 1}.$$

Beweis: i) Es sei S ein Schaltkreis über $\mathcal{S}$, der f berechnet und $D(S) = D_{\mathcal{S}}(f)$ erfüllt. Hat eines der Gatter von S einen Ausgangsgrad $k > 1$ so ersetzen wir dieses Gatter durch k Kopien, die dann nur noch den Ausgangsgrad 1 haben. Wenn wir dies für alle Gatter durchführen, so erhalten wir einen Schaltkreis S', dessen Gatter alle den Ausgangsgrad 1 haben, der f berechnet und für den ebenfalls $D(S') = D_{\mathcal{S}}(f)$ gilt. Außerdem hat S' bei Fortlassung der Eingangsknoten eine Baumstruktur, bei der alle Kanten in Richtung Wurzel gerichtet sind. Aufgrund unserer Voraussetzung, dass $\mathcal{S}$ nur zweistellige Funktionen enthält, ist S' ohne Eingangsknoten sogar ein Binärbaum der Tiefe $D(S') - 1$ (da die Kanten zu den Eingangsknoten gerade 1 zur Tiefe des Schaltkreises beitragen). Da für einen Binärbaum der Tiefe d gilt, dass er höchstens $2^{d+1} - 1$ Knoten enthält, erhalten wir für die Anzahl $C(S')$ der Gatter von S' die Beziehung

$$C(S') \leq 2^{D(S')} - 1$$

und daraus durch Umstellen und Logarithmieren (zur Basis 2)

$$\log(C(S') + 1) \leq D(S') = D_{\mathcal{S}}(f).$$

Unter Beachtung der nach Definition geltenden Beziehung $L_{\mathcal{S}}(f) \leq C(S')$ folgt hieraus die Behauptung.

ii) Wir beweisen die Aussage durch vollständige Induktion über $L_{\mathcal{S}}(f)$.

Ist $L_{\mathcal{S}}(f) \leq 2$, so besteht der längenminimale Schaltkreis aus höchstens zwei Gattern. Damit gilt auch $D_{\mathcal{S}}(f) \leq 2$ und wegen $k(\mathcal{S}) \geq 2$ folgt die gewünschte Relation.

Es sei nun T ein Schaltkreis über $\mathcal{S}$, der f berechnet, bei Fortlassen der Eingangsknoten ein Binärbaum ist und $C(T) = L_{\mathcal{S}}(f) \geq 3$ erfüllt. Mit T_1 und T_2 bezeichnen wir die beiden Schaltkreise, die aus T entstehen, wenn wir den Ausgangsknoten entfernen. T_1 berechne die Funktion f_1 und T_2 berechne f_2. Ohne Beschränkung der Allgemeinheit gelte noch $C(T_1) \leq C(T_2)$. Wegen $C(T_1) + C(T_2) + 1 = C(T)$ folgen daher

$$C(T_1) \leq \frac{C(T)-1}{2} \leq C(T_2) \text{ und } C(T_2) \leq C(T) - 2\,. \tag{1.11}$$

Außerdem gilt nach Konstruktion noch

$$D_{\mathcal{S}}(f) \leq 1 + \max\{D_{\mathcal{S}}(f_1), D_{\mathcal{S}}(f_2)\}. \tag{1.12}$$

Ferner sei T_0 ein hinsichtlich der Gatterzahl minimaler Teilbaum von T_2, dessen Blätter auch Blätter von T_2 sind (die Blätter haben im Schaltkreis nur noch Eingangsknoten als Vorgänger), mit

$$C(T_0) \geq \frac{C(T_2)}{3}\,.$$

Wegen der Minimalität von T_0 können die beiden aus T_0 durch Streichen der Wurzel entstehenden Bäume höchstens $\left\lceil \frac{C(T_2)}{3} \right\rceil - 1$ Gatter enthalten, woraus

$$C(T_0) \leq 2 \cdot \frac{C(T_2)}{3} + 1 \leq \frac{2(C(T)-2)}{3} + 1 \leq \frac{2C(T)-1}{3} \tag{1.13}$$

folgt. Möge T_0 die Funktion f_0 berechnen, und für $i \in \{0,1\}$ möge $f_{2,i}$ die Funktion sein, die durch den Schaltkreis $T_{2,i}$ berechnet wird, der aus T_2 durch Ersetzen von T_0 durch die Konstante k_i entsteht. Dann ergibt sich

$$\begin{aligned} C(T_{2,i}) &= C(T_2) - C(T_0) + 1 \leq C(T_2) - \tfrac{C(T_2)}{3} + 1 \leq \tfrac{2C(T_2)}{3} + 1 \\ &\leq \tfrac{2C(T)-2}{3} + 1 \leq \tfrac{2C(T)-1}{3} \end{aligned} \tag{1.14}$$

für $i \in \{0,1\}$. Außerdem erhalten wir

$$f_2(x_1,\ldots,x_n) = \overline{f_0(x_1,\ldots,x_n)} f_{2,0}(x_1,\ldots,x_n) \vee f_0(x_1,\ldots,x_n) f_{2,1}(x_1,\ldots,x_n),$$

denn bei $f_0(x_1,\ldots,x_n) = 0$ gilt $f_2(x_1,\ldots,x_n) = f_{2,0}(x_1,\ldots,x_n)$, und analog gilt auch $f_2(x_1,\ldots,x_n) = f_{2,1}(x_1,\ldots,x_n)$ bei $f_0(x_1,\ldots,x_n) = 1$. Aus dieser Darstellung resultiert

$$D_{\mathcal{S}}(f_2) \leq D_{\mathcal{S}}(\overline{x}y \vee xz) + \max\{D_{\mathcal{S}}(f_0), D_{\mathcal{S}}(f_{2,0}), D_{\mathcal{S}}(f_{2,1})\}. \tag{1.15}$$

Aus (1.13) und (1.14) und der Induktionsvoraussetzung ergibt sich

$$D_{\mathcal{S}}(g) \leq k(\mathcal{S}) \cdot \log(L_{\mathcal{S}}(g)+1) \leq k(\mathcal{S}) \cdot \log\Big(\frac{2C(T)-1}{3} + 1\Big)$$

für $g \in \{f_0, f_{2,0}, f_{2,1}\}$ und hieraus mittels (1.15)

$$D_{\mathcal{S}}(f_2) \leq D_{\mathcal{S}}(\overline{x}y \vee xz) + k(\mathcal{S}) \cdot \log\Big(\frac{2C(T)-1}{3} + 1\Big). \tag{1.16}$$

Weiterhin erhalten wir aus der Induktionsvoraussetzung und (1.11) noch

$$D_{\mathcal{S}}(f_1) \leq k(\mathcal{S}) \cdot \log\Big(\frac{C(T)-1}{2} + 1\Big),$$

woraus unter Beachtung von (1.12) und (1.16) dann

$$\begin{aligned}
D_{\mathcal{S}}(f) &\leq 1 + D_{\mathcal{S}}(\overline{x}y \vee xz) + k(\mathcal{S}) \cdot \log\big(\frac{2C(T)-1}{3} + 1\big) \\
&\leq 1 + D_{\mathcal{S}}(\overline{x}y \vee xz) + k(\mathcal{S}) \cdot \log\big(\frac{2C(T)+2}{3}\big) \\
&\leq 1 + D_{\mathcal{S}}(\overline{x}y \vee xz) + k(\mathcal{S}) \cdot \log\big(\frac{2(C(T)+1)}{3}\big) \\
&\leq 1 + D_{\mathcal{S}}(\overline{x}y \vee xz) + k(\mathcal{S}) \cdot \big(\log(\frac{2}{3}) + \log(C(T)+1)\big) \\
&= 1 + D_{\mathcal{S}}(\overline{x}y \vee xz) + k(\mathcal{S}) \log(\frac{2}{3}) + k(\mathcal{S}) \log(C(T)+1) \\
&= 1 + D_{\mathcal{S}}(\overline{x}y \vee xz) + k(\mathcal{S})(1 - \log(3)) + k(\mathcal{S}) \log(C(T)+1) \\
&= 1 + D_{\mathcal{S}}(\overline{x}y \vee xz) + \frac{1 + D_{\mathcal{S}}(\overline{x}y \vee xz)}{\log(3) - 1}(1 - \log(3)) + k(\mathcal{S}) \log(C(T)+1) \\
&= k(\mathcal{S}) \log(C(T)+1)
\end{aligned}$$

und damit die Behauptung folgen. □

Als erstes bemerken wir, dass die Voraussetzun(en) $(\{k_0, k_1\} \subseteq)\mathcal{S} \subseteq B_2$ durch einen Übergang zu einer vollständigen Menge mit dieser Bedingung und daraus resultierender Beachtung von konstanten Faktoren erreicht werden kann. Außerdem stellt $k_0, k_1 \in \mathcal{S}$ schaltungstechnisch keine Einschränkung dar, da dies durch Eingangsknoten mit konstanter Eingabe relativ einfach realisiert werden kann.

Als zweites bemerken wir, dass $k(B_2) \approx 5.13$ gilt und dass der Wert $k(\mathcal{S})$ auch für andere $\mathcal{S}$ relativ klein ist. Daher wird die Tiefe relativ gut durch den Logarithmus der Länge approximiert.

Wir kommen nun zu Relationen zwischen Größe und Tiefe bzw. Länge.

Satz 1.20 *Für jede vollständige Menge $\mathcal{S} \subseteq B_2$ und jede Funktion $f \in B$ gelten*

$$C_{\mathcal{S}}(f) \leq L_{\mathcal{S}}(f) \quad \textit{und} \quad \log(C_{\mathcal{S}}(f)+1) \leq D_{\mathcal{S}}(f).$$

Beweis: Die erste Relation folgt wegen $L_{\mathcal{S}}(f) = C_{1,\mathcal{S}}(f)$ aus (1.8). Die zweite Relation folgt aus der ersten und Satz 1.19 i). □

Ohne Beweis geben wir noch die Abschätzung in der anderen Richtung. Für den technisch aufwendigen Beweis verweisen wir auf [21].

Satz 1.21 *Für jede vollständige Menge $\mathcal{S} \subseteq B$ gibt es eine Konstante $k'(\mathcal{S})$ derart, dass für jede Funktion $f \in B$*

$$k'(\mathcal{S}) \cdot D_{\mathcal{S}}(f) \cdot \log(D_{\mathcal{S}}(f)) \leq C_{\mathcal{S}}(f)$$

gilt. □

Durch Umformen erhalten wir aus Satz 1.21 die Relation

$$D_{\mathcal{S}}(f) \leq \frac{C_{\mathcal{S}}(f)}{k'(\mathcal{S}) \cdot \log(D_{\mathcal{S}}(f))}. \tag{1.17}$$

Beachten wir noch, dass aus Satz 1.20 $\log(\log(C_{\mathcal{S}}(f))) \leq \log(D_{\mathcal{S}}(f))$ folgt, so erhalten wir aus (1.17)

$$D_{\mathcal{S}}(f) \leq \frac{C_{\mathcal{S}}(f)}{k'(\mathcal{S}) \cdot \log(\log(C_{\mathcal{S}}(f)))}$$

für vollständige Mengen $\mathcal{S} \subseteq B_2$.

Für die Beziehung zwischen Länge und Größe ergibt sich mittels Satz 1.19 sofort

$$\log(L_{\mathcal{S}}(f)) \leq \frac{C_{\mathcal{S}}(f)}{k'(\mathcal{S}) \cdot \log(\log(C_{\mathcal{S}}(f)))}$$

für vollständige Mengen $\mathcal{S} \subseteq B_2$.

Übungsaufgaben

1. a) Beweisen Sie, dass $C_{\{\wedge,\vee,\neg\}}(f) \leq 4C_{\{\oplus,\wedge,k_1\}}(f)$ für alle Booleschen Funktionen f gilt.
   b) Geben Sie Boolesche Funktionen f_1, f_2 und f_3 an, für die

   $$C_{\{\wedge,\vee,\neg\}}(f_i) = C_{\{\oplus,\wedge,k_1\}}(f_i) + i$$

   gilt.

2. Beweisen Sie, dass sich die Abschätzung aus Satz 1.19 i) nicht verbessern lässt (d. h., zeigen Sie, dass es eine Funktion gibt, bei der in Satz 1.19 i) die Gleichheit gilt).

3. Berechnen Sie $k(\mathcal{S})$ für $\mathcal{S} = \{\wedge, \vee, \neg\}$ und $\mathcal{S} = \{\oplus, \wedge, k_1\}$.

1.5. Asymptotische Komplexität – untere und obere Schranken

In diesem Abschnitt wollen wir Schranken für die Komplexität $C_{\mathcal{S}}(f)$ bestimmen.

Sollen die Schranken für alle Booleschen Funktionen gelten, so ist die untere Schranke offenbar durch 1 gegeben. Dies folgt daraus, dass jede Funktion f aus der Menge $\mathcal{S}$ der Basisfunktionen durch einen Schaltkreis realisiert wird, der neben den Eingangsknoten nur den mit f markierten Knoten enthält. Damit gilt $C_{\mathcal{S}}(f) = 1$.

Diese untere Schranke lässt sich aber deutlich verbessern, wenn wir eine einfache Forderung an die zu realisierende Funktion stellen.

Wir sagen, dass $f \in B_n$ von der Variablen x_i wesentlich abhängt, wenn es Werte a_j mit $1 \leq j \leq n$, $j \neq i$, derart gibt, dass

$$f(a_1, a_2, \ldots, a_{i-1}, 0, a_{i+1}, a_{i+2}, \ldots, a_n) \neq f(a_1, a_2, \ldots, a_{i-1}, 1, a_{i+1}, a_{i+2}, \ldots, a_n)$$

gilt. Bei einer Änderung der wesentlichen Variablen ändert sich also auch der Wert der Funktion.

Satz 1.22 *Für jede Boolesche Funktion $f \in B_n$, die von allen Variablen wesentlich abhängt, gilt $C_{B_2}(f) \geq n - 1$.*

Beweis: Es sei S ein Schaltkreis über B_2, der f berechnet und $C(S) = C_{B_2}(f)$ erfüllt. Zur Vereinfachung setzen wir $C(S) = c$. Wir berechnen die Anzahl k der Kanten von S auf zwei Arten. Zum einen gehen in jedes Gatter von S genau zwei Kanten. Daher gilt $k = 2c$. Zum anderen geht aus jedem Eingangsknoten mindestens eine Kante aus, da f von jeder Variablen wesentlich abhängt. Weiterhin geht auch von jedem Gatter mit Ausnahme des Endgatters mindestens eine Kante aus. Somit ergibt sich $k \geq n + c - 1$ und damit $2c \geq n + c - 1$, woraus die Behauptung sofort folgt. □

Aus Satz 1.22 folgt sofort, dass es keine Schranke gibt, durch die die Komplexität aller Funktionen nach oben beschränkt wird, da die Komplexität mit wachsender Stelligkeit mindestens linear in der Stelligkeit wächst.

Wir betrachten daher als Komplexitätsmaße Funktionen, bei denen eine Abhängigkeit von der Stelligkeit besteht. Dies entspricht dem Vorgehen bei der Komplexität von Berechnungen, bei der Funktionen betrachtet werden, die von der Größe der Eingabe (z. B. Wortlänge) abhängen.

Definition 1.23 *Für ein Komplexitätsmaß $K \in \{C, L, D\}$ und eine vollständige Menge $\mathcal{S} \subseteq B$ von Basisfunktionen definieren wir die Funktion $K_{\mathcal{S}} : \mathbb{N} \to \mathbb{N}$ vermöge*

$$K_{\mathcal{S}}(n) = \max\{K_{\mathcal{S}}(f) \mid f \in B_n\}\,.$$

Ziel dieses Abschnitts ist es, die Funktion $K_{\mathcal{S}}$ zu bestimmen. Die genaue Ermittlung wird uns aber kaum gelingen. Daher sind wir an unteren Schranken $K^u_{\mathcal{S}}(n)$ und oberen Schranken $K^o_{\mathcal{S}}(n)$ für $K_{\mathcal{S}}(n)$, d. h. Funktionen mit

$$K^u_{\mathcal{S}}(n) \leq K_{\mathcal{S}}(n) \leq K^o_{\mathcal{S}}(n)$$

interessiert, mittels derer dann zumindest eine asymptotische Bestimmung von $K_{\mathcal{S}}$ möglich sein kann.

Aufgrund der Definition 1.23 und der Ergebnisse des vorhergehenden Abschnitts ist klar, dass für zwei Mengen $\mathcal{S}$ und $\mathcal{S}'$ Konstanten c_1 und c_2 mit

$$c_1 K_{\mathcal{S}}(n) \leq K_{\mathcal{S}'}(n) \leq c_2 K_{\mathcal{S}}(n)$$

existieren. Zur asymptotischen Beschreibung kann man sich daher ohne Beschränkung der Allgemeinheit auf $\mathcal{S} = B_2$ beschränken. Zur Abkürzung setzen wir noch $K_{B_2} = K$ für $K \in \{C, D, L\}$.

Wir bestimmen zuerst obere Schranken.

Satz 1.24 *Es gibt eine natürliche Zahl n_0 derart, dass für jede natürliche Zahl $n \geq n_0$*

$$C(n) \leq \frac{2^n}{n} + o\Big(\frac{2^n}{n}\Big)$$

gilt.

Beweis: Wir geben zuerst eine neue Darstellung Boolescher Funktionen an und schätzen dann die Komplexität eines Schaltkreises auf der Basis dieser Darstellung ab. Die Darstellung als wesentlicher Schlüssel des Beweises geht auf O. B. Lupanov zurück und heißt (k, s)-Darstellung.

Wir wählen zuerst eine natürliche Zahl k mit $1 \leq k \leq n$ und teilen die Menge der n Variablen in zwei Mengen ein, von denen die erste Menge M_1 aus den ersten $n-k$ Variablen $x_1, x_2, \ldots, x_{n-k}$ und die zweite Menge M_2 aus den restlichen k Variablen $x_{n-k+1}, x_{n-k+2}, \ldots, x_n$ besteht. Wir bilden eine Tabelle, deren Zeilen aus den 2^k Tupeln $\beta_1, \beta_2, \ldots, \beta_{2^k}$, die durch Belegung der Variablen aus M_2 gebildet werden können, und deren Spalten analog aus den 2^{n-k} Tupeln $\alpha_1, \alpha_2, \ldots, \alpha_{2^{n-k}}$ bestehen, die den Belegungen der Variablen aus M_1 entsprechen. Der Funktionswert $f(a_1, \ldots, a_{n-k}, a_{n-k+1}, \ldots, a_n)$ steht dann im Schnittpunkt der Spalte zu $(a_1, \ldots, a_{n-k})$ und der Zeile zu $(a_{n-k+1}, \ldots, a_n)$.

Wir wählen als nächstes eine natürliche Zahl s mit $1 \leq s \leq 2^k$ und teilen die Zeilen in Blöcke aus jeweils s Zeilen ein. Offenbar gibt es $p = \lceil \frac{2^k}{s} \rceil$ derartige Blöcke, bei denen der letzte weniger als s Zeilen haben kann. Wir nehmen im folgenden an, dass alle Blöcke gleich groß sind und überlassen die notwendigen Modifikationen bei einem kleineren letzten Block dem Leser. Die Blöcke bezeichnen wir mit $A_1, A_2, \ldots, A_p$. Dabei besteht der Block A_i, $1 \leq i \leq p$, aus den Tupeln $\beta_{(i-1)s+1}, \beta_{(i-1)s+2}, \ldots, \beta_{is}$.

Für $1 \leq i \leq p$ setzen wir

$$f(x_1, x_2, \ldots, x_{n-k}, A_i) = \begin{pmatrix} f(x_1, x_2, \ldots, x_{n-k}, \beta_{(i-1)s+1}) \\ f(x_1, x_2, \ldots, x_{n-k}, \beta_{(i-1)s+2}) \\ \vdots \\ f(x_1, x_2, \ldots, x_{n-k}, \beta_{is}) \end{pmatrix}.$$

Offenbar ist $f(x_1, x_2, \ldots, x_{n-k}, A_i)$ ein s-dimensionaler Vektor über $\{0, 1\}$.

Eine Veranschaulichung der Darstellung ist in Abbildung 1.9 gegeben.

Für einen s-dimensionalen Vektor $w = (w_1, w_2, \ldots, w_s)$ über $\{0, 1\}$, eine natürliche Zahl i, $1 \leq i \leq p$, und eine n-stellige Funktion $f \in B_n$ definieren wir die Menge $A(i, w, f)$ durch

$$A(i, w, f) = \{\alpha \mid \alpha \in \{0, 1\}^{n-k}, f(\alpha, A_i) = w\}$$

und die Funktionen $f_{i,w}$, $f^{(1)}_{i,w}$ und $f^{(2)}_{i,w}$ durch

$$f_{i,w}(x_1, x_2, \ldots x_n) = \begin{cases} f(x_1, \ldots, x_n), & (x_1, \ldots, x_{n-k}) \in A(i, w, f), (x_{n-k+1}, \ldots, x_n) \in A_i, \\ 0, & \text{sonst}, \end{cases}$$

$$f^{(1)}_{i,w}(x_1, x_2, \ldots x_n) = \begin{cases} 1, & (x_1, \ldots, x_{n-k}) \in A(i, w, f), \\ 0, & \text{sonst}, \end{cases}$$

$$f^{(2)}_{i,w}(x_1, x_2, \ldots x_n) = \begin{cases} 1, & w_t = 1 \text{ und } (x_{n-k+1}, \ldots, x_n) = \beta_{(i-1)s+t}, \\ 0, & \text{sonst}. \end{cases}$$

	$x_{n-k+1}\dots x_n$	α_1	$\cdots$	$\alpha_{2^{n-k}}$	x_1 $\vdots$ x_{n-k}
A_1	β_1 $\vdots$ β_s				s
$\vdots$	$\vdots$				
A_i	$\beta_{(i-1)s+1}$ $\vdots$ β_{is}		w	w'	s
$\vdots$	$\vdots$				
A_{p-1}	$\beta_{(p-2)s+1}$ $\vdots$ $\beta_{(p-1)s}$				s
A_p	$\beta_{(p-1)s+1}$ $\vdots$ β_{2^k}				s'

Abbildung 1.9: (k,s)-Darstellung einer Funktion (hier haben wir den allgemeinen Fall mit einem kleineren letzten Block notiert)

Wir bemerken, dass die Funktionen $f_{i,w}^{(1)}$ und $f_{i,w}^{(2)}$ nur von den ersten $n-k$ bzw. den letzten k Variablen abhängen. Offenbar gelten dann

$$\begin{aligned}
f_{i,w}(x_1,x_2,\dots,x_n) &= f_{i,w}^{(1)}(x_1,x_2,\dots,x_n) \wedge f_{i,w}^{(2)}(x_1,x_2,\dots,x_n)\,,\\
f(x_1,x_2,\dots,x_n) &= \bigvee_{i=1}^{p} \bigvee_{w\in\{0,1\}^s} f_{i,w}(x_1,x_2,\dots,x_n)\,,\\
f(x_1,x_2,\dots,x_n) &= \bigvee_{i=1}^{p} \bigvee_{w\in\{0,1\}^s} f_{i,w}^{(1)}(x_1,\dots,x_n) \wedge f_{i,w}^{(2)}(x_1,\dots,x_n)\\
&= \bigvee_{i=1}^{p} \bigvee_{w\in\{0,1\}^s} f_{i,w}^{(1)}(x_1,\dots,x_{n-k}) \wedge f_{i,w}^{(2)}(x_{n-k+1},\dots,x_n).
\end{aligned}$$

Wir nutzen die letzte Darstellung zum Konstruieren eines Schaltkreises S für f. Der resultierende Schaltkreis S ist in Abbildung 1.10 zu sehen. Dabei bedeuten D_1 und D_2 Schaltkreise zum Berechnen aller Konjunktionen von der Form $x_1^{a_1}x_2^{a_2}\dots x_{n-k}^{a_{n-k}}$ bzw. $x_{n-k+1}^{a_{n-k+1}}x_{n-k+2}^{a_{n-k+2}}\dots x_n^{a_n}$. Außerdem stellen E_1 und E_2 Schaltkreise zum Berechnen aller Alternativen dieser Konjunktionen dar, die zur Berechnung von $f_{i,w}^{(1)}$ bzw. $f_{i,w}^{(2)}$ durch konjunktive Normalformen erforderlich sind. Der Schaltkreis G berechnet alle Konjunktionen $f_{i,w} = f_{i,w}^{(1)} \wedge f_{i,w}^{(2)}$ für $1 \le i \le p$ und $w \in \{0,1\}^s$. Der Schaltkreis F berechnet dann die Alternative aller $f_{i,w}$. Entsprechend dieser Bedeutung sind D_1, D_2, E_1, E_2, G, und F Schaltkreise mit vielen Eingabe- und/oder Ausgabeknoten. Wir haben in Abbildung 1.10 aber stets nur eine Ausgabe und/oder Eingabe angegeben.

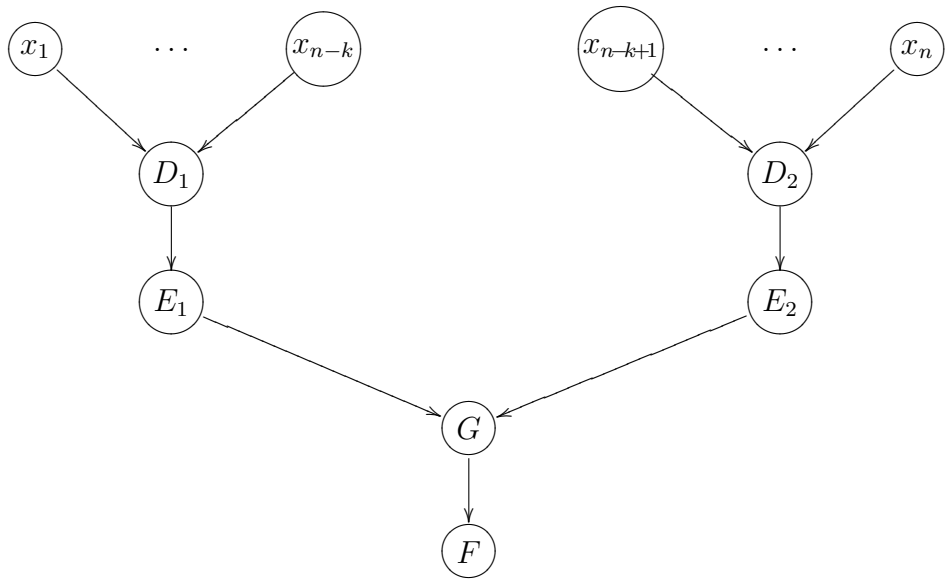


Abbildung 1.10: Schaltkreis entsprechend der (k, s)-Darstellung

Wir ermitteln nun die Komplexitäten der einzelnen Schaltkreise.

- $C(D_1) \leq (n-k)2^{n-k}$
  Da x^a die Negation sein kann, benutzen wir für jede Variable noch ihre Negation. Dies erfordert $n-k$ Gatter. Anschließend konstruieren wir jede der Konjunktionen $x_1^{a_1}x_2^{a_2}\ldots x_{n-k}^{a_{n-k}}$ unter Verwendung von $(n-k-1)$ $\wedge$-Gattern. Da wir insgesamt 2^{n-k} Konjunktionen berechnen müssen, ergibt sich insgesamt eine Komplexität von höchstens

$$(n-k)+(n-k-1)2^{n-k} = (n-k)-2^{n-k}+(n-k)2^{n-k} \leq (n-k)2^{n-k}\,.$$

- $C(D_2) \leq k2^k$
  Dies kann analog zur Betrachtung für D_1 gezeigt werden.

- $C(E_1) \leq \left(\frac{2^k}{s}+1\right)\cdot 2^{n-k}$
  Jede Funktion $f_{i,w}^{(1)}$ ist die Alternative derjenigen Konjunktion, die zu einer Spalte gehören, in der durch den i-ten Block w erzeugt wird. Da die Spalten zu w und w' mit $w \neq w'$ disjunkt sind, wird keine Konjunktion für zwei verschiedene Wörter benutzt. Folglich sind höchstens 2^{n-k} $\vee$-Gatter erforderlich. Da wir dies für alle i, $1 \leq i \leq p$, machen müssen, sind insgesamt höchstens $p2^{n-k}$ Gatter notwendig. Die Aussage folgt nun wegen $p \leq \left(\frac{2^k}{s}+1\right)$.

- $C(E_2) \leq s\cdot 2^s \cdot \left(\frac{2^k}{s}+1\right)$
  Da w höchstens s Einsen enthält, erfordert jedes w die Alternative von höchstens $s-1$ Konjunktionen für $f_{i,w}^{(2)}$. Da dies für alle w und für alle i zu tun ist, benötigen wir insgesamt höchstens $(s-1)2^s p \leq s2^s p$ Gatter.

- $C(G) \leq \left(\frac{2^k}{s}+1\right)\cdot 2^s$
  Dies folgt einfach daraus, dass wir für jedes Paar (i, w) eine Konjunktion benötigen und es p Möglichkeiten für i und 2^s Möglichkeiten für w gibt.

- $C(F) \leq \big(\frac{2^k}{s} + 1\big) \cdot 2^s$
  Dies folgt einfach daraus, dass wir für alle in G konstruierten $\big(\frac{2^k}{s} + 1\big) \cdot 2^s$ Konjunktionen alternativ miteinander verbinden müssen.

Insgesamt erhalten wir daher

$$\begin{aligned} C(S) &\leq (n-k)2^{n-k} + k2^k + \big(\frac{2^k}{s} + 1\big)2^{n-k} + s2^s\big(\frac{2^k}{s} + 1\big) + 2^s\big(\frac{2^k}{s} + 1\big) + 2^s\big(\frac{2^k}{s} + 1\big) \\ &= (n-k)2^{n-k} + k2^k + \big(\frac{2^k}{s} + 1\big)(2^{n-k} + s2^s + 2^{s+1})\,. \end{aligned}$$

Wir wählen nun

$$k = \lceil 3\log(n) \rceil \quad \text{und} \quad s = n - \lceil 5\log(n) \rceil$$

und schätzen $C(S)$ ab. Es gelten

$$\begin{aligned} \log(n-k) + (n-k) &\leq \log(n-k) + n - 3\log(n) \\ &\leq \log(n) + 1 + n - 3\log(n) \\ &= n + 1 - 2\log(n) \\ &\leq n + 1 - \log(n^2)\,, \end{aligned}$$

und daraus gewinnen wir durch Exponieren für den ersten Summanden

$$(n-k)2^{n-k} \leq \frac{2^{n+1}}{n^2}\,. \tag{1.18}$$

Wegen $k \leq 3\log(n) + 1 = \log(n^3) + 1$ erhalten wir durch Exponieren $2^k \leq 2^{\log(n^3)+1} = 2n^3$ und damit für den zweiten Summanden

$$k2^k \leq 2n^3(3\log(n) + 1)\,. \tag{1.19}$$

Wegen

$$\begin{aligned} \log\big(\frac{s}{2^k}\big) = \log(s) - k \leq \log(n - 5\log(n)) - 3\log(n) &\leq \log(n) - 3\log(n) \\ &= -2\log(n) = \log(\frac{1}{n^2}) \end{aligned}$$

ergibt sich

$$\frac{2^k}{s} + 1 = \frac{2^k}{s}\big(1 + \frac{s}{2^k}\big) \leq \frac{2^k}{s}\big(1 + \frac{1}{n^2}\big)\,. \tag{1.20}$$

Wegen

$$s + k - n + \log(s+2) \leq n - 5\log(n) + 3\log(n) + 1 - n + \log(n - 5\log(n) + 2) \leq 1 - \log(n)$$

für $5\log(n) \geq 2$ erhalten wir zuerst

$$2^{s+k-n}(s+2) \leq \frac{2}{n}$$

und damit

$$2^{n-k} + s2^s + 2^{s+1} = 2^{n-k}(1 + s2^{s+k-n} + 2^{s+1+k-n}) = 2^{n-k}(1 + 2^{s+k-n}(s+2)) \\ \leq 2^{n-k}(1 + \frac{2}{n})\,. \tag{1.21}$$

Unter Ausnutzung von (1.18), (1.19), (1.20) und (1.21) erhalten wir aus der obigen Abschätzung für $C(S)$ die Beziehung

$$\begin{aligned} C(S) &\leq \frac{2^{n+1}}{n^2} + 2n^3(3\log(n)+1) + \frac{2^k}{s}\big(1+\frac{1}{n^2}\big)2^{n-k}\big(1+\frac{2}{n}\big) \\ &= \frac{2^n}{n}\cdot\frac{2}{n} + 2n^3(3\log(n)+1) + \frac{2^n}{s}\big(1+\frac{2}{n}+\frac{1}{n^2}+\frac{2}{n^3}\big) \\ &= \frac{2^n}{n}\cdot\frac{2}{n} + 2n^3(3\log(n)+1) + \frac{2^n}{n-\lceil 5\log(n)\rceil}\big(1+\frac{2}{n}+\frac{1}{n^2}+\frac{2}{n^3}\big) \\ &= \frac{2^n}{n}\cdot\frac{2}{n} + 2n^3(3\log(n)+1) + \frac{2^n}{n(1-\frac{\lceil 5\log(n)\rceil}{n})}\big(1+\frac{2}{n}+\frac{1}{n^2}+\frac{2}{n^3}\big) \\ &= \frac{2^n}{n}\cdot\frac{2}{n} + 2n^3(3\log(n)+1) + \frac{2^n}{n(1-\frac{\lceil 5\log(n)\rceil}{n})}\big(1+\frac{2}{n}+\frac{1}{n^2}+\frac{2}{n^3}\big)\,. \end{aligned}$$

Hieraus folgt sofort

$$\lim_{n\to\infty} \frac{C(S)}{\frac{2^n}{n}} \leq 1$$

und daher die obere Schranke

$$\frac{2^n}{n} + o(\frac{2^n}{n})\,,$$

womit die Behauptung bewiesen ist. □

Der Schaltkreis für eine Funktion f entsprechend dem vorstehendem Beweis verwendet (in den Teilen D_1, D_2, E_1 und E_2) Knoten, deren Ausgangsgrad beliebig groß wird. Daher kann er nicht zum Konstruieren von längenoptimalen Schaltkreisen verwendet werden. Das Auffächern der Knoten in mehrere Knoten, die dann jeweils den Ausgangsgrad 1 haben, führt zu einem zu großen Anwachsen der Knotenzahl. Um eine Reduktion durchzuführen, beachtet man, dass die Funktionen $f_{i,w}^{(1)}$ und $f_{i,w}^{(2)}$ nur an wenigen Stellen den Wert 1 annehmen. Unter Beachtung dieses Faktes gelang es dem russischen Mathematiker Oleg B. Lupanov, den folgenden Satz zu beweisen, worauf wir hier verzichten.

Satz 1.25 *Es gibt eine natürliche Zahl $n_0 \geq 1$ derart, dass für jede natürliche Zahl $n \geq n_0$*

$$L(n) \leq \frac{2^{n+1}}{\log(n)} + o\big(\frac{2^n}{\log(n)}\big)$$

gilt. □

Satz 1.26 *Für jede natürliche Zahl $n \geq 2$ gilt*

$$D(n) \leq n + \lceil \log(n) \rceil\,.$$

Beweis: Es sei $f \in B_n$ eine n-stellige Boolesche Funktion. Wegen

$$\#(f^{-1}(1)) + \#(f^{-1}(0)) = 2^n$$

gilt $\#(f^{-1}(1)) \leq 2^{n-1}$ oder $\#(f^{-1}(0)) \leq 2^{n-1}$.

Es sei zuerst $\#(f^{-1}(1)) \leq 2^{n-1}$. Wir betrachten nun einen Schaltkreis zur Berechnung von f auf der Basis der disjunktiven Normalform. Dazu berechnen wir erst einmal $\overline{x_j}$ für jede Variable, wofür offenbar ein Beitrag 1 zur Tiefe geleistet wird. Die $\#(f^{-1}(1))$ alternativ verknüpften Konjunktionen bestehen jeweils aus n Elementen x_i oder $\overline{x_j}$ mit $1 \leq i, j \leq n$. Folglich hat jede Konjunktion die Tiefe $\lceil \log(n) \rceil$ und deren Alternative erfordert noch einmal die Tiefe

$$\lceil \log(\#(f^{-1}(1))) \rceil \leq \log(2^{n-1}) = n - 1\,.$$

Damit ist insgesamt höchstens die Tiefe $1 + \lceil \log(n) \rceil + n - 1$ erforderlich.

Gilt $\#(f^{-1}(0)) \leq 2^{n-1}$, so verwenden wir zur Konstruktion des Schaltkreises die konjunktive Normalform, wofür sich in analoger Weise höchstens die Tiefe $\lceil \log(n) \rceil + n$ ergibt. □

McColl und Paterson ([15]) haben Satz 1.26 dahingehend verschärft, dass es eine natürliche Zahl $n_0 \geq 1$ derart gibt, dass für jede natürliche Zahl $n \geq n_0$ die Relation $D(n) \leq n+1$ gilt. Zum Beweis dieser Aussage wird ein Graph der Tiefe $n+1$ konstruiert, aus dem die Schaltkreise für die unterschiedlichen Funktionen nur durch Variation der Markierung gewonnen werden. Wir verzichten auf den Beweis und geben auch ohne Beweis die folgende Verschärfung von Gaskov an, die bis auf konstante additive Terme optimal ist ([4]).

Satz 1.27 *Es gibt eine natürliche Zahl $n_0 \geq 1$ derart, dass für jede natürliche Zahl $n \geq n_0$*

$$D(n) \leq n - \log(\log(n)) + 2 + o(1)$$

gilt. □

Wir kommen nun zur Bestimmung unterer Schranken. Dazu ermitteln wir zuerst die Anzahl der n-stelligen Booleschen Funktionen, deren Komplexität höchstens b ist. Dann wählen wir b in Abhängigkeit von n passend und zeigen, dass die zugehörige Anzahl kleiner als die Anzahl $2^{(2^n)}$ aller n-stelligen Booleschen Funktionen ist. Folglich muss es eine Funktion geben, deren Komplexität größer als das gewählte b ist.

Für natürliche Zahlen $n \geq 1$ und $b \geq 1$ und ein Komplexitätsmaß $K \in \{C, D, L\}$ setzen wir

$$K(n, b) = \#(\{f \mid f \in B_n, K_{B_2}(f) \leq b\})\,.$$

Lemma 1.28 *Für natürliche Zahlen $n \geq 1$ und $b \geq 1$ gelten*

$$L(n, b) \leq n^{b+1} 64^b \quad \textit{und} \quad C(n, b) \leq \frac{b \cdot 16^b \cdot (n+b)^{2b}}{b!}\,.$$

Beweis: Wir beweisen zuerst die Aussage für L. Weil der fan-out der Knoten, die mit einer Basisfunktion markiert sind, höchstens 1 ist, können die Schaltkreise als umgekehrte binäre Bäume angesehen werden; der Ausgangsknoten des Schaltkreises entspricht der Wurzel und die Vorgängerknoten im Schaltkreis sind die Nachfolgerknoten im Baum.

Wir stellen zuerst fest, dass die Anzahl der binären Bäume mit b inneren Knoten (und einer beliebigen Anzahl von Blättern) höchstens 4^b ist. Dies folgt daraus, dass wir bei der Wurzel beginnend, jeweils vier Möglichkeiten für die Nachfolger haben (beides sind Blätter, nur der linke bzw. rechte Nachfolger ist ein Blatt, beide Nachfolger sind innere Knoten).

Weiterhin ist $b+1$ die maximale Zahl von Blättern. Dies kann mittels vollständiger Induktion über b leicht bewiesen werden.

Wir können nun jedes Blatt mit einer der n Variablen markieren, wofür sich unter Beachtung der maximalen Blattzahl offenbar höchstens n^{b+1} Möglichkeiten ergeben. Ferner kann jeder der b inneren Knoten mit einer der 16 Funktionen aus B_2 belegt werden. Hierfür gibt es höchstens 16^b Möglichkeiten.

Insgesamt erhalten wir daher $4^b n^{b+1} 16^b$ mögliche Schaltkreise mit b inneren Knoten und n Eingangsknoten. Da verschiedene Schaltkreise sogar gleiche Funktionen berechnen können, folgt die Aussage des Satzes für L.

Wir gehen bei der Aussage für C zuerst genauso vor. In diesem Fall gibt es höchstens $(b+n)^2$ Möglichkeiten zur Wahl der Nachfolger (im graphentheoretischen Sinn bzw. Vorgänger im Sinn des Schaltkreises) eines inneren Knoten, denn jeder Nachfolger kann einer der b inneren Knoten oder einer der n Eingangsknoten sein. Damit gibt es $(b+n)^{2b}$ mögliche Graphen. Da wir jeden der b inneren Knoten mit einer der 16 Funktionen aus B_2 markieren können, erhalten wir höchstens $16^b(b+n)^{2b}$ verschiedene Schaltkreise. Wir haben noch den Ausgabeknoten zu spezifizieren. Hierfür kommt jeder der b inneren Knoten in Frage. Daher gibt es $b \cdot 16^b(b+n)^{2b}$ mögliche Schaltkreise zur Berechnung von Funktionen. Jede der berechenbaren Funktionen wird aber $b!$ mal berechnet, da bei einer anderen Benennung der Knoten und einer entsprechend geänderten Markierung keine Änderung der Funktion erfolgt. Folglich werden höchstens soviel verschiedene Funktionen berechnet, wie im Satz angegeben ist. □

Wir wollen noch eine abgeschwächte Form der zweiten Aussage aus Lemma 1.28 angeben, die uns im Folgenden reicht und durch rein mathematische Umformungen erhalten wird. Es handelt sich zum einen um eine Abschwächung, weil die Aussage nur für hinreichend große n gilt, und zum anderen ist es eine Abschwächung, weil die darin angegebene Schranke für die Anzahl der Funktionen größer als die in Lemma 1.28 ist.

Folgerung 1.29 *Es gibt eine natürliche Zahl n_0 derart, dass für alle $n \geq n_0$*

$$C(n,b) \leq be^n(16e(n+b))^b \tag{1.22}$$

gilt.[2]

Beweis: Wir benutzen die beiden folgenden aus der Mathematik bekannten Relationen, die für hinreichend große n (d. h. für $n \geq n_0$) gelten:

$$b! \geq \left(\frac{b}{e}\right)^b \quad \text{und} \quad \left(\frac{n+b}{b}\right)^b \leq e^n\,.$$

[2]Hierbei ist e die Basis der natürlichen Logarithmen. Es gilt $e = 2,7138\ldots$

Damit folgt aus Lemma 1.28

$$\begin{aligned} C(n,b) &\leq \frac{b16^b(n+b)^{2b}}{b!} \leq b16^b\frac{(n+b)^{2b}}{(\frac{b}{e})^b} \\ &= b16^be^b(n+b)^b(\frac{n+b}{b})^b \leq b16^be^b(n+b)^be^n \\ &= be^n(16e(n+b))^b\,, \end{aligned}$$

was gerade die Behauptung besagt. □

Satz 1.30 *Für jede natürliche Zahl $n \geq 1$ gilt*

$$\frac{2^n}{\log(n)}(1-\delta(n)) < L(n)$$

mit

$$\delta(n) = \frac{\log(n)}{2^n} + 6\frac{1}{\log(n)}\,.$$

Beweis: Wir haben zu zeigen, dass es eine Funktion $f \in B_n$ gibt, deren Länge bei einer Beschreibung durch eine Formel größer als $\frac{2^n}{\log(n)}(1-\delta(n))$ ist.

Dazu setzen wir $b = \frac{2^n}{\log(n)}(1-\delta(n))$ und zeigen, dass $L(n,b) < 2^{2^n}$. Hieraus folgt die Existenz der gewünschten Form sofort, da es 2^{2^n} Funktionen in B_n gibt.

Die Aussage $L(n,b) < 2^{2^n}$ ist gleichwertig zu $\log(L(n,b)) < 2^n$. Nach Lemma 1.28 und unserer Wahl von b gilt für hinreichend großes n

$$\begin{aligned} \log(L(n,b)) &\leq (b+1)\log(n) + 6b \\ &= \Big(\frac{2^n}{\log(n)}(1-\delta(n)) + 1\Big)\log(n) + 6\frac{2^n}{\log(n)}(1-\delta(n)) \\ &< 2^n(1-\delta(n)) + \log(n) + 6\frac{2^n}{\log(n)} \\ &= 2^n - 2^n\Big(\frac{\log(n)}{2^n} + 6\frac{1}{\log(n)}\Big) + \log(n) + 6\frac{2^n}{\log(n)} \\ &= 2^n, \end{aligned}$$

womit auch $L(n,b) < 2^{2^n}$ nachgewiesen ist. □

Satz 1.31 *Es gibt eine natürliche Zahl n_0 derart, dass für alle $n \geq n_0$*

$$n - \log(\log(n)) - 2 < D(n)$$

gilt.

Beweis: Nach Satz 1.19 und Satz 1.30 gibt es eine Boolesche Funktion f, für deren Tiefe

$$\begin{aligned} D(f) &> \log\Big(\frac{2^n}{\log(n)}(1-\delta(n))+1\Big) \\ &\geq \log\Big(\frac{2^n}{\log(n)}(1-\delta(n))\Big) \\ &= n - \log(\log(n)) + \log(1-\delta(n)) \end{aligned}$$

gilt. Für hinreichend großes n ist $1-\delta(n) \geq \frac{1}{2}$ und damit $\log(1-\delta(n)) \geq -2$. Folglich ist die Beziehung

$$D(f) > n - \log(\log(n)) - 2$$

erfüllt. □

Satz 1.32 *Für jede (beliebig kleine) Zahl δ gibt es eine natürliche Zahl n_0 derart, dass für alle $n \geq n_0$*

$$\frac{2^n}{n}(1-\delta) < C(n)$$

gilt.

Beweis: Wir gehen wie beim Beweis von Satz 1.30 vor, d. h. wir zeigen, dass die Anzahl der Funktionen, deren Komplexität höchstens $\frac{2^n}{n}(1-\delta)$ ist, kleiner als die Anzahl 2^{2^n} aller Funktionen aus B_n ist. Dazu wählen wir $b = \frac{2^n}{n}(1-\delta)$ und n so groß, dass nach Folgerung 1.29 die Ungleichung (1.22) gilt. Beachten wir noch $b \leq 2^n$, so erhalten wir

$$\begin{aligned} C(n,b) &\leq 2^n e^n (16e(n+b))^b \leq (2e)^n (16e(n+b))^b \leq 8^n (16e(n+b))^b = 2^{3n}(16e(n+b))^b \\ &= 2^{3n}\Big(16e\big(n+\frac{2^n}{n}(1-\delta)\big)\Big)^{\frac{2^n}{n}(1-\delta)}. \end{aligned}$$

Da für große n offenbar $16e\big(n+\frac{2^n}{n}(1-\delta)\big) \leq 2^n$ gilt, ergibt sich

$$C(n,b) \leq 2^{3n}(2^n)^{\frac{2^n}{n}(1-\delta)} = 2^{3n+2^n(1-\delta)}.$$

Hieraus folgt

$$\frac{C(n,b)}{2^{2^n}} \leq \frac{2^{3n+2^n(1-\delta)}}{2^{2^n}} = 2^{3n-\delta 2^n} < 1,$$

da für hinreichend große n der Exponent $3n-\delta 2^n$ negativ ausfällt. Damit haben wir

$$C(n, \frac{2^n}{n}(1-\delta)) < 2^{2^n}$$

gezeigt. □

Kombinieren wir die erhaltenen oberen und unteren Schranken aus den Sätzen 1.24, 1.25, 1.27, 1.30, 1.31 und 1.32, so ergeben sich die Relationen

$$\frac{2^n}{n}(1-\delta) \leq C(n) \leq \frac{2^n}{n} + o\Big(\frac{2^n}{n}\Big)\,,$$

$$\frac{2^n}{\log(n)}(1-\delta(n)) \leq L(n) \leq \frac{2^{n+1}}{\log(n)} + o\Big(\frac{2^n}{\log(n)}\Big)\,,$$

$$n - \log(\log(n)) - 2 \leq D(n) \leq n - \log(\log(n)) + 2 + o(1)\,,$$

aus denen die folgenden Aussagen zum asymptotischen Verhalten der Funktionen $C(n)$, $L(n)$ und $D(n)$ folgen.

Satz 1.33 *Es gelten die folgenden Beziehungen:*

i) $C(n) = \Theta(\frac{2^n}{n})$,

ii) $L(n) = \Theta(\frac{2^n}{\log(n)})$,

iii) $D(n) = \Theta(n - \log(\log(n)))$. □

Wenn wir statt der von uns nicht bewiesenen Aussagen aus Satz 1.27 die von uns gezeigte schwächere Relation aus Satz 1.26 verwenden, so ergibt sich nur $D(n) = O(n)$.

In den Beweisen von Satz 1.30 und Satz 1.32 haben wir die Existenz von Funktionen mit großer Länge bzw. großer Komplexität nachgewiesen. Der Beweis sagt aber zum einen nicht aus, wie viele Funktionen mit großer Länge bzw. Komplexität es gibt, und zum anderen ist es ein reiner Existenzbeweis und gibt keinen Hinweis auf eine konkrete Funktion mit großer Länge bzw. Komplexität. Wir wollen hierzu einige Anmerkungen machen.

Im Beweis von Satz 1.32 haben wir gezeigt, dass

$$\frac{C(n, \frac{2^n}{n}(1-\delta))}{2^{2^n}} \leq 2^{3n-\delta 2^n}$$

gilt. Für den Grenzwert für $n \to \infty$ ergibt sich daraus

$$\lim_{n\to\infty} \frac{C(n, \frac{2^n}{n}(1-\delta))}{2^{2^n}} = 0\,.$$

Der Anteil der Funktionen, deren Komplexität höchstens $\frac{2^n}{n}(1-\delta)$ beträgt, geht also mit wachsendem n gegen 0, d. h. für jedes (beliebig kleine) δ haben fast alle[3] Funktionen aus B_n eine Komplexität, die größer als $\frac{2^n}{n}(1-\delta)$ ist. Somit geht die Wahrscheinlichkeit, dass eine zufällig gewählte Funktion aus B_n eine größere Komplexität als $\frac{2^n}{n}(1-\delta)$ hat, mit wachsendem n gegen 1.

Um so überraschender ist es, dass bis heute keine Funktionen bekannt sind, deren Komplexität groß ist. Das bisher beste Resultat zur Existenz konkreter komplizierter Funktionen ist der folgende Satz.

[3]Wir sagen, dass eine Aussage für fast alle Elemente einer Menge M gilt, falls M disjunkte Vereinigung von Mengen M_n ist und der Anteil der Elemente aus M_n, die die Aussage erfüllen, für $n \to \infty$ gegen 1 konvergiert.

Eine Funktion $f \in B_{2^k+3k+3}$ sei wie folgt definiert. Wir teilen die 2^k+3k+3 Variablen von f in der folgenden Weise ein

$$\begin{aligned}
\underline{x} &= (x_1, x_2, \ldots, x_{2^k}),\\
\underline{a} &= (x_{2^k+1}, x_{2^k+2}, \ldots, x_{2^k+k}),\\
\underline{b} &= (x_{2^k+k+1}, x_{2^k+k+2}, \ldots, x_{2^k+2k}),\\
\underline{c} &= (x_{2^k+2k+1}, x_{2^k+2k+2}, \ldots, x_{2^k+3k}),\\
p &= x_{2^k+3k+1},\\
q &= x_{2^k+3k+2},\\
r &= x_{2^k+3k+3},
\end{aligned}$$

bezeichnen mit a, b und c die Zahlen, deren Dualdarstellungen durch $\underline{a}$, $\underline{b}$ und $\underline{c}$ gegeben sind, wenn wir die Tupel als Dualdarstellung interpretieren, und setzen

$$\begin{aligned}
f(x_1, x_2, \ldots x_{2^k+3k+3}) &= f(\underline{x}, \underline{a}, \underline{b}, \underline{c}, p, q, r)\\
&= \left[q \wedge [(x_a \wedge x_b) \vee (p \wedge x_b \wedge x_c^r)]\right] \vee \overline{q} \wedge (x_a \oplus x_b).
\end{aligned}$$

Satz 1.34 *Es gilt* $C(f) \geq 3 \cdot 2^k - 3$. □

Dieser Satz gibt nur ein lineares Wachstum (mit dem Faktor 3) in der Anzahl von $2^k + 3k + 3$ der Variablen. Man beachte, dass auf der andere Seite Funktionen, die von allen Variablen wesentlich abhängen, lineares Wachstum aufweisen (siehe Satz 1.22).

Die obige Aussage, dass fast alle Funktionen eine große Komplexität haben, gilt für die Länge und Tiefe in entsprechender Weise. Dies kann durch Wahl von

$$\delta(n) = \frac{1}{\log(\log(n))}$$

für L wie oben leicht nachgewiesen werden und folgt dann für die Tiefe mittels Satz 1.19.

Übungsaufgaben

1. Zeigen Sie, dass die Methode zum Beweis von Satz 1.26 angewandt auf das Maß C eine zu große obere Schranke verglichen mit Satz 1.24 liefert.

2. Für natürliche Zahlen $n \geq 1$ und $b \geq 1$ setzen wir

   $$L'(n, b) = \#(\{f \mid f \in B_n, K_{\{\wedge,\vee,\neg\}}(f) \leq b\}).$$

   Bestimmen Sie $L'(3, 2)$ und geben Sie eine Abschätzung für $L'(n, b)$ an.

1.6. Minimierung von Schaltkreisen

Die bisherigen Ergebnisse haben Schranken für die Komplexität, Länge und Tiefe Boolescher Funktionen ergeben. Jedoch ist noch keine Aussage dazu getroffen worden, wie man einen günstigen Schaltkreis für eine Funktion erhält. In diesem Abschnitt soll dazu ein Beitrag geleistet werden. Wir werden allerdings nicht versuchen, einen günstigen

Schaltkreis aus der Funktion direkt heraus zu konstruieren, sondern wir werden zuerst von der disjunktiven Normalform ausgehend eine kostengünstige Variante der disjunktiven Normalform als Darstellung der Funktion erzeugen und diese dann in einen Schaltkreis umsetzen. Kostengünstig bezieht sich hier also auf ein für diese Zwecke einzuführendes Komplexitätsmaß, das aber unter Berücksichtigung der Darstellung eine Nähe zur Größe bzw. Länge aufweist.

Definition 1.35 *Es sei X eine (unendliche) Menge von Variablen.*

i) *K-Ausdrücke sind wie folgt induktiv definiert.*

- *Für jede Variable $x \in X$ sind x und $\overline{x}$ K-Ausdrücke.*
- *Ist U ein K-Ausdruck und kommt weder die Variable x selbst noch ihre Negation $\overline{x}$ in U vor, so sind $U \wedge x$ und $U \wedge \overline{x}$ auch K-Ausdrücke.*
- *Andere K-Ausdrücke gibt es nicht.*

ii) *KA-Ausdrücke sind wie folgt induktiv definiert.*

- *Jeder K-Ausdruck ist ein KA-Ausdruck.*
- *Sind V_1 und V_2 zwei KA-Ausdrücke, so ist $V_1 \vee V_2$ ein KA-Ausdruck.*
- *Andere KA-Ausdrücke gibt es nicht.*

iii) *V ist ein KA-Ausdruck für $f \in B_n$, falls $f = V$ gilt.*

Offensichtlich ist ein K-Ausdruck eine Konjunktion von paarweise verschiedenen einfachen bzw. negierten Variablen. Ein KA-Ausdruck ist dann eine Alternative von K-Ausdrücken. Damit ist jede disjunktive Normalform ein KA-Ausdruck. Somit gibt es nach Satz 1.2 auch für jede Funktion einen KA-Ausdruck.

Definition 1.36 *Die Kosten eines KA-Ausdrucks sind induktiv wie folgt definiert:*
– *Für $x \in X$ gilt $k(x) = k(\overline{x}) = 1$.*
– *Für $x \in X$ und einen K-Ausdruck U, der weder x noch $\overline{x}$ enthält, gilt*

$$k(U \wedge x) = k(U \wedge \overline{x}) = k(U) + 1.$$

– *Sind V_1 und V_2 KA-Ausdrücke, so gilt $k(V_1 \vee V_2) = k(V_1) + k(V_2)$.*

Bei den Kosten zählen wir die Anzahl der einfachen bzw. negierten Variablen im KA-Ausdruck, wobei wir mehrfach auftretende Variablen auch mehrfach zählen.

Wir merken an, dass die Kosten im Wesentlichen der Größe des Schaltkreises entsprechen, den man aus der Darstellung der Funktion als KA-Ausdruck gewinnt. Wir betrachten dazu einen KA-Ausdruck

$$V = V_1 \vee V_2 \vee \cdots \vee V_n$$

mit den K-Ausdrücken $V_1, V_2, \ldots, V_n$. Für $1 \leq i \leq n$ habe der K-Ausdruck V_i die Kosten k_i. Dann gilt

$$k(V) = \sum_{i=1}^{n} k_i \,.$$

Ferner sei m die Anzahl der verschiedenen Variablen, die in einfacher oder negierter Form im Ausdruck V vorkommen.

Wir konstruieren nun den folgenden Schaltkreis. Die m in V einfach oder negiert vorkommenden Variablen entsprechen den Eingangsknoten. Für jede der Variablen erzeugen wir zuerst die Negation. Für $1 \leq i \leq n$ konstruieren wir dann einen Schaltkreis S_i für V_i, wofür wir höchstens $k_i - 1$ $\wedge$-Gatter benötigen. Unter Verwendung von höchstens $n-1$ $\vee$-Gattern erhalten wir dann einen Schaltkreis S für V. Offenbar gilt

$$C(S) = m + \sum_{i=1}^{n}(k_i - 1) + (n-1) = m - 1 + \sum_{i-1}^{n} k_i = m - 1 + k(V)\,.$$

Der so konstruierte Schaltkreis hat die Eigenschaft, dass auf jedem Weg von einem Eingangsknoten zum Ausgangsknoten die Folge der Markierungen der inneren Knoten die Form $-^r \wedge^s \vee^t$ mit $0 \leq r \leq 1$, $0 \leq s$ und $0 \leq t$ haben. Bei Beschränkung auf derartige Schaltkreise ist der oben angegebene Schaltkreis sogar optimal. Damit stimmen bei Beschränkung auf solche Schaltkreise Komplexität der Funktion und Kosten des KA-Ausdrucks bis auf die um 1 verringerte Anzahl der Variablen überein.

Wir werden uns in diesem Abschnitt daher damit beschäftigen, zu einer Funktion einen kostengünstigen KA-Ausdruck zu ermitteln.

Definition 1.37

i) *Ein K-Ausdruck U heißt Implikant einer Funktion f, falls für jede Belegung der Variablen der K-Ausdruck U nur dann den Wert 1 annimmt, wenn auch f den Wert 1 annimmt.*

ii) *Ein K-Ausdruck U heißt Primimplikant von f, wenn U ein Implikant von f ist und bei Streichung einer beliebigen (einfachen oder negierten) Variable in U ein K-Ausdruck entsteht, der kein Implikant von f ist.*

Zu einem Ausdruck A mit Variablen $x_1, x_2, \ldots, x_n$ und einem Wert $w \in \{0,1\}$ bezeichne $A^{-1}(w)$ die Menge aller Tupel $(a_1, a_2, \ldots, a_n)$, für die der Ausdruck A unter der Belegung α mit $\alpha(x_i) = a_i$ für $1 \leq i \leq n$ den Wert w annimmnt. Ein K-Ausdruck U also genau dann ein Implikant einer Funktion f, wenn $U^{-1}(1) \subseteq f^{-1}(1)$ gilt.

Die Primimplikanten sind nach Definition die kostenmäßig minimalen Implikanten.

Mit $I(f)$ bzw. $PI(f)$ bezeichnen wir die Menge der Implikanten bzw. Primimplikanten von f.

Es sei

$$V = U_1 \vee U_2 \vee \cdots \vee U_r \tag{1.23}$$

ein KA-Ausdruck für eine Funktion f, wobei U_i für $1 \leq i \leq r$ ein K-Ausdruck ist. Dann gilt offenbar

$$f^{-1}(1) = V^{-1}(1) = U_1^{-1}(1) \cup U_2^{-1}(1) \cup \cdots \cup U_r^{-1}(1)\,. \tag{1.24}$$

Hieraus folgt sofort, dass jeder der K-Ausdrücke U_i, $1 \leq i \leq r$, ein Implikant von f ist.

Es sei U ein Implikant von f. Falls U kein Primimplikant ist, gibt es eine Variable x derart, dass x oder $\overline{x}$ in U vorkommt und der K-Ausdruck U', der aus U durch Streichen von x bzw. $\overline{x}$ entsteht, ein Implikant von f ist. Außerdem gilt $U^{-1}(1) \subset (U')^{-1}(1)$. Wir können diesen Prozess fortsetzen, solange der neu konstruierte Implikant U' kein Primimplikant ist. Daher gibt es zu jedem Implikanten U von f einen Primimplikanten W von f mit

$$U^{-1}(1) \subseteq W^{-1}(1) \subseteq f^{-1}(1) \text{ und } k(W) \leq k(U). \tag{1.25}$$

Ersetzen wir im KA-Ausdruck V aus (1.23) jeden K-Ausdruck U_i durch den zugehörigen Primimplikanten W_i, so erhalten wir den KA-Ausdruck

$$V' = W_1 \vee W_2 \vee \cdots \vee W_r\,,$$

für den wegen (1.24) und (1.25)

$$\begin{aligned} f^{-1}(1) &= U_1^{-1}(1) \cup U_2^{-1}(1) \cup \cdots \cup U_r^{-1}(1) \\ &\subseteq W_1^{-1}(1) \cup W_2^{-1}(1) \cup \cdots \cup W_r^{-1}(1) \\ &\subseteq f^{-1}(1) \end{aligned}$$

gilt. Daraus ergibt sich die Gleichheit $f^{-1}(1) = W_1^{-1}(1) \cup W_2^{-1}(1) \cup \cdots \cup W_r^{-1}(1)$. Daher ist V' ebenfalls ein KA-Ausdruck für f. Wegen (1.25) gilt noch $k(V') \leq k(V)$.

Damit haben wir das folgende Lemma bewiesen.

Lemma 1.38 *Ein hinsichtlich der Kosten minimaler KA-Ausdruck für eine Boolesche Funktion f ist die Alternative von Primimplikanten von f.* □

Wegen Lemma 1.38 sind für die Gewinnung kostengünstiger KA-Ausdrücke die Primimplikanten von besonderen Interesse. Daher suchen wir einen Algorithmus zum Bestimmen aller Primimplikanten einer Funktion f. Ein solcher Algorithmus stammt von QUINE und MCCLUSKEY. Er ist in Abbildung 1.11 angegeben.

Wir machen darauf aufmerksam, dass der Wert von i bei Eintreten der Abbruchbedingung der WHILE-Schleife die zu vereinigenden Mengen festlegt.

Wir beweisen nun die Korrektheit des Algorithmus von QUINE und MCCLUSKEY. Dazu reicht es zu zeigen, dass

- Q_i die Menge aller Implikanten U von f mit $k(U) = i$ ist und
- P_i die Menge aller Primimplikanten U von f mit $k(U) = i$ ist.

Für n ist die Aussage gültig. Zum einen besteht Q_n nach Konstruktion aus allen K-Ausdrücken $t_{\underline{a}}$ mit $\underline{a} \in f^{-1}(1)$. Wegen $m_{\underline{a}}^{-1}(1) = \{\underline{a}\} \subseteq f^{-1}(1)$ ist $t_{\underline{a}}$ auch Implikant von f.

Zum anderen enthält ein K-Ausdruck U der Länge n jede der Variablen. Daher ist $U^{-1}(1)$ einelementig. Es sei $U^{-1}(1) = \{\underline{a}\}$. Wegen $m_{\underline{a}}^{-1}(1) = \{\underline{a}\}$, gilt $U = t_{\underline{a}}$. Ist U noch Implikant von f, erfüllt also $U^{-1}(1) \subseteq f^{-1}(1)$, so ergibt sich noch $\underline{a} \in f^{-1}(1)$. Damit ist $U \in Q_n$ gezeigt.

Für $i < n$ ergibt sich die Gültigkeit aus dem folgenden Lemma.

Lemma 1.39 *Ein K-Ausdruck U, der weder x noch $\overline{x}$ enthält, ist genau dann ein Implikant für f, wenn $U \wedge x$ und $U \wedge \overline{x}$ Implikanten von f sind.*

Eingabe: Boolesche Funktion $f \in B_n$ in Form der Tabelle (entsprechend Abb. 1.1), bei der jedem n-Tupel $\underline{a}$ der Funktionswert $f(\underline{a})$ zugeordnet wird

Ausgabe: Menge $PI(f)$ der Primimplikanten von f

$Q_n := \{t_{\underline{a}} \mid \underline{a} \in f^{-1}(1) \text{ und } t_{\underline{a}}(x_1, x_2, \ldots, x_n) = x_1^{a_1} \wedge x_2^{a_2} \wedge \cdots \wedge x_n^{a_n}\}$;
$i := n$;
WHILE $Q_i \neq \emptyset$
    BEGIN
        $i := i - 1$;
        $Q_i := \{U \mid$ es gibt eine Variable x derart, dass x und $\overline{x}$
                nicht in U vorkommen und $U \wedge x, U \wedge \overline{x} \in Q_{i+1}$ $\}$;
        $P_{i+1} := \{U \mid U \in Q_{i+1}$, alle V in Q_i sind nicht Teilausdruck von U $\}$
    END ;
$PI(f) := \bigcup_{k=i+1}^{n} P_k$

Abbildung 1.11: Algorithmus zum Bestimmen der Primimplikanten (von QUINE und MCCLUSKEY)

Beweis: Es sei zuerst $U \in I(f)$. Falls $U \wedge x$ für eine Belegung $\underline{a}$ den Wert 1 annimmt, so nimmt auch U auf $\underline{a}$ den Wert 1 an. Folglich gilt auch $f(\underline{a}) = 1$. Damit ist gezeigt, dass $U \wedge x$ ein Implikant von f ist. Analog weisen wir nach, dass auch $U \wedge \overline{x}$ ein Implikant von f ist.

Es seien nun $U \wedge x$ und $U \wedge \overline{x}$ Implikanten von f. Falls bei einer Belegung $\underline{a}$ der K-Ausdruck U den Wert 1 annimmt, so nimmt mindestens einer der K-Ausdrücke $U \wedge x$ oder $U \wedge \overline{x}$ bei $\underline{a}$ den Wert 1 an, da entweder x oder $\overline{x}$ bei $\underline{a}$ den Wert 1 zugewiesen bekommen. Damit ist in jedem Fall $f(\underline{a}) = 1$, womit bewiesen ist, dass U ein Implikant von f ist. □

Die Aussage für P_k, $i \leq k \leq n$ (man beachte, dass i der Wert ist, bei dem die Schleife abgebrochen wird), folgt direkt aus der Definition des Primimplikanten.

Beispiel 1.40 *Wir betrachten eine Funktion f, die durch die Tabelle*

a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

gegeben sei. Wegen

$$f^{-1}(1) = \{(0,0,1), (0,1,1), (1,0,0), (1,0,1), (1,1,0)\}$$

ergeben sich zuerst

$$Q_3 = \{\overline{a}\overline{b}c,\ \overline{a}bc,\ a\overline{b}\overline{c},\ a\overline{b}c,\ ab\overline{c}\}$$

und daraus entsprechend dem Algorithmus

$$\begin{aligned} Q_2 &= \{\overline{a}c,\ \overline{b}c,\ a\overline{b},\ a\overline{c}\}\,, \\ P_3 &= \emptyset\,, \\ Q_1 &= \emptyset\,, \\ P_2 &= Q_2\,. \end{aligned}$$

Die Abbruchbedingung für die WHILE-Schleife ist mit $i = 1$ erreicht, und folglich erhalten wir

$$PI(f) = P_2 \cup P_3 = \{\overline{a}c, \overline{b}c, a\overline{b}, a\overline{c}\}\,.$$

Wir schätzen jetzt die Komplexität des Algorithmus von Quine und McCluskey ab. Dazu bemerken wir zuerst, dass sich Q_n in höchstens $O(n2^n)$ Schritten erzeugen läßt. Ferner gibt es 3^n K-Ausdrücke über n Variablen, denn jede Variable kann einfach oder negiert oder gar nicht vorkommen. Zur Feststellung, ob ein K-Ausdruck in Q_i liegt, müssen wir Q_{i+1} zweimal durchmustern. Wenn wir die Ausdrücke in Q_{i+1} geordnet aufschreiben, erfordert das Durchsuchen mittels binärer Suche höchstens $\log(3^n) = n\log(3)$ Schritte. Daher sind für die Bestimmung von Q_i aus Q_{i+1} maximal $O(n3^n)$ Schritte notwendig. Da höchstens n Mengen ermittelt werden müssen, ergibt sich eine Gesamtkomplexität des Algorithmus von höchstens $O(n^2 3^n)$.

Man beachte, dass die Größe der Eingabe $O(n2^n)$ beträgt, da die Tafel $(n+1)2^n$ Einträge hat. Somit ist die Komplexität des Algorithmus wegen

$$n^2 3^n = n^2(2^{\log(3)})^n = n^2(2^n)^{\log(3)} \leq (n2^n)^2$$

höchstens quadratisch in Bezug auf die Größe der Eingabe.

Mit der Kenntnis aller Primimplikanten ist die Bestimmung des kostengünstigsten KA-Ausdrucks für eine Funktion aber noch nicht abgeschlossen. Die Alternative aller Primimplikanten muss nämlich nicht die kostengünstigste Variante sein. So gilt z. B. für die Funktion aus Beispiel 1.40

$$f(a,b,c) = \overline{a}c \vee \overline{b}c \vee a\overline{b} \vee a\overline{c} = \overline{a}c \vee \overline{b}c \vee a\overline{c}\,,$$

wobei der erste der beiden KA-Ausdrücke der Alternative aller Primimplikanten von f entspricht. Es sind also nur einige der Primimplikanten der Funktion erforderlich. Wir geben daher einen Algorithmus zur Auswahl von Primimplikanten an.

Unter der PI-Tafel einer Funktion f verstehen wir eine Matrix, deren Zeilen den Primimplikanten und deren Spalten den Tupeln aus $f^{-1}(1)$ entsprechen. Wir schreiben in den Schnittpunkt der Zeile zu U und der Spalte zu $\underline{a}$ den Wert $U(\underline{a})$.

Der Algorithmus aus Abbildung 1.12 reduziert nun schrittweise die Tafel durch Streichen von Zeilen und/oder Spalten. Das Streichen einer Zeile wird von der Aufnahme des zugehörigen Primimplikanten in einen kostengünstigen KA-Ausdruck V begleitet.

Eingabe: PI-Tafel einer Funktion f

Ausgabe: Teilmenge R von Primimplikanten von f und reduzierte PI-Tafel

$R := \emptyset$;
$T_0 :=$ Matrix mit einer Zeile und Spalte und Eintrag 0 ;
$T_1 :=$ PI-Tafel von f ;
$i := 1$;
WHILE $T_i \neq T_{i-1}$
    BEGIN $M_i := \{\underline{a} \mid$ zu $\underline{a}$ gehörende Spalte enthält genau eine $1\}$;
        $N_i := \{U \mid U$ ist Primimplikant in $T_i, U(\underline{a}) = 1$ für ein $\underline{a} \in M_i\}$;
        $R := R \cup N_i$;
        T_i' entstehe aus T_i durch Streichen aller Zeilen $U \in N_i$ und
            Streichen aller Spalten zu $\underline{b}$ mit $U(\underline{b}) = 1$ für ein $U \in N_i$;
        T_{i+1} entstehe aus T_i' durch
            Streichen aller Spalten c, für die eine Spalte c' mit $c' \leq c$ existiert;
        $i := i + 1$
    END

Abbildung 1.12: Algorithmus zum Reduzieren einer PI-Tafel

Die Korrektheit des Algorithmus folgt aus den zwei nachfolgenden Bemerkungen und der Tatsache, dass es ausreicht, wenn

$$\text{zu jedem } \underline{a} \text{ ein Primimplikant } U \text{ mit } U(\underline{a}) = 1 \text{ existiert.} \tag{1.26}$$

Bemerkung 1: Die Spalte zu $\underline{a} \in f^{-1}(1)$ enthalte genau eine 1, und die 1 stehe in der Zeile zu U. Dann gilt für alle anderen Primimplikanten U' die Relation $\underline{a} \notin (U')^{-1}(1)$. Wegen (1.24) muss daher jeder KA-Ausdruck V, der eine mehrfache Alternative von Primimplikanten ist, U als einen der Primimplikanten enthalten. Daher nehmen wir U in die Menge R der Primimplikanten für den kostengünstigen KA-Ausdruck auf. Die Spalten zu einem $\underline{b}$ mit $U(\underline{b}) = 1$ können gestrichen werden, da für sie U bereits die Forderung aus (1.26) absichert.

Bemerkung 2: Für die Spalten c' und c zu den Tupeln $\underline{a'}$ und $\underline{a}$ gelte die Beziehung $c' \leq c$, d. h. für jeden Primimplikanten U folgt aus $U(\underline{a'}) = 1$ auch $U(\underline{a}) = 1$. Wir benötigen ein U mit $U(\underline{a'}) = 1$. Dieser Primimplikant U erfüllt dann auch $U(\underline{a}) = 1$ und sichert damit die Forderung (1.26) für $\underline{a}$ ebenfalls ab.

Beispiel 1.40 (Fortsetzung) *Als PI-Tafel für f erhalten wir*

	$(0,0,1)$	$(0,1,1)$	$(1,0,0)$	$(1,0,1)$	$(1,1,0)$
$\overline{a}c$	1	1	0	0	0
$\overline{b}c$	1	0	0	1	0
$a\overline{b}$	0	0	1	1	0
$a\overline{c}$	0	0	1	0	1

Wir wenden den Algorithmus zum Reduzieren der Tafel an. Die Spalten zu $(0,1,1)$ *und* $(1,1,0)$ *enthalten jeweils genau ein 1, die in den Zeilen zu* $\overline{a}c$ *bzw.* $a\overline{c}$ *stehen. Damit ergibt*

sich

$$R = \{\overline{a}c, a\overline{c}\}\,. \tag{1.27}$$

Weiterhin entsteht T_1' durch Streichen der Zeilen zu $\overline{a}c$ und $a\overline{c}$ und aller Spalten, in denen eine 1 in den gestrichenen Zeilen steht. Dies liefert

	$(1,0,1)$
$\overline{b}c$	1
$a\overline{b}$	1

als Tafel. Da T_1' nur ein Spalte enthält, gilt trivialerweise $T_2 = T_1'$. Da die einzige Spalte von T_2 zwei Einsen enthält, erhalten wir $T_2' = T_3 = T_2$. Damit ist die Abbruchbedingung erreicht.

Der Algorithmus ist polynomial bez. der Stelligkeit n der Funktion, der Anzahl r der Zeilen (d. h. Anzahl der Primimplikanten, die höchstens 3^n ist) und der Anzahl s der Spalten (d. h. $\#(f^{-1}(1))$ und damit höchstens 2^n). Die Notwendigkeit eines Streichens ist durch das Durchmustern aller Elemente der Tafel (d. h. von höchstens $2^n 3^n = 6^n$ Elementen) feststellbar, und auch das Streichen selbst erfordert höchstens den Aufwand 6^n. Da höchstens 2^n Spalten und höchstens 3^n Zeilen gestrichen werden können, ergibt sich höchstens ein Zeitaufwand von $(2^n + 3^n)2 \cdot 6^n$, der polynomial in 2^n und 3^n ist. Bezüglich der Größe der ursprünglichen Eingabe (Tafel für f) ist dies auch polynomial.

Wir müssen nun noch eine Auswahl unter den Primimplikanten der reduzierten PI-Tafel vornehmen. Dies muss so geschehen, dass (1.26) erfüllt ist und die Kosten minimal ausfallen.

Beispiel 1.40 (Fortsetzung) Die beiden Primimplikanten $\overline{b}c$ und $a\overline{b}$ haben beide die gleichen Kosten, und es reicht, einen der beiden Primimplikanten zu wählen. Wenn wir $\overline{b}c$ wählen, ergibt sich der minimale KA-Ausdruck

$$\overline{a}c \vee a\overline{c} \vee \overline{b}c$$

mit den Gesamtkosten 6. Bei Wahl von $a\overline{b}$ erhalten wir den KA-Ausdruck

$$\overline{a}c \vee a\overline{c} \vee a\overline{b}$$

für f, der ebenfalls die Kosten 6 hat.

Beispiel 1.40 zeigt, dass ein kostenminimaler KA-Ausdruck nicht eindeutig bestimmt ist.

Jedoch ließ sich ein minimaler KA-Ausdrucks in Beispiel 1.40 aus der reduzierten PI-Tafel sehr einfach bestimmen. Wir wollen nun zeigen, dass dies im allgemeinen eine schwierige Aufgabe ist.

Satz 1.41 *Das Problem des Ermittelns eines minimalen KA-Ausdrucks für eine beliebige Funktion f aus der reduzierten PI-Tafel von f ist NP-vollständig.*

Beweis: Wir weisen zuerst nach, dass die Minimierung durch einen nichtdeterministischen Algorithmus in polynomialer Zeit in der Anzahl der Spalten und Zeilen der PI-Tafel erreicht werden kann. Dazu reicht es offenbar zu zeigen, dass es einen nichtdeterministischen Algorithmus gibt, der in polynomialer Zeit einen KA-Ausdruck aus den Primimplikanten der PI-Tafel bestimmt, dessen Kosten kleiner einer vorgegebenen Konstanten k sind. Wir haben diesen Algorithmus dann nur der Reihe nach für $k = 1$, $k = 2$, ... durchzuführen. Durch das erste k, für das die Antwort positiv ausfällt, wird der minimale Wert für die Kosten gegeben, und der (als existent angenommene) Algorithmus liefert einen zugehörigen KA-Ausdruck. Da die Kosten der Alternative aller Primimplikanten der Tafel eine polynomiale Schranke für die minimalen Kosten liefert, ist insgesamt nur ein polynomialer Aufwand zum Bestimmen eines minimalen KA-Ausdrucks erforderlich.

Es sei r die Anzahl der Primimplikanten der Tafel. Nichtdeterministisch läßt sich jede mögliche Alternative von Primimplikanten, wovon es 2^r gibt (der Primimplikant wird in die Alternative aufgenommen oder nicht) in $O(r)$ Schritten erzeugen. Die Kosten einer Alternative sind auch in $O(r)$ Schritten berechenbar. Folglich können wir in $O(r)$ Schritten nichtdeterministisch ermitteln, ob ein KA-Ausdruck existiert, dessen Kosten höchstens k sind.

Wir zeigen nun, dass das Überdeckungsproblem

Gegeben: r nichtleere Teilmengen $S_1, S_2, \ldots, S_r$ von $M = \{1, 2, \ldots, s\}$ mit $\bigcup_{i=1}^{r} S_i = M$,

Gesucht: Mengen $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ derart, dass $\bigcup_{j=1}^{k} S_{i_j} = M$ und $\bigcup_{l=1}^{k-1} S_{u_l} \subset M$ für jede Wahl von $S_{u_1}, S_{u_2}, \ldots, S_{u_{k-1}}$

auf die Bestimmung eines minimalen KA-Ausdrucks aus einer reduzierten PI-Tafel in polynomialer Zeit reduziert werden kann. Die Behauptung des Satzes folgt dann aus der bekannten Tatsache, dass das Überdeckungsproblem NP-vollständig ist.

Wir interpretieren das Überdeckungsproblem durch eine Tafel. Dazu assoziieren wir mit jeder Zahl i, $1 \leq i \leq s$, eine Spalte und mit jeder Menge S_j, $1 \leq j \leq r$, eine Zeile. Wir tragen in den Schnittpunkt der Zeile zu S_j und der Spalte zu i eine 1 ein, falls $i \in S_j$ gilt. In allen anderen Fällen tragen wir eine 0 ein. Enthält die Spalte zu i genau eine 1, sagen wir in der Zeile zu S_j, so muss S_j berücksichtigt werden, da sonst i nicht in der Vereinigung liegt. Daher können wir eine Reduktion der zugehörigen Tafel wie bei der PI-Tafel vornehmen. Die reduzierte Tafel hat dann die Eigenschaft, dass in jeder Spalte mindestens zwei Einsen stehen, keine zwei Spalten miteinander bez. $\leq$ vergleichbar sind und jede Zeile mindestens eine 1 enthält (da Zeilen, die nur aus Nullen bestehen, nichts zu M beitragen). Offensichtlich ist auch das Überdeckungsproblem für Mengen S_i, $1 \leq i \leq r$ und M, deren Beschreibung eine reduzierte Tafel ist, bereits NP-vollständig, da die Reduktion in polynomialer Zeit (in der Anzahl der Zeilen und Spalten) vorgenommen werden kann.

Wir zeigen nun, dass es zu jeder Tafel T, die einem reduzierten Überdeckungsproblem entspricht, eine Funktion f gibt, deren reduzierte PI-Tafel mit T übereinstimmt und deren Primimplikanten alle die gleichen Kosten haben. Wegen der gleichen Kosten

aller Primimplikanten, reicht es eine minimale Zahl von Primimplikanten zu bestimmen, deren Alternative f entspricht, da diese Alternative dann ein minimaler KA-Ausdruck ist. Damit ist eine Reduktion des Überdeckungsproblems auf die Bestimmung minimaler KA-Ausdrücke gegeben.

Es sei n die Anzahl der Zeilen von T. Die Spalten von T sind dann Elemente von $\{0,1\}^n$. Ferner sei S die Menge der Spalten von T. Wir definieren die Funktion $f \in B_{n+2}$ durch

$$f(a_1,\ldots,a_n,b_1,b_2) = \begin{cases} 1, & \text{für } a_1+\cdots+a_n \neq 0, b_1 = b_2 = 0, \\ 1, & \text{für } 0^n \neq (a_1,\ldots,a_n) \notin S, a_1 \oplus \cdots \oplus a_n = 0, b_1 = 1, b_2 = 0, \\ 1, & \text{für } 0^n \neq (a_1,\ldots,a_n) \notin S, a_1 \oplus \cdots \oplus a_n = 1, b_1 = 0, b_2 = 1, \\ 0, & \text{sonst.} \end{cases}$$

Die Menge der Primimplikanten von f ist dann durch

$$\begin{aligned} PI(f) = &\{x_i \cdot \overline{x_{n+1}} \cdot \overline{x_{n+2}} \mid 1 \leq i \leq n\} \\ &\cup \{m_{(a_1,\ldots,a_n)}\overline{x_{n+1}} \mid a_1+\cdots+a_n \neq 0, (a_1,\ldots,a_n) \notin S, a_1 \oplus \cdots \oplus a_n = 0\} \\ &\cup \{m_{(a_1,\ldots,a_n)}\overline{x_{n+2}} \mid a_1+\cdots+a_n \neq 0, (a_1,\ldots,a_n) \notin S, a_1 \oplus \cdots \oplus a_n = 1\} \end{aligned}$$

gegeben. Dies ist wie folgt einzusehen.

Aus der Definition von f folgt sofort, dass die K-Ausdrücke aus $PI(f)$ alle Implikanten f sind.

Es sei nun w ein Implikant von f. Da $f(\underline{a},1,1) = 0$ für alle $\underline{a} \in \{0,1\}$ gilt, muss mindestens einer (oder beide) der K-Ausdrücke $\overline{x_{n+1}}$ und $\overline{x_{n+2}}$ in w vorkommen. Kommen beide diese Ausdrücke in w vor, so muss wegen $f(0,\ldots,0,0,0) = 0$ mindestens eine der Variablen x_i, $1 \leq i \leq n$, unnegiert in w kommen, womit w eine Verlängerung von $x_i\overline{x_{n+1}x_{n+2}}$ ist.

Kommt $\overline{x_{n+1}}$ in w vor und kommt $\overline{x_{n+2}}$ nicht in w vor, so enthält w jede Variable. Angenommen, w enthält weder x_i noch $\overline{x_i}$. Wir wählen nun $a_1, a_2, \ldots, a_n$ so, dass

$$w(a_1, a_2, \ldots, a_n, 0, 1) = 1\,.$$

Da x_i und $\overline{x_i}$ in w nicht vorkommen, so gilt auch

$$w(a_1, \ldots, a_{i-1}, \overline{a_i}, a_{i+1}, \ldots, a_n, 0, 1) = 1.$$

Damit ergibt sich aus der Implikantendefinition

$$f(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n, 0, 1) = f(a_1, \ldots, a_{i-1}, \overline{a_i}, a_{i+1}, \ldots, a_n, 0, 1) = 1.$$

Aus der Definition von f erhalten wir noch

$$a_1 \oplus \cdots \oplus a_{i-1} \oplus a_i \oplus a_{i+1} \oplus \cdots \oplus a_n = a_1 \oplus \cdots \oplus a_{i-1} \oplus \overline{a_i} \oplus a_{i+1} \oplus \cdots \oplus a_n = 1,$$

woraus der Widerspruch $a_i = \overline{a_i}$ folgt. Wenn w aber jede Variable x_i, $1 \leq i \leq n$, enthält, muss $w = m_{\underline{a}}\overline{x_{n+1}}$ für ein $\underline{a} \in \{0,1\}^n$ sein. Wegen $w(\underline{a},0,1) = f(\underline{a},0,1) = 1$ sind $\underline{a} \neq 0^n$ und $\underline{a} \notin S$ gültig. Damit ist w einer der Ausdrücke aus $PI(f)$.

Durch analoge Schlüsse kann $w \in PI(f)$ für den Fall zeigen, dass $\overline{x_{n+2}}$ aber nicht $\overline{x_{n+1}}$ in w vorkommt.

Wir reduzieren nun die Tafel der Primimplikanten. Die Spalte zu $(\underline{a}, 0, 1)$ mit $\underline{a} \neq 0^n$, $\underline{a} \notin S$ und $\bigoplus_{i=1}^n a_i = 1$ enthält nur in der Zeile zu $m_{\underline{a}}\overline{x_{n+1}}$ eine Eins. Damit streichen wir sowohl diese Zeile und Spalte, als auch die Spalte zu $(\underline{a}, 0, 0)$, in deren Zeile zu $m_{\underline{a}}\overline{x_{n+1}}$ ebenfalls eine Eins steht. Auf diese Weise streichen wir alle Zeilen zu Primimplikanten der Form $m_{\underline{a}}\overline{x_{n+1}}$ und alle Spalten zu $(\underline{a}, 0, 0)$ und $(\underline{a}, 0, 1)$ mit $\underline{a} \neq 0^n$, $\underline{a} \notin S$ und $\bigoplus_{i=1}^n a_i = 1$.

Analog kommt es zur Streichung aller Zeilen zu Primimplikanten der Form $m_{\underline{a}}\overline{x_{n+2}}$ und aller Spalten zu $(\underline{a}, 0, 0)$ und $(\underline{a}, 1, 0)$ mit $\underline{a} \neq 0^n$, $\underline{a} \notin S$ und $\bigoplus_{i=1}^n a_i = 0$.

Damit verbleiben die Primimplikanten $m_i = x_i\overline{x_{n+1}}\overline{x_{n+2}}$, $1 \leq i \leq n$, und die Spalten zu $(\underline{a}, 0, 0)$ mit $\underline{a} \in S$. Da nun noch $m_i(a_1, \ldots, a_i, \ldots, a_n, 0, 0) = a_i$ gilt, stimmt die Spalte zu $(\underline{a}, 0, 0)$ mit $(a_1, a_2, \ldots, a_n) = \underline{a}$ überein, d. h. die Spalte zu $(\underline{a}, 0, 0)$ ist eine Spalte von T. Damit ist gezeigt, dass die durch diesen Reduktionsschritt entstandene Tafel T ist. Da T nicht weiter reduzierbar ist, muss T die reduzierte PI-Tafel von f sein. □

Die bisherigen Ergebnisse belegen, dass das Auffinden kostenminimaler KA-Ausdrücke zu einer Funktion eine nicht einfache Aufgabe ist. Es ist daher nicht anzunehmen, dass die Konstruktion minimaler Schaltkreise für f bez. der Größe oder Länge einfach ist.

Abschließend wollen wir anhand eines Beispiels zeigen, wieweit die minimalen Kosten und die minimale Größe voneinander abweichen können.

Beispiel 1.42 *Wir betrachten die Funktion*

$$f(x_1, x_2, \ldots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n\,.$$

Offensichtlich ist f durch einen Schaltkreis berechenbar, der aus $n-1$ $\oplus$-Gattern besteht. Damit gibt es für jede vollständige Menge $\mathcal{S}$ nach Satz 1.17 eine Konstante $c_{\mathcal{S}}$ mit

$$C_{\mathcal{S}}(f) \leq c_{\mathcal{S}}(n-1)\,.$$

Andererseits können wir wie im vorhergehenden Beweis zeigen, dass jeder Primimplikant von f jede Variable enthält (denn Tupel die sich nur an einer Stelle unterscheiden nehmen bei f verschiedene Werte an). Damit ist

$$U = \bigvee_{\underline{a} \in f^{-1}(1)} m_{\underline{a}}\,,$$

der (eindeutig bestimmte) kostenminimalen KA-Ausdruck zu f. Dessen Kosten betragen aber

$$k(U) = n \cdot 2^{n-1}\,,$$

da es 2^{n-1} Tupel gibt, auf denen f den Wert 1 annimmt (wir geben die ersten $n-1$ Werte des Tupels beliebig vor und ergänzen die letzte Komponente so, dass sich bei f der Wert 1 ergibt).

Übungsaufgaben

1. Bestimmen Sie die Kosten des KA-Ausdrucks $A = x_1\overline{x_2}x_3 \vee x_1x_2\overline{x_3} \vee \overline{x_1x_2}x_3$. Ist A ein kostenminimaler Ausdruck für die durch A gegebene Funktion?

2. Gegeben sei die Funktion $f(x_1, x_2, x_3, x_4, x_5)$, für die $f(x_1, x_2, x_3, x_4, x_5) = 1$ genau dann gilt, wenn $x_1 + x_2 + x_3 = x_4 + x_5$ (normale Addition) gilt. Bestimmen Sie alle Implikanten und Primimplikanten von f.

3. Geben Sie die Funktion „`if` x `then` y `else` z“ an und bestimmen Sie alle Implikanten und Primimplikanten.

4. Leiten Sie einen kostenminimalen KA-Ausdruck für die Funktion $f(x_1, x_2, x_3, x_4)$ mit $f(x_1, x_2, x_3, 0) = x_1 \oplus x_2 \oplus x_3$ und $f(x_1, x_2, x_3, 1) = \overline{f(x_1, x_2, x_3, 0)}$ her.

1.7. Verzweigungsprogramme und ihre Komplexität

Gegenstand dieses Abschnitts sind Verzweigungsprogramme, die ein weiteres Modell zur Berechnung von Booleschen Funktionen darstellen. Nach den Definitionen der Verzeigungsprogramme, der von ihnen berechneten Funktion und Komplexitätsmaßen auf Verzweigungsprogrammen geben wir einige Abschätzungen für die Komplexitäten.

Definition 1.43

i) *Ein Verzweigungsprogramm ist ein gerichteter Baum*[4] *mit Kanten- und Knotenmarkierungen und drei Sorten von Knoten:*

- *genau einer Quelle, die mit einer Booleschen Variablen markiert ist, deren Eingangsgrad 0 und Ausgangsgrad 2 betragen und von den beiden vom Knoten ausgehenden Kanten ist die eine mit 0 und die andere mit 1 markiert,*
- *inneren Knoten, die mit einer Booleschen Variablen markiert sind, deren Eingangsgrad mindestens* 1 *ist, deren Ausgangsgrad 2 ist und von den beiden vom Knoten ausgehenden Kanten ist die eine mit 0 und die andere mit 1 markiert, und*
- *genau zwei Senken, die mit 0 bzw. 1 markiert sind und deren Ausgangsgrade 0 sind.*

ii) *Es sei G ein Verzweigungsprogramm, dessen innere Knoten mit x_i, $1 \leq i \leq n$, markiert sind. Wir ordnen G wie folgt eine Funktion f_G zu: Es sei*

$$\underline{a} = (a_1, a_2, \ldots, a_n) \in \{0,1\}^n.$$

Wir beginnen mit der Quelle und folgen stets bei einem Knoten x_i, $1 \leq i \leq n$, der mit a_i markierten Kante. Der Funktionswert $f_G(\underline{a})$ ist die Markierung der erreichten Senke.

[4]Ein gerichteter Graph heißt gerichteter Baum, falls er keinen gerichteten Kreis enthält.

Wir machen darauf aufmerksam, dass zwei verschiedene innere Knoten mit der gleichen Variablen markiert sein dürfen.

Die Quelle ist die Wurzel des Baumes, und die Senken sind die (beiden einzigen) Blätter des Baumes. Nach der Definition ist klar, dass jeder (maximal lange) Weg in einem Verzweigungsprogramm von der Quelle zu einer Senke führt. Außerdem gibt es bei vorgegebenem Tupel $\underline{a}$ genau einen Weg von der Quelle zu einer Senke, wenn man so vorgeht, wie es die Berechnung eines Funktionswertes erfordert.

Beispiel 1.44 *Wir betrachten den Graphen G aus Abbildung 1.13, der offensichtlich ein Verzweigungsprogramm ist.*

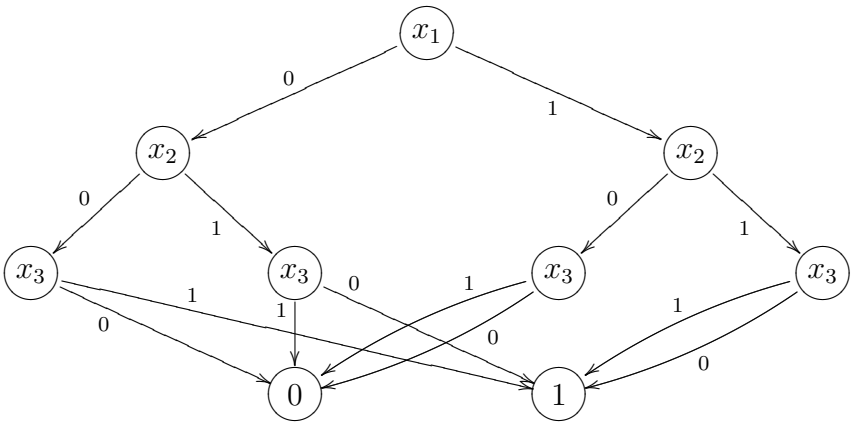


Abbildung 1.13: Verzweigungsprogramm G

Betrachten wir das Tupel $(0,1,0)$*, so beginnt der zugehörige Weg in der Wurzel* x_1*, geht entlang der mit* 0 *markierten Kante zum linken der mit* x_2 *markierten Knoten, dann entlang der mit* 1 *markierten Kante zum (von links gerechnet) zweiten mit* x_3 *markierten Knoten und schließlich entlang der mit* 0 *markierten Kante zur mit* 1 *markierten Senke. Bei abwechselnder Angabe von Knoten und Kantenmarkierung erhalten wir* $x_1 - 0 - x_2 - 1 - x_3 - 0 - 1$ *als Beschreibung des Weges. Damit ergibt sich* $f_G(0,1,0) = 1$*. Entsprechend wird für das Tupel* $(1,0,0)$ *der Weg* $x_1 - 1 - x_2 - 0 - x_3 - 0 - 0$ *und folglich* $f_G(1,0,0) = 0$ *erhalten. Insgesamt ergibt sich die Tafel*

x_1	x_2	x_3	$f_G(x_1,x_2,x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

die durch die Funktion

$$f_G(x_1,x_2,x_3) = (\overline{x_1} \wedge (x_2 \oplus x_3)) \vee (x_1 \wedge x_2)$$

beschrieben ist.

Definition 1.45

i) *Für ein Verzweigungsprogramm G sind die Größe $VC(G)$ und die Tiefe $VD(G)$ als die um 1 erhöhte Anzahl der inneren Knoten von G bzw. die Tiefe von G definiert.*

ii) *Für eine Boolesche Funktion f und $K \in \{C, D\}$ setzen wir*

$$VK(f) = \min\{VK(G) \mid G \text{ berechnet } f\}\,.$$

Die Größe $VC(G)$ gibt die Anzahl der Knoten an, die mit einer Variablen markiert sind.

Beispiel 1.44 (Fortsetzung) *Für das Verzweigungsprogramm G aus Abb. 1.13 ergeben sich $VC(G) = 7$ und $VD(G) = 3$. Damit gelten auch $VC(f_G) \leq 7$ und $VD(f_G) \leq 3$. Bezüglich der Größe ist das Verzweigungsprogramm G aus Abb. 1.13 nicht optimal für f_G, da das Verzweigungsprogramm G' aus Abb. 1.14 mit $VC(G') = 5$ auch f_G berechnet. (Da in G vom dritten und vierten Knoten mit der Markierung x_3 (von links nach rechts gelesen) beide Kanten zu den Senken 0 bzw. 1 gehen, können wir diese Knoten weglassen und direkt zu den Senken gehen, ohne den Funktionswert zu ändern. So entsteht G' aus G.)*

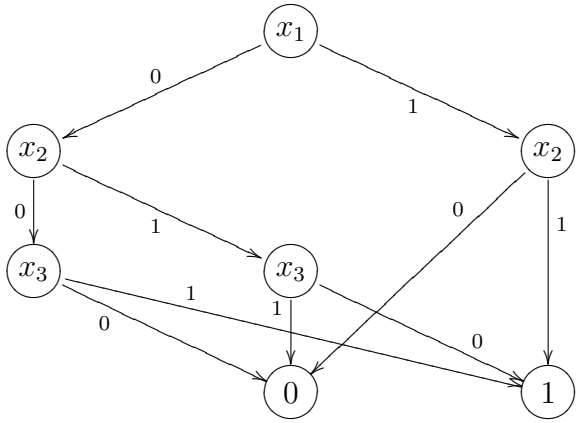


Abbildung 1.14: Verzweigungsprogramm G'

Damit haben wir $VC(f_G) \leq 5$ und $VD(f_G) \leq 3$. Ohne Beweis merken wir an, dass diese Abschätzungen optimal sind, d. h. es gelten

$$VC(f_G) = 5 \quad \textit{und} \quad VD(f_G) = 3\,.$$

Verzweigungsprogramme entsprechen Programmen, bei denen nur Anweisungen der Form

IF $x_i = 0$ THEN GOTO a ELSE GOTO b ,
$f := 0$,
$f := 1$

sind. Bei dieser Interpretation entspricht die Tiefe der Rechenzeit des Programms und die Größe der Anzahl der IF-THEN-ELSE-Anweisungen.

Wir geben nun eine obere Schranke für die Komplexität Boolescher Funktionen bei Realisierung durch Verzweigungsprogramme. Im Wesentlichen gewinnen wir die Schranken durch eine Verallgemeinerung der Konstruktion aus Abb. 1.13, bei der zuerst nach der Belegung von x_1, dann nach der von x_2 usw. gefragt wird.

Satz 1.46 *Für jede Boolesche Funktion $f \in B_n$ gelten*

$$VC(f) \leq 2^n - 1 \quad \textit{und} \quad VD(f) \leq n\,.$$

Beweis: Wir konstruieren einen vollständigen ungerichteten binären Baum der Tiefe n. Von den beiden Kanten, die von einem Knoten ausgehen, der kein Blatt ist, markieren wir eine mit 0 und die andere mit 1. Wir markieren die Wurzel (Quelle) mit x_1. Ferner markieren wir die Nachfolger eines Knotens, der mit x_i, $1 \leq i \leq n-1$, markiert ist, beide mit x_{i+1}. Ist die Folge der Kantenmarkierungen auf dem Weg von der Wurzel zu einem Nachfolger z eines mit x_n markierten Knotens $a_1 a_2 \ldots a_n$, so markieren wir z mit $f(a_1, a_2, \ldots, a_n)$.

Um aus dem binären Baum ein Verzweigungsprogramm G zu erhalten, führen wir Richtungen von mit x_i markierten Knoten zu mit x_{i+1} markierten Knoten und von mit x_n markierten Knoten zu den Blättern ein und identifizieren dann alle mit 0 und alle mit 1 markierten Blätter. Nach Konstruktion wird durch dieses Verzweigungsprogramm offenbar die Funktion f berechnet. Ferner gelten

$$VD(G) = n \quad \text{und} \quad VC(G) = 1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1\,,$$

womit der Satz bewiesen ist. □

Die unteren Schranken gewinnen wir durch eine Simulation von Verzweigungsprogrammen durch Schaltkreise.

Satz 1.47 *Für jede vollständige Menge $\mathcal{S}$ gibt es zwei Konstanten c_1 und c_2 derart, dass für jede Boolesche Funktion $f \in B_n$*

$$C_{\mathcal{S}}(f) \leq c_1 \cdot (VC(f) + 2) \quad \textit{und} \quad D_{\mathcal{S}}(f) \leq c_2 \cdot (VD(f) + 1)$$

gelten.

Beweis: Wir betrachten zuerst die Größen C bzw. VC. Es sei G ein Verzweigungsprogramm, das optimal für die Funktion $f \in B_n$ ist, d. h. es gilt $VC(G) = VC(f)$.

Mit s bezeichnen wir die Funktion, die durch

$$s(x, y, z) = \overline{x}y \vee xz$$

definiert ist. Nimmt x den Wert 0 an, so gibt s den Wert von y aus. Nimmt x dagegen den Wert 1 an, so wird von s der Wert von z ausgegeben.

Wir konstruieren aus G wie folgt einen Schaltkreis S. Wir verwenden die n mit $x_1, x_2, \ldots, x_n$ markierten Eingangsknoten. Anstelle eines inneren Knotens, der in G mit x_i markiert ist, benutzen wir in S einen Knoten der mit s markiert ist, und als Eingaben

dieses Knotens verwenden wir x_i für x und für y bzw. z die Knoten, zu denen in G mit 0 bzw. 1 bewertete Kanten führen. Analog verfahren wir mit der Quelle. Die Senke, die mit $a \in \{0,1\}$ markiert ist, ersetzen wir im Schaltkreis durch einen mit der konstanten Funktion k_a markierten Knoten, dessen Eingänge von gewissen Eingangsknoten kommen. Als Ausgabeknoten von S verwenden wir den Knoten, der der Wurzel von G entspricht. (Die inneren Knoten des Schaltkreises S entsprechen den Knoten (einschließlich der Quelle und den Senken) des Verzweigungsprogramms, wobei die Richtung der Kanten umgekehrt wird, und zusätzlich Kanten zu den Eingabeknoten gezogen werden.)

Wegen der Markierung der inneren Knoten des Schaltkreises mit s wird bei $x_i = 0$ der Wert weitergeleitet, der von dem Knoten kommt, der auch in G bei $x_i = 0$ erreicht wird. Entsprechendes gilt für $x_i = 1$ ebenfalls. Folglich wird bei der Eingabe $(a_1, a_2, \ldots, a_n)$ im Schaltkreis der Wert ausgegeben, der im Verzweigungsprogramm erreicht wird. Somit berechnet der Schaltkreis auch f.

Offensichtlich gilt $C(S) = VC(G) + 2$.

Der konstruierte Schaltkreis benutzt die Funktionen s, k_0 und k_1 und ist deshalb unter Umständen kein Schaltkreis über $\mathcal{S}$. Wir ersetzen jeden in S mit s, k_0 bzw. k_1 markierten Knoten durch einen Schaltkreis, der die entsprechende Funktion berechnet. Der so erhaltene Schaltkreis S' ist ein Schaltkreis über $\mathcal{S}$. Mit

$$c_1 = \max\{C_{\mathcal{S}}(s), C_{\mathcal{S}}(k_0), C_{\mathcal{S}}(k_1)\}$$

erhalten wir offenbar

$$C(S') \leq c_1 \cdot C(S) = c_1(VC(G) + 2)$$

und damit

$$C_{\mathcal{S}}(f) \leq c_1(VC(f) + 2)\,.$$

Für die Tiefe kann ein analoger Beweis gegeben werden. □

Wir bemerken, dass sich für $\mathcal{S} = B_2$ die Werte $c_1 = 3$ und $c_2 = 2$ ergeben.

In Analogie zu den Betrachtungen zur Komplexität auf der Basis von Schaltkreisen setzen wir

$$VK(n) = \max\{VK(f) \mid f \in B_n\}$$

für $K \in \{C, D\}$.

Unter Verwendung der Aussagen des Abschnitts 1.5. erhalten wir damit den folgenden Satz.

Satz 1.48 *Für hinreichend großes n gelten*

$$\frac{2^n}{3n} - 2 \leq VC(n) \leq 2^n - 1$$

und

$$\frac{n - \log(n) - c}{2} - 1 \leq VD(n) \leq n\,,$$

wobei c eine Konstante ist.

Übungsaufgaben

1. Bestimmen Sie ein Verzweigungsprogramm für die Funktion $f(x_1, x_2, x_3, x_4)$ mit $f(x_1, x_2, x_3, 0) = x_1 \oplus x_2 \oplus x_3$ und $f(x_1, x_2, x_3, 1) = \overline{f(x_1, x_2, x_3, 0)}$.

2. Bestimmen Sie Verzweigungsprogramme für die Funktion $x_1\overline{x_2}x_3 \vee x_1x_2\overline{x_3} \vee \overline{x_1x_2}x_3$, die minimal bez. Größe bzw. Tiefe sind.

1.8. Schaltkreise versus Turing-Maschinen

In diesem Abschnitt wollen wir einen Zusammenhang zwischen der Zeitkomplexität für die Entscheidung einer Sprache durch deterministische Turing-Maschinen und der Komplexität von Schaltkreisen herstellen. Dabei setzen wir voraus, dass der Leser über Grundkenntnisse über Turing-Maschinen aus der Grundvorlesung Theoretische Informatik verfügt. Als Referenzen geben wir die Lehrbücher [18], [20] und [22] an.

Zuerst wollen wir dabei klären, welches Turing-Maschinen-Modell wir verwenden wollen. Wir betrachten deterministische Turing-Maschinen $M = (X, Z, z_0, Q, \delta)$, wobei X das Eingabealphabet, Z die Menge der Zustände, z_0 den Anfangszustand, Q die Menge der Endstände und δ die Überführungsfunktion bezeichnen. Die Überführungsfunktion δ ist eine Funktion von $(Z \setminus Q) \times (X \cup \{*\})$ in $Z \times (X \cup \{*\}) \times \{R, N, L\}$, wobei das Symbol $*$ dafür steht, dass eine Zelle leer ist, und R und L bzw. N stehen für die Bewegungen des Lese/Schreibkopfes der Maschine nach rechts und links bzw. für ein Verharren des Kopfes an seiner Stelle.

Wir sagen, dass eine Turing-Maschine M ein Wort $w = a_1a_2 \dots a_n$ mit $a_i \in X$ für $1 \leq i \leq n$, akzeptiert, wenn folgende Bedingungen erfüllt sind:

1. zu Beginn der Arbeit steht w auf dem Band,

2. zu Beginn der Arbeit ist M im Zustand z_0,

3. zu Beginn der Arbeit steht der Kopf über der Zelle, in der a_1 steht,

4. die Maschine stoppt in einem Endzustand,

5. bei Beendigung der Arbeit steht auf dem Band nur ein ausgezeichnetes Symbol $1 \notin X$,

6. dieses Symbol steht in der Zelle, in der zu Beginn a_1 stand; der Kopf befindet sich erneut über dieser Zelle.

Ein Wort w wird von M abgelehnt, wenn in Punkt 5 der vorstehenden Definition der Akzeptanz 1 durch $0 \notin X$ ersetzt wird und ansonsten die Bedingungen der Akzeptanzdefinition erhalten bleiben.

Da die Symbole 1 und 0 nur zum Unterscheiden von Akzeptanz und Ablehnung eines Wortes benutzt werden, führen wir sie bei der Angabe der Bestandteile der Turing-Maschine nicht gesondert auf.

Wir sagen, dass M eine Sprache $L \subset X^+$ entscheidet, wenn M jedes Wort aus L akzeptiert und jedes Wort aus $X^* \setminus L$ ablehnt.

Es sei $t : \mathbb{N} \to \mathbb{N}$ eine Funktion. Wir sagen, dass M eine Sprache L in der Zeit t entscheidet, wenn M ein Wort w der Länge n nach höchstens $t(n)$ Schritten akzeptiert oder ablehnt.

Dieses Konzept der Entscheidbarkeit unterscheidet sich von dem Üblichen, bei dem nur verlangt wird, dass die Maschine bei Akzeptanz in einen Zustand aus der Menge $F \subset Q$ gelangt, während bei Ablehnung ein Zustand aus $Q \setminus F$ erreicht wird. Es ist aber leicht zu zeigen, dass beide Konzepte äquivalent sind.

Ähnliches gilt auch für die Komplexität. Dabei setzen wir der Einfachheit halber voraus, dass für die Komplexität t im klassischen Fall $t(n) \geq n$ gilt. Dann erfolgt bei unserer Definition eine Entscheidung der Sprache in höchstens der Zeit t' mit $t'(n) = 5 \cdot t(n) + 4$. Dies ist wie folgt zu sehen: Wir markieren im ersten Schritt die Zelle, in der der erste Buchstabe des Wortes zu Beginn steht (indem wir eine gestrichene Version des Alphabets in dieser Zelle verwenden und erzeugen dann Markierungen für den Anfang und das Ende des Wortes, die wir während der Arbeit mitverschieben. Dies erfordert höchstens $|w|+4 \leq t(|w|)+4$ Schritte. Dann arbeiten wir das Wort entsprechend klassischer Methode ab. Dies erfordert höchstens $t(|w|)$ Schritte. Dann löschen wir alle Buchstaben auf dem Band und schreiben in die markierte Zelle je nach erreichtem Endzustand eine 1 oder 0. Dies erfordert höchstens drei Läufe über das Wort auf dem Band bei Erreichen des Endzustandes. Da das Wort auf dem Band höchstens die Länge $t(|w|)$ aufweist, erhalten wir für die abschließende Phase eine Komplexität von höchstens $2 \cdot t(|w|)$. Durch Addition ergibt sich das gewünschte Resultat.

Daher unterscheiden sich die beiden Definitionen hinsichtlich der Komplexitätsresultate nicht, wenn man nur am asymptotischen Verhalten der Komplexität interessiert ist.

Um einen Zusammenhang mit Schaltkreisen herzustellen, benötigen wir eine Definition der Entscheidung von Sprachen durch Schaltkreise. Eine naheliegende Variante wäre die folgende: Für einen Schaltkreis S, der eine Boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$ berechnet, ist die Menge $f^{-1}(1)$ die Menge der akzeptierten Wörter und $f^{-1}(0)$ die Menge der abgelehnten Wörter. Diese Definition hat zwei Nachteile: Zum einen erfassen wir nur Wörter über $\{0,1\}$ und zum anderen haben alle Wörter die gleiche Länge n. Während der erste Nachteil unter Verwendung von geeigneten Kodierungen einer (beliebigen) Menge X durch Dualwörter behoben werden kann, ist der zweite nur unter Verwendung einer unendlichen Menge von Schaltkreisen S_n, $n \geq 1$, vermeidbar, wobei S_n eine n-stellige Boolesche Funktion realisiert.

Es sei $\mathcal{F} = (S_1, S_2, \ldots, S_n, \ldots)$ eine unendliche Folge von Schaltkreisen, bei der S_n für $n \geq 1$ eine n-stellige Boolesche Funktion f_n realisiert. Wir sagen, dass $\mathcal{F}$ eine Sprache $L \subset \{0,1\}^*$ entscheidet, wenn

$$L \cap \{0,1\}^n = f_n^{-1}(1) \text{ für } n \geq 1$$

gilt.

Umgekehrt können wir für jede Sprache $L \subset \{0,1\}^*$ und jede natürliche Zahl $n \geq 1$ die Menge $L_n = L \cap \{0,1\}^n$ betrachten. Ferner wird durch L_n eindeutig eine n-stellige Boolesche Funktion f_n durch $L_n = f_n^{-1}(1)$ definiert. Ferner gibt es für $n \geq 1$ zu f_n einen Schaltkreis S_n, der f_n berechnet. Damit gibt es zu jeder Sprache $L \subset \{0,1\}^+$ eine Folge $\mathcal{F} = (S_1, S_2, \ldots)$ von Schaltkreisen, die L entscheidet.

Es seien $\mathcal{F} = (S_1, S_2, \ldots, S_n, \ldots)$ eine unendliche Folge von Schaltkreisen, bei der S_n für $n \geq 1$ eine n-stellige Boolesche Funktion f_n realisiert, und $g : \mathbb{N} \to \mathbb{N}$ eine Funktion. Wir sagen, dass $\mathcal{F}$ eine Sprache $L \subset \{0,1\}^*$ mit der Komplexität g entscheidet, wenn $\mathcal{F}$ die Sprache L entscheidet und überdies $C(S_n) \leq g(n)$ für $n \geq 1$ gilt.

Wir wollen nun zuerst zeigen, dass es ist nicht möglich ist, aus der Entscheidung einer Sprache durch Schaltkreise mit einer kleinen Komplexität auch auf eine einfache Entscheidung der Sprache durch Turing-Maschinen zu schließen. Genauer gesagt, wollen wir beweisen, dass es eine Folge von Schaltkreisen gibt, die eine sogar unentscheidbare Sprache mit linearer Komplexität entscheiden kann.

Dazu sei $h : \mathbb{N} \to \{0,1\}$ eine totale und nicht algorithmisch berechenbare Funktion (die Existenz derartiger Funktionen wurde in der Grundvorlesung zur Theoretischen Informatik gezeigt). Wir definieren dann für $n \geq 1$ die Sprache

$$L_n = \begin{cases} \{0^n\}, & \text{falls } h(n) = 0, \\ \{0^{n-1}1\}, & \text{falls } h(n) = 1 \end{cases}$$

und setzen

$$L = \bigcup_{i \geq 1} L_i \,.$$

Offensichtlich ist L unentscheidbar, denn wäre L entscheidbar, so könnte h algorithmisch berechnet werden, da $h(n) = 0$ genau dann gilt, wenn $0^n \in L$ gilt.

Andererseits gehört zu L_n die Funktion f_n mit

$$f_n(\underbrace{0,0,\ldots,0}_{n\text{-mal}}) = \begin{cases} 1, & h(n) = 0, \\ 0, & h(n) = 1, \end{cases}$$

$$f_n(\underbrace{0,0,\ldots,0}_{(n-1)\text{-mal}},1) = \begin{cases} 1, & h(n) = 1, \\ 0, & h(n) = 0, \end{cases}$$

$f_n(x_1, x_2, \ldots, x_n) = 0$ in allen anderen Fällen.

Hieraus folgt sofort

$$f_n(x_1, x_2, \ldots, x_n) = \begin{cases} \overline{x_1} \wedge \overline{x_2} \wedge \cdots \wedge \overline{x_{n-1}} \wedge \overline{x_n}, & h(n) = 0, \\ \overline{x_1} \wedge \overline{x_2} \wedge \cdots \wedge \overline{x_{n-1}} \wedge x_n, & h(n) = 1 \end{cases}$$

für $n \geq 1$. Wir realisieren nun die Funktionen f_n durch Schaltkreise S_n, indem wir einfach die angegebene Formel umsetzen. Wir benötigen dann für S_n jeweils $(n-1)$ $\wedge$-Gatter und n bzw. $n-1$ Negationsgatter. Somit ergibt sich

$$C(S_n) = \begin{cases} 2n-1, & h(n) = 0, \\ 2n-2, & h(n) = 1. \end{cases}$$

Somit ist L mit linearer Komplexität durch Schaltkreise entscheidbar.

Wir wollen nun die umgekehrte Situation betrachten, d. h. wir nehmen an, dass eine Sprache L mit einer gewissen Zeitkomplexität durch Turing-Maschinen akzeptiert werden

kann und fragen nach der Komplexität der Entscheidung durch Schaltkreise. Wir werden zeigen, dass hier im Wesentlichen höchstens ein Ansteigen um einen logarithmischen Faktor entsteht.

Dabei wollen wir wie folgt vorgehen. Wir kodieren zuerst die Eingaben und Zustände durch Tupel über $\{0,1\}$. Genauer gesagt, gelten

$$X = \{x_1, x_2, \ldots, x_n\} \text{ und } Z = \{z_0, z_1, \ldots, z_m\}$$

und

$$r = \lfloor \log_2(n) \rfloor \text{ und } s = \lfloor \log_2(m) \rfloor ,$$

so kodieren wir

$$x_i \text{ durch } (\alpha_0, \alpha_1, \ldots, \alpha_r) \text{ mit } \sum_{j=0}^{r} \alpha_j \cdot 2^j = i \text{ und } \alpha_j \in \{0,1\} \text{ für } 0 \leq j \leq r,$$

$$z_i \text{ durch } (\beta_0, \beta_1, \ldots, \beta_s) \text{ mit } \sum_{j=0}^{s} \beta_j \cdot 2^j = i \text{ und } \beta_j \in \{0,1\} \text{ für } 0 \leq j \leq s.$$

Ferner kodieren wir $*$ durch das Tupel $(0,0,\ldots,0) \in \{0\}^{r+1}$, das der Zahl 0 entspricht, d. h. wir betrachten $*$ als x_0. Fassen wir nun die Überführungsfunktion δ als ein Tripel $(\delta_1, \delta_2, \delta_3)$ mit

$$\begin{aligned}
&\delta_1 : (Z \setminus Q) \times (X \cup \{*\}) \to Z ,\\
&\delta_2 : (Z \setminus Q) \times (X \cup \{*\}) \to X \cup \{*\} ,\\
&\delta_3 : (Z \setminus Q) \times (X \cup \{*\}) \to \{R, N, L\}
\end{aligned}$$

auf, so können wir unter Beachtung der Kodierungen Boolesche Funktionen

$$\begin{aligned}
&\delta_1' : \{0,1\}^{s+1} \times \{0,1\}^{r+1} \to \{0,1\}^{s+1}\\
&\delta_2' : \{0,1\}^{s+1} \times \{0,1\}^{r+1} \to \{0,1\}^{r+1}
\end{aligned}$$

konstruieren, die das lokale Verhalten der Turing-Maschine bez. der Zustandsänderung und der Änderung des Bandes erfassen. (Analoges könnte man auch für die dritte Funktion vornehmen, indem man R, N und L binär kodiert, aber dies wird sich als nicht notwendig erweisen.)

Wir stellen uns nun vor, dass wir sowohl für die Änderung der Zustände als auch für die Zelleninhalte eine Simulation mittels der Funktionen δ_1' und δ_2' vornehmen. Dann wird in jedem Takt der Zustand entsprechend der Funktion δ_1' geändert, wobei die ersten $s+1$ Komponenten dem aktuellen Zustand entsprechen und die letzten $r+1$ Komponenten den Inhalt der Zelle widerspiegeln, über der der Kopf gerade steht, während das Ergebnis eine Kodierung des nächsten Zustandes ist. Analog verhält es sich mit δ_2', nur dass das Ergebnis den veränderten Inhalt der Zelle angibt, über der der Kopf stand. Die Schwierigkeit der Simulation besteht nun darin, dass wir vorab nicht wissen, wo der Kopf steht. Wäre dies für jeden Takt im Voraus bekannt, so wüssten wir, welche Funktion mit welchen Eingabegrößen bei der Simulation benutzt werden muss.

Dies führt zu folgender Definition.

Definition 1.49 *Eine Turing-Maschine M heißt bewegungsuniform, wenn die Position des Kopfes nach i Schritten nur von der Länge n des Eingabewortes (und nicht vom Wort selbst) abhängt.*

Nach dieser Definition sind die Kopfbewegungen für alle Wörter der gleichen Länge identisch.

Es sei M eine bewegungsuniforme Turing-Maschine. Mit $pos(i, n)$ bezeichnen wir die Position des Kopfes nach i Schritten bei Eingabe eines Wortes der Länge n. Hierbei stehe der erste Buchstabe des Eingabewortes zu Beginn in der ersten Zelle, d. h. es gilt $pos(0, n) = 1$ für alle $n \geq 1$.

Damit ist unser Ziel erreicht. Bei einer bewegungsuniformen Turing-Maschine haben wir im i-ten Schritt als Eingabe für δ_1' und δ_2' den Zustand nach $i-1$ Schritten und den Inhalt der Zelle $pos(i-1, n)$ zu benutzen. Dabei ist zu bemerken, dass wir den Zustand durch δ_1' stets aktualisieren und durch δ_2' auch immer den aktuellen Inhalt der Zellen kennen.

Wir illustrieren diese Idee anhand folgenden Beispiels. Wir betrachten die Turing-Maschine

$$M = (\{a, b, c\}, \{z_0, r, r', r'', r''', f, f', f'', q\}, z_0, \{q\}, \delta)\,,$$

wobei δ durch folgende Tabelle gegeben ist:

	z_0	r	r'	r''	r'''	f	f'	f''
$*$	$(q, 0, N)$	$(r', *, L)$		$(r''', *, R)$	$(q, 1, N)$	$(f', *, L)$	$(f'', *, R)$	$(q, 0, N)$
a	(r, a, R)	(r, a, R)	$(f', *, L)$	$(r'', *, R)$		(f, a, L)	$(f', *, L)$	
b	(f, b, R)	(r, b, R)	$(r'', *, L)$	$(r'', *, L)$		(f, b, L)	$(f', *, L)$	
c	(f, c, R)	(r, c, R)	$(f', *, L)$	$(r'', *, L)$		(f, c, L)	$(f', *, L)$	

Die Turing-Maschine M geht beim Lesen eines as in den Zustand r (r und seine gestrichenen Versionen stehen für *richtig*) und läuft nach rechts zum Wortende und steht dann im Zustand r' über dem letzten Buchstaben. Ist dieser ein b, so wird im Zustand r'' unter Löschen aller Buchstaben an den Wortanfang gelaufen. Ist das $*$ vor dem Wort erreicht, wird der Kopf um eine Zelle nach rechts bewegt, schreibt dort eine 1 und stoppt. Ist der erste Buchstabe kein a, so geht M in den Zustand f (für *falsch*), bewegt sich an das Wortende, unter Löschen wieder zum Wortanfang, schreibt dort eine 0 und stoppt. Ist der erste Buchstabe ein a, aber der letzte Buchstabe kein b, so wird durch Übergang in den Zustand f' noch zum Wortanfang gegangen, eine 0 geschrieben und gestoppt.

Damit akzeptiert M genau die Menge der Wörter w, deren erster Buchstabe ein a und deren letzter Buchstabe ein b sind.

Offensichtlich bewegt sich der Kopf nach rechts bis zur Zelle nach dem letzten Buchstaben, dann zurück nach links zur Zelle vor dem ersten Buchstaben und dann noch eine Zelle nach rechts. Daher ist die Kopfbewegung nur von der Länge des Wortes abhängig. Die Maschine M ist also bewegungsuniform.

Die zugehörige Schaltung für das Wort ab oder für ein beliebiges Eingabewort der Länge 2 ist in Abbildung 1.15 zu sehen, wobei wir auf die Kodierungen durch Tupel über der Menge $\{0, 1\}$ verzichten und daher mit δ_1 und δ_2 operieren. Neben den beiden

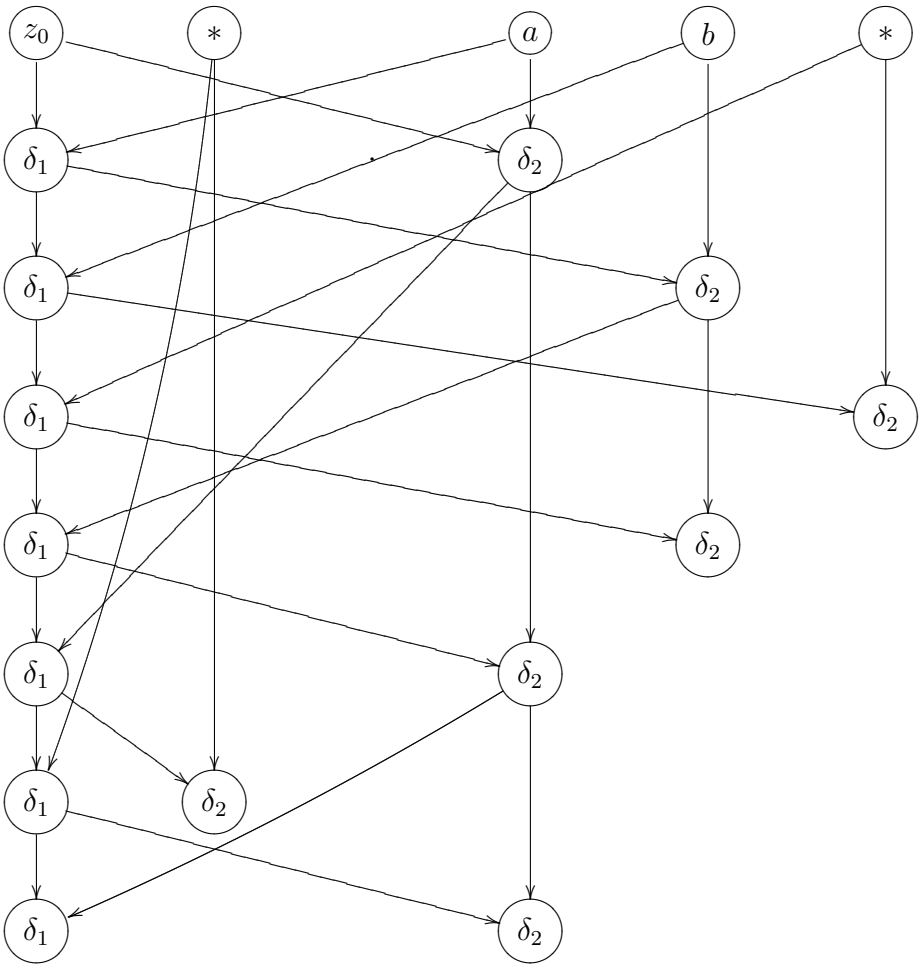


Abbildung 1.15: Schaltung zur Abarbeitung eines Wortes der Länge 2

Eingabesymbolen sind auch die Zellen davor und dahinter zu berücksichtigen. Die Ausgabe erfolgt beim untersten δ_2.

Somit wird jeder Schritt der Turing-Maschine durch zwei Schaltkreise, die δ_1' und δ_2' realisieren, simuliert. Gehen wir von einer Turing-Maschine mit einer Zeitkomplexität t aus, so benötigen wir $t(n)$ Schaltkreise für δ_1' und $t(n)$ Schaltkreise für δ_2' bei einer Eingabe der Länge n. Unter Verwendung von $C(\delta_1') = c_1$ und $C(\delta_2') = c_2$ ergibt sich damit eine Entscheidung der Sprache durch Schaltkreise der Komplexität $(c_1 + c_2)t(n)$.

Vorstehende Überlegungen ergeben daher den folgenden Satz.

Satz 1.50 *Wird eine Sprache L durch eine bewegungsuniforme Turing-Maschine in der Zeit t entschieden, so gibt es eine Konstante c und eine Folge von Schaltkreisen, die L mit der Komplexität t' mit $t'(n) = ct(n)$ entscheidet.* □

Um die Voraussetzung der Bewegungsuniformität zu elimieren, benutzen wir den folgenden Satz, den wir ohne Beweis angeben (für einen Beweis verweisen wir auf [21]).

Satz 1.51 *Zu jeder Turing-Maschine, die eine Sprache L in der Zeit t entscheidet, gibt es eine bewegungsuniforme Turing-Maschine, die L in $O(t(n)\log(t(n)))$ entscheidet.* □

Zusammengefasst ergibt sich somit der folgende Satz.

Satz 1.52 *Wird eine Sprache L durch eine Turing-Maschine in der Zeit t entschieden, so gibt es eine Folge von Schaltkreisen, die L mit der Komplexität $O(t(n)\log(t(n)))$ entscheidet.* □

In der obigen Simulation einer bewegungsuniformen Turing-Maschine durch einen Schaltkreis hat der Schaltkreis eine Tiefe von $O(t(n))$. Wir wollen nun eine andere Simulation vornehmen, bei der eine geringere Tiefe erreicht wird. Dabei wird sich herausstellen, dass eine Beziehung zwischen der Tiefe und der Raumkomplexität besteht. Um die Raumkomplexität zu definieren, betrachten wir Turing-Maschinen, die neben dem Band, auf dem die Eingabe steht, noch ein Arbeitsband haben. Ferner soll die Eingabe nur gelesen werden können, und alle „Rechnungen“ erfolgen auf dem Arbeitsband. Die Raumkomplexität wird dann als Anzahl der Zellen des Arbeitsbandes, die während der Arbeit der Maschine, benutzt werden, definiert. Dies wird gemacht, um nicht das reine Lesen einer Eingabe der Länge n, was mit Benutzung von n Zellen verbunden ist, zu messen, da dann in der Regel die Raumkomplexität mindestens linear ist. Die Einzelheiten der Definition überlassen wir dem Leser.

Wenn wir für die Wörter der Länge n die Funktion $f(x_1, x_2, \ldots, x_n) = x_1 \wedge x_2 \wedge \cdots \wedge x_n$ benutzen, d. h. 1^n ist das einzige akzeptierte Wort der Länge n, so erfordert der zugehörige Schaltkreis die Tiefe $\lceil \log(n) \rceil$. Daher ist zu erwarten, dass die Tiefe von

$$l(n) = \max\{s(n), \lceil \log(n) \rceil\}$$

abhängig sein wird, wobei s die Raumkomplexität sei.

Wenn wir die Konfiguration der Turing-Maschine $M = (X, Z, z_0, Q, \delta)$ durch

$$(q, i, a_1, a_2, \ldots, a_{s(n)}, j)$$

beschreiben, wobei q der aktuelle Zustand, i die Position des Lesekopfes auf dem Eingabeband, a_r der Inhalt der Zelle i des Arbeitsbandes ist, $1 \leq r \leq s(n)$, und j die Position des Lese/Schreibkopfes auf dem Arbeitsband ist, so gibt es höchstens

$$k(n) = \#(Z) \cdot n \cdot \#(X)^{s(n)} \cdot s(n)$$

verschiedene Konfigurationen. Gilt nun $t(n) > k(n)$, so wird mindestens eine Konfiguration während der Arbeit mindestens zweimal erreicht. Dann geht M aber in eine Schleife und beendet die Arbeit nicht, wie bei der Entscheidung einer Sprache gefordert wird. Folglich haben wir noch $t(n) \leq k(n)$. Daraus resultiert

$$\log((t(n)) \leq log(k(n)) = \log(\#(Z)) + \log(n) + s(n) \cdot \log(\#(X)) + \log(s(n)) = O(l(n)).$$

Satz 1.53 *Wenn L durch eine deterministische Turing-Maschine in der Zeit $t(n)$ mit dem Platzbedarf $s(n)$ entschieden wird, so gilt*

$$D(f_n) = O(l(n) \cdot \log(t(n)) = O(l(n)^2)$$

für die Funktion f_n mit $f_n^{-1}(1) = L \cap \{0,1\}^n$.

Beweis: Wir variieren die Turing-Maschine so, dass sie in einem Stoppzustand q^* nicht anhält, sondern ohne Konfigurationsänderung weiterarbeitet. Dadurch erreichen wir, dass wir das Ergebnis an der Konfiguration nach $t(n)$ Schritten ablesen können.

Es sei $p(w)$ die Anzahl der Konfigurationen, die bei Eingabe von w erreicht werden können. Dann definieren wir eine binäre quadratische $(p(w), p(w))$-Matrix $A(w)$ durch $a_{k,k'} = 1$, falls die Konfiguration $k = (q, i, a_1, a_2, \ldots, a_{s(n)}, j)$ in einem Schritt in die Konfiguration k' überführt wird. Der Wert $a_{k,k'}$ kann durch einen Schaltkreis konstanter Tiefe berechnet werden, weil nur eine Abhängigkeit vom i-ten Symbol von w, a_j und q vorliegt. Offenbar gilt $p(n) \leq k(n)$.

Es sei $A(w)^f$ die f-te Potenz von $A(w)$. Nach Definition ergibt sich genau dann $a^f_{k,k'} = 1$, wenn k in f Schritten in k' überführt wird. Die Berechnung von $A(w)^{t(n)}$ kann nun durch einen Schaltkreis der Tiefe $O(\lceil \log(t(n)) \rceil)$ erfolgen, wobei jedes Gatter eine Matrixmultiplikation realisiert. Die Berechnung eines Elementes der Matrix $C = A \cdot B$ erfolgt nach der Formel

$$c_{k,k'} = \bigvee_{k''} a_{k,k''} \wedge b_{k'',k'}.$$

Daher erfordert die Berechnung eines Matrizenprodukts für jedes Element die Tiefe

$$O(\log(p(w)))$$

und damit auch insgesamt nur die Tiefe $O(\log(p(w)))$.

Es seien k_0 die Anfangskonfiguration und K_a die Menge der akzeptierenden Konfigurationen. Dann haben wir

$$f_n(w) = \bigvee_{k \in K_a} a^{t(n)}_{k_0,k}.$$

Ausgehend von $A(w)^{t(n)}$ erfordert dies noch einmal einen Schaltkreis der Tiefe

$$O(\log(\#(K_a))).$$

Damit ergibt sich für den Gesamtschaltkreis eine Tiefe von

$$\begin{aligned} O(1 + \log(t(n)) \log(p(w)) + \log(\#(K_a))) &\leq O(\log(t(n)) \cdot \log(k(n)) + \log(k(n))) \\ &\leq O(\log(t(n)) \cdot l(n)), \end{aligned}$$

womit die Behauptung bewiesen ist. □

Übungsaufgaben

1. Konstruieren Sie eine Folge $\mathcal{F}$ von Schaltkreisen zum Entscheiden der Sprache $\{ (ab)^n \mid n \geq 1; a, b \in \{0, 1\} \}$ und bestimmen Sie die Komplexität des Entscheidens.

2. Konstruieren Sie eine Folge $\mathcal{F}$ von Schaltkreisen zum Entscheiden der Sprache $\{ a^n b^n \mid n \geq 1; a, b \in \{0, 1\} \}$ und bestimmen Sie die Komplexität des Entscheidens.

Kapitel 2

Kolmogorov-Komplexität

In diesem Abschnitt behandeln wir die minimale Komplexität zur Beschreibung eines Wortes.

Dabei sei V ein Alphabet mit mindestens zwei Buchstaben. Die Wörter sollen dann – wenn nicht ausdrücklich anders festgelegt – immer über V gebildet werden. Die in diesem Abschnitt betrachteten Turing-Maschinen sollen stets deterministisch sein.

Definition 2.1 *Die Komplexität $K_A(x)$ eines Wortes $x \in V^+$ bezüglich eines Algorithmus A ist die Länge der kürzesten Eingabe $p \in \{0,1\}^+$ mit $A(p) = x$, d. h. in formalisierter Form*

$$K_A(x) = \min\{|p| \mid p \in \{0,1\}^+,\ A(p) = x\}.$$

Falls kein p mit $A(p) = x$ existiert, so setzen wir $K_A(x) = \infty$.

Wenn wir den Algorithmus A_{id} betrachten, der die identische Abbildung von $\{0,1\}^+$ in $\{0,1\}^+$ realisiert, so gilt wegen $A_{id}(x) = x$ und $A_{id}(y) \neq x$ für alle $y \neq x$ offenbar, $K_{A_{id}}(x) = |x|$.

Für ein Wort $a_1a_2 \dots a_m \in \{0,1,2,\dots,k-1\}^*$ definieren wir

$$val_k(a_1a_2 \dots a_m) = a_1k^{m-1} + a_2k^{m-2} + \dots + a_{m-2}k^2 + a_{m-1}k + a_m,$$

d. h. $a_1a_2 \dots a_m$ ist die k-äre Darstellung von $val_k(a_1a_2 \dots a_m)$. Es sei $A_{2,3}$ der Algorithmus, der $\{0,1\}^+$ in $\{0,1,2\}^+$ wie folgt abbildet. Für ein Wort $x_1x_2 \dots x_n \in \{0,1\}^+$ berechnen wir $m = val_2(x_1x_2 \dots x_n)$. Dann setzen wir

$$A_{2,3}(x_1x_2 \dots x_n) = y_1y_2 \dots y_k \in \{0,1,2\}^+,$$

wobei
- für $val_2(x_1x_2 \dots x_n) = 0$ die Beziehungen $y_1 = 0$ und $k = 1$ gelten,
- für $val_2(x_1x_2 \dots x_n) \neq 0$ die Beziehungen $y_1 \neq 0$ und $m = val_3(y_1y_2 \dots y_k)$ erfüllt sind.

Es sei $x = 120 \in \{0,1,2\}^+$ gegeben. Dann ergibt sich $val_3(120) = 15$. Damit gilt für jedes Wort 0^n1111 mit $n \geq 0$, dass es durch $A_{2,3}$ auf 120 abgebildet wird, da $val_2(0^n1111) = 15$ ist. Folglich haben wir $K_{A_{2,3}}(x) = |1111| = 4$.

Analog erhalten wir für das Wort $y = 1021$ den Wert $val_3(1021) = 34$ und damit $K_{A_{2,3}}(y) = |100010| = 6$ da 100010 das kürzeste Wort z über $\{0,1\}$ mit $val_2(z) = 34$ ist.

Falls das Wort x' mit 0 beginnt, aber von 0 verschieden ist, so gibt es kein Wort $z' \in \{0,1\}^+$, das durch $A_{2,3}$ auf x' abgebildet wird.

Allgemein erhalten wir

$$K_{A_{2,3}}(x) = \begin{cases} 1, & \text{für } x = 0, \\ \lceil \log_2(val_3(x)) \rceil, & \text{für } x = ax',\ a \in \{1,2\},\ x' \in \{0,1,2\}^*, \\ \infty, & \text{sonst.} \end{cases}$$

Offensichtlich hängt die Komplexität $K_A(x)$ bez. A in entscheidendem Maße vom Algorithmus A ab. Wir wollen nun solche A suchen, die bis auf additive Konstanten eine minimale Größe der Komplexität liefern. Formalisiert wird dies durch die folgende Definition.

Definition 2.2 *Ein Algorithmus A_1 ist asymptotisch nicht schlechter als ein Algorithmus A_2, wenn es eine Konstante c_{A_2} so gibt, dass*

$$K_{A_1}(x) \leq K_{A_2}(x) + c_{A_2}$$

für alle $x \in V^*$ gilt.

Wir wollen zeigen, dass es Algorithmen gibt, die asymptotisch besser als jeder andere Algorithmus sind.

Eine Turing-Maschine $U = (X, Z, z_0, Q, \delta)$ (mit der Eingabemenge X, der Menge Z von Zuständen, dem Anfangszustand z_0, der Menge Q von Stoppzuständen und der Überführungsfunktion δ) heißt *universell*, wenn sie auf einer Beschreibung einer beliebigen Turing-Maschine M und einer Eingabe w für M den Wert $f_M(w)$ berechnet, wobei f_M die von M induzierte Funktion ist.

Wir bemerken hier nur, dass es universelle Turing-Maschinen gibt. Dazu schreiben wir eine gegebene Turingmaschine

$$M = (\{x_1, x_2, \ldots x_n\}, \{z_0, z_1, \ldots, z_m\}, z_0, \{z_r, z_{r+1}, \ldots, z_m\}, \delta')$$

ausführlich als

$$\begin{array}{l} *, x_1, x_2, \ldots, x_n; z_0, z_1, \ldots, z_m; z_r, z_{r+1}, \ldots, z_m; \\ (*, z_0, u_1, v_1, R_1), (*, z_1, u_2, v_2, R_2), \ldots, (*, z_{r-1}, u_r, v_r, R_r), \\ (x_1, z_0, u_{r+1}, v_{r+1}, R_{r+1}), (x_1, z_1, u_{r+2}, v_{r+2}, R_{r+2}), \ldots, (x_1, z_{r-1}, u_{2r}, v_{2r}, R_{2r}), \\ \quad \ldots \\ (x_n, z_0, u_{nr+1}, v_{nr+1}, R_{nr+1}), \ldots, (x_n, z_{r-1}, u_{n(r+1)}, v_{n(r+1)}, R_{n(r+1)}), \end{array}$$

wobei $*$ das Blanksymbol bezeichnet und für ein Eingabesymbol x und einen Zustand z das Tupel (x, z, u, v, R) durch $\delta'(x, z) = (u, v, R)$ festgelegt ist. Damit wird die Turing-Maschine M durch ein Wort über $B = \{*, x_1, x_2, \ldots, x_n, z_0, z_1, \ldots, z_m, R, L, N, (,), ;\} \cup \{, \}$ beschrieben. Auf diesem Wort kann U immer nachsehen, welche Aktion bei Eingabe von w zu welchem Zeitpunkt auszuführen ist, und diese Aktion dann auf der Eingabe ausführen.

Im Folgenden sprechen wir von universellen Algorithmen, wenn sie durch universelle Turing-Maschinen gegeben sind.

Satz 2.3 *Es sei U ein universeller Algorithmus und $x \in V^*$. Dann gilt*

$$K_U(x) \leq K_A(x) + c_A$$

für jeden Algorithmus A, wobei c_A eine nur von A (und nicht von x) abhängende Konstante ist (d. h. U ist asymptotisch nicht schlechter als jeder andere Algorithmus A).

Beweis: Es seien U ein universeller Algorithmus, A ein beliebiger Algorithmus und s_A die Beschreibung von A, die U neben p erhalten muss, um $A(p)$ zu berechnen. Es sei c_A die Länge der Beschreibung s_A. Aus der Eingabe $s_A p$ der Länge $c_A + |p|$ berechnet U den Wert $A(p)$.

Es sei nun p ein Wort kürzester Länge mit $A(p) = x$. Dann gelten $K_A(x) = |p|$ und $U(s_A p) = A(p) = x$. Damit erhalten wir

$$K_U(x) = \min\{|y| \mid U(y) = x\} \leq |s_A p| = |p| + c_A = K_A(x) + c_A.$$

□

Aus Satz 2.3 folgt sofort, dass für zwei universelle Algorithmen U_1 und U_2 eine Konstante c mit

$$|K_{U_1}(x) - K_{U_2}(x)| \leq c$$

existiert. Die Komplexitäten bez. der beiden universellen Algorithmen unterscheiden sich also höchstens um eine Konstante. Daher ist es im Folgenden nicht wesentlich, welchen universellen Algorithmus wir wählen.

Definition 2.4 *Es sei U ein (fest gewählter) universeller Algorithmus. Dann definieren wir die Kolmogorov-Komplexität $K(x)$ durch $K(x) = K_U(x)$.*

Entsprechend der Definition ist die Kolmogorov-Komplexität für Wörter über V definiert. Wenn $V = \{0, 1, 2, \ldots, k-1\}$, so können wir eine natürliche Zahl m durch ihre k-äre Darstellung $d_k(m)$, d. h. durch ein Wort über V, angeben. Die Kolmogorov-Komplexität $K(m)$ von m kann daher als Kolmogorov-Komplexität von $d_k(m)$ definiert werden, d. h.

$$K(m) = K(d_k(m)).$$

Die Länge des Wortes $d_k(m)$ bezeichnen wir auch mit $|m|$. Dann gilt

$$|m| = \lceil \log_k(m) \rceil.$$

Wir geben nun einige Eigenschaften der Kolmogorov-Komplexität an.

Lemma 2.5 *Es sei $V = \{0, 1\}$.*

1. *Es gibt eine Konstante c derart, dass für alle $x \in \{0,1\}^+$ die Beziehung*

   $$K(x) \leq |x| + c$$

   gilt.

2. *Für jede natürliche Zahl n gibt es eine Konstante c derart, dass*

$$2^{n-c} \leq \#(\{x \mid x \in \{0,1\}^+, K(x) \leq n\}) < 2^{n+1}$$

gilt.

3. *Für jede berechenbare Funktion f gibt es eine Konstante c_f derart, dass für alle x, für die $f(x)$ definiert ist,*

$$K(f(x)) \leq K(x) + c_f$$

gilt.

4. *Für $n \in \mathbb{N}$ sei V_n eine endliche Menge mit maximal 2^n Elementen. Ferner sei die Menge $\{(x,n) \mid x \in V_n\}$ rekursiv aufzählbar. Dann existiert eine Konstante c derart, dass $K(x) \leq n + c$ für alle $n \in \mathbb{N}$ und alle Elemente $x \in V_n$ gilt.*

Beweis: 1. Da die Kolmogorov-Komplexität auf einem universellen Algorithmus beruht, erhalten wir für den Algorithmus A_{id} aus Lemma 2.3

$$K(x) \leq K_{A_{id}}(x) + c_{A_{id}} = |x| + c_{A_{id}}$$

und damit die gewünschte Aussage mit $c = c_{A_{id}}$.

2. Es seien $A_n = \{x \in \{0,1\}^+ \mid K(x) \leq n\}$ und $B_n = \{y \in \{0,1\}^* \mid |y| \leq n\}$. Für jedes Wort x gilt: Falls x in A_n liegt, so gibt es eine Beschreibung y_x zu x in B_n. Für je zwei unterschiedliche Wörter x_1, x_2 aus A_n gilt $y_{x_1} \neq y_{x_2}$ (unterschiedliche Beschreibungen liefern verschiedene Wörter). Daraus folgt: $\#(A_n) \leq \#(B_n)$. Beachten wir noch, dass es 2^i Wörter der Länge i über dem Alphabet $\{0,1\}$ gibt, so ergibt sich

$$\#(A_n) \leq 2^0 + 2^1 + 2^2 + \cdots + 2^{n-1} + 2^n = 2^{n+1} - 1 < 2^{n+1}.$$

Ferner erhalten wir aus der ersten Aussage, dass für eine gewisse Konstante c alle Wörter der Länge $n - c$ eine Kolmogorov-Komplexität haben, die höchstens $n - c + c = n$ ist. Damit gibt es mindestens 2^{n-c} Wörter, deren Komplexität höchstens n beträgt. Folglich gilt $2^{n-c} \leq \#(A_n)$.

3. Es sei U der universelle Algorithmus mit $K_U(x) = K(x)$. Da f berechenbar ist, gibt es eine Beschreibung s_f der Funktion f, dass U auf der Eingabe $s_f x$ den Wert $f(x)$ berechnet, d. h. es gilt $U(s_f x) = f(x)$. Es sei nun p ein kürzestes Wort mit $U(p) = x$ und damit auch $K(x) = |p|$. Wir verwenden nun die Eingabe $s_f p$ für U. Dann gibt es einen Algorithmus A, der bei der Eingabe s_f zuerst analog zu U das Wort p in x überführt, d. h. wir erhalten $s_f x$, und dann erneut analog zu U das Wort $s_f x$ in $f(x)$ überführt. Dann gilt $K_A(f(x)) \leq |s_f| + |p|$ da $s_f p$ nicht ein kürzestes Wort sein muss, dass durch A in $f(x)$ überführt wird. Beachten wir noch Lemma 2.3, so erhalten wir

$$K(f(x)) = K_U(f(x)) \leq K_A(f(x)) + c_A \leq |p| + |s_f| + c_A = K(x) + |s_f| + c_A$$

und damit die Aussage mit $c = |s_f| + c_A$.

4. Bei der Aufzählung der Elemente (y,m) sei für gegebenes n und $x \in V_n$ das Paar (x,n) das i-te Element mit zweiter Komponente n. Dann ist $i \leq 2^n - 1$ (wenn wir die

Zählung bei 0 beginnen). Damit gibt es $a_j \in \{0,1\}$, $0 \leq j \leq n-1$, mit $\sum_{i=0}^{n-1} a_j 2^j = i$. Wir betrachten nun den Algorithmus A, der bei der Eingabe von $a_{n-1}a_{n-2}\ldots a_0$ (die sowohl n als auch i als enthält) die Aufzählung der Elemente (y,m) vornimmt und x ausgibt, wenn (x,n) das i-te Paar mit zweiter Komponente n ist. Dann gilt $A(a_{n-1}a_{n-2}\ldots a_0) = x$ und somit $K_A(x) \leq n$. Unter Verwendung von Lemma 2.5 erhalten wir

$$K(x) \leq K_A(x) + c_A \leq n + c_A$$

für jedes $x \in V_n$. □

Wir bemerken, dass „fast alle“ Kolmogorov-Komplexitäten von Wörtern der Länge n über $\{0,1\}$ in einem Intervall von $n-c$ bis $n+c$ liegen, wobei c eine passende Konstante ist. Nach der ersten Aussage von Lemma 2.5 erfüllt jedes Wort der Länge n die Relation $K(x) \leq n+c'$ für eine Konstante c'. Es sei nun $c \geq c'$. Dann gilt erst recht $K(x) \leq n+c$ für alle x der Länge n. Die Anzahl der Wörter mit $K(x) < n-c$ ist sicher durch die Anzahl der Wörter über $\{0,1\}$ mit einer Länge $< n-c$, d. h. durch 2^{n-c} beschränkt. Damit ist der Anteil der Wörter der Länge n mit $n-c \leq K(x) \leq n+c$ bezogen auf alle Wörter der Länge n mindestens

$$\frac{2^n - 2^{n-c}}{2^n} = 1 - 2^{-c}.$$

Für hinreichend großes c liegt der Anteil dann nahe 1.

Lemma 2.6 *Zu jeder berechenbaren Funktion $h : V^+ \to \mathbb{N}$ mit $h(x) \leq K(x)$ für alle $x \in V^+$ gibt es eine Konstante C derart, dass $h(x) \leq C$ für alle $x \in V^+$ gilt.*

Beweis: Ohne Beschränkung der Allgemeinheit sei $V = \{0,1,2,\ldots,k-1\}$ (für ein k). Es sei h eine Funktion mit den geforderten Eigenschaften. Angenommen, dass keine konstante obere Schranke C für h existiert, dann gibt es zu jeder Zahl m ein Wort x_m mit

$$m < h(x_m) \leq K(x_m).$$

Zur k-ären Darstellung $d_k(m)$ für eine gegebene Zahl m kann ein Wort x_m mit $m < h(x_m) \leq K(x_m)$ sogar berechnet werden. Dazu zählen wir alle Wörter über V auf und berechnen dann der Reihe nach die Werte $h(x)$, bis ein x mit $h(x) < m$ gefunden wird. Es sei nun $f : V^+ \to V^+$ die Funktion, die $d_k(m)$ das so berechnete Wort x_m zuordnet. Wir erhalten

$$m < k(x_m) = k(f(d_k(m))) \leq K(f(d_k(m))). \tag{2.1}$$

Da f berechenbar ist, gibt es nach der dritten Aussage von Lemma 2.5 eine Konstante c_f mit

$$K(f(d_k(m))) \leq K(d_k(m)) + c_f. \tag{2.2}$$

Nach der ersten Aussage von lemma 2.5 haben wir noch

$$K(d_k(m)) \leq \lceil |d_k(m)| + c = \log_k(m) \rceil + c \tag{2.3}$$

für eine konstante c'. Aus (2.1), (2.2) und (2.3) erhalten wir

$$m < K(f(d_k(m))) \leq \lceil \log_k(m) \rceil + c + c_f < \log_k(m) + c'$$

mit der Konstanten $c' = c + c_f + 1$. Dies ist für große m aber offensichtlich nicht möglich. Folglich ist unsere Annahme falsch und es gibt eine obere Schranke für h. □

Folgerung 2.7 *Die Kolmogorov-Komplexität ist keine berechenbare Funktion.*

Beweis: Wir nehmen an, dass die Kolmogorov-Komplexität berechenbar sei. Es gilt $K(x) \leq K(x)$ für alle Wörter x. Nach Lemma 2.6 gibt es eine konstante obere Schranke C für die Werte von $K(x)$. Dies widerspricht der oben festgestellten Aussage, dass es nur endlich viele Wörter gibt, für die die Kolmogorov-Komplexität kleiner als C ist. □

Für die Angabe einer weiteren Eigenschaft der Kolmogorov-Komplexität benötigen wir die folgende Definition.

Definition 2.8 *Eine Funktion $F : V^+ \to \mathbb{N}$ heißt von oben rekursiv aufzählbar, falls es eine totale berechenbare Funktion $k : V^+ \times \mathbb{N} \to \mathbb{N}$ gibt, für die*

$$k(x,0) \geq k(x,1) \geq k(x,2) \geq \cdots \text{ und } F(x) = \lim_{n\to\infty} k(x,n)$$

für alle $x \in V^+$ gelten.

Da die Werte $k(x,m)$ ganzzahlig sind, gibt es offensichtlich für jedes $x \in V^+$ eine natürliche Zahl n_x derart, dass $k(x,n) = F(x)$ für alle $n \geq n_x$ gilt.

Lemma 2.9 *Die Kolmogorov-Komplexität ist von oben rekursiv aufzählbar.*

Beweis: Es sei U der universelle Algorithmus, der zur Definition der Kolmogorov-Komplexität verwendet wird. Wir definieren nun die Funktion k im Wesentlichen dadurch, dass $k(x,n)$ die Länge des kürzesten Wortes $y \in \{0,1\}^+$ ist, das durch höchstens n Schritte in x überführt werden kann. Das „im Wesentlichen“ liegt darin begründet, dass wir für kleine n nicht absichern können, dass x durch U in höchstens n Schritten überhaupt berechnet werden kann. Deshalb nutzen wir für diese Fälle die Schranke aus der ersten Aussage von Lemma 2.5. Daher setzen wir

$$k(x,n) = \min\{|x|+c,\ \min\{|y| \mid U \text{ berechnet } x \text{ bei Eingabe von } y \text{ in} \leq n \text{ Schritten}\}\}.$$

Wegen der Forderung, dass höchstens n Schritte verwendet werden dürfen, folgt sofort $k(x,n-1) \geq k(x,n)$. Da bei der Definition der Kolmogorov-Komplexität keine Beschränkung der Schrittzahl vorgenommen wird, gilt auch noch $\lim_{n\to\infty} k(x,n) = K(x)$. Damit erfüllt das so konstruierte k die geforderten Eigenschaften, und K ist als von oben rekursiv aufzählbar nachgewiesen. □

Das folgende Lemma gibt eine Art Umkehrung von Lemma 2.9 an, denn es besagt, dass jede von oben rekursiv aufzählbare Funktion (mit einer gewissen Eigenschaft) bis auf eine additive Konstante die Kolmogorov-Komplexität ist.

Lemma 2.10 *Es seien $K' : \{0,1\}^+ \to \mathbb{N}$ eine von oben rekursiv aufzählbare Funktion und C eine Konstante, für die $\#(\{x \mid K'(x) \leq n\}) \leq C2^n$ für alle n erfüllt ist. Dann gibt es eine Konstante c derart, dass $K(x) \leq K'(x) + c$ für alle $x \in \{0,1\}^+$ gilt.*

Beweis: Da K' von oben rekursiv aufzählbar ist, gibt es eine Funktion k' mit

$$k'(x,0) \geq k'(x,1) \geq k'(x,2) \geq \ldots \text{ und } K'(x) = \lim_{n\to\infty} k'(x,n)$$

für alle $x \in \{0,1\}^+$. Ferner sei

$$W_n = \{x \mid x \in \{0,1\}^+, K'(x) < n\}.$$

Wir zeigen jetzt, dass die Relation $R\{(x,n) \mid x \in W_n\}$ aufzählbar ist. Die Aufzählung kann nämlich wie folgt geschehen: Entsprechend einer Aufzählung der Menge $\{(x,m) \mid x \in \{0,1\}^+, m \in \mathbb{N}\}$[1] werden der Reihe nach die Werte $k'(x,m)$ ermittelt, und (x,n) wird zur Aufzählung von R hinzugefügt, wenn $k'(x,m) < n$ gilt. Wegen $K'(x) \leq k'(x,m)$ für $m \in \mathbb{N}_0$ und $\lim_{m\to\infty} k'(x,m) = K'(x)$ werden genau die Elemente von R aufgezählt.

Wir setzen

$$n' = n + \lceil \log_2(C) \rceil \quad \text{und} \quad V_{n'} = W_n.$$

Dann gilt

$$\#(V_{n'} \leq C2^n \leq 2^{n'}.$$

Aus der vierten Aussage von Lemma 2.5 erhalten wir

$$K(x) \leq n' + c' = n + \lceil \log_2(C) \rceil + c' \tag{2.4}$$

mit einer gewissen Konstanten c' für alle $x \in V_{n'} = W_n$.

Es sei nun $n = K'(x) + 1$. Dann gilt $x \in W_n$. Aus (2.4) ergibt sich daher

$$K(x) \leq n + \lceil \log_2(C) \rceil + c' = K'(x) + 1 + \lceil \log_2(C) \rceil + c' = K'(x) + c$$

mit $c = 1 + \lceil \log_2(C) \rceil + c'$. □

Der folgende Satz besagt, dass die Kolmogorov-Komplexität durch einige ihrer oben angegebenen Eigenschaften charakterisiert wird.

Satz 2.11 *Es sei $K' : \{0,1\}^* \to \mathbb{N}$ eine Funktion mit folgenden Eigenschaften:*

1. *Für jede berechenbare Funktion f gibt es eine Konstante c_f derart, dass*

   $$K'(f(x)) \leq K'(x) + c_f$$

   für alle x gilt, für die $f(x)$ definiert ist.

2. *Die Funktion K' ist von oben rekursiv aufzählbar.*

[1] Eine solche Aufzählung kann analog zur Aufzählung der positiven rationalen Zahlen, d. h. der Paare (a,b) mit $a,b \in \mathbb{N}$, konstruiert werden.

3. Es existieren Konstanten c und C derart, dass

$$c2^n \leq \#(\{x \mid x \in \{0,1\}^+,\ K'(x) < n\}) \leq C2^n$$

für alle $n \in \mathbb{N}$ erfüllt ist.

Dann unterscheidet sich K' von der Kolmogorov-Komplexität höchstens um einen konstanten additiven Term.

Beweis: Aus den Eigenschaften 2 und 3 und Lemma 2.10 erhalten wir sofort

$$K(x) \leq K'(x) + \overline{c} \tag{2.5}$$

für eine gewisse Konstante $\overline{c}$.

Da K' nach Eigenschaft 2 von oben rekursiv aufzählbar ist, können wir wie im Beweis von Lemma 2.10 alle Wörter $x \in \{01\}^+$ mit $K'(x) < n$ konstruieren. Wegen Eigenschaft 3 gibt es davon mindestens $c2^n$ Wörter. Wir wählen (die Konstante) d nun minimal mit der Eigenschaft $2^{n-d} \leq c2^n$. Für die natürlichen Zahlen erzeugen wir entsprechend ihrer natürlichen Ordnung der Reihe nach die Mengen S_n von Elementen mit $k'(x) < n$, wobei wir abbrechen, wenn 2^{n-d} Elemente erhalten haben. Dann gilt offensichtlich

$$S_1 \subset S_2 \subset S_3 \subset \dots\ .$$

Für $i \in \mathbb{N}$ setzen wir $T_i = S_{i+1} \setminus S_i$. Nach Konstruktion sind die Mengen T_i, $i \in \mathbb{N}$, paarweise disjunkt. Außerdem gilt

$$\#(T_i) = \#(S_{i+1}) - \#(S_i) = 2^{(i+1)-d} - 2^{i-d} = 2^{i-d}.$$

Da $T_i \subseteq S_{i+1}$ gilt, gilt

$$K'(x) < n + 1 \text{ für alle } x \in T_n. \tag{2.6}$$

Wir betrachten nun eine beliebige Funktion f, die die 2^{n-d} Wörter aus T_n eineindeutig auf die 2^{n-d} Wörter über $\{0,1\}$ der Länge $n-d$ abbildet. Wegen der Eigenschaft 1 erhalten wir

$$K'(f(x)) \leq K'(x) + c_f \text{ für alle } x \in T_n. \tag{2.7}$$

Betrachten wir nun $y \in \{0,1\}^+$ mit $|y| = n - d$, so gibt es ein x mit $f(x) = y$. Für ein solches y gilt dann wegen (2.7) und (2.6)

$$K'(y) = K(f(x)) \leq K'(x) + c_f < n + 1 + c_f = |y| + d + 1 + c_f. \tag{2.8}$$

Da $n - d$ alle natürlichen Zahlen durchläuft, gilt (2.8) sogar für alle y.

Es sei U der universelle Algorithmus, auf dem die Kolmogorov-Komplexität beruht. Zu einem Wort $x \in \{0,1\}^+$ sei p_x ein kürzestes Wort aus $\{0,1\}^+$ mit $U(p_x) = x$. Dann gilt $K(x) = |p_x|$. Wegen der Eigenschaft 1 und (2.8) erhalten wir

$$\begin{aligned} K'(x) = K'(U(p_x)) &\leq K'(p_x) + c_U \leq |p_x| + d + 1 + c_f + c_U = K(x) + d + 1 + c_f + c_U \\ &= K(x) + c \end{aligned} \tag{2.9}$$

mit einer Konstanten $c = d + 1 + c_f + c_U$.

Aus (2.5) und (2.9) folgt die Behauptung. □

Wir wollen nun eine Anwendung der Kolmogorov-Komplexität auf ein zahlentheoretisches Problem geben und dadurch die Brauchbarkeit des Konzepts demonstrieren.

Bereits im Altertum (z. B. in den „Elementen“ EUKLIDS, ca. 300 v. Chr.) war bekannt, dass es unendlich viele Primzahlen gibt. Dagegen war es offen, wie die Primzahlen verteilt sind, oder anders gefragt, wie viele Primzahlen kleiner als eine vorgegebene natürliche Zahl sind. Wir definieren $\pi(n)$ als die Anzahl der Primzahlen p mit $p \leq n$. In den Jahren 1795 und 1798 vermuteten C. F. GAUSS (1777–1855) und A. M. LEGENDRE (1752–1833), dass $\pi(n)$ angenähert durch $n/\ln(n)$ beschrieben wird, wobei ln den natürlichen Logarithmus (zur Basis e) bezeichnet. Der russische Mathematiker P. L. TSCHEBYCHEFF (1821–1894) zeigte im Jahre 1851, dass

$$0.929 \leq \frac{\pi(n)}{n/\ln(n)} \leq 1.106$$

gilt. Im Jahre 1896 bewiesen der französische und der belgische Mathematiker J. S. HADAMARD (1865–1963) und CH. J. DE LA VALLEE POUSSIN (1866–1962) die folgende – heute meist Primzahlsatz genannte – Aussage.

Satz 2.12 *(Primzahlsatz)*

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln(n)} = 1 \ .$$

Alle bekannten Beweise für den Primzahlsatz sind sehr kompliziert und lang. Außerdem setzen sie viele Vorkenntnise aus der Zahlentheorie voraus.

Wir wollen jetzt mittels der Kolmogorov-Komplexität einen (einfachen) Beweis für eine nur etwas schwächere Form des Primzahlsatzes geben. Dafür benötigen wir die folgende Aussage.

Lemma 2.13 *Es sei $n_1, n_2, n_3, \ldots$ eine unendliche Folge natürlicher Zahlen mit den Eigenschaften*

$$n_i \leq n_{i+1} \quad \textit{und} \quad K(n_i) \geq \frac{\lceil \log_2(n_i) \rceil}{2},$$

$n \geq 1$. Weiterhin sei q_i, $i \geq 1$, die größte Primzahl, die die Zahl n_i teilt. Dann ist die Menge $Q = \{q_i \mid i \geq 1\}$ unendlich.

Beweis: Wir nehmen an, dass Q endlich ist (und werden einen Widerspruch herleiten). Es seien p die größte Primzahl in Q und $p_1, p_2, \ldots, p_m$ die Primzahlen mit $p_i \leq p$. Dann hat jede Zahl n_i eine Primzahldarstellung

$$n_i = p_1^{r_{i,1}} p_2^{r_{i,2}} \ldots p_m^{r_{m,1}}$$

mit gewissen Zahlen $r_{j,i} \in \mathbb{N}_0$. Daher existiert ein Algorithmus A, der zu einer Eingabe $(r_{i,1}, r_{i,2}, \ldots, r_{i,m})$ die Binärdarstellung von n_i berechnet. Dann gibt es nach der ersten Aussage von Lemma 2.5 eine Konstante c mit

$$K(n_i) \leq \lceil \log_2(r_{i,1}) \rceil + \lceil \log_2(r_{i,2}) \rceil + \cdots + \lceil \log_2(r_{i,m}) \rceil + c.$$

Wegen $\log_2(r_{i,j}) \leq \log_2(n_i)$ liefert dies

$$K(n_i) \leq m \cdot \lceil \log_2(\log_2(n_i)) \rceil + c.$$

Aufgrund der Voraussetzung $K(n_i) \geq \frac{\lceil \log_2(n_i) \rceil}{2}$ erhalten wir

$$\frac{\lceil \log_2(n_i) \rceil}{2} \leq K(n_i) \leq m \cdot \lceil \log_2(\log_2(n_i)) \rceil + c$$

für alle $i \geq 1$. Dies ist aber unmöglich, da m und c konstante Werte sind. □

Satz 2.14 *Für unendlich viele $n \in \mathbb{N}$ gilt*

$$\frac{n}{32 \log_2(n) \cdot (\log_2(\log_2(n)))^2} \leq \pi(n) \ .$$

Beweis: Mit $bin(n)$ bezeichnen wir die Binärdarstellung von n, d. h. $bin(n) = d_2(n)$. Es sei p_j die j-te Primzahl entsprechend der natürlichen Ordnung der natürlichen Zahlen.

Es seien $n \in \mathbb{N}$ und p_m die größte Primzahl, die n teilt. Dann lässt sich n einfach aus p_m und $\frac{n}{p_m}$ durch Multiplikation gewinnen. Wegen der Existenz einer Aufzählung aller Primzahlen, kann n auch aus den Zahlen m und $\frac{n}{p_m}$ bestimmt werden, indem man aus der Aufzählung und m die Primzahl p_m ermittelt und dann mit $\frac{n}{p_m}$ multipliziert. Als Eingabe ist das Wort $bin(m)bin(\frac{n}{p_m})$ aber nicht verwendbar, da keine Information vorliegt, wo $bib(m)$ aufhört. Deshalb setzen wir für k mit der Binärdarstellung $bin(k) = a_1 a_2 \dots a_{\lceil \log_2(k) \rceil}$

$$Bin(m) = a_1 0 a_2 0 \dots a_{\lceil \log_2(k) \rceil - 1} 0 a_{\lceil \log_2(k) \rceil} 1.$$

Wir betrachten nun $Bin(n)bin(\frac{n}{p_m})$. Dieses Wort ermöglicht die Berechnung von n aus m und $\frac{n}{p_m}$, da die erste 1 an einer geraden Position anzeigt, dass die Binärdarstellung von m gerade aus den Buchstaben davor an ungerader Position gebildet wird. Es gilt

$$|Bin(m)bin(\frac{n}{p_m})| = 2\lceil \log_2(m) \rceil + \lceil \log_2(\frac{n}{p_m}) \rceil.$$

Wir verkürzen diese Eingabe zur Berechnung von n noch weiter (die Länge der kürzesten Eingabe liefert die Kolmogorov-Komplexität). Dazu betrachten wir

$$w = Bin(\lceil \log_2(m) \rceil) bin(m) bin(\frac{n}{p_m}).$$

Tatsächlich ermöglicht $Bin(\lceil \log_2(m) \rceil)$ die Kenntnis, dass die ersten $\lceil \log_2(m) \rceil$ Positionen nach der ersten 1 an gerader Position gerade $bin(m)$ sind und die restlichen Werte dann $bin(\frac{n}{p_m})$ angeben. Offenbar gilt

$$|w| = 2 \cdot \lceil \log_2(\lceil \log_2(m) \rceil) \rceil + \lceil \log_2(m) \rceil + \lceil \log_2(\frac{n}{p_m}) \rceil.$$

Dieser Prozess kann beliebig weiter iteriert werden. Wir führen nur noch einen durch. Wir betrachten also

$$v(m, \frac{n}{p_m}) = Bin(\lceil \log_2(\lceil \log_2(m) \rceil) \rceil) bin(\lceil \log_2(m) \rceil) bin(m) bin(\frac{n}{p_m})$$

mit der Länge

$$|v(m, \frac{n}{p_m})| = 2 \cdot \lceil \log_2(\lceil \log_2(\lceil \log_2(m) \rceil) \rceil) \rceil + \lceil \log_2(\lceil \log_2(m) \rceil) \rceil + \lceil \log_2(m) \rceil + \lceil \log_2(\frac{n}{p_m}) \rceil. \tag{2.10}$$

Wir betrachten nun die Zahlen n mit $2^i \leq n \leq 2^{i+1} - 1$. Dies sind 2^i Zahlen. Jede dieser Zahlen hat eine Binärdarstellung der Länge $i+1$. Nach der zweiten Aussage von Lemma 2.5 gibt es höchsten 2^{i-2} Zahlen mit einer Kolmogorov-Komplexität $\leq (i-3)$. Damit gilt für mehr die Hälfte der Zahlen n mit $2^i \leq n \leq 2^{i+1} - 1$ die Beziehung $K(n) \geq \lceil \log_2(n) \rceil - 2$.

Da die Länge von $v(m, \frac{n}{p_m})$ den Term $\lceil \log_2(\frac{n}{p_m}) \rceil = \lceil \log_2(n) - \log -2(p_m) \rceil$ enthält und $m < p_m$ gilt, ist es leicht zu sehen, dass auch für mehr als die Hälfte der Zahlen n mit $2^i \leq n \leq 2^{i+1} - 1$ die Beziehung $|v(m, \frac{n}{p_m})| \geq \lceil \log_2(n) \rceil - 1$ gilt.

Somit gibt es für jedes $i \in \mathbb{N}$ eine Zahl n_i mit

$$2^i \leq n_i \leq 2^{i+1} - 1,\ K(n_i) \geq \lceil \log_2(n) \rceil - 2 \text{ und } |v(m, \frac{n_i}{p_m})| \geq \lceil \log_2(n_i) \rceil - 2. \tag{2.11}$$

Ausführlich lautet die letzte der Beziehungen wegen (2.10)

$$\lceil \log_2(n_i) \rceil - 1 \leq 2 \cdot \lceil \log_2(\lceil \log_2(\lceil \log_2(m) \rceil) \rceil) \rceil + \lceil \log_2(\lceil \log_2(m) \rceil) \rceil + \lceil \log_2(m) \rceil + \lceil \log_2(\frac{n_i}{p_m}) \rceil,$$

woraus

$$\begin{aligned} \lceil \log_2(p_m) \rceil &\leq 2 \cdot \lceil \log_2(\lceil \log_2(\lceil \log_2(m) \rceil) \rceil) \rceil + \lceil \log_2(\lceil \log_2(m) \rceil) \rceil + \lceil \log_2(m) \rceil + 1 \\ &\leq 2 \cdot \log_2(\log_2(\log_2(m))) + \log_2(\log_2(m)) + \log_2(m) + 5 \end{aligned}$$

folgt (wobei die 5 dadurch entsteht, dass wir jedes Mal nicht zur nächsten Zahl übergehen und dadurch eine Verkleinerung um 4 eintreten kann), woraus sich

$$p_m \leq 32 \cdot m \cdot log_2(m) \cdot (\log_2(\log_2(m)))^2 \tag{2.12}$$

ergibt.

Nach (2.11) erfüllt die Folge $n_1, n_2, \ldots$ die Voraussetzungen des Lemmas 2.13. Damit gilt (2.12) für unendlich viele natürliche Zahlen m. Somit gibt es unter den ersten $32 \cdot m \cdot log_2(m) \cdot (\log_2(\log_2(m)))^2$ natürlichen Zahlen mindestens m Primzahlen, d. h.

$$\pi(32 \cdot m \cdot log_2(m) \cdot (\log_2(\log_2(m)))^2) \geq m.$$

Wir setzen nun $n = 32 \cdot m \cdot log_2(m) \cdot (\log_2(\log_2(m)))^2$. Es ergibt sich

$$\pi(n) \geq m = \frac{n}{32 \cdot log_2(m) \cdot (\log_2(\log_2(m)))^2}.$$

Wegen $n \geq m$ ergibt sich

$$\pi(n) \geq \frac{n}{32 \cdot log_2(n) \cdot (\log_2(\log_2(n)))^2}$$

und damit die gewünschte Aussage. □

Übungsaufgaben

1. Beweisen Sie, dass es Wörter $x \in \{0,1\}^*$ mit $K(x) \geq |x|$ gibt.

2. Man verweise nach, dass 99% aller Wörter mit mindestens 8 Buchstaben eine Kolmogorov-Komplexität von mehr als $n - 8$ haben.

3. Zeigen Sie, dass es keine Turing-Maschine gibt, die zu einer beliebigen natürlichen Zahl n ein Wort x_n der Länge n mit einer Kolmogorov-Komplexität von mindestens $\frac{n}{2}$ ausgibt.

Kapitel 3

Beschreibungskomplexität von Grammatiken

3.1. Definitionen

Wir geben in diesem Abschnitt erst eine kurze Wiederholung von Begriffen zur Theorie der formalen Grammatiken, vor allem um Bezeichnungen zu klären, und führen danach drei Maße für die Beschreibungskomplexität von Grammatiken ein.

Für ein Alphabet V bezeichnen wir mit V^* die Menge aller Wörter über V (einschließlich des Leerworts λ) und mit V^+ die Menge aller nichtleeren Wörter über V. Für ein Wort $w \in V^*$ und eine Menge $U \subseteq V$ bezeichnen $|w|$ die Länge des Wörtes w und $|w|_U$ die Anzahl der Vorkommen von Buchstaben aus U in w. Für $a \in V$ schreiben wir einfach $|w|_a$ anstelle von $|w|_{\{a\}}$. Eine Sprache über V ist eine Teilmenge von V^*. Eine Sprache L heißt λ-frei, falls $L \subseteq V^+$ gilt.

Eine *Regelgrammatik* (oder kurz *Grammatik*) ist ein Quadrupel

$$G = (N, T, P, S),$$

wobei

- N und T endliche, disjunkte Alphabete sind, deren Vereinigung wir mit V bezeichnen,
- P eine endliche Teilmenge von $(V^* \setminus T^*) \times V^*$ ist, und
- $S \in N$ gilt.

Die Menge N ist das Alphabet der Nichtterminale oder Hilfssymbole oder Variablen und T das der Terminale. Im Folgenden werden wir meist große lateinische Buchstaben zur Bezeichnung der Nichtterminale und kleine lateinische Buchstaben für die Terminale verwenden, wobei die Buchstaben auch indiziert sein können. Die Elemente aus P heißen Regeln. Meistens werden wir das Paar (α, β) aus P in der Form $\alpha \to \beta$ schreiben, da diese Notation der Anwendung von Regeln als Ersetzung entspricht. Das Symbol S heißt Axiom oder Startwort.

Wir sagen, dass in der Grammatik $G = (N, T, P, S)$ aus dem Wort $\gamma \in V^+$ das Wort $\gamma' \in V^*$ *direkt erzeugt* wird, wenn

$$\gamma = \gamma_1 \alpha \gamma_2, \quad \gamma' = \gamma_1 \beta \gamma_2, \quad \alpha \to \beta \in P$$

für gewisse $\gamma_1, \gamma_2 \in V^*$ gelten. Wir schreiben dann

$$\gamma \Longrightarrow \gamma'.$$

Falls die bei der Erzeugung verwendete Regel $p = \alpha \to \beta$ betont werden soll, so schreiben wir $\gamma \Longrightarrow_p \gamma'$. Durch $\Longrightarrow$ wird offenbar eine Relation definiert. Wie üblich kann hiervon der reflexive und transitive Abschluss $\Longrightarrow^*$ gebildet werden, d. h. es gilt

$$\gamma \Longrightarrow^* \gamma'$$

genau dann, wenn es eine natürliche Zahl $n \geq 0$ und Wörter $\delta_0, \delta_1, \delta_2, \ldots, \delta_{n-1}, \delta_n$ mit

$$\gamma = \delta_0 \Longrightarrow \delta_1 \Longrightarrow \delta_2 \Longrightarrow \cdots \Longrightarrow \delta_{n-1} \Longrightarrow \delta_n = \gamma'$$

gibt (im Fall $n = 0$ gilt $\gamma = \gamma'$, und im Fall $n = 1$ haben wir $\gamma \Longrightarrow \gamma'$); $\gamma \Longrightarrow^* \gamma'$ gilt genau dann, wenn γ' durch iterierte Anwendung von (im Allgemeinen verschiedenen) Regeln aus γ entsteht. Gilt $\gamma \Longrightarrow^* \gamma'$, so sagen wir auch, γ' ist aus γ (in mehreren Schritten) *ableitbar* oder *erzeugbar*.

Ein Wort $w \in V^*$ heißt *Satzform* von G, wenn $S \Longrightarrow^* w$ gilt, d. h. wenn w aus S erzeugt werden kann. Mit $S(G)$ bezeichnen wir die Menge aller Satzformen von G.

Für eine Grammatik $G = (N, T, P, S)$ ist die *von G erzeugte Sprache* $L(G)$ durch

$$L(G) = \{w \mid w \in T^* \text{ und } S \Longrightarrow^* w\}$$

definiert. Die von G erzeugte Sprache besteht also aus allen Satzformen von G, die nur Terminale enthalten.

Für eine Grammatik G und ein Nichtterminal A von G setzen wir noch

$$L(G, A) = \{w \mid w \in T^* \text{ und } A \Longrightarrow^* w\}.$$

$L(G, A)$ ist also die Sprache, die bei Verwendung von A als Axiom erzeugt wird. Offenbar gilt $L(G) = L(G, S)$.

Zwei Grammatiken sind einander äquivalent, wenn sie die gleiche Sprache erzeugen.

Wir sagen, die Regelgrammatik $G = (N, T, P, S)$ ist

- *monoton*, wenn für alle Regeln $\alpha \to \beta \in P$ die Bedingung $|\alpha| \leq |\beta|$ erfüllt ist, wobei als Ausnahme $S \to \lambda$ zugelassen ist, wenn $|\beta'|_S = 0$ für alle Regeln $\alpha' \to \beta' \in P$ gilt,
- *kontextfrei*, wenn alle Regeln in P von der Form $A \to w$ mit $A \in N$ und $w \in V^*$ sind,
- *regulär*, wenn alle Regeln in P von der Form $A \to wB$ oder $A \to w$ mit $A, B \in N$ und $w \in T^*$ sind.

Eine kontextfreie Grammatik $G = (N, T, P, S)$ ist

- *reduziert*, wenn für jedes $A \in N$ sowohl eine Ableitung $S \Longrightarrow^* w_1 A w_2$ als auch eine Ableitung $A \Longrightarrow^* w$ mit $w_1, w_2 \in V^*$ und $w \in T^*$ existieren,

- λ-*frei*, wenn alle Regeln in P von der Form $A \to w$ mit $w \in V^+$ sind, wobei als Ausnahme $S \to \lambda$ zugelassen ist, wenn $|v|_S = 0$ für alle Regeln $B \to v \in P$ gilt,

- in *Chomsky-Normalform*, wenn alle Regeln in P von der Form $A \to BC$ oder $A \to a$ mit $A, B, C \in N$ und $a \in T$ sind, wobei als Ausnahme $S \to \lambda$ zugelassen ist, wenn $|v|_S = 0$ für alle Regeln $D \to v \in P$ gilt.

Bei einer reduzierten Grammatik lässt sich also jede Satzform so weiter ableiten, dass ein Wort über der Terminalmenge entsteht.

Eine reguläre Grammatik $G = (N, T, P, S)$ ist in *regulärer Normalform*, wenn alle Regeln aus P von der Form $A \to aB$ oder $A \to a$ mit $A, B \in N$ und $a \in T$ sind, wobei als Ausnahme $S \to \lambda$ zugelassen ist, wenn $|v|_S = 0$ für alle Regeln $C \to v \in P$ gilt.

Es ist bekannt, dass es zu jeder kontextfreien Grammatik G eine reduzierte kontextfreie Grammatik G_1, eine kontextfreie λ-freie Grammatik G_2 und eine Grammatik G_3 in Chomsky-Normalform derart gibt, dass $L(G) = L(G_1) = L(G_2) = L(G_3)$ gilt. Außerdem gibt es zu jeder regulären Grammatik H eine Grammatik H' in regulärer Normalform, so dass $L(H) = L(H')$ erfüllt ist.

Im Folgenden sei eine abzählbar unendliche Menge $\mathcal{U}$ gegeben und für jede Grammatik $G = (N, T, P, S)$ gelte stets $N \subset \mathcal{U}$ und $T \subset \mathcal{U}$. Wir bezeichnen mit

RE	die Menge aller Regelgrammatiken,
MON	die Menge aller monotonen Grammatiken,
CF	die Menge aller kontextfreien Grammatiken,
CF-λ	die Menge aller kontextfreien λ-freien Grammatiken,
redCF	die Menge aller reduzierten kontextfreien Grammatiken,
ChCF	die Menge aller kontextfreien Grammatiken in Chomsky-Normalform,
REG	die Menge aller regulären Grammatiken,
nfREG	die Menge aller Grammatiken in regulärer Normalform.

Für eine Menge X von Grammatiken (im Folgenden wird stets

$$X \in \{RE, MON, CF, CF\text{-}\lambda, redCF, ChCF, REG, nfREG\}$$

gelten) bezeichnen wir mit $\mathcal{L}(X)$ die Menge aller Sprachen, die von Grammatiken aus X erzeugt werden können.

Es gilt

$$\begin{aligned}\mathcal{L}(nfREG) &= \mathcal{L}(REG)\\ &\subset \mathcal{L}(CF) = \mathcal{L}(ChCF) = \mathcal{L}(CF\text{-}\lambda) = \mathcal{L}(redCF)\\ &\subset \mathcal{L}(MON)\\ &\subset \mathcal{L}(RE).\end{aligned}$$

Für eine Grammatik $G = (N, T, P, S)$ führen wir die folgenden drei Komplexitätsmaße ein:

$$\begin{aligned} \mathrm{Var}(G) &= \#(N), \\ \mathrm{Prod}(G) &= \#(P), \\ \mathrm{Symb}(G) &= \sum_{\alpha\to\beta\in P} (|\alpha| + |\beta| + 1). \end{aligned}$$

Es sei nun X eine Menge von Grammatiken. Wir erweitern die obigen Komplexitätsmaße auf Sprachen $L \in \mathcal{L}(X)$ mittels

$$\begin{aligned} \mathrm{Var}_X(L) &= \min\{\mathrm{Var}(G) \mid L = L(G),\ G \in X\}, \\ \mathrm{Prod}_X(L) &= \min\{\mathrm{Prod}(G) \mid L = L(G),\ G \in X\}, \\ \mathrm{Symb}_X(L) &= \min\{\mathrm{Symb}(G) \mid L = L(G),\ G \in X\}. \end{aligned}$$

Wir sagen, eine Grammatik $G \in X$ ist *minimal* für eine Sprache $L \in \mathcal{L}(X)$ bezüglich eines Komplexitätsmaßes $K \in \{\,\mathrm{Var}, \mathrm{Prod}, \mathrm{Symb}\,\}$, falls $L(G) = L$ und $K(G) = K_X(L)$ gelten.

Beispiel 3.1 *Wir betrachten die Grammatik*

$$G_1 = (\{S, U, X, X', Y, Z\}, \{a, b\}, P_1, S)$$

mit

$$\begin{aligned} P_1 = \{&S \to YX'aZ,\ YX' \to YX,\ YX' \to b^2U,\ Ua \to aU,\ UZ \to b^2, \\ &Xa \to aaX,\ XZ \to X'Z,\ aX' \to X'a\}. \end{aligned}$$

Es ist leicht zu sehen, dass die Satzformen von G_1 eine der folgenden Formen

$$S,\ Ya^{2n}Xa^mZ,\ Ya^kX'a^lZ,\ b^2a^kUa^lZ,\ b^2a^{2^r}b^2$$

mit $n \geq 0$, $m \geq 0$, $2n + 2m = 2^r$, $k \geq 0$, $l \geq 0$, $k + l = 2^r$, $r \geq 0$ haben, und dass umgekehrt jedes Wort dieser Form auch Satzform ist. Damit gilt

$$L_1 = L(G_1) = \{b^2a^{2^r}b^2 \mid r \geq 0\}.$$

Als Komplexitätsmaße von G_1 erhält man durch einfaches Auszählen

$$\mathrm{Var}(G_1) = 6, \quad \mathrm{Prod}(G_1) = 8 \quad \textit{und} \quad \mathrm{Symb}(G_1) = 43.$$

Damit ergibt sich auch

$$\mathrm{Var}_{MON}(L_1) \leq 6, \quad \mathrm{Prod}_{MON}(L_1) \leq 8 \quad \textit{und} \quad \mathrm{Symb}_{MON}(L_1) \leq 43.$$

Die monotone Grammatik

$$G_1' = (\{S, U, X, X'\}, \{a, b, Y, Z\}, P_1, S)$$

erzeugt offensichtlich auch L_1, denn wir erhalten die gleichen Ableitungen wie in G_1. Der Unterschied zwischen G_1 und G_1' liegt darin, dass wir bei G_1' die Symbole Y und Z als

Terminale auffassen (obwohl sie in keinem Wort der Sprache vorkommen). Für G_1' *erhalten wir*

$$\mathrm{Var}(G_1') = 4, \quad \mathrm{Prod}(G_1) = 8 \quad \textit{und} \quad \mathrm{Symb}(G_1) = 43.$$

Folglich ergibt sich für die Anzahl der Variablen die Verschärfung $\mathrm{Var}_{MON}(L_1) \leq 4$.

Beispiel 3.2 *Es sei* r *eine beliebige positive natürliche Zahl. Für die kontextfreie Grammatik*

$$G_2 = (\{S, A\}, \{a, b\}, \{S \to A^r,\ A \to aA,\ A \to Ab,\ A \to ab\}, S)$$

ergeben sich offenbar

$$L_{2,r} = L(G_2) = \{a^n b^m \mid n \geq 1,\ m \geq 1\}^r$$

und

$$\mathrm{Var}(G_2) = 2, \quad \mathrm{Prod}(G_2) = 4 \quad \textit{und} \quad \mathrm{Symb}(G_2) = 14 + r.$$

$L_{2,r}$ *ist für jedes* $r \geq 1$ *eine reguläre Sprache, da z. B. die reguläre Grammatik*

$$G_2' = (\{A_i \mid 1 \leq i \leq r\} \cup \{B_i \mid 1 \leq i \leq r\}, \{a, b\}, P_2, A_1)$$

mit

$$\begin{aligned} P_2 = {} & \{A_i \to aA_i \mid 1 \leq i \leq r\} \cup \{A_i \to aB_i \mid 1 \leq i \leq r\} \\ & \cup \{B_i \to bB_i \mid 1 \leq i \leq r\} \cup \{B_i \to bA_{i+1} \mid 1 \leq i \leq r-1\} \cup \{B_r \to b\} \end{aligned}$$

ebenfalls $L_{2,r}$ *erzeugt. Wegen* $\mathrm{Var}(G_2') = 2r$ *gilt* $\mathrm{Var}_{REG}(L_{2,r}) \leq 2r$. *Wir wollen nun zeigen, dass hinsichtlich der Anzahl der Nichtterminale* G_2' *sogar optimal ist.*

Es sei daher $G = (N, \{a, b\}, P, S)$ *eine (beliebige) reguläre Grammatik mit* $L(G) = L_{2,r}$ *und* $\mathrm{Var}(G) = \mathrm{Var}_{REG}(L_{2,r})$.

Wir bemerken zuerst, dass für jedes $A \in N$ *mit* $A \neq S$ *eine Ableitung der Form* $A \Longrightarrow^* uA$ *mit* $u \neq \lambda$ *existiert. Angenommen, es gibt ein* $A \neq S$, *für das es keine Ableitung dieses Typs gibt. Dann gibt es nur endlich viele Wörter über* T, *die von* A *(in beliebig vielen Schritten) erzeugt werden können, d. h.* $L(G, A)$ *ist endlich. Jeder Regel* $p = X \to vA$ *mit* $v \in T^*$ *ordnen wir die Menge*

$$P_p = \{X \to vz \mid z \in L(G, A)\}$$

zu, und setzen $G_A = (N_A, T, P_A, S)$ *mit*

$$N_A = N \setminus \{A\} \quad \textit{und} \quad P_A = \{\, p \,|\, p = B \to vC \in P, B \neq A, C \neq A \,\} \cup \bigcup_{\substack{p = B \to w \in P \\ B \neq A}} P_p.$$

Da wir jedes Vorkommen von A *in einer rechten Seite einer Regel durch ein terminales Wort ersetzen, dass aus* A *erzeugt werden kann, ist leicht zu sehen, dass*

$$L(G_A) = L(G) = L_{2,r}$$

gilt. Da aber $\mathrm{Var}(G_A) = \mathrm{Var}(G) - 1 = \mathrm{Var}_{REG}(L_{2,r}) - 1$ *gilt, erhalten wir einen Widerspruch.*

Es sei nun

$$S \Longrightarrow^* vA \Longrightarrow^* vuA \Longrightarrow^* vuw$$

eine Ableitung eines Terminalwortes. Falls für die Ableitung $A \Longrightarrow^* uA$ *eine der Beziehungen* $u = u_1abu_2$ *oder* $u = u_1bau_2$ *gilt, so liefert die Ableitung*

$$S \Longrightarrow^* vA \Longrightarrow^* vuA \Longrightarrow^* vuuA \Longrightarrow^* vu^3A \Longrightarrow^* \cdots \Longrightarrow^* vu^{2r}A \Longrightarrow^* vu^{2r}w$$

ein Wort in dem mindestens $2r$*-mal ein Wechsel von* a *nach* b *oder von* b *nach* a *erfolgt. Dies steht im Widerspruch zur Struktur der Wörter in* $L_{2,r}$*, bei denen höchstens* r *derartige Wechsel auftreten. Folglich gilt bei* $A \Longrightarrow^* uA$ *entweder* $u \in \{a\}^+$ *oder* $u \in \{b\}^+$*. Somit erfordert jeder Wechsel von* a *nach* b *oder von* b *nach* a *mindestens einen Wechsel der Nichtterminale. Weiterhin sind diese Nichtterminale paarweise voneinander verschieden, da sonst eine Ableitung* $A \Longrightarrow^* uA$ *derart existiert, dass* ab *oder* ba *ein Teilwort von* u *ist. Damit hat* G *mindestens* $2r$ *Nichtterminale. Deshalb gilt* $\mathrm{Var}_{REG}(L_{2,r}) \geq 2r$*.*

Aufgrund der regulären Grammatik G_2' *mit* $L(G_2') = L_{2,r}$ *und* $\mathrm{Var}(G_2') = 2r$ *erhalten wir*

$$\mathrm{Var}_{REG}(L_{2,r}) = 2r.$$

Das Beispiel 3.2 belegt, dass kontextfreie Grammatiken erheblich effizienter als reguläre Grammatiken bei der Beschreibung von Sprachen sein können. Wir definieren daher einige mögliche Relationen zwischen den Komplexitätsmaßen.

Es seien $K \in \{\mathrm{Var}, \mathrm{Prod}, \mathrm{Symb}\}$ ein Komplexitätsmaß, sowie X und Y zwei Mengen von Grammatiken, für die $\mathcal{L}(X) \subseteq \mathcal{L}(Y)$ gilt. Wir schreiben

- $X \approx_K Y$, falls es eine Konstante k derart gibt, dass $|K_X(L) - K_Y(L)| \leq k$ für alle $L \in \mathcal{L}(X)$ gilt,
- $Y \leq_K^1 X$, falls es eine Folge L_n, $n \geq 1$, von Sprachen mit $L_n \in \mathcal{L}(X)$ derart gibt, dass $\lim_{n\to\infty}(K_X(L_n) - K_Y(L_n)) = \infty$ gilt,
- $Y \leq_K^2 X$, falls es eine Folge L_n, $n \geq 1$, von Sprachen $L_n \in \mathcal{L}(X)$ derart gibt, dass $\lim_{n\to\infty} \frac{K_X(L_n)}{K_Y(L_n)} = \infty$ gilt,
- $Y \leq_K^3 X$, falls es eine Folge L_n, $n \geq 1$, von Sprachen $L_n \in \mathcal{L}(X)$ und eine Konstante k derart gibt, dass $K_X(L_n) \geq n$ und $K_Y(L_n) \leq k$ gilt.

Intuitiv bedeutet die approximative Gleichheit $X \approx_K Y$, dass sich die Komplexitäten $K_X(L_n)$ und $K_Y(L_n)$ einer Sprache, die von Grammatiken aus X und Y erzeugt werden kann, höchstens um eine Konstante unterscheiden können. Die Relation $\leq_K^1$ bedeutet, dass der Unterschied zwischen den beiden Komplexitäten $K_X(L_n)$ und $K_Y(L_n)$ beliebig groß werden kann. Bei $\leq_K^2$ wird der Unterschied zwischen den Komplexitäten nicht nur beliebig groß, sondern für jede Konstante c gilt für hinreichend großes n sogar

$$c \cdot K_Y(L_n) \leq K_X(L_n) - K_Y(L_n),$$

also ist der Unterschied durch keine lineare Funktion in $K_Y(L_n)$ beschränkt. Bei $\leq_K^3$ kann die Differenz $K_X(L_n) - K_Y(L_n)$ durch keinerlei Funktion mehr beschränkt werden, da für jede Funktion f die Differenz größer als der maximale der Werte $f(1), f(2), \ldots, f(k)$ wird. Offenbar gelten

$$\text{aus } \leq_K^2 \text{ folgt } \leq_K^1,$$
$$\text{aus } \leq_K^3 \text{ folgt } \leq_K^2.$$

Wegen der in Beispiel 3.2 gezeigten Beziehungen

$$\mathrm{Var}_{CF}(L_{2,r}) = 2 \quad \text{und} \quad \mathrm{Var}_{REG}(L_{2,r}) = 2r \geq r$$

für $r \geq 1$ gilt in dieser Terminologie

$$CF \leq_{\mathrm{Var}}^3 REG. \tag{3.1}$$

Für den Nachweis einer Relation vom Typ $X \leq_K^i Y$, $i \in \{1, 2, 3\}$ benötigt man Sprachen aus $\mathcal{L}(Y)$ beliebig großer Komplexität. Eine Verschärfung dieser Forderung besteht darin, dass jede beliebige positive natürliche Zahl im Wertevorrat von K_Y auftaucht.

Definition 3.3 *Wir sagen, dass ein Komplexitätsmaß $K \in \{\mathrm{Var}, \mathrm{Prod}, \mathrm{Symb}\}$ bezüglich einer Menge Y von Grammatiken vollständig ist, wenn es für jede natürliche Zahl $n \geq 1$ (bei* Symb *$n \geq 3$) eine Sprache $L_n \in \mathcal{L}(Y)$ derart gibt, dass $K_Y(L_n) = n$ gilt.*

Übungsaufgaben

1. Geben Sie zwei Mengen X und Y von Grammatiken mit $\mathcal{L}(X) \subseteq \mathcal{L}(Y)$ an, für die $Y \leq_{\mathrm{Prod}}^1 X$ aber nicht $Y \leq_{\mathrm{Prod}}^2 X$ gilt.

2. Zu einer natürlichen Zahl $i \geq 1$ sei L_i die Sprache

   $$L_i = \{\, a^{ki} b^{ki} \mid k \geq 1 \,\}.$$

   Ermitteln Sie die Werte $\mathrm{Var}_{CF}(L_i)$, $\mathrm{Prod}_{CF}(L_i)$ und $\mathrm{Symb}_{CF}(L_i)$.

3.2. Anzahl der Nichtterminale

In diesem Abschnitt untersuchen wir die Anzahl der zur Erzeugung einer Sprache nötigen Nichtterminale. Wir behandeln zuerst die Beziehungen zwischen den Maßen Var_X und Var_Y für verschiedene Klassen X und Y von Grammatiken.

Lemma 3.4 *Für je zwei Mengen X und Y von Grammatiken mit $X \subseteq Y$ und jede Sprache $L \in \mathcal{L}(X)$ gilt* $\mathrm{Var}_Y(L) \leq \mathrm{Var}_X(L)$.

Beweis: Es sei $G \in X$ eine Grammatik mit $L(G) = L$ und $\mathrm{Var}(G) = \mathrm{Var}_X(L)$. Da auch $G \in Y$ nach Voraussetzung gilt, erhalten wir $\mathrm{Var}_Y(L) \leq \mathrm{Var}(G) = \mathrm{Var}_X(L)$. □

Lemma 3.5

i) *Für jede kontextfreie Sprache L gilt* $\mathrm{Var}_{CF}(L) = \mathrm{Var}_{redCF}(L)$.

ii) *Für jede kontextfreie λ-freie Sprache L gilt* $\mathrm{Var}_{CF}(L) = \mathrm{Var}_{CF\text{-}\lambda}(L)$.

Beweis: i) Da jede reduzierte kontextfreie Grammatik auch eine kontextfreie Grammatik ist, gilt nach Lemma 3.4

$$\mathrm{Var}_{CF}(L) \leq \mathrm{Var}_{redCF}(L). \tag{3.2}$$

Ist umgekehrt $L \in \mathcal{L}(CF)$ und $G = (N, T, P, S)$ eine kontextfreie Grammatik mit $L = L(G)$ und $\mathrm{Var}(G) = \mathrm{Var}_{CF}(L)$, so konstruieren wir die zugehörige reduzierte Grammatik G' durch Streichen aller Nichtterminale A von N, für die keine Ableitung der Form $S \Longrightarrow^* uAv$ oder $A \Longrightarrow^* w \in T^*$ existiert, und aller Regeln, in denen A vorkommt. Folglich gilt $\mathrm{Var}(G') \leq \mathrm{Var}(G)$. Dies impliziert

$$\mathrm{Var}_{redCF}(L) \leq \mathrm{Var}(G') \leq \mathrm{Var}(G) = \mathrm{Var}_{CF}(L),$$

woraus mit (3.2) sofort die Behauptung folgt.

ii) Der Beweis verläuft analog. Wir haben nur zu beachten, dass die Standardkonstruktion einer λ-freien Grammatik aus einer Grammatik, die eine λ-freie Sprache erzeugt, die Anzahl der Nichtterminale nicht erhöht. □

Folgerung 3.6 *Für $X \in \{CF, redCF, CF\text{-}\lambda, MON, RE\}$ gilt $X \leq^3_{\mathrm{Var}} REG$.*

Beweis: Wegen $CF \subseteq RE$ und $CF\text{-}\lambda \subseteq MON$ folgt die Aussage aus (3.1) und den Lemmata 3.4 und 3.5. □

Satz 3.7 *Das Maß* Var *ist bezüglich REG vollständig.*

Beweis: Es sei $n \geq 1$ eine gerade Zahl. Dann setzen wir

$$L_n = L_{2,\frac{n}{2}} = (\{a\}^+\{b\}^+)^{\frac{n}{2}}$$

und erhalten aus Beispiel 3.2 $\mathrm{Var}_{REG}(L_n) = n$.

Für ungerades $n \geq 1$ setzen wir

$$L_n = (\{a\}^+\{b\}^+)^{\frac{n-1}{2}}\{a\}^+$$

und können dafür wie in Beispiel 3.2 $\mathrm{Var}_{REG}(L_n) = n$ nachweisen. □

Bevor wir das Analogon zu Satz 3.7 für kontextfreie Sprachen angeben und beweisen, zeigen wir zwei Lemmata, die es uns erlauben, zukünftig gewisse Eigenschaften von bezüglich der Anzahl der Nichtterminale minimalen Grammatiken voraus zu setzen.

Lemma 3.8

i) *Zu einer kontextfreien Grammatik $G = (N, T, P, S)$ mit* $\mathrm{Var}(G) = \mathrm{Var}_{CF}(L(G))$ *gibt es zu jedem Nichtterminal $A \in N$ mit $A \neq S$ eine Regel $A \to uAv$ mit $uv \neq \lambda$ in P.*

ii) *Für eine kontextfreie Grammatik* $G = (N, T, P, S)$ *mit* $\mathrm{Var}(G) = \mathrm{Var}_{CF}(L(G))$ *ist für jedes* $A \in N$ *mit* $A \neq S$ *die Sprache* $L(G, A)$ *unendlich.*

Beweis: i) Wir nehmen an, dass es ein Nichtterminal $A \in N$, $A \neq S$, so gibt, dass die rechten Seiten aller Regeln mit linker Seite A den Buchstaben A nicht enthalten. Wir setzen

$$P_A = \{w \mid A \to w \in P\}.$$

Wir zerlegen die rechten Seiten einer jeden Regel $p = B \to u_1 A u_2 A \dots u_r A u_{r+1} \in P$ mit $B \neq A$ so, dass $u_i \in (V \setminus \{A\})^*$ für $1 \leq i \leq r+1$ ist, und setzen

$$P_p = \{B \to u_1 w_1 u_2 w_2 \dots u_r w_r u_{r+1} \mid r \geq 0,\ w_i \in P_A,\ 1 \leq i \leq r\}.$$

(Falls die rechte Seite von p kein A enthält, ist $P_p = \{p\}$.) Wir betrachten nun die (kontextfreie) Grammatik

$$G' = (N \setminus \{A\}, T, \bigcup_{\substack{p=B\to z\in P\\ B\neq A}} P_p, S).$$

Es ist leicht zu sehen, dass $L(G') = L(G)$ gilt, denn die in G bzw. G' möglichen Ableitungen

$$\begin{aligned} S &\Longrightarrow^* v_1 B v_2 \Longrightarrow v_1 u_1 A u_2 A \dots u_r A u_{r+1} v_2 \\ &\Longrightarrow v_1 u_1 w_1 u_2 A \dots u_r A u_{r+1} v_2 \\ &\Longrightarrow v_1 u_1 w_1 u_2 w_2 u_3 A \dots u_r A u_{r+1} v_2 \\ &\ \ \vdots \\ &\Longrightarrow v_1 u_1 w_1 u_2 w_2 \dots u_r w_r u_{r+1} v_2 \end{aligned}$$

und

$$S \Longrightarrow^* v_1 B v_2 \Longrightarrow v_1 u_1 w_1 u_2 w_2 \dots u_r w_r u_{r+1} v_2$$

sind gleichwertig. Da offenbar

$$\mathrm{Var}_{CF}(L(G)) = \mathrm{Var}_{CF}(L(G')) \leq \mathrm{Var}(G') = \mathrm{Var}(G) - 1 = \mathrm{Var}_{CF}(L(G)) - 1$$

gilt, erhalten wir einen Widerspruch.

ii) Der Beweis wird analog zu dem für den Teil i) geführt, wobei wir nur P_A durch (die nach Annahme endliche Menge) $L(G, A)$ ersetzen. □

Satz 3.9 *Das Maß* Var *ist bezüglich CF vollständig.*

Beweis: Wir betrachten die Sprachen

$$\begin{aligned} L_1 &= \{a\}, \\ L_2 &= \{a\}^+\{b\}^+\{a\}^+, \\ L_n &= \bigcup_{i=1}^{n-1} \{a^i b\}^+ \quad \text{für} \quad n \geq 3. \end{aligned}$$

Da die kontextfreien Grammatiken

$$G_1 = (\{S\}, \{a\}, \{S \to a\}, S),$$
$$G_2 = (\{S, A\}, \{a, b\}, \{S \to aS,\ S \to Sa,\ S \to aAa,\ A \to bA, A \to b\}, S),$$
$$G_n = (\{S, A_1, A_2, \ldots, A_{n-1}\}, \{a, b\}, \bigcup_{i=1}^{n-1} \{S \to A_i,\ A_i \to a^i b A_i,\ A_i \to a^i b\}, S),\ n \geq 3,$$

mit

$$\mathrm{Var}(G_1) = 1, \quad \mathrm{Var}(G_2) = 2 \quad \text{und} \quad \mathrm{Var}(G_n) = n \text{ für } n \geq 3$$

die Sprachen L_1, L_2 und L_n für $n \geq 3$ erzeugen, ist

$$\mathrm{Var}_{CF}(L_1) \leq 1, \quad \mathrm{Var}_{CF}(L_2) \leq 2 \quad \text{und} \quad \mathrm{Var}_{CF}(G_n) \leq n \text{ für } n \geq 3 \tag{3.3}$$

bereits gezeigt.

Es gilt $\mathrm{Var}_{CF}(L_1) \geq 1$ offensichtlich, da jede L_1 erzeugende Grammatik mindestens ein Nichtterminal (das Axiom) hat. Folglich gilt $\mathrm{Var}_{CF}(L_1) = 1$.

Angenommen, es gibt eine kontextfreie Grammatik $G = (\{S\}, \{a, b\}, P, S)$ mit einem einzigen Nichtterminal S und $L_2 = L(G)$. Wegen Lemma 3.5 können wir ferner ohne Beschränkung der Allgemeinheit annehmen, dass G reduziert ist. Wenn es eine Satzform gibt, in der S mindestens zweimal vorkommt, so gibt es in G eine Ableitung

$$S \Longrightarrow^* xSySz \Longrightarrow^* xabayabaz \Longrightarrow^* x'abay'abaz' = w$$

mit $x', y', z' \in T^*$. Die Struktur von w widerspricht aber der der Wörter von L_2, womit wir einen Widerspruch erhalten haben.

Folglich enthalten alle Satzformen von G höchstens ein S. Es sei nun p die maximale Länge der rechten Seiten von Regeln in P. Die Sprache L_2 enthält das Wort $ab^{2p}a$. Die Ableitung dieses Wortes erfordert daher die Existenz von Regeln der Form $S \to b^r S b^s$ oder $S \to b^t$ mit $r + s > 0$ bzw. $t > 0$, da sonst der Abstand von zwei Vorkommen von a durch p beschränkt wäre. Im ersten Fall gibt es in G die Ableitung $S \Longrightarrow b^r S b^s \Longrightarrow^* b^r abab^s$ und im zweiten Fall die Ableitung $S \Longrightarrow b^t$, die beide nicht zu Wörtern in L_2 führen. Erneut erhalten wir einen Widerspruch, der beweist, dass wir mindestens zwei Nichtterminale zum Erzeugen von G benötigen. Das bedeutet $\mathrm{Var}_{CF}(L_2) \geq 2$, woraus mit (3.3) dann $\mathrm{Var}_{CF}(L_2) = 2$ folgt.

Wir behandeln nun den Fall $n \geq 3$. Es sei $G = (N, T, P, S)$ eine kontextfreie Grammatik mit $L(G) = L_n$ und $\mathrm{Var}(G) = \mathrm{Var}_{CF}(L_n)$. Wegen Lemma 3.5 können wir annehmen, dass G λ-frei und reduziert ist.

Es sei A ein Nichtterminal mit $A \neq S$. Dann gibt es nach Lemma 3.8 eine Regel $A \to uAv$ mit $uv \neq \lambda$. Da G reduziert ist, gibt es in G die Ableitungen

$$\begin{aligned} S &\Longrightarrow^* z_1 A z_2 \Longrightarrow z_1 u A v z_2 \Longrightarrow z_1 u^2 A v^2 z_2 \Longrightarrow \cdots \Longrightarrow z_1 u^m A v^m z_2 \\ &\Longrightarrow^* z_1'(u')^m w (v')^m z_2' \end{aligned} \tag{3.4}$$

von Wörtern in $L(G)$, bei denen m eine beliebige positive natürliche Zahl und $w \in L(G, A)$ sind sowie (wegen der λ-Freiheit) $|(u')^m| \geq m$ oder $|(v')^m| \geq m$ gelten. Wir wählen nun

$m \geq 2n$. Dann muss $(u')^m$ oder $(v')^m$ ein Teilwort $z = ba^ib$ für ein $i \in \{1, 2, \ldots, n-1\}$ enthalten. Ohne Beschränkung der Allgemeinheit nehmen wir $(u')^m = x_1ba^ibx_2$ an.

Nach Lemma 3.8 ist $L(G, A)$ unendlich. Es sei $y \in L(G, A)$ mit $|y| \geq n + 1$. Dann gibt es eine Zerlegung $y = y_1ba^jby_2$ mit gewissen Wörtern $y_1, y_2 \in T^*$. Nach (3.4) gilt dann $z_1'x_1ba^ibx_2y_1ba^jby_2(v')^mz_2' \in L(G) = L_n$. Aufgrund der Struktur der Wörter aus L_n erhalten wir $i = j$. Daher gibt es für jedes $A \neq S$ genau ein $i \in \{1, 2, \ldots, n-1\}$ derart, dass alle Wörter in $L(A, G)$, deren Länge mindestens $2n$ ist, Teilwörter aus $\{a^ib\}^+$ haben.

Die gleichen Überlegungen gelten auch für S, falls es eine Ableitung $S \Longrightarrow^* t_1St_2$ mit $t_1, t_2 \in V^*$ und $t_1t_2 \neq \lambda$ gibt. Damit gibt es keine Ableitungen dieser Form, da $L(G, S) = L_n$ gilt.

Angenommen, $N \setminus \{S\}$ enthält weniger als $n - 1$ Nichtterminale. Dann gibt es eine natürliche Zahl $j \in \{0, 1, 2, \ldots, n-1\}$ so, dass G nur endlich viele Wörter aus $\{a^jb\}^+$ erzeugen kann. Dies widerspricht $L(G) = L_n$, womit $\mathrm{Var}_{CF}(L_n) \geq n$ gezeigt ist. Mit (3.3) folgt die Behauptung. □

Bezüglich der beliebigen Regelgrammatiken liegt eine grundsätzlich andere Situation als bei regulären und kontextfreien Grammatiken vor. Hier ist die Anzahl der zum Erzeugen einer Sprache nötigen Nichtterminale generell beschränkt.

Satz 3.10 *Für jede Sprache* $L \in \mathcal{L}(RE)$ *gilt* $\mathrm{Var}_{RE}(L) \leq 4$.

Beweis: Zu jeder Sprache $L \in \mathcal{L}(RE)$ gibt es eine Grammatik $G = (N, T, P, S)$, die $L(G) = L$ erfüllt und nur Regeln der Form $\alpha \to \beta$ und $A \to a$ mit $\alpha, \beta \in N^*$, $A \in N$, $a \in T$ hat.[1] Wir betrachten die folgende Grammatik

$$G' = (\{S', \#, [,]\}, N \cup T, P_1 \cup P_2 \cup P_3 \cup P_4, S')$$

mit

$$\begin{aligned}
P_1 &= \{S' \to [\#S],\ [\#] \to \lambda\},\\
P_2 &= \bigcup_{A \in N} \{\#A \to A\#,\ A\# \to \#A\},\\
P_3 &= \{\, \#\alpha \to \#\beta \mid \alpha \to \beta \in P;\ \alpha, \beta \in N^* \,\},\\
P_4 &= \{\, [A \to a[\mid A \to a \in P;\ A \in N;\ a \in T \,\}.
\end{aligned}$$

Die Nichtterminale [und] fungieren im Wesentlichen als Markierungen des linken bzw. rechten Endes der Satzform. Jede Ableitung in G' beginnt mit einer Anwendung der Regel $S' \to [\#S]$, da dies die einzige Regel für das Axiom ist, und endet mit einer Anwendung der Regel $[\#] \to \lambda$, da dies die einzige Regel ist, bei der auf der rechten Seite kein Nichtterminal von G' steht. Mittels der Regeln aus P_2 darf das Symbol # nach links und rechts verschoben werden. Die Regeln aus P_3 erlauben die Simulation der Ableitungen in G. Wegen der Regeln aus P_4 dürfen wir den linken Endmarker nach rechts verschieben, wobei jede Verschiebung aber mit einer gleichzeitigen Simulation einer terminierenden Regel aus P verbunden ist. Aus diesen Erläuterungen folgt, dass jede nichtterminale Satzform von G' die Form $u[v_1\#v_2]$ hat, wobei $u \in T^*$ gilt und uv_1v_2 eine Satzform von G ist. Der

[1] Sollte eine Grammatik nicht in dieser Form vorliegen, ersetze man zuerst jedes Vorkommen eines Terminals a in Regeln durch ein neues Nichtterminal a' und füge dann die Regeln $a' \to a$ hinzu.

letzte Schritt einer Ableitung in G' hat daher die Form $u[\#] \Longrightarrow u$ mit $u \in L(G)$. Folglich erhalten wir $L(G') = L(G) = L$. Wegen $\mathrm{Var}(G') = 4$ folgt die Behauptung. □

Für monotone Grammatiken können wir keine generelle Beschränkung der Anzahl der Nichtterminale wie in Satz 3.10 angeben, aber wir geben eine Schranke an, die nur von der Größe des Alphabets abhängt, über dem die Sprache definiert ist.

Satz 3.11 *Für jede Sprache $L \in \mathcal{L}(MON)$ mit $L \subseteq T^*$ gilt* $\mathrm{Var}_{MON}(L) \leq 3\#(T) + 1$.

Beweis: Es sei L eine Sprache über dem Alphabet T. Dann gibt es zu je drei Buchstaben a, b und c eine Menge L_{abc} derart, dass die Sprache $L_{abc}\{abc\}$ alle Wörter mit einer Länge von mindestens vier enthält, die auf abc enden. Damit lässt sich L wie folgt zerlegen:

$$L = \{\, w \mid w \in L,\ |w| \leq 3 \,\} \cup \bigcup_{(a,b,c)\in T^3} L_{abc}\{abc\}.$$

Wir benutzen ohne Beweis den folgenden Satz: *Ist L eine monotone Sprache, so gibt es zu jedem Wort $abc \in T^3$ eine monotone Grammatik G_{abc} mit $L(G_{abc}) = L_{abc}$.* Es sei jeweils $G_{abc} = (N_{abc}, T, P_{abc}, S_{abc})$. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass

- die Mengen N_{abc} der Nichttermiale dieser Grammatiken paarweise disjunkt sind,
- alle Grammatiken nur Regeln der Form $\alpha \to \beta$ und $A \to a$ mit $\alpha, \beta \in N_{abc}^*$, $A \in N_{abc}$ und $a \in T$ haben.

Wir setzen dann

$$\begin{aligned}
N' &= \bigcup_{(a,b,c)\in T^3} N_{abc},\\
P' &= \bigcup_{(a,b,c)\in T^3} P_{abc},\\
G' &= (\{S'\} \cup \bigcup_{a\in T} \{[^a, \#^a,]^a\}, N' \cup T, P_1 \cup P_2 \cup P_3 \cup P_4, S'),\\
P_1 &= \{\, S' \to w \mid w \in L(G),\ |w| \leq 3 \,\} \cup \bigcup_{(a,b,c)\in T^3} \{S' \to [^a\#^b S_{abc}]^c,\ [^a\#^b]^c \to abc\},\\
P_2 &= \bigcup_{A\in N'} \bigcup_{a\in T} \{\#^a A \to A\#^a,\ A\#^a \to \#^a A\},\\
P_3 &= \{\, \#^a\alpha \to \#^a\beta \mid \alpha \to \beta \in P',\ a \in T \,\},\\
P_4 &= \{\, [^b A \to a[^b \mid A \to a \in P',\ b \in T \,\}.
\end{aligned}$$

Wegen der Disjunktheit der Mengen der Nichtterminale und Regeln erhält man in Analogie zum Beweis von Satz 3.10

$$L(G') = \{\, w \mid w \in L,\ |w| \leq 3 \,\} \cup \bigcup_{(a,b,c)\in T^3} L_{abc}\{abc\} = L.$$

Wegen $\mathrm{Var}(G') = 3\#(T) + 1$ folgt die Behauptung. □

Folgerung 3.12 *Für $X \in \{MON, RE\}$ gilt $X \leq^3_{\mathrm{Var}} CF$.*

Beweis: Wir betrachten die Sprachen $L_n \subseteq \{a,b\}^*$, $n \geq 1$, aus dem Beweis von Satz 3.9. Nach diesem Beweis gilt $\mathrm{Var}_{CF}(L_n) = n$. Aus den Sätzen 3.10 und 3.11 erhalten wir $\mathrm{Var}_{RE}(L_n) \leq 4$ und $\mathrm{Var}_{MON}(L_n) \leq 7$. □

Lemma 3.13 *Es sei w ein Wort der Länge n über T. Dann gilt $\mathrm{Var}_{nfREG}(\{w\}) = n$.*

Beweis: Es sei $w = a_1 a_2 \dots a_n$ mit $a_i \in T$, $1 \leq i \leq n$. Dann erzeugt die Grammatik

$$G = (\{A_i \mid 1 \leq i \leq n\}, T, \{A_i \to a_i A_{i+1} \mid 1 \leq i \leq n-1\} \cup \{A_n \to a_n\}, A_1)$$

in regulärer Normalform offenbar die Sprache $\{a_1 a_2 \dots a_n\} = \{w\}$. Wegen $\mathrm{Var}(G) = n$ ist die Beziehung $\mathrm{Var}_{nfREG}(\{w\}) \leq n$ bereits gezeigt.

Es sei nun $H = (N, T, P, S)$ eine Grammatik in regulärer Normalform, die die Sprache $L(H) = \{w\}$ erzeugt. Jede Ableitung von w hat offensichtlich die Form

$$\begin{aligned} S = B_1 &\Longrightarrow a_1 B_2 \Longrightarrow a_1 a_2 B_3 \Longrightarrow \cdots \Longrightarrow a_1 a_2 \dots a_{i-1} B_i \Longrightarrow \cdots \Longrightarrow a_1 a_2 \dots a_{n-1} B_n \\ &\Longrightarrow a_1 a_2 \dots a_{n-1} a_n = w. \end{aligned} \tag{3.5}$$

Angenommen, es wären zwei der Nichtterminale in solch einer Ableitung gleich, also $B_i = B_j$ für gewisse i und j mit $1 \leq i < j \leq n$. Dann gibt es eine Ableitung

$$B_i \Longrightarrow a_i B_{i+1} \Longrightarrow \cdots \Longrightarrow a_i a_{i+1} \dots a_{j-1} B_j,$$

und folglich die Ableitung

$$B_i \Longrightarrow^* a_i a_{i+1} \dots a_{j-1} B_i.$$

Außerdem folgen aus (3.5) die Ableitungen

$$S \Longrightarrow^* a_1 a_2 \dots a_{i-1} B_i \quad \text{und} \quad B_i = B_j \Longrightarrow^* a_j a_{j+1} \dots a_n.$$

Damit gibt es in H die Ableitung

$$\begin{aligned} S &\Longrightarrow^* a_1 a_2 \dots a_{i-1} B_i \\ &\Longrightarrow^* a_1 a_2 \dots a_{i-1} a_i \dots a_{j-1} B_i \\ &\Longrightarrow^* a_1 a_2 \dots a_{i-1} a_i \dots a_{j-1} a_i \dots a_{j-1} B_i \\ &\Longrightarrow^* a_1 a_2 \dots a_{i-1} a_i \dots a_{j-1} a_i \dots a_{j-1} a_j \dots a_n \end{aligned}$$

eines Wortes der Länge $i-1+(j-i)+(j-i)+(n-(j-1)) = n+(j-i) > n$. Dies widerspricht $L(H) = \{w\}$. Folglich sind alle Nichtterminale in der Ableitung (3.5) paarweise verschieden. Daher gilt $\mathrm{Var}(H) \geq n$ für jede Grammatik H in regulärer Normalform mit $L(H) = \{w\}$, womit auch $\mathrm{Var}_{nfREG}(\{w\}) \geq n$ gezeigt ist. □

Folgerung 3.14 *$REG \leq^3_{\mathrm{Var}} nfREG$.*

Beweis: Es seien n eine natürliche Zahl und w ein Wort der Länge n über T. Die reguläre Grammatik $G' = (\{S\}, T, \{S \to w\}, S)$ erzeugt $\{w\}$, und damit gilt $\mathrm{Var}_{REG}(\{w\}) = 1$ (da jede Grammatik mindestens ein Nichtterminal hat). Nun folgt die Behauptung aus Lemma 3.13. □

Folgerung 3.14 besagt, dass wir den Übergang zur regulären Normalform mit einer drastischen Erhöhung der Anzahl der Nichtterminale erkaufen. Die Aussagen des obigen Lemmas 3.5 zeigen dagegen, dass der Übergang zur reduzierten Grammatik bzw. zur λ-freien Grammatik kein Erhöhung der Anzahl der Nichtterminale mit sich führt. Ohne Beweis geben wir nun das entsprechende Resultat für den Fall der Chomsky-Normalform an, bei der ebenfalls eine beliebig große Erhöhung eintreten kann.

Lemma 3.15 *$CF \leq^3_{\mathrm{Var}} ChCF$.* □

Zwischen den Klassen der kontextfreien Grammatiken in Chomsky-Normalform und der regulären Grammatiken besteht folgender Zusammenhang.

Lemma 3.16 *$ChCF \leq^2_{\mathrm{Var}} REG$.*

Beweis: Es sei $X_n = \{a_1, a_2, \ldots, a_n\}$ zu jeder natürlichen Zahl $n \geq 1$ ein Alphabet. Wir betrachten die Sprachen $L_n = (\{a_1\}^+\{a_2\}^+ \cdots \{a_n\}^+)^n$ für $n \geq 1$. Jede der Sprachen L_n ist regulär und wird beispielsweise von folgender Grammatik G_n erzeugt:

$$G_n = (N_n, X_n, P_n, S_{1,1})$$

mit

$$\begin{aligned}
N_n &= \{S_{1,1}, S_{1,2}, \ldots, S_{1,n}, S_{2,1}, S_{2,2}, \ldots, S_{2,n}, \ldots, S_{n,1}, S_{n,2}, \ldots, S_{n,n}\},\\
P_n &= \{\, S_{i,j} \to a_j S_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq n \,\}\\
&\quad \cup \{\, S_{i,j} \to a_j S_{i,j+1} \mid 1 \leq i \leq n, 1 \leq j \leq n-1 \,\}\\
&\quad \cup \{\, S_{i,n} \to a_n S_{i+1,1} \mid 1 \leq i \leq n-1 \,\} \cup \{\, S_{n,n} \to a_n \,\}.
\end{aligned}$$

Jede Ableitung in G_n hat folgende Form:

$$\begin{aligned}
S_{1,1} &\Longrightarrow^{n_{1,1}} a_1^{n_{1,1}} S_{1,1} \Longrightarrow a_1^{n_{1,1}+1} S_{1,2}\\
&\Longrightarrow^{n_{1,2}} a_1^{n_{1,1}+1} a_2^{n_{1,2}} S_{1,2} \Longrightarrow a_1^{n_{1,1}+1} a_2^{n_{1,2}+1} S_{1,3}\\
&\;\;\vdots\\
&\Longrightarrow^{n_{1,n-1}} a_1^{n_{1,1}+1} a_2^{n_{1,2}+1} \ldots a_{n-1}^{n_{1,n-1}} S_{1,n-1} \Longrightarrow a_1^{n_{1,1}+1} a_2^{n_{1,2}+1} \ldots a_{n-1}^{n_{1,n-1}+1} S_{1,n}\\
&\Longrightarrow^{n_{1,n}} a_1^{n_{1,1}+1} a_2^{n_{1,2}+1} \ldots a_n^{n_{1,n}} S_{1,n} \Longrightarrow a_1^{n_{1,1}+1} a_2^{n_{1,2}+1} \ldots a_n^{n_{1,n}+1} S_{2,1}\\
&\;\;\vdots\\
&\Longrightarrow^{n_{n,1}+1} a_1^{n_{1,1}+1} a_2^{n_{1,2}+1} \ldots a_n^{n_{1,n}+1} \ldots a_1^{n_{n,1}+1} S_{n,2}\\
&\Longrightarrow^{n_{n,2}+1} a_1^{n_{1,1}+1} a_2^{n_{1,2}+1} \ldots a_n^{n_{1,n}+1} \ldots a_1^{n_{n,1}+1} a_2^{n_{n,2}+1} S_{n,3}\\
&\;\;\vdots\\
&\Longrightarrow^{n_{n,n-1}+1} a_1^{n_{1,1}+1} a_2^{n_{1,2}+1} \ldots a_n^{n_{1,n}+1} \ldots a_1^{n_{n,1}+1} a_2^{n_{n,2}+1} \ldots a_{n-1}^{n_{n,n-1}+1} S_{n,n}\\
&\Longrightarrow^{n_{n,n}+1} a_1^{n_{1,1}+1} a_2^{n_{1,2}+1} \ldots a_n^{n_{1,n}+1} \ldots a_1^{n_{n,1}+1} a_2^{n_{n,2}+1} \ldots a_n^{n_{n,n}+1}.
\end{aligned}$$

Wegen $\mathrm{Var}(G_n) = n^2$ gilt $\mathrm{Var}_{REG}(L_n) \leq n^2$. Zum Erzeugen der beliebig vielen a_1 am Anfang ist ein Nichtterminal erforderlich. Zum Erzeugen der anschließenden a_2 ist ein anderes Nichtterminal nötig, da sonst a_1 und a_2 gemischt auftreten würden. Analog argumentiert man für jeden der ersten n Blöcke. Für die nächsten a_1 benötigt man ein weiteres Nichtterminal. Wenn man das gleiche wie für die ersten a_1 nähme, könnte man die Ableitung beliebig oft wiederholen, wodurch auch das Wort $(a_1 a_2 \cdots a_n)^{2n} \notin L_n$ ableitbar wäre. Da für jeden Block ein neues Nichtterminal benötigt wird, gilt $\mathrm{Var}_{REG}(L_n) \geq n^2$. Zusammen ergibt sich $\mathrm{Var}_{REG}(L_n) = n^2$.

Wir betrachten nun die folgenden kontextfreien Grammatiken G'_n in Chomsky-Normalform:

$$G'_1 = (\{S_1\}, X_1, \{S_1 \to S_1 S_1, S_1 \to a_1\}, S_1)$$

sowie für $n \geq 2$

$$G'_n = (N'_n, X_n, P'_n, S_1)$$

mit

$$\begin{aligned}
N'_n &= \{S_1, S_2, \ldots, S_n, S'_1, S'_2, \ldots, S'_{n-1}, A_2, A_3, \ldots, A_n, A'_1, A'_2, \ldots, A'_{n-1}\},\\
P'_n &= \{\, S_i \to S'_i S_{i+1} \mid 1 \leq i \leq n-1 \,\}\\
&\quad \cup \{\, S'_i \to A'_1 A_2 \mid 1 \leq i \leq n-1 \,\}\\
&\quad \cup \{\, S_n \to A'_1 A_2 \,\}\\
&\quad \cup \{\, A_i \to A'_i A_{i+1} \mid 2 \leq i \leq n-1 \,\}\\
&\quad \cup \{\, A'_i \to A'_i A'_i, A'_i \to a_i \mid 1 \leq i \leq n-1 \,\}\\
&\quad \cup \{\, A_n \to A_n A_n, A_n \to a_n \,\}.
\end{aligned}$$

Für jede natürliche Zahl $n \geq 1$ gilt $L(G'_n) = L_n$ und $\mathrm{Var}(G'_n) \leq 4n$. Folglich gilt auch $\mathrm{Var}_{ChCF}(L_n) \leq 4n$. Wegen

$$\lim_{n\to\infty} \frac{\mathrm{Var}_{REG}(L_n)}{\mathrm{Var}_{ChCF}(L_n)} \geq \lim_{n\to\infty} \frac{1}{4} n = \infty$$

erhalten wir $ChCF \leq^2_{\mathrm{Var}} REG$. □

Ob dieses Ergebnis optimal ist, oder ob sogar $ChCF \leq^3_{\mathrm{Var}} REG$ gilt, ist uns nicht bekannt.

Zusammenfassend erhalten wir folgende Beziehung zwischen den Klassen bezüglich des Maßes Var:

$$MON \leq^3_{\mathrm{Var}} CF \approx_{\mathrm{Var}} redCF \approx_{\mathrm{Var}} CF\text{-}\lambda \leq^3_{\mathrm{Var}} REG \leq^3_{\mathrm{Var}} nfREG$$

und

$$CF \leq^3_{\mathrm{Var}} ChCF \leq^2_{\mathrm{Var}} REG.$$

Uns sind zur Zeit außer dem trivialen Fakt $\mathrm{Var}_{RE}(L) \leq \mathrm{Var}_{MON}(L)$ (siehe Lemma 3.4) keine Aussagen über das Verhältnis zwischen RE und MON hinsichtlich der Anzahl der Nichtterminale bekannt.

Bisher haben wir stets ausgehend von einer Sprache die zugehörige minimale Anzahl von Nichtterminalen für die Erzeugung der Sprache bestimmt. Man kann das Problem auch umgekehrt betrachten. Gegeben sei eine Zahl n, und wir versuchen die Menge der Sprachen zu bestimmen, die mit höchstens n Nichtterminalen erzeugt werden können. Wir geben zwei Resultate dieser Art an.

Lemma 3.17 *Für eine reguläre Sprache L gilt genau dann $\mathrm{Var}_{REG}(L) = 1$, wenn es zwei endliche Mengen R_1 und R_2 von Wörtern so gibt, dass $L = R_1^* R_2$ gilt.*

Beweis: Es sei $G = (N, T, P, S)$ eine reguläre Grammatik mit $\mathrm{Var}(G) = 1$, d. h. $N = \{S\}$. Dann gibt es in P nur Regeln der Form $S \to wS$ und $S \to w$ mit $w \in T^*$. Wir setzen

$$R_1 = \{w \mid w \in T^*,\ S \to wS \in P\},$$
$$R_2 = \{w \mid w \in T^*,\ S \to w \in P\}.$$

Jede Ableitung in G hat die Form

$$S \Longrightarrow w_1 S \Longrightarrow w_1 w_2 S \Longrightarrow \cdots \Longrightarrow w_1 w_2 \ldots w_t S \Longrightarrow w_1 w_2 \ldots w_t v$$

mit $t \geq 0$, $w_1, w_2, \ldots, w_t \in R_1$ und $v \in R_2$. Folglich gilt $L(G) \subseteq R_1^* R_2$. Umgekehrt ist leicht zu sehen, dass für jedes Wort $w_1' w_2' \ldots w_s' v' \in R_1^* R_2$ eine Ableitung in G existiert, so dass sogar $L(G) = R_1^* R_2$ gilt.

Ist umgekehrt $L \subset T^*$ und $L = R_1^* R_2$ für gewisse endliche Sprachen R_1 und R_2, so wird L offenbar von der (regulären) Grammatik

$$G' = (\{S\}, T, \{S \to wS \mid w \in R_1\} \cup \{S \to w \mid w \in R_2\}, S)$$

erzeugt, womit $\mathrm{Var}_{REG}(L) = 1$ gezeigt ist. □

Lemma 3.18 *Für eine Sprache L gilt genau dann $\mathrm{Var}_{REG}(L) \leq 2$, wenn es endliche Sprachen $R_1, R_2, R_3, R_4, R_5, R_6$ so gibt, dass*

$$L = (R_1^* R_2 R_4^* R_5)^* (R_1^* R_3 \cup R_1^* R_2 R_4^* R_6)$$

gilt.

Beweis: Es sei $G = (\{S, A\}, T, P, S)$. Mit den Mengen

$$R_1 = \{w \mid S \to wS \in P\},$$
$$R_2 = \{w \mid S \to wA \in P\},$$
$$R_3 = \{w \mid S \to w \in P\},$$
$$R_4 = \{w \mid A \to wA \in P\},$$
$$R_5 = \{w \mid A \to wS \in P\},$$
$$R_6 = \{w \mid A \to w \in P\}.$$

kann analog zum vorhergehenden Beweis gezeigt werden, dass $L(G)$ die gewünschte Form hat. Auch der Beweis für $\mathrm{Var}_{REG}(L) \leq 2$ für Sprachen der angegebenen Form ist analog leicht zu führen. □

Man beachte, dass bei Grammatiken mit einem Nichtterminal die Mengen R_2, R_4, R_5 und R_6 leer sind. Dadurch geht eine Sprache mit der Darstellung aus Lemma 3.18 dann in eine Sprache mit der Darstellung $R_1^* R_3$ über, die der aus Lemma 3.17 entspricht.

Lemma 3.19 *Eine Sprache L über einem einelementigen Alphabet $\{a\}$ ist genau dann regulär, wenn es endliche Mengen $U_1 \subseteq \{a\}^*$ und $U_2 \subseteq \{a\}^*$ sowie eine natürliche Zahl $p > 0$ mit $L = U_1 \cup U_2\{a^p\}^*$ gibt.*

Beweis: Ist L endlich, so erhalten wir mit $U_1 = L$ und $U_2 = \emptyset$ die gewünschte Form.

Es sei nun L eine unendliche reguläre Sprache über einem Alphabet $\{a\}$. Dann gibt es einen deterministischen endlichen Automaten $\mathcal{A} = (\{a\}, Z, z_0, \delta, F)$ mit dem Eingabesymbol a, einer Menge $Z = \{z_0, z_1, \ldots, z_n\}$ von Zuständen, dem Anfangszustand z_0, einer Menge $F \subseteq Z$ von akzeptierenden Zuständen und einer Überführungsfunktion $\delta : Z \times X \to Z$, der L akzeptiert.

Die Überführungsfunktion δ ist durch den Graphen in Abbildung 3.1 charakterisiert.

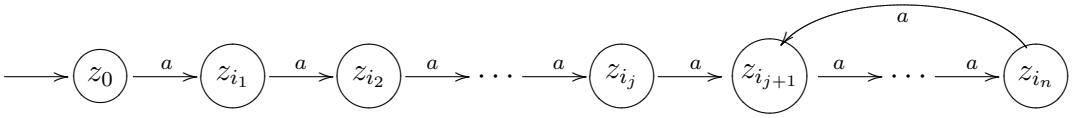


Abbildung 3.1: Transitionsgraph von $\mathcal{A}$

Für die akzeptierte Sprache gilt $L = L_1 \cup L_2$ mit

$$L_1 = \{\, a^k \mid 0 \le k \le j \text{ und } z_{i_k} \in F \,\} \text{ und}$$
$$L_2 = \{\, a^k \mid j+1 \le k \le n \text{ und } z_{i_k} \in F \,\}\{a^{n-j}\}^*.$$

Mit $U_1 = L_1$, $U_2 = \{\, a^k \mid j+1 \le k \le n \text{ und } z_{i_k} \in F \,\}$ und $p = n - j$ erhalten wir die gewünschte Form. □

Folgerung 3.20 *Für jede reguläre Sprache L über einem Alphabet, das aus genau einem Buchstaben besteht, gilt* $\mathrm{Var}_{REG}(L) \le 2$.

Beweis: Nach Lemma 3.19 gibt es zwei endliche Sprachen U_1 und U_2 sowie eine natürliche Zahl p so, dass $L = U_1 \cup U_2\{a^p\}^*$ gilt. Mit $R_1 = \emptyset$, $R_2 = U_2$, $R_3 = U_1$, $R_4 = \{a^p\}$, $R_5 = \emptyset$ und $R_6 = \{\lambda\}$ geht die Darstellung von L durch U_1, U_2 und a^p in die Form aus Lemma 3.18 über, womit die Aussage bewiesen ist. □

Mit jedem Komplexitätsmaß $K \in \{\mathrm{Var}, \mathrm{Prod}, \mathrm{Symb}\}$ ist auch das Problem der algorithmischen Berechnung von $K(G)$ bzw. $K_X(L(G))$ (für eine Menge X von Grammatiken) verbunden. Die folgenden Probleme sind in diesem Zusammenhang von Interesse:

- Bestimme zu einer Grammatik G den Wert $K(G)$.
- Bestimme zu einer Grammatik G aus der Menge X den Wert $K_X(L(G))$.
- Entscheide für eine Grammatik G aus X und eine natürliche Zahl $n \ge 1$, ob $K_X(L(G)) = n$ gilt.
- Entscheide für eine Grammatik G aus X, ob $K(G) = K_X(L(G))$ gilt, d. h. entscheide ob G eine bez. K minimale Grammatik aus X ist.

- Ermittle zu einer Grammatik G aus X eine bez. K minimale Grammatik G' aus X für $L(G)$, d. h. eine Grammatik $G' \in X$ mit $L(G') = L(G)$ und $K(G') = K_X(L(G))$.

Das erste Problem betrifft die Komplexität der Grammatik, während die anderen die Komplexität der erzeugten Sprache betreffen.

Wir wollen nun den Status der Entscheidbarkeit oben genannter Probleme bez. des Maßes Var untersuchen.

Satz 3.21 *Zu jeder der Mengen $X \in \{nfREG, REG, CF, ChCF, CF\text{-}\lambda, MON, RE\}$ und jeder Grammatik $G \in X$ gibt es einen Algorithmus, der* $\mathrm{Var}(G)$ *berechnet.*

Beweis: Wir haben die Anzahl der Nichtterminale der angegebenen Grammatik zu ermitteln. Dies kann durch Zählen während eines Durchlaufs durch die Grammatikbeschreibung geschehen. □

Satz 3.22 *Es gibt keinen Algorithmus, der für eine kontextfreie Grammatik G entscheidet, ob* $\mathrm{Var}_{CF}(L(G)) = 1$ *gilt.*

Beweis: Es seien zwei Mengen $X = \{x_1, x_2, \ldots, x_n\}$ und $Y = \{y_1, y_2, \ldots, y_n\}$ von nichtleeren Wörtern über dem Alphabet $\{a, b\}$ gegeben. Wir betrachten nun folgende Sprachen

$$\begin{aligned}
L_X &= \{ba^{i_1}ba^{i_2}\ldots ba^{i_{k-1}}ba^{i_k}cx_{i_k}x_{i_{k-1}}\ldots x_{i_2}x_{i_1} \mid k \geq 1,\ 1 \leq i_j \leq n \text{ für } 1 \leq j \leq k\},\\
L_Y &= \{ba^{i_1}ba^{i_2}\ldots ba^{i_{k-1}}ba^{i_k}cy_{i_k}y_{i_{k-1}}\ldots y_{i_2}y_{i_1} \mid k \geq 1,\ 1 \leq i_j \leq n \text{ für } 1 \leq j \leq k\},\\
L_{X,Y} &= \{wcv^R \mid w \in L_X,\ v \in L_Y\},\\
L_s &= \{w_1cw_2cw_2^Rcw_1^R \mid w_1, w_2 \in \{a,b\}^*\},\\
L(X,Y) &= \{a,b,c\}^* \setminus (L_{X,Y} \cap L_s).
\end{aligned}$$

Aus den gegebenen Mengen X und Y kann eine kontextfreie Grammatik $G(X,Y)$ mit $L(G(X,Y)) = L(X,Y)$ konstruiert werden. [2]

Das Postsche Korrespondenzproblem (PKP) bez. X und Y besteht darin, festzustellen, ob es eine Folge $i_1 i_2 \ldots i_k$ mit $1 \leq i_j \leq n$ für $1 \leq j \leq k$ so gibt, dass

$$x_{i_1}x_{i_2}\ldots x_{i_k} = y_{i_1}y_{i_2}\ldots y_{i_k}$$

gilt. Die Folge $i_1 i_2 \ldots i_k$ heißt dann Lösung des PKPs. Es ist bekannt, dass das PKP für beliebige (gleichmächtige) Mengen X und Y von Wörtern über $\{a, b\}$ unentscheidbar ist.

Folgende Aussagen setzen die obigen Sprachen in Relation zum PKP. $L_{X,Y} \cap L_s \neq \emptyset$ ist genau dann gültig, wenn das PKP für X und Y eine Lösung hat. Folglich ist $L(X,Y) = \{a,b,c\}^*$ genau dann gültig, wenn das PKP bez. X und Y keine Lösung hat. Somit ist $L(G(X,Y)) = \{a,b,c\}^*$ genau dann gültig, wenn das PKP bez. X und Y keine Lösung

[2] Wir verzichten auf einen vollständigen Beweis dieser Aussage und bemerken nur folgende Fakten: Es lassen sich leicht deterministische Kellerautomaten angeben, die $L_{X,Y}$ bzw. L_s akzeptieren. Da die Menge der deterministischen kontextfreien Sprachen unter Komplementbildung abgeschlossen ist, ergibt sich, dass auch $\{a,b,c\}^* \setminus L_{X,Y}$ und $\{a,b,c\}^* \setminus L_s$ kontextfrei sind. Als Vereinigung dieser Mengen ist auch $L(X,Y)$ kontextfrei. Da zu den Operationen entsprechende Konstruktionen der zugehörigen Kellerautomaten bzw. Grammatiken existieren und der Übergang von Kellerautomaten zu Grammatiken und umgekehrt jeweils algorithmisch vollzogen werden können, lässt sich $G(X,Y)$ algorithmisch bestimmen.

hat. Damit ergibt sich aus der Unentscheidbarkeit des PKPs die Unentscheidbarkeit der Frage, ob $L(G(X,Y))$ die Menge aller Wörter über $\{a,b,c\}$ ist.

Gilt $L(X,Y) = \{a,b,c\}^*$, so wird $L(X,Y)$ offenbar von der Grammatik

$$G(X,Y) = (\{S\}, \{a,b,c\}, \{S \to aS,\ S \to bS,\ S \to cS,\ S \to \lambda\}, S)$$

erzeugt, woraus wir sofort $\mathrm{Var}_{CF}(L(X,Y)) = 1 = \mathrm{Var}_{CF}(L(G(X,Y))$ erhalten.

Wir werden nun zeigen, dass bei $L(X,Y) \neq \{a,b,c\}^*$ die Beziehung

$$\mathrm{Var}_{CF}(L(X,Y)) = \mathrm{Var}_{CF}(L(G(X,Y))) \geq 2$$

gilt. Daher würde eine Entscheidbarkeit der Frage, ob $\mathrm{Var}_{CF}(L(G)) = 1$ ist, die Entscheidbarkeit von $L(G(X,Y)) = \{a,b,c\}^*$ implizieren. Vom letzteren Problem wissen wir aber schon, dass es unentscheidbar ist. Daher muss es unentscheidbar sein, ob $\mathrm{Var}_{CF}(L(G)) = 1$ gilt.

Wir nehmen nun an, dass es eine kontextfreie Grammatik $H = (\{S\}, \{a,b,c\}, P, S)$ mit nur einem Nichtterminal gibt, die $L(X,Y) \neq \{a,b,c\}^*$ erzeugt. Wegen $L_{X,Y} \cap L_s \neq \emptyset$ gibt es eine Lösung $i_1 i_2 \dots i_{k-1} i_k$ des PKPs. Wir setzen

$$I = ba^{i_1}ba^{i_2}\dots ba^{i_k},\ X = x_{i_1}x_{i_2}\dots x_{i_k},\ J = I^R \text{ und } Y = X^R.$$

Dann gilt

$$I^m cX^m cY^m cJ^m \in L_{X,Y} \cap L_s \quad \text{für alle } m \geq 1 \tag{3.6}$$

und folglich $w_m = I^{m-1}cX^mcY^mcJ^m \in L(X,Y)$ für alle $m \geq 1$. Wir betrachten eine Ableitung von $w = w_m$ mit $m \geq \max\{|u| \mid S \to u \in P\}$. Dann gibt es ein α so, dass $S \Longrightarrow \alpha \Longrightarrow^* w$ und $\alpha = \alpha_0 S \alpha_1$ und $w = \alpha_0 w_0 w_1$ für gewisse $\alpha_0, w_0, w_1 \in T^*$ mit $S \Longrightarrow^* w_0$, $\alpha_1 \Longrightarrow^* w_1$ und $\alpha_0 w_1 \neq \lambda$ gelten. Aufgrund der Wahl von m enthält α_0 kein c. Daher gibt es Wörter z_0 und z_1 so, dass $z_0 z_1 = I$ und $\alpha_0 = I^{i_0} z_0$ gelten. Wegen $\alpha_0 w_0 w_1 = w$ erhalten wir

$$\alpha_0 z_1 z_0 w_0 w_1 = I^m cX^m cY^m cJ^m. \tag{3.7}$$

Weiterhin gilt $z_1 z_0 w_0 \in L(X,Y)$. Dies ist klar, falls $z_1 z_0 w_0$ keine drei Vorkommen des Buchstaben c hat. Kommen aber drei c vor, so muss

$$z_1 z_0 w_0 = z_1 z_0 u_1 c u_2 c u_3 c u_4 \tag{3.8}$$

sein. Aus (3.7) folgt dann $\alpha_0 z_1 z_0 u_1 c u_2 c u_3 c u_4 w_1 = I^m cX^m cY^m cJ^m$ und damit

$$\alpha_0 z_1 z_0 u_1 = I^m,\ u_2 = X^m,\ u_3 = Y^m \text{ und } u_4 w_1 = J^m. \tag{3.9}$$

Nehmen wir nun an, dass $z_1 z_0 w_0 \notin L(X,Y)$ gilt. Dann muss $z_1 z_0 w_0 \in L_{X,Y} \cap L_s$ sein, woraus mit (3.8) und (3.9) $z_1 z_0 u_1 c X^m c Y^m c u_4 \in L_{X,Y} \cap L_s$ folgt. Beachten wir die Struktur der Wörter in $L_{X,Y} \cap L_s$, so erhalten wir $z_1 z_0 u_1 = I^m$ und $u_4 = J^m$. Damit ergibt sich aus (3.9) $\alpha_0 = \lambda$ und $w_1 = \lambda$. Dies ist ein Widerspruch zu $\alpha_0 w_1 \neq \lambda$.

Deshalb gibt es eine Ableitung $S \Longrightarrow^* z_1 z_0 w_0$. Hieraus resultiert die Ableitung

$$S \Longrightarrow \alpha_0 S \alpha_1 \Longrightarrow^* \alpha_0 z_1 z_0 w_0 \alpha_1 \Longrightarrow^* \alpha_0 z_1 z_0 w_0 w_1.$$

Mittels (3.9) ergibt dies $I^m cX^m cY^m cJ^m \in L(H) = L(X,Y)$ im Widerspruch zu (3.6). □

Folgerung 3.23 *Für eine gegebene kontextfreie Grammatik G und eine natürliche Zahl $n \geq 1$ ist es unentscheidbar, ob $\mathrm{Var}_{CF}(L(G)) = n$ gilt.*

Beweis: Für $n = 1$ stimmt die Behauptung mit der Aussage von Satz 3.22 überein und ist damit bewiesen.

Es sei nun $n \geq 2$. Wir betrachten anstelle von $G(X, Y)$ eine Grammatik $G_n(X, Y)$ mit

$$L(G_n(X, Y)) = L(X, Y) \cup \{de^2\}^* \cup \{de^3\}^* \cup \cdots \cup \{de^n\}^*.$$

Unter Verwendung von Satz 3.9 und den Betrachtungen beim Beweis von Satz 3.22 zeigt man sehr leicht, dass $\mathrm{Var}_{CF}(L(G_n(X, Y)) = n$ genau dann gilt, wenn $L(X, Y) = \{a, b, c\}^*$ ist, also wenn das PKP bez. X und Y keine Lösung hat. □

Folgerung 3.24 *Es gibt keinen Algorithmus, der zu einer kontextfreien Grammatik G den Wert $\mathrm{Var}_{CF}(L(G))$ berechnet.*

Beweis: Angenommen, es gäbe einen solchen Algorithmus. Dann kann $\mathrm{Var}_{CF}(L(G))$ algorithmisch ermittelt werden und durch Vergleich mit einer gegebenen natürlichen Zahl n kann $\mathrm{Var}_{CF}(L(G)) = n$ algorithmisch entschieden werden.

Dies widerspricht Folgerung 3.23. □

Folgerung 3.25 *Es gibt keinen Algorithmus, der zu einer kontextfreien Grammatik G eine bez.* Var *minimale kontextfreie Grammatik G' für $L(G)$, d. h. G' erfüllt $L(G') = L(G)$ und $\mathrm{Var}(G') = \mathrm{Var}_{CF}(L(G))$, liefert.*

Beweis: Angenommen es gäbe einen solchen Algorithmus. Dann bestimmen wir zu G zuerst die minimale Grammatik G' und dann von dieser $\mathrm{Var}(G')$. Dies liefert wegen $\mathrm{Var}(G') = \mathrm{Var}_{CF}(L(G))$ einen Algorithmus zur Berechnung von $\mathrm{Var}_{CF}(L(G))$, der nach Folgerung 3.24 nicht existiert. □

Satz 3.26 *Es ist unentscheidbar, ob für eine gegebene kontextfreie Grammatik G die Beziehung $\mathrm{Var}(G) = \mathrm{Var}_{CF}(L(G))$ gilt.*

Beweis: Wir betrachten erneut das PKP bezüglich der Mengen $X = \{x_1, x_2, \ldots, x_n\}$ und $Y = \{y_1, y_2, \ldots, y_n\}$ über dem Alphabet $\{a, b\}$ und dazu die Grammatik

$$H(X, Y) = (\{S, S', T, T', R\}, \{a, b, c, d\}, P, S)$$

mit

$$\begin{aligned} P = \{&S \to aSa,\ S \to bSb,\ S \to bT'RT'a,\ S \to aT'RT'b,\\ &S \to aTRa,\ S \to bTRb,\ S \to aRTa,\ S \to bRTb,\\ &T \to Ta,\ T \to Tb,\ T \to a,\ T \to b,\\ &T' \to T,\ T' \to \lambda,\\ &R \to cSc,\ R \to cS'c,\ R \to dTd\}\\ &\cup \{S' \to x_i S' y_i^R \mid 1 \leq i \leq n\} \cup \{S' \to x_i dT dy_i^R \mid 1 \leq i \leq n\}. \end{aligned}$$

Es ist leicht zu sehen, dass

$$\begin{aligned} L = L(H(X,Y)) = \{u_1 c u_2 c \dots c u_{k-1} c u_k d w_0 d v_k c v_{k-1} c \dots c v_2 c v_1 \mid \\ w_0 \in \{a,b\}^+,\ k \geq 1,\ u_i, v_i \in \{a,b\}^+ \text{ für } 1 \leq i \leq k, \\ u_j \neq v_j^R \text{ für } 1 \leq j < k \text{ und entweder } u_k \neq v_k^R \text{ oder} \\ u_k = x_{i_1} x_{i_2} \dots x_{i_l} = y_{i_1} y_{i_2} \dots y_{i_l} = v_k^R \text{ für gewisse } i_1, i_2, \dots, i_l\}. \end{aligned}$$

Dabei tritt der Fall $u_k = v_k^R$ offenbar nur ein, wenn es eine Lösung des PKPs bez. X und Y gibt ($i_1 i_2 \dots i_l$ ist gerade eine Lösung).

Wir stellen nun erst einmal fest, dass die Nichtterminale T' und R eigentlich nicht benötigt werden, wie im Beweis von Lemma 3.8 gezeigt wurde. Entsprechend der dortigen Konstruktion erhalten wir eine Grammatik $H'(X,Y)$ mit 3 Nichtterminalen und $L(H'(X,Y)) = L$. Wir zeigen nun

$$\mathrm{Var}_{CF}(L) = \begin{cases} 2, & \text{wenn das PKP bez. } X \text{ und } Y \text{ keine Lösung hat,} \\ 3, & \text{wenn das PKP bez. } X \text{ und } Y \text{ eine Lösung hat.} \end{cases} \tag{3.10}$$

Habe das PKP bez. X und Y zuerst keine Lösung. Dann können wir auf das Nichtterminal S' und alle zugehörigen Regeln verzichten, da man aus S' wegen der Nichtexistenz einer Lösung nur Wörter der Form $u_k d w_0 d v_k$ mit $u_k \neq v_k^R$ erzeugen kann, die auch aus S erzeugt werden können. Damit gilt $\mathrm{Var}_{CF}(L) \leq 2$.

Angenommen, es wäre $\mathrm{Var}_{CF}(L) = 1$. Dann gibt es eine kontextfreie Grammatik $H' = (\{A\}, \{a,b,c,d\}, P', A)$ mit $L(H') = L$. Wegen $w_m = a d a^m d a^m \in L$ für alle $m \geq 2$ gibt es eine Ableitung $A \Longrightarrow \alpha \Longrightarrow^* a d a^m d a^m$ von w_m in H'. Es sei m hinreichend groß. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass $\alpha \neq A$ gilt. Enthält α zweimal den Buchstaben A, so ist aus α und damit aus A ein Wort mit mindestens vier Vorkommen von d ableitbar. Ein solches Wort liegt aber nicht in L. Folglich erhalten wir $\alpha = \alpha_1 A \alpha_2$ für gewisse $\alpha_1 \in T^*$ und $\alpha_2 \in T^*$ (da $\alpha \neq w_m$ wegen der Wahl von m sein muss). Wegen der hinreichend großen Wahl von m muss auch $\alpha_2 = a^j$ mit $0 \leq j < m$ gelten.

Ist $\alpha_1 = \lambda$, so ist $j \geq 1$. Damit erhalten wir wegen $a^{j+1} d a d a \in L$ die Ableitung $A \Longrightarrow A a^j \Longrightarrow^* a^{j+1} d a d a a^j \notin L$, womit ein Widerspruch erreicht wurde.

Ist $\alpha_1 = a$, so muss aus A ein Wort abgeleitet werden, dass mit d beginnt. Dies widerspricht erneut der Form der Wörter in L.

Gilt $\alpha_1 = a d a^r$, so erhalten wir die Ableitung

$$A \Longrightarrow a d a^r A a^j \Longrightarrow^* a d a^r a d a^m d a^m a^j \notin L.$$

Weitere Möglichkeiten gibt es aufgrund der Wahl von m für α_1 nicht, womit gezeigt ist, dass zur Erzeugung von L mindestens 2 Nichtterminale benötigt werden.

Analog zeigt man, dass drei Nichtterminale erforderlich sind, wenn das PKP eine Lösung hat (der Beweis ist aber erheblich umfangreicher, siehe [7]).

Wegen (3.10) erhalten wir, dass $G(X,Y)$ nach Ersetzen von T' und R' genau dann minimal ist, wenn das PKP bez. X und Y eine Lösung hat. Da letzteres unentscheidbar ist, ist also auch unentscheidbar, ob G minimal ist. □

Hinsichtlich regulärer Grammatiken in Normalform stellt sich die Situation völlig anders dar. Alle oben erwähnten Entscheidungsprobleme werden entscheidbar. Wir beweisen

zuerst, dass es einen Algorithmus gibt, der zu einer Grammatik G in Normalform eine für $L(G)$ minimale Grammatik in regulärer Normalform bestimmt.

Satz 3.27 *Es gibt einen Algorithmus, der zu einer gegebenen regulären Grammatik G in Normalform eine reguläre Grammatik G' in Normalform bestimmt, für die $L(G') = L(G)$ und $\mathrm{Var}(G') = \mathrm{Var}_{nfREG}(L(G))$ gelten.*

Beweis: Es sei $G = (N, T, P, S)$ gegeben. Wir bestimmen zuerst $n = \mathrm{Var}(G)$. Nach Satz 3.21 ist dies algorithmisch möglich. Wir geben nun eine Konstruktion aller Grammatiken mit k Nichtterminalen, $k < n$, und dem Terminalalphabet. Dazu setzen wir

$$N_k = \{A_1, A_2, \ldots, A_k\}.$$

Weiterhin wählen wir für jede natürliche Zahl i mit $1 \leq i \leq k$ eine Teilmenge $M(k,i)$ von T und für jedes Paar (i,j) mit $1 \leq i,j \leq k$ eine Teilmenge $M(k,i,j)$ von T und setzen

$$\begin{aligned} P(k,i) &= \{A_i \to x \mid x \in M(k,i)\}, \\ P(k,i,j) &= \{A_i \to xA_j \mid x \in M(k,i,j)\}. \end{aligned}$$

Wir betrachten die Grammatik

$$H = \left(N_k, T, \bigcup_{1 \leq i \leq k} P(k,i) \cup \bigcup_{1 \leq i,j \leq k} P(k,i,j), A_1\right).$$

Es können alle Grammatiken mit k Nichtterminalen in dieser Weise erhalten werden. Beginnend mit $k = 1$ testen wir alle möglichen Grammatiken H, ob $L(H) = L(G)$ gilt. Trifft dies auf eine Grammatik H zu, so ist $\mathrm{Var}_{nfREG}(L(G)) = \mathrm{Var}(H)$. Gilt dagegen $L(H) \neq L(G)$ für alle H, so setzen wir mit $k = 2$ fort. Haben wir auch für $k = \mathrm{Var}(G) - 1$ noch keine zu G äquivalente Grammatik, so ist G selbst eine minimale Grammatik für die Sprache $L(G)$. □

Aus Satz 3.27 erhalten wir durch einfache Schlüsse, dass auch die anderen Entscheidbarkeitsprobleme für reguläre Grammatiken in Normalform entscheidbar sind.

Folgerung 3.28

1. *Es gibt einen Algorithmus, der zu gegebener regulärer Grammatik G in Normalform $\mathrm{Var}_{nfREG}(L(G))$ bestimmt.*

2. *Es ist entscheidbar, ob eine gegebene Grammatik G in regulärer Normalform minimal für $L(G)$ ist, d. h. ob $\mathrm{Var}(G) = \mathrm{Var}_{nfREG}(L(G))$ gilt.* □

Es ist den Autoren nicht bekannt, welchen Entscheidbarkeitsstatus die betrachteten Probleme bei beliebigen regulären Grammatiken haben.

Übungsaufgaben

1. Man beweise Lemma 3.15.

2. Man gebe einen ausführlichen Beweis von Lemma 3.18 an.

3. Es sei $T_n = \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n\}$ für $n \geq 2$. Außerdem sei

$$L_n = \{\, a_1^{p_1} a_2^{p_2} \cdots a_n^{p_n} b_1^{p_1} b_2^{p_2} \cdots b_n^{p_n} \mid p_i \geq 1, 1 \leq i \leq n \,\}$$

   für $n \geq 2$. Jede dieser Sprachen ist nicht kontextfrei. Man weise nach, dass für jede natürliche Zahl $n \geq 2$ zum Erzeugen der Sprache L_n mittels einer monotonen Grammatik zwei Nichtterminale ausreichen.

4. a) Man beweise die Verbesserung der Aussage aus Satz 3.10 zu $\mathrm{Var}_{RE}(L) \leq 2$.
   b) Man beweise $\mathrm{Var}_{MON}(L) \leq \#(T)+1$ als Verbesserung der Aussage aus Satz 3.11.

3.3. Anzahl der Regeln

Wir beginnen mit Lemmata, die denen vom Beginn von Abschnitt 2.2 entsprechen.

Lemma 3.29 *Für je zwei Mengen X und Y von Grammatiken mit $X \subseteq Y$ und jede Sprache $L \in \mathcal{L}(X)$ gilt* $\mathrm{Prod}_Y(L) \leq \mathrm{Prod}_X(L)$.

Beweis: Der Beweis wird analog zu dem von Lemma 3.4 geführt. □

Lemma 3.30

i) *Es sei $X \in \{nfREG, REG, ChCF, CF\text{-}\lambda, CF\}$. Zu jeder Sprache $L \in \mathcal{L}(X)$ gibt es eine reduzierte Grammatik $G = (N, T, P, S)$, die die Sprache L erzeugt und für die* $\mathrm{Prod}_X(L) = \mathrm{Prod}(G)$ *gilt.*

ii) *Es sei $X \in \{REG, CF\text{-}\lambda, CF\}$. Ist $G \in X$ eine reduzierte Grammatik mit den Eigenschaften* $\mathrm{Prod}(G) = \mathrm{Prod}_X(L(G))$ *und $\#(L(G)) \geq 2$, so gilt für jedes Nichtterminal A von G, dass auch $L(G, A)$ mindestens zwei Wörter enthält.*

Beweis: i) Der Beweis erfolgt analog zu dem von Lemma 3.5 i).

ii) Angenommen, es gibt ein Nichtterminal A mit $\#(L(G, A)) \leq 1$. Wegen der vorausgesetzten Reduziertheit gilt $L(G, A) \neq \emptyset$. Somit enthält $L(G, A)$ genau ein Wort w. Wegen der Reduziertheit gibt es auch eine Regel mit linker Seite A. Ersetzen wir A in allen rechten Seiten durch w und Streichen alle Regeln mit linker Seite A, so entsteht eine Grammatik G' mit weniger Regeln und $L(G') = L(G)$, was nach Voraussetzung nicht möglich ist. □

Wir bestimmen nun für einige Sprachen die Anzahl der nötigen Regeln bei Verwendung gewisser Typen von Grammatiken.

Lemma 3.31 *Zu $n \geq 1$ sei*

$$L_n = \{a^{2^i} \mid 1 \leq i \leq n\}.$$

Dann gilt

$$\mathrm{Prod}_{CF}(L_n) = n.$$

Beweis: Offensichtlich erzeugt die Grammatik

$$G_n = (\{S\}, \{a\}, \{S \to a^{2^i} \mid 1 \leq i \leq n\}, S)$$

die Sprache L_n, womit $\mathrm{Prod}_{CF}(L_n) \leq n$ nachgewiesen ist.

Nach Lemma 3.30 können wir voraussetzen, dass es eine reduzierte Grammatik H_n mit $L(H_n) = L_n$ und $\mathrm{Prod}(H_n) = \mathrm{Prod}_{CF}(L_n)$ gibt, bei der jedes Nichtterminal mindestens zwei Wörter erzeugt.

Wir zeigen zuerst, dass es in H_n keine Ableitung $S \Longrightarrow^* xAyBz$ mit Nichtterminalen A und B und terminalen Wörtern x, y und z gibt. Angenommen, dies wäre der Fall. Dann können wir aus A zwei Wörter a^{i_1} und a^{i_2} und aus B zwei Wörter a^{j_1} und a^{j_2} mit $i_1 < i_2$ und $j_1 < j_2$ ableiten. Setzen wir noch $s = |xyz|$, so sind aus S die Wörter a^{s_1}, a^{s_2}, a^{s_3} und a^{s_4} mit

$$\begin{aligned} s_1 &= s + i_1 + j_1, \\ s_2 &= s + i_2 + j_1, \\ s_3 &= s + i_1 + j_2, \\ s_4 &= s + i_2 + j_2 \end{aligned}$$

ableitbar. Offensichtlich gelten weiterhin

$$s_4 - s_3 = s_2 - s_1 > 0 \quad \text{und} \quad s_3 > s_1. \tag{3.11}$$

Beachten wir noch, dass die Wörter $a^{s_i} \in L_n$ liegen, so muss $s_i = 2^{t_i}$ für $1 \leq i \leq 4$ gelten. Somit ergibt sich aus (3.11)

$$2^{t_4} = 2^{t_3} + (2^{t_2} - 2^{t_1}) = 2^{t_1}(2^{t_3 - t_1} + 2^{t_2 - t_1} - 1),$$

womit wir einen Widerspruch erhalten haben, da wegen $t_3 - t_1 \geq 1$ und $t_2 - t_1 \geq 1$ die ungerade Zahl $2^{t_3 - t_1} + 2^{t_2 - t_1} - 1$ ein Teiler von 2^{t_4} ist.

Wir haben damit gezeigt, dass es keine Satzformen gibt, die mindestens zwei Nichtterminale enthalten (da die restlichen Nichtterminale wegen der Reduziertheit zu terminalen Wörtern abgeleitet werden können). Nehmen wir daher nun an, dass es zwei Ableitungen verschiedener Wörter gibt, in denen das gleiche Nichtterminal auftritt. Es seien also

$$S \Longrightarrow^* x_1 A y_1 \quad \text{und} \quad S \Longrightarrow^* x_2 A y_2$$

zwei derartige Ableitungen. Wir setzen $u_1 = |x_1 y_1|$ und $u_2 = |x_2 y_2|$. Weiterhin sei wie oben $\{a^{i_1}, a^{i_2}\} \subseteq L(G, A)$. Dann erhalten wir, dass die Wörter a^{v_i}, $1 \leq i \leq 4$, mit

$$v_1 = u_1 + i_1,\ v_2 = u_1 + i_2,\ v_3 = u_2 + i_1 \text{ und } v_4 = u_2 + i_2$$

in L_n liegen. Ohne Beschränkung der Allgemeinheit können wir $u_1 < u_2$ und $i_1 < i_2$ annehmen und wie oben ein Widerspruch herleiten.

Damit ist gezeigt, dass die Ableitungen der Wörter a^{2^i} und a^{2^j}, $1 \leq i < j \leq n$, abgesehen von S keine gemeinsamen Nichtterminale haben. Erfolgt die Ableitung direkt, d. h. mittels $S \to a^{2^i}$ bzw. über ein Nichtterminal A_i, so wird jeweils mindestens eine Regel benutzt, die für einen anderen Wert j nicht in Frage kommt. Daher gibt es mindestens n Regeln, womit $\mathrm{Prod}_{CF}(L_n) \geq n$ ebenfalls gezeigt ist. □

Lemma 3.32 *Zu $n \geq 2$ sei*

$$L_n = \{a^{2^i} \mid 1 \leq i \leq n\} \cup \{a^{j2^{n+1}} \mid j \geq 1\}.$$

Dann gelten

$$\mathrm{Prod}_{MON}(L_n) \leq 12 \quad \textit{und} \quad \mathrm{Prod}_{CF}(L_n) \geq n.$$

Beweis: Die monotone Grammatik

$$G_n = (\{S, \#, A, A', B, C\}, \{a\}, P_n, S)$$

mit

$$\begin{aligned} P_n = \{&S \to aa,\ S \to \#Aa\#,\ S \to C,\\ &\#A \to \#aaA',\ A'a \to aaA',\ A'\# \to Aa\#,\ aA \to Aa,\\ &\#A \to aB,\ Ba \to aB,\ B\# \to aa,\\ &C \to Ca^{2^{n+1}},\ C \to a^{2^{n+1}}\} \end{aligned}$$

erzeugt L_n, wie man leicht nachrechnet. Damit gilt $\mathrm{Prod}_{MON}(L_n) \leq 12$.

Um die Aussage bezüglich kontextfreier Grammatiken zu beweisen, gehen wir wie beim Beweis des vorhergehenden Lemmas vor. Wir haben nur jeweils mehr Fälle zu unterscheiden. So erhalten wir wie beim Beweis des Lemmas 3.31 die Relationen

$$\begin{aligned} s_1 &= s + i_1 + j_1,\\ s_2 &= s + i_2 + j_1,\\ s_3 &= s + i_1 + j_2,\\ s_4 &= s + i_2 + j_2, \end{aligned}$$

nur dass jetzt nicht alle der s_i, $1 \leq i \leq 4$, Potenzen von 2 sein müssen.

Sind alle s_i Potenzen von 2, erhalten wir wie oben einen Widerspruch.

Wir betrachten nun den Fall, dass $s_i \leq 2^n$ für $i \in \{1, 2, 3\}$ und $s_4 \geq 2^{n+1}$ gelten. Damit erhalten wir $s_i = 2^{t_i}$ mit $1 \leq t_i \leq n$ für $i \in \{1, 2, 3\}$ und $s_4 = j2^{n+1}$ für ein $j \geq 1$. Ist $j = 1$ haben wir es wiederum mit vier Zweierpotenzen zu tun und wir erhalten analog einen Widerspruch. Es sei daher $j \geq 2$ Weiter ergibt sich erneut wegen $s_4 - s_3 = s_2 - s_1$die Beziehung

$$2^{n+2} \leq j2^{n+1} = 2^{t_3} + 2^{t_2} - 2^{t_1} \leq 2^{t_3} + 2^{t_2} \leq 2^n + 2^n = 2^{n+1},$$

womit der gewünschte Widerspruch hergestellt ist.

Die Herleitung der Widersprüche in den anderen Fällen bleibt dem Leser überlassen (siehe [17]).

Damit erhalten wir wie im Beweis von Lemma 3.31, dass mindestens n Regeln erforderlich sind. □

Die Aussage des folgenden Lemmas unterscheidet sich von denen der bisherigen Lemmata, weil in ihm nicht nur eine Aussage zur Anzahl der nötigen Regeln für eine Sprache gemacht wird, sondern eine diesbezügliche Aussage für *alle* endlichen Sprachen getroffen wird.

Lemma 3.33 *Es seien L eine endliche Sprache, die mehr als n Wörter enthält, und G eine reguläre Grammatik mit $L(G) = L$. Dann gilt* $\text{Prod}(G) > \log_2(n) + 1$.

Beweis: Wie beweisen die zum Lemma äquivalente Formulierung: *Erzeugt eine reguläre Grammatik G mit höchstens n Regeln eine endliche Sprache L, so enthält L höchstens 2^{n-1} Wörter.*

Wir beweisen diese Aussage mittels vollständiger Induktion über die Anzahl n der Regeln.

Ist $n = 1$, so erzeugt $G = (N, T, P, S)$ genau ein Wort, wenn die Regel die Form $S \to w \in T^*$ hat, oder die leere Menge, wenn die einzige Regel aus P auf der rechten Seite mindestens ein Nichtterminal hat. In beiden Fällen gilt $\#(L(G)) \leq 1 = 2^0$.

Es sei nun $n \geq 2$ und es gelte die Aussage für alle Werte i mit $i < n$. Wir zeigen, dass sie dann auch für n gilt. Es sei also $G = (N, T, P, S)$ eine reguläre Grammatik mit n Regeln.

Wir nehmen zuerst an, dass es eine Ableitung $S \Longrightarrow^* uS \Longrightarrow^* uv$ mit $uv \in T^*$ gibt. Ist $u \neq \lambda$, so können wir $u^n v$ für alle $n \geq 1$ ableiten. Dies impliziert, dass $L(G)$ im Widerspruch zu unserer Voraussetzung unendlich ist. Ist dagegen $u = \lambda$, so gibt es ein Nichtterminal A ($A = S$ ist zugelassen) mit $A \to S$. Um Unendlichkeit der Sprache zu vermeiden, darf es dann auch keine Ableitung $S \Longrightarrow^* u'A$ mit $u' \neq \lambda$ geben. Folglich wird die erzeugte Sprache nicht verändert, wenn wir die Regel $A \to S$ streichen. Die dadurch entstandene Grammatik G' enthält nur $n - 1$ Regeln und erzeugt ebenfalls $L(G)$. Nach Induktionsvoraussetzung gilt dann

$$\#(L(G)) = \#(L(G')) \leq 2^{n-2} < 2^{n-1}.$$

Wir nehmen nun an, dass S in keiner Ableitung – außer als Startsymbol – vorkommt. Es seien $S \to w_1$, $S \to w_2$, …, $S \to w_r$ die Regeln von P mit linker Seite S. Dann kommt S in keinem der Wörter w_i, $1 \leq i \leq r$ vor.

Gilt $r = n$, so ist

$$L(G) = \{z \mid z \in \{w_1, w_2, \ldots, w_r\},\ z \in T^*\},$$

da für Nichtterminale in den Wörtern w_i keine Regeln zur Weiterableitung existieren. Damit gilt

$$\#(L(G)) \leq r = n \leq 2^{n-1}$$

für $n \geq 2$.

Es sei nun $r < n$. Zu i mit $1 \leq i \leq r$ setzen wir

$$L_i = \{w \mid w_i \Longrightarrow^* w,\ w \in T^*\}.$$

Offenbar gilt

$$L = \bigcup_{i=1}^{r} L_i.$$

Gilt $w_i \in T^*$ und damit $L_i = \{w_i\}$ für ein i, $1 \leq i \leq n$, so erhalten wir $\#(L_i) = 1 \leq 2^{n-r-1}$.

Gilt $w_i = v_i A$ für ein i, $1 \leq i \leq n$, so setzen wir

$$L_i' = \{v \mid A \Longrightarrow^* v,\ v \in T^*\}.$$

Nach Konstruktion gilt dann

$$L_i = \{v_i v \mid v \in L_i'\}$$

und damit $\#(L_i) = \#(L_i')$. Beachten wir noch, dass in den Ableitungen der Wörter aus L_i' aus A das Startsymbol S nicht mehr vorkommt und daher höchstens die $n-r$ Regeln aus $P \setminus \{S \to w_1, S \to r_2, \ldots, S \to w_r\}$ zur Anwendung kommen, so ergibt sich nach Induktionsvoraussetzung $\#(L_i) = \#(L_i') \leq 2^{n-r-1}$.

Somit ergibt sich

$$\#(L(G)) = \#\left(\bigcup_{i=1}^{r} L_i\right) \leq r \cdot 2^{n-r-1} \leq 2^r \cdot 2^{n-r-1} = 2^{n-1}. \qquad \square$$

Lemma 3.34 *Zu $n \geq 1$ sei*

$$L_n = \{a^{2^n}, a^{2^n+1}, a^{2^n+2}, \ldots, a^{2^{n+1}}\}.$$

Dann gelten

$$\mathrm{Prod}_{CF}(L_n) \leq 3 \quad \textit{und} \quad \mathrm{Prod}_{REG}(L_n) \geq n.$$

Beweis: Die kontextfreie Grammatik

$$G_n = (\{S, A\}, \{a\}, \{S \to A^{2^n},\ A \to a,\ A \to a^2\}, S)$$

mit drei Regeln erzeugt offenbar L_n, womit $\mathrm{Prod}_{CF}(L_n) \leq 3$ gezeigt ist.

Zu $n \geq 1$ hat die Sprache L_n genau $2^n + 1$ Elemente. Damit gilt nach Lemma 3.33 für jede reguläre Grammatik G, die L_n erzeugt, $\mathrm{Prod}(G) \geq \log_2(2^n + 2) \geq n$, woraus $\mathrm{Prod}_{REG}(L_n) \geq n$ folgt. $\square$

Lemma 3.35 *Zu $n \geq 1$ sei $L_n = \{a^n\}$. Dann gelten*

$$\mathrm{Prod}_{REG}(L_n) = 1 \quad \textit{und} \quad \mathrm{Prod}_{nfREG}(L_n) \geq n.$$

Beweis: Die reguläre Grammatik $G = (\{S\}, \{a\}, \{S \to a^n\}, S)$ erzeugt L_n, und somit gilt $\mathrm{Prod}_{REG}(L_n) = 1$.

Andererseits folgt aus Lemma 3.13, dass zur Erzeugung von L_n durch reguläre Grammatiken in Normalform mindestens n Nichtterminale nötig sind. Da für jedes dieser Nichtterminale auch eine Regel erforderlich ist, werden mindestens n Regeln zum Erzeugen von L_n durch reguläre Grammatiken in Normalform benötigt. □

Aus den obigen Abschätzungen der notwendigen Anzahl von Regeln für gewisse Sprachen ergeben sich die folgenden Sätze.

Satz 3.36 Prod *ist bez. CF und REG vollständig.*

Beweis: Für *CF* folgt die Aussage sofort aus Lemma 3.31.

Sie gilt für *REG* ebenfalls. Zum einen ist die zum Erzeugen von L_n im Beweis von Lemma 3.31 verwendete Grammatik regulär, woraus $\mathrm{Prod}_{REG}(L_n) \leq n$ folgt. Zum anderen gilt aber sicher $n = \mathrm{Prod}_{CF}(L_n) \leq \mathrm{Prod}_{REG}(L_n)$. □

Satz 3.37 $MON \leq^3_{\mathrm{Prod}} CF \leq^3_{\mathrm{Prod}} REG \leq^3_{\mathrm{Prod}} nfREG$.

Beweis: $MON \leq^3_{\mathrm{Prod}} CF$ folgt aus Lemma 3.32. $CF \leq^3_{\mathrm{Prod}} REG$ folgt aus Lemma 3.34. $REG \leq^3_{\mathrm{Prod}} nfREG$ folgt aus Lemma 3.35. □

Wir haben bisher keine Aussagen zu Einschränkungen der kontextfreien Grammatiken gemacht. Aus Lemma 3.30 folgt sofort, dass

$$CF \approx_{\mathrm{Prod}} redCF$$

gilt. Dagegen ergibt sich für λ-freie Grammatiken die Relation

$$CF \leq^3_{\mathrm{Prod}} CF\text{-}\lambda,$$

auf deren Beweis wir verzichten. Man beachte, dass hinsichtlich der Maße Var und Prod das Verhältnis zwischen *CF* und *CF*-λ sehr unterschiedlich ist.

Lemma 3.38 $CF \leq^3_{\mathrm{Prod}} ChCF$ *und* $CF\text{-}\lambda \leq^3_{\mathrm{Prod}} ChCF$.

Beweis: Wir betrachten die Sprachen $L_n = \{ a^{2^n} \}$ für $n \geq 0$. Jede dieser Sprachen wird durch die folgende λ-freie Grammatik $G_n = (\{S\}, \{a\}, \{S \to a^{2^n}\}, S)$ erzeugt. Damit gilt $\mathrm{Prod}_{CF\text{-}\lambda}(L_n) = 1$. Da mindestens eine Regel nötig ist, gilt auch $\mathrm{Prod}_{CF}(L_n) = 1$. Mit Grammatiken in Chomsky-Normalform wird in jedem Ableitungsschritt die Satzform jedoch nur um höchstens ein Zeichen länger. Durch eine Regel erhält man eine Satzform mit höchstens zwei Buchstaben. Falls diese Regel noch einmal angewendet werden kann, entsteht eine Sprache mit unendlich vielen Wörtern. Folglich benötigen wir eine zweite Regel. Mit ihr erhalten wir eine Satzform mit höchstens vier Buchstaben. Auch diese Regel kann kein zweites Mal verwendet werden, da wir eine endliche Sprache erzeugen wollen. Folglich benötigt man mindestens n Regeln, um eine Satzform der Länge 2^n zu erzeugen und eine weitere Regel zum Terminieren (dabei ändert sich die Länge der Satzform nicht). Eine

kontextfreie Grammatik in Chomsky-Normalform mit genau $n+1$ Regeln zum Erzeugen der Sprache L_n ist $G'_n = (\{ S_0, S_1, \ldots, S_n \}, \{a\}, P', S_1)$ mit

$$P' = \{ S_i \to S_{i+1}S_{i+1} \,|\, 0 \leq i \leq n-1 \} \cup \{ S_n \to a \}.$$

Also gilt $\text{Prod}_{ChCF}(L_n) = n+1$. Daraus folgen die Behauptungen. □

Analog zum Lemma 3.16 kann man die folgenden Beziehungen zeigen.

Lemma 3.39 *ChCF* $\leq^2_{\text{Prod}}$ *REG und CF-*λ $\leq^2_{\text{Prod}}$ *REG.*

Wir kommen nun zu den Entscheidbarkeitsfragen bez. des Maßes Prod. Offensichtlich gilt der folgende Satz.

Satz 3.40 *Für jede Grammatik G ist* $\text{Prod}(G)$ *berechenbar.* □

Wir beweisen nun folgenden Satz.

Satz 3.41

i) *Es ist entscheidbar, ob für eine gegebene kontextfreie Grammatik G die Beziehung* $\text{Prod}_{CF}(L(G)) = 1$ *gilt.*

ii) *Es ist unentscheidbar, ob für eine gegebene kontextfreie Grammatik G und eine gegebene Zahl* $n \geq 2$ *die Beziehung* $\text{Prod}_{CF}(L(G)) = n$ *gilt.*

Beweis: i) Enthält G nur eine einzige Regel, so ist $L(G)$ entweder leer oder enthält genau ein Wort. Für eine kontextfreie Grammatik ist es entscheidbar, ob sie leer bzw. endlich ist. Ferner liefert im Fall der Endlichkeit die Konstante aus dem Pumping-Lemma, die aus G berechenbar ist, eine obere Schranke für die Wörter aus $L(G)$. Wir testen nun alle Wörter bis zu dieser Länge, ob sie von G erzeugt werden. Werden mindestens zwei Wörter erzeugt, so ist $\text{Prod}_{CF}(L(G)) \neq 1$.

ii) Es sei $n = 2$. Wir betrachten erneut die Sprachen $L_{X,Y}$ und L_s aus dem Beweis von Satz 3.22 zu gegebenen Mengen X und Y. Ferner sei die Abbildung $h : \{a,b,c\}^* \to \{a,b\}^*$ durch

$$h(\lambda) = \lambda,\ h(a) = ab,\ h(b) = aabb,\ h(c) = aaabbb,$$
$$h(x_1x_2) = h(x_1)h(x_2) \text{ für } x_1, x_2 \in \{a,b,c\}^*$$

gegeben. Weiterhin betrachten wir die Grammatik

$$G = (\{S\}, \{a,b\}, \{S \to SaSb,\ S \to \lambda\}, S)$$

und die von G erzeugte Sprache L und setzen

$$L' = L \setminus h(L_{X,Y} \cap L_s).$$

Wir zeigen

$$\text{Prod}_{CF}(L') \begin{cases} = 2, & \text{wenn das PKP bez. } X \text{ und } Y \text{ keine Lösung hat,} \\ \geq 3, & \text{wenn das PKP bez. } X \text{ und } Y \text{ eine Lösung hat.} \end{cases}$$

Hat das PKP bez. X und Y keine Lösung, so gilt $L' = L$ und damit nach Konstruktion $\mathrm{Prod}_{CF}(L') = 2$.

Hat das PKP eine Lösung $i_1 i_2 \dots i_k$, so liegt

$$w = ba^{i_1}b \dots ba^{i_k}cx_{i_k} \dots x_{i_1}cy_{i_1} \dots y_{i_k}ca^{i_k}b \dots a^{i_1}b$$

in $L_{X,Y} \cap L_s$. Ist $h(w)$ in L, so sind wegen der Ableitungen

$$\begin{aligned}
&S \Longrightarrow SaSb \Longrightarrow^* waSb \Longrightarrow wab,\\
&S \Longrightarrow SaSb \Longrightarrow^* waSb \Longrightarrow waSaSbb \Longrightarrow^* waabb,\\
&S \Longrightarrow SaSb \Longrightarrow^* waSb \Longrightarrow waSaSbb \Longrightarrow waSaSaSbbb \Longrightarrow^* waaabbb
\end{aligned}$$

in G auch $wab, waabb, waaabbb$ in L. Wegen $\lambda \in L$ folgt daher

$$\{ab, aabb, aaabbb\}^* \subset L.$$

Wegen $h(w) \in \{ab, aabb, aaabbb\}^*$ folgt damit $L' \subset L$. Andererseits enthält L' aber sicher alle Wörter aus L mit einer Länge ≤ 30, da die Wörter in $L_{X,Y} \cap L_s$ mindestens drei c, zwei a und zwei b enthalten. Angenommen, es gibt eine Grammatik $H = (N, T, P, S)$ zur Erzeugung von L' mit nur 2 Regeln. Wegen $\lambda \in L'$ muss es eine Regel $A \to \lambda$ geben. Ist $A \neq S$, so muss die zweite Regel $S \to A^k$ für ein $k \geq 1$ sein, da sonst λ nicht erzeugt werden kann. Dann gilt aber $L(H) = \{\lambda\}$ im Widerspruch zu $L(H) = L' \neq \{\lambda\}$. Folglich ist eine der beiden Regeln $S \to \lambda$. Da jede Anwendung der Regeln $S \to SaSb$ und $S \to \lambda$ aus G die gleiche Anzahl von as und bs produziert, muss dies auch für die Regeln aus P gelten. Ferner muss $ab \in L'$ herleitbar sein. Damit muss die zweite Regel in P die Form $S \to S^r a S^s b S^t$ haben.

Es seien zuerst $r \geq 1$, $s \geq 1$ und $t \geq 0$. Dann gibt es die Ableitung

$$S \Longrightarrow S^r a S^s b S^t \Longrightarrow^* SaS^s b S^t \Longrightarrow^* SaSb.$$

Verwenden wir stets diese Ableitung anstelle der Regel selbst, so folgt offenbar $L \subseteq L(H)$. Dies widerspricht wegen $L' \subset L$ der Forderung $L(H) = L'$.

Bei $r = t = 0$ und $s \geq 1$ gilt $abab \in L'$ aber $abab \notin L(H)$. In den Fällen $s = 0$ und $r = 0$, $t \geq 1$ oder $r \geq 1$, $t = 0$ oder $r \geq 1$, $t \geq 1$ ergibt sich $L(H) = \{ab\}^*$. In diesen Fällen erhalten wir wegen $aabb \in L'$ jeweils einen Widerspruch zu $L(H) = L'$.

Bei $r = s = t = 0$ ergibt sich $L(H) = \{ab, \lambda\}$. Dies widerspricht ebenfalls $L(H) = L'$.

Für den verbleibenden Fall $r = 0$, $s \geq 1$, $t \geq 1$ können wir wie oben zeigen, dass $\{ab, aabb, aaabbb\}^* \subseteq L(H)$ gilt. Wegen $h(w) \in \{ab, aabb, aaabbb\}^*$ ist damit $h(w) \in L(H)$, während $h(w) \notin L'$ gilt.

Da wir in allen Fällen einen Widerspruch hergeleitet haben, sind zum Erzeugen von L' mindestens drei Regeln nötig.

Es sei $n = 3$. In diesem Fall setzen wir

$$L'' = \{a, b\}^* \setminus h(L_{X,Y} \cap L_s)$$

und beweisen analog

$$\mathrm{Prod}_{CF}(L'') \begin{cases} = 3, & \text{wenn das PKP bez. } X \text{ und } Y \text{ keine Lösung hat,} \\ \geq 4, & \text{wenn das PKP bez. } X \text{ und } Y \text{ eine Lösung hat.} \end{cases}$$

Es sei nun $n \geq 4$. Wir setzen $m = n - 3$ und

$$U_m = \{d^{2^i} \mid 1 \leq i \leq m\}.$$

Wegen Lemma 3.31 ist eine bez. Prod minimale Grammatik für U_m dadurch gegeben, dass wir die Wörter aus U_m jeweils in einem Schritt direkt aus dem Axiom Z herleiten. Es ist leicht zu sehen, dass dann

$$\text{Prod}_{CF}(L' \cup U_m) = \text{Prod}_{CF}(L') + m + 1$$

gilt (zusätzlich zu den Wörtern aus U_m erzeugen wir aus Z noch das Axiom der minimalen Grammatik für L'). Damit gilt

$$\text{Prod}_{CF}(L' \cup U_m) \begin{cases} = n, & \text{wenn das PKP bez. } X \text{ und } Y \text{ keine Lösung hat,} \\ > n, & \text{wenn das PKP bez. } X \text{ und } Y \text{ eine Lösung hat.} \end{cases}$$

Damit haben wir nachgewiesen, dass es für jede natürliche Zahl $n \geq 2$ unentscheidbar ist, ob eine gegebene kontextfreie Grammatik G die Bedingung $\text{Prod}_{CF}(L(G)) = n$ erfüllt. □

Wie bei den Betrachtungen zur Anzahl der Nichtterminale lassen sich nun aus Satz 3.41 die folgenden Aussagen ableiten.

Folgerung 3.42

i) *Es gibt keinen Algorithmus, der zu einer beliebigen kontextfreien Grammatik G den Wert $\text{Prod}_{CF}(L(G))$ berechnet.*

ii) *Es gibt keinen Algorithmus, der zu einer beliebigen kontextfreien Grammatik G eine bez.* Prod *minimale kontextfreie Grammatik G' für $L(G)$ liefert.* □

Es ist noch offen, ob es entscheidbar ist, ob eine gegebene kontextfreie Grammatik G bez. Prod minimal für $L(G)$ ist.

In Analogie zu den Beweisen bezüglich des Maßes Var können wir nun zeigen, dass die angegebenen Probleme alle entscheidbar sind, wenn wir uns auf reguläre Grammatiken in Normalform beschränken.

Satz 3.43

i) *Es gibt einen Algorithmus, der zu einer gegebenen regulären Grammatik G in Normalform eine reguläre Grammatik G' in Normalform bestimmt, für die $L(G') = L(G)$ und $\text{Prod}(G') = \text{Prod}_{nfREG}(L(G))$ gelten.*

ii) *Es gibt einen Algorithmus, der zu gegebener regulärer Grammatik G in Normalform $\text{Prod}_{nfREG}(L(G))$ bestimmt.*

iii) *Es ist entscheidbar, ob eine gegebene Grammatik G in regulärer Normalform minimal für $L(G)$ ist, d. h. ob $\text{Prod}(G) = \text{Prod}_{nfREG}(L(G))$ gilt.* □

Erneut ist es offen, ob diese Probleme für reguläre Grammatiken entscheidbar oder unentscheidbar sind.

Übungsaufgaben

1. Man beweise Lemma 3.39 ausführlich.

2. Zu jeder natürlichen Zahl $n > 0$ sei $L_n = \{ w \mid w \in \{a, b, c\}^*,\ |w| = n \}$. Man beweise die Aussagen $\text{Prod}_{CF}(L_n) \leq 4$ und $\text{Prod}_{nfREG}(L_n) = 3n$.

3.4. Anzahl der Symbole

Wir untersuchen jetzt Grammatiken hinsichtlich des Komplexitätsmaßes Symb.

Lemma 3.44 *Für je zwei Mengen X und Y von Grammatiken mit $X \subseteq Y$ und jede Sprache $L \in \mathcal{L}(X)$ gilt* $\text{Symb}_Y(L) \leq \text{Symb}_X(L)$.

Beweis: Der Beweis wird analog zu dem von Lemma 3.4 geführt. □

Lemma 3.45 *Für jede natürliche Zahl n und jede Sprache L, die aus genau einem Wort der Länge n besteht, gilt* $\text{Symb}_{REG}(L) = n + 2$.

Beweis: Es gelte $L = \{w\}$ für ein Wort $w \in T^*$ der Länge n.

Da die Grammatik $G = (\{S\}, T, \{S \to w\}, S)$ nur das Wort w erzeugt, gilt offenbar $\text{Symb}_{REG}(L) \leq \text{Symb}(G) = n + 2$.

Es sei nun $H = (N, T, P, S)$ eine reguläre Grammatik, die minimal für L ist. Enthält sie die Regel $S \to w$, so gilt $\text{Symb}(H) \geq n + 2$.

Enthält P die Regel $S \to w$ nicht, so gibt es für w eine Ableitung

$$\begin{aligned} S = A_0 &\Longrightarrow w_1A_1 \Longrightarrow w_1w_2A_2 \Longrightarrow \cdots \Longrightarrow w_1w_2 \ldots w_{m-1}A_{m-1} \\ &\Longrightarrow w_1w_2 \ldots w_{m-1}w_m = w, \end{aligned}$$

wobei der Reihe nach die Regeln

$$S \to w_1A_1, \quad A_{i-1} \to w_iA_i \text{ für } 2 \leq i \leq m-1, \quad A_{m-1} \to w_m$$

angewendet werden. Falls alle A_i, $1 \leq i \leq m-1$, voneinander und von S verschieden sind, so ergibt sich

$$\text{Symb}_{REG}(L) = \text{Symb}(H) \geq |w_m| + 2 + \sum_{i=1}^{m-1}(|w_i| + 3) = |w| + 3m - 1 \geq n + 2.$$

Es sei $A_i = A_j$ mit $0 \leq i < j \leq m - 1$. Wir betrachten die Ableitung

$$A_i \Longrightarrow^* w_{i+1}w_{i+2} \ldots w_jA_j.$$

Gilt $w_{i+1}w_{i+2} \ldots w_j \neq \lambda$, so kann dieser Teil iteriert werden, und wir können ein Wort der Länge $k > n$ erzeugen, was $H(L) = \{w\}$ widerspricht. Ist dagegen, $w_{i+1}w_{i+2} \ldots w_j = \lambda$, so streichen wir in P die Regel $A_i \to A_{i+1}$ und erhalten P'. Offensichtlich gilt dann wegen der Existenz der Ableitung

$$\begin{aligned} S &\Longrightarrow w_1A_1 \Longrightarrow w_1w_2A_2 \Longrightarrow \cdots \Longrightarrow w_1w_2 \ldots w_iA_i = w_1w_2 \ldots w_iA_j \\ &\Longrightarrow^* w_1w_2 \ldots w_{m-1}w_m = w \end{aligned}$$

für die Grammatik $H' = (N, T, P', S)$ ebenfalls $L(H) = \{w\}$. Da H' drei Symbole weniger als H hat, ist H für L nicht minimal wie vorausgesetzt.

Damit ist auch $\text{Symb}(H) \geq n + 2$ gezeigt. □

Hieraus ergibt sich sofort das folgende Resultat.

Folgerung 3.46 Symb *ist bezüglich REG vollständig.* □

Ohne Beweis geben wir an, dass diese Aussage auch für kontextfreie Grammatiken richtig ist.

Satz 3.47 *Das Maß* Symb *ist bezüglich CF vollständig.* □

Satz 3.48 *Es gelten* $CF \leq^2_{\text{Symb}} REG$, $CF\text{-}\lambda \leq^2_{\text{Symb}} REG$ *und* $ChCF \leq^2_{\text{Symb}} REG$.

Beweis: Wir betrachten für $n \geq 1$ die Sprache $L_n = \{a^{2^n}\}$. Nach Lemma 3.45 ergibt sich $\text{Symb}_{REG}(L_n) = 2^n + 2$. Andererseits erzeugt die kontextfreie Grammatik

$$G_n = (\{A_0, A_1, \ldots, A_{n-1}\}, \{a\}, P_n, A_0)$$

mit

$$P_n = \{A_{n-1} \to a^2\} \cup \bigcup_{i=0}^{n-2} \{A_i \to A_{i+1}^2\}$$

die Sprache L_n. Damit gilt $\text{Symb}_{CF}(L_n) \leq \text{Symb}(G_n) = 4n$. Da die Grammatiken G_n auch λ-frei sind, gilt $\text{Symb}_{CF\text{-}\lambda}(L_n) \leq 4n$. Nimmt man bei einer Grammatik G_n noch ein neues Nichtterminal A_n hinzu und ersetzt die Regel $A_{n-1} \to a^2$ durch die beiden Regeln $A_{n-1} \to A_n^2$ und $A_n \to a$, so erhält man eine Grammatik G'_n in Chomsky-Normalform, die ebenfalls die Sprache L_n erzeugt. Damit gilt $\text{Symb}_{ChCF}(L_n) \leq \text{Symb}(G'_n) = 4n + 3$. Daraus folgen die Behauptungen. □

Zu jeder natürlichen Zahl k gibt es nur endlich viele Grammatiken G (bis auf Umbenennungen der Symbole), für die $\text{Symb}(G) \leq k$ gilt. Folglich gibt es in keiner Sprachfamilie X eine unendliche Folge von Sprachen L_n mit $\text{Symb}_X(L_n) \leq k$. Daher gilt für keine zwei Grammatikklassen X und Y die Beziehung $X \leq^3_{\text{Symb}} Y$.

Wir betrachten nun den Übergang von kontextfreien Grammatiken zu λ-freien kontextfreien Grammatiken.

Satz 3.49 *Für jede kontextfreie Sprache* L *gilt* $\text{Symb}_{CF\text{-}\lambda}(L) \leq 10 \cdot \text{Symb}_{CF}(L)$.

Beweis: Es sei $G = (N, T, P, S)$ eine kontextfreie Grammatik, die bez. Symb minimal für L ist. Wir setzen

$$E = \{A \mid A \in N,\ A \Longrightarrow^* \lambda\}.$$

Für jede Regel $p = A \to \alpha \in P$ mit $\alpha \neq \lambda$ definieren nun eine Menge $\varphi(p)$ von Regeln in der folgenden Weise:

- Enthält α kein Symbol aus E, so setzen wir $\varphi(p) = \{p\}$. Damit gilt

  $$\mathrm{Symb}(\varphi(A \to \alpha)) = \mathrm{Symb}(A \to \alpha),$$

  wobei sich hier Symb auf die Anzahl der in der oder den angegebenen Regeln vorkommenden Symbole bezieht.

- Ist $\alpha = E_1 E_2 \dots E_k \in E^+$, so besteht $\varphi(p)$ aus den Regeln

  $$\begin{aligned} A &\to E_1 \mid E_1 R_2 \mid R_2, \\ R_2 &\to E_2 \mid E_2 R_3 \mid R_3, \\ &\vdots \\ R_{k-2} &\to E_{k-2} \mid E_{k-2} R_{k-1} \mid R_{k-1}, \\ R_{k-1} &\to E_{k-1} \mid E_{k-1} E_k \mid E_k \text{ und} \\ S &\to \lambda, \text{ falls } A = S, \end{aligned}$$

  wobei $R_1, R_2, \dots, R_{k-1}$ neue Nichtterminale sind, die auch von den neuen Nichtterminalen anderer Regeln verschieden sind. Man rechnet leicht nach, dass

  $$\mathrm{Symb}(\varphi(A \to \alpha)) \le 10(k-1) + 2 \le 10(k+2) = 10 \cdot \mathrm{Symb}(A \to \alpha)$$

  gilt.

- Falls α sowohl Elemente aus E als auch welche aus $(N \cup T) \setminus E$ enthält, so stellen wir zuerst fest, dass es Wörter $u_i \in ((N \cup T) \setminus E)^*$, $0 \le i \le n$, und $\alpha_j \in E^+$, $1 \le j \le n$, mit $u_k \neq \lambda$ für $1 \le k \le n-1$ und $\alpha = u_0 \alpha_1 u_1 \alpha_2 u_2 \dots \alpha_n u_n$ gibt. Wir setzen nun

  $$\varphi(p) = \{A \to u_0 A_1 u_1 A_2 u_2 \dots A_n u_n\} \cup \bigcup_{i=1}^{n} \varphi(A_i \to \alpha_i),$$

  wobei erneut die A_i, $1 \le i \le n$, neue Nichtterminale sind und $\varphi(A_i \to \alpha_i)$, $1 \le i \le n$, wie im vorhergehenden Punkt definiert wird. Dann gilt

  $$\begin{aligned} \mathrm{Symb}(\varphi(p)) &\le n + 2 + \sum_{i=0}^{n} |u_i| + 10 \sum_{i=1}^{n} |\alpha_i| \\ &\le 10(2 + \sum_{i=0}^{n} |u_i| + \sum_{i=1}^{n} |\alpha_i|) \\ &= 10 \cdot \mathrm{Symb}(p). \end{aligned}$$

Es seien nun $P' = \bigcup_{p \in P} \varphi(p)$ und N' die Menge aller Nichtterminale, die in Regeln von P' vorkommen. Dann erfüllt die λ-freie Grammatik $G' = (N', T, P', S)$ die Bedingungen $L(G') = L(G)$ und $\mathrm{Symb}(G') \le 10 \cdot \mathrm{Symb}(G)$, woraus die Behauptung folgt. □

Aus dem Satz 3.49 ergibt sich auf Grund der Definition die folgende Aussage.

Folgerung 3.50 *CF* $\leq^1_{\text{Symb}}$ *CF-*λ.

Wir betrachten nun den Übergang von (λ-freien) kontextfreien Grammatiken zu kontextfreien Grammatiken in Chomsky-Normalform.

Satz 3.51 *CF* $\leq^2_{\text{Symb}}$ *ChCF und CF-*λ $\leq^2_{\text{Symb}}$ *ChCF.*

Beweis: Zu jeder natürlichen Zahl $n \geq 1$ seien $T_n = \{ a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n, c \}$ ein Alphabet und $L_n = \{ a_1^{p_1} a_2^{p_2} \ldots a_n^{p_n} c b_n^{p_n} b_{n-1}^{p_{n_1}} \ldots b_1^{p_1} \mid p_i \geq 0, 1 \leq i \leq n \}$ eine Sprache über dem Alphabet T_n. Die folgende Grammatik erzeugt die Sprache L_n:

$$
\begin{aligned}
G_n &= (\{ S_1, S_2, \ldots, S_n \}, T_n, P_n, S_1) \text{ mit} \\
P_n &= \{ S_i \to a_i S_i b_i \mid 1 \leq i \leq n \} \\
&\quad \cup \{ S_i \to S_{i+1} \mid 1 \leq i \leq n-1 \} \\
&\quad \cup \{ S_n \to c \}.
\end{aligned}
$$

Es gilt $\text{Symb}(G_n) = 8n$. Da die Grammatik G_n kontextfrei und λ-frei ist, gilt

$$\text{Symb}_{CF}(L_n) \leq 8n \text{ und } \text{Symb}_{CF\text{-}\lambda}(L_n) \leq 8n.$$

Es sei $G'_n = (N'_n, T_n, P'_n, S')$ eine kontextfreie Grammatik in Chomsky-Normalform, die die Sprache L_n erzeugt. Da die Sprache nicht endlich ist, gibt es eine Regel $S' \to A_1 B_1$ in P'_n und es gibt Ableitungen $A_1 \Longrightarrow^* a_1 u_1$ und $B_1 \Longrightarrow^* v_1 b_1$ mit $a_1 u_1 v_1 b_1 \in L_n$. Da es in der Sprache L_n auch Wörter ohne a_1 gibt, enthält P'_n eine Regel $S' \to A_2 B_2$, zu der Ableitungen $A_2 \Longrightarrow^* a_2 u_2$ und $B_2 \Longrightarrow^* v_2 b_2$ mit $a_2 u_2 v_2 b_2 \in L_n$ existieren. Es gilt $A_1 \neq A_2$, da es sonst die Ableitung $S' \to A_2 B_1 \Longrightarrow^* a_2 u_2 v_1 b_1$ gäbe, was aber $a_2 u_2 v_1 b_1 \notin L_n$ widerspricht. Aus dem gleichen Grund gilt $B_1 \neq B_2$. Analog existieren in P'_n auch die Regeln $S' \to A_i B_i$ für $i = 3, 4, \ldots, n$ mit $A_i \neq A_j$ und $B_i \neq B_j$ für $1 \leq i \leq n$, $1 \leq j \leq n$ und $i \neq j$.

Wegen $a_1 a_2 u_{12} v_{21} b_2 b_1 \in L_n$ für geeignete Wörter u_{12} und v_{21} gibt es in P'_n eine Regel $S' \to A'_1 B'_1$ mit $A'_1 B'_1 \Longrightarrow^* a_1 a_2 u_{12} v_{21} b_2 b_1$. Es gibt zwei Fälle:

1. Es existiert auch eine Ableitung $A'_1 B'_1 \Longrightarrow^* a_1 a_3 u_{13} v_{31} b_3 b_1$ mit $a_1 a_3 u_{13} v_{31} b_3 b_1 \in L_n$. In diesem Falle gilt

   (a) $A'_1 \Longrightarrow^* a_1$ sowie $B'_1 \Longrightarrow^* a_2 u_{12} v_{21} b_2 b_1$ und $B'_1 \Longrightarrow^* a_3 u_{13} v_{31} b_3 b_1$ oder

   (b) $A'_1 \Longrightarrow^* a_1 a_2 u_{12} v_{21} b_2$ und $A'_1 \Longrightarrow^* a_1 a_3 u_{13} v_{31} b_3$ sowie $B'_1 \Longrightarrow^* b_1$,

   da sonst für A'_1 und B'_1 jeweils mindestens zwei unterschiedlich terminierende Ableitungen existieren würden und $A'_1 B'_1$ damit zu einem Wort führen würde, das nicht in L_n liegt. Im Falle (a) muss während des Ableitens von B'_1, im Falle (b) von A'_1 sowohl eine Regel $C_{12} \to A_{12} B_{12}$ als auch eine Regel $C_{13} \to A_{13} B_{13}$ angewendet werden können, um beide Terminalwörter erhalten zu können.

2. Es gibt keine Ableitung $A'_1 B'_1 \Longrightarrow^* a_1 a_3 u_{13} v_{31} b_3 b_1$ mit $a_1 a_3 u_{13} v_{31} b_3 b_1 \in L_n$. In diesem Falle ist eine andere Regel $S' \to A'_{13} B'_{13}$ nötig, um das Wort $a_1 a_3 u_{13} v_{31} b_3 b_1$ abzuleiten.

Indem die Argumentation fortgesetzt wird, erhalten wir, dass mindestens für jedes Zahlenpaar (i,j) mit $1 \leq i < n$ und $i < j \leq n$ vier Symbole nötig sind. Folglich gilt

$$\text{Symb}_{ChCF}(L_n) \geq \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 4 = \sum_{i=1}^{n-1}(4(n-i)) = 4n(n-1) - 4\sum_{i=1}^{n-1} i = 2n^2 - 2n,$$

woraus die Beziehungen

$$\lim_{n\to\infty} \frac{\text{Symb}_{ChCF}(L_n)}{\text{Symb}_{CF}(L_n)} = \infty \quad \text{und} \quad \lim_{n\to\infty} \frac{\text{Symb}_{ChCF}(L_n)}{\text{Symb}_{CF\text{-}\lambda}(L_n)} = \infty$$

folgen. □

Der Beweis der folgenden Satzes kann ähnlich geführt werden. Wir überlassen ihn dem Leser als Übungsaufgabe.

Satz 3.52 *REG* $\leq^2_{\text{Symb}}$ *nfREG.*

Hinsichtlich der Entscheidbarkeitsfragen beginnen wir erneut mit dem trivialen Resultat, dass die Anzahl von Symbolen von Grammatiken berechenbar ist.

Satz 3.53 *Für eine beliebige kontextfreie Grammatik G ist* Symb(G) *berechenbar.* □

Für die Entscheidbarkeitsfragen bezüglich der kontextfreien Sprachen geben wir ohne Beweis den folgenden Satz an.

Satz 3.54

i) *Es gibt keinen Algorithmus, der zu einer beliebigen kontextfreien Grammatik G eine kontextfreie Grammatik G′ bestimmt, die für L(G) minimal bezüglich* Symb *ist.*

ii) *Es gibt keinen Algorithmus, der zu einer beliebigen kontextfreien Grammatik G den Wert* $\text{Symb}_{CF}(L(G))$ *bestimmt.*

iii) *Es ist unentscheidbar, ob für eine gegebene kontextfreie Grammatik G und eine gegebene natürliche Zahl* $n \geq 8$ *die Beziehung* $\text{Symb}_{CF}(L(G)) = n$ *gilt; bis* $n = 7$ *ist die Frage entscheidbar.*

iv) *Es ist unentscheidbar, ob eine gegebene kontextfreie Grammatik G minimal für L(G) ist, d. h. ob* $\text{Symb}(G) = \text{Symb}_{CF}(L(G))$ *gilt.* □

Wir merken an, dass die Entscheidbarkeit $\text{Symb}_{CF}(L(G)) = n$ für $n \leq 7$ – wie bei Prod – daraus resultiert, dass bei $n \leq 7$ nur sehr spezielle reguläre Mengen erzeugt werden können.

Für Symb erhalten wir bereits für reguläre Grammatiken die Entscheidbarkeit der Probleme. Bei Var und Prod ist die Entscheidbarkeit nur für reguläre Grammatiken in Normalform nachgewiesen, während für beliebige reguläre Grammatiken die Entscheidbarkeit offen ist.

Satz 3.55

i) *Es gibt einen Algorithmus, der zu einer gegebenen regulären Grammatik G eine reguläre Grammatik G' bestimmt, für die $L(G') = L(G)$ und $\text{Symb}(G') = \text{Symb}_{REG}(L(G))$ gelten.*

ii) *Es gibt einen Algorithmus, der zu gegebener regulärer Grammatik G $\text{Symb}_{REG}(L(G))$ bestimmt.*

iii) *Es ist entscheidbar, ob eine gegebene reguläre Grammatik G minimal für $L(G)$ ist, d. h. ob $\text{Symb}(G) = \text{Symb}_{REG}(L(G))$ gilt.*

Beweis: i) Es sei eine Grammatik $G = (N, T, P, S)$ gegeben. Wir berechnen zuerst den Wert $\text{Symb}(G)$. Dann konstruieren wir alle regulären Grammatiken H mit Nichtterminalen aus der Menge $\{A_1, A_2, \ldots, A_{\text{Symb}(G)}\}$ und der Terminalmenge T, die die Beziehung $\text{Symb}(H) \leq \text{Symb}(G)$ erfüllen. Dies sind offensichtlich nur endlich viele Grammatiken. Es sei M die Menge aller dieser regulären Grammatiken. Dann bestimmen wir

$$k = \min\{\text{Symb}(H) \mid H \in M,\ L(H) = L(G)\}.$$

Jede Grammatik G' aus M mit $L(G') = L(G)$ und $\text{Symb}(G') = k$ ist eine minimale Grammatik für $L(G)$.

ii) Der im Teil i) dieses Beweises erhaltene Wert k ist $\text{Symb}_{REG}(L(G))$.

iii) Die Aussage folgt aus i). □

Übungsaufgaben

1. Warum ist der Beweis von Satz 3.55 nicht auf kontextfreie Grammatiken übertragbar?

2. Man gebe eine minimale reguläre Grammatik zum Erzeugen der Sprache

   $$L = \{\, a^{2n} \mid n \geq 1 \,\} \cup \{\, b^{2n+1} \,\} n \geq 0$$

   an.

3. Ist die monotone Grammatik

   $$G = (\{S, B, B'\}, \{a, b, c\}, P, S)$$

   mit

   $$P = \{\, S \to abc,\ S \to aBc,\ aB \to aabB',\ bB \to Bb \,\}$$
   $$\cup \{\, B'b \to bB',\ B'c \to Bcc,\ B'c \to bcc \,\}$$

   für die Sprache $\{\, a^n b^n c^n \mid n \geq 1 \,\}$ minimal bezüglich der Anzahl der Symbole und der monotonen Grammatiken?

4. Man beweise Satz 3.52.

Kapitel 4

Beschreibungskomplexität endlicher Automaten

Die Ausführungen in diesem Kapitel stammen im Wesentlichen aus den Büchern [9] und [10] (englisch bzw. deutsch).

4.1. Algebraische Charakterisierungen von regulären Sprachen

Ein deterministischer *endlicher Automat* ist ein Quintupel

$$\mathcal{A} = (X, Z, z_0, \delta, F),$$

wobei X und Z Alphabete sind, z_0 ein Element von Z ist, δ eine Funktion von $Z \times X$ in Z ist, und F eine nichtleere Teilmenge von Z ist. Das Alphabet X ist die Menge der Eingabesymbole, Z ist die Menge der Zustände, z_0 ist der Anfangszustand, δ ist die Überführungsfunktion und F ist die Menge der akzeptierenden Zustände.

Im Unterschied dazu ist bei nichtdeterministischen endlichen Automaten die Überführungsfunktion als Abbildung in die Potenzmenge von Z erklärt (einem Paar $(z, x) \in Z \times X$ wird nicht ein einzelner Zustand zugeordnet sondern eine Menge von Zuständen). Im weiteren Verlauf werden wir stets deterministische endliche Automaten betrachten.

Durch

$$\begin{aligned} \delta^*(z, \lambda) &= z, \text{ für } z \in Z, \\ \delta^*(z, wa) &= \delta(\delta^*(z, w), a) \text{ für } w \in X^*,\ a \in X \end{aligned}$$

erweitern wir δ zu einer Funktion δ^* von $Z \times X^*$ in Z.

Die von $\mathcal{A}$ *akzeptierte* Sprache ist durch

$$T(\mathcal{A}) = \{w \in X^* \mid \delta^*(z_0, w) \in F\}$$

definiert.

Es ist bekannt, dass eine Sprache L genau dann regulär ist, wenn es einen endlichen Automaten $\mathcal{A}$ gibt, der L akzeptiert.

Wir wollen zuerst zwei algebraische Charakterisierungen regulärer Sprachen herleiten. Dazu benötigen wir die folgenden Definitionen.

Es seien X ein Alphabet und $R \subseteq X^*$. Eine Äquivalenzrelation $\sim$ auf der Menge X^* heißt *Rechtskongruenz*, falls für beliebige Wörter $x, y \in X^*$ und $a \in X$ aus $x \sim y$ auch $xa \sim ya$ folgt. Bei Multiplikation von rechts werden äquivalente Wörter wieder in äquivalente Wörter überführt.

Eine Äquivalenzrelation $\sim$ auf X^* heißt *Verfeinerung* der Menge R, wenn für beliebige Wörter $x, y \in X^*$ aus $x \sim y$ folgt, dass $x \in R$ genau dann gilt, wenn auch $y \in R$ gilt. Dies bedeutet, dass mit einem Wort $x \in R$ auch alle zu x äquivalenten Wörter in R liegen. Folglich liegt die zu $x \in R$ gehörige Äquivalenzklasse vollständig in R. Hieraus resultiert die Bezeichnung Verfeinerung für diese Eigenschaft.

Für eine Äquivalenzrelation $\sim$ bezeichnen wir die Anzahl der Äquivalenzklassen von $\sim$ als Index von $\sim$ und bezeichnen sie mit $Ind(\sim)$. Die Äquivalenzrelation $\sim$ hat *endlichen Index*, falls $Ind(\sim)$ endlich ist.

Wir nennen eine Äquivalenzrelation $\sim$ eine *R-Relation*, falls sie eine Rechtskongruenz mit endlichem Index ist und eine Verfeinerung von R ist.

Beispiel 4.1 *Es seien $\mathcal{A} = (X, Z, z_0, \delta, F)$ ein endlicher Automat und $R = T(\mathcal{A})$. Wir definieren auf X^* eine Relation $\sim_{\mathcal{A}}$ durch*

$$x \sim_{\mathcal{A}} y \quad \text{genau dann, wenn} \quad \delta^*(z_0, x) = \delta^*(z_0, y)$$

gilt. Offenbar ist $\sim_{\mathcal{A}}$ eine Äquivalenzrelation. Wir zeigen nun, dass $\sim_{\mathcal{A}}$ eine R-Relation ist.

Es sei $x \sim_{\mathcal{A}} y$. Dann gilt nach Definition $\delta^*(z_0, x) = \delta^*(z_0, y)$. Damit erhalten wir

$$\delta^*(z_0, xa) = \delta(\delta^*(z_0, x), a) = \delta(\delta^*(z_0, y), a) = \delta^*(z_0, ya)$$

und somit $xa \sim_{\mathcal{A}} ya$, womit nachgewiesen ist, dass $\sim_{\mathcal{A}}$ eine Rechtskongruenz ist.

Es sei erneut $x \sim_{\mathcal{A}} y$. Ferner sei $x \in R$. Dies bedeutet $\delta^*(z_0, x) = \delta^*(z_0, y)$ und $\delta^*(z_0, x) \in F$. Folglich gilt $\delta^*(z_0, y) \in F$, woraus $y \in R$ folgt. Analog folgt aus $y \in R$ auch $x \in R$. Damit ist $\sim_{\mathcal{A}}$ Verfeinerung von R.

Es sei $x \in X^*$. Ferner sei $\delta^*(z_0, x) = z$. Dann ergibt sich für die zu x bez. $\sim_{\mathcal{A}}$ gehörende Äquivalenzklasse $[x]_{\mathcal{A}}$

$$\begin{aligned}[x]_{\mathcal{A}} &= \{y \mid x \sim_{\mathcal{A}} y\} \\ &= \{y \mid \delta^*(z_0, x) = \delta^*(z_0, y)\} \\ &= \{y \mid \delta^*(z_0, y) = z\}.\end{aligned}$$

Es sei nun $z \in Z$ ein von z_0 erreichbarer Zustand. Dann gibt es ein Wort x mit $\delta^*(z_0, x) = z$ und die Beziehungen

$$\begin{aligned}\{y \mid \delta^*(z_0, y) = z\} &= \{y \mid \delta^*(z_0, y) = \delta^*(z_0, x)\} \\ &= \{y \mid y \sim_{\mathcal{A}} x\} \\ &= [x]_{\mathcal{A}}\end{aligned}$$

sind gültig. Daher gibt es eine eineindeutige Beziehung zwischen den Äquivalenzklassen von $\sim_{\mathcal{A}}$ und den Zuständen von $\mathcal{A}$, die vom Anfangszustand erreichbar sind. Dies bedeutet, dass die Anzahl der Zustände von $\mathcal{A}$ mit dem Index von $\sim_{\mathcal{A}}$ übereinstimmt. Wegen der Endlichkeit der Zustandsmenge Z ist also auch der Index von $\sim_{\mathcal{A}}$ endlich.

Beispiel 4.2 *Für eine Sprache $R \subseteq X^*$ definieren wir auf der Menge X^* die Relation $\sim_R$ wie folgt: $x \sim_R y$ gilt genau dann, wenn für alle $w \in X^*$ das Wort xw genau dann in R ist, falls dies auch für yw der Fall ist. Wir zeigen, dass $\sim_R$ eine Rechtskongruenz ist, die R verfeinert.*

Es seien dazu $x \sim_R y$, $a \in X$ und $w \in X^*$. Dann ist $aw \in X^*$ und nach Definition der Relation $\sim_R$ gilt $xaw \in R$ genau dann, wenn $yaw \in R$ gültig ist. Dies liefert, da w beliebig ist, die Aussage $xa \sim_R ya$. Somit ist $\sim_R$ eine Rechtskongruenz.

Wählen wir $w = \lambda$, so ergibt sich aus $x \sim_R y$ nach Definition, dass $x \in R$ genau dann gilt, wenn auch $y \in R$ erfüllt ist. Deshalb ist $\sim_R$ eine Verfeinerung von R.

Wir bemerken, dass $\sim_R$ nicht notwendigerweise einen endlichen Index haben muss. Dazu betrachten wir

$$R = \{a^n b^n \mid n \geq 0\}$$

und zwei Wörter a^k und a^ℓ mit $k \neq \ell$. Wegen $a^k b^k \in R$ und $a^\ell b^k \notin R$ sind offensichtlich a^ℓ und a^k nicht äquivalent. Somit gibt es mindestens soviel Äquivalenzklassen wie Potenzen von a und damit soviel wie natürliche Zahlen. Dies zeigt die Unendlichkeit des Index.

Ziel dieses Abschnitts ist eine Charakterisierung der regulären Sprachen durch Äquivalenzrelationen mit den oben definierten Eigenschaften.

Satz 4.3 *Für eine Sprache $R \subseteq X^*$ sind die folgenden Aussagen gleichwertig:*

i) *R ist regulär.*

ii) *Es gibt eine R-Relation.*

iii) *Die Relation $\sim_R$ (aus Beispiel 4.2) hat endlichen Index.*

Beweis: i) $\Longrightarrow$ ii). Zu einer regulären Sprache R gibt es einen endlichen Automaten $\mathcal{A}$ mit $T(\mathcal{A}) = R$. Dann können wir entsprechend Beispiel 4.1 die Relation $\sim_{\mathcal{A}}$ konstruieren. Diese ist nach Beispiel 4.1 eine R-Relation.

ii) $\Longrightarrow$ iii) Nach Voraussetzung gibt es eine R-Relation $\sim$ mit endlichem Index. Wir beweisen nun $Ind(\sim_R) \leq Ind(\sim)$.

Dazu zeigen wir zuerst, dass aus $x \sim y$ auch $xw \sim yw$ für alle $w \in X^*$ folgt. Für $w = \lambda$ ist dies klar. Für $w \in X$ ist es klar, da $\sim$ eine Rechtskongruenz ist. Es sei die Aussage nun schon für $|w| < n$ bewiesen. Wir betrachten das Wort v der Länge n. Dann gibt es ein w der Länge $n-1$ und ein $a \in X$ mit $v = wa$. Nach Induktionsvoraussetzung haben wir $xw \sim yw$. Nach der Definition der Rechtskongruenz folgt daraus $xwa \sim ywa$ und damit $xv \sim yv$.

Es sei nun $x \sim y$ und $w \in X^*$. Dann gilt auch $xw \sim yw$. Da $\sim$ eine R-Relation ist, folgt $xw \in R$ gilt genau dann, wenn $yw \in R$ gilt. Nach Definition $\sim_R$ bedeutet dies aber $x \sim_R y$. Damit ist gezeigt, dass $x \sim y$ auch $x \sim_R y$ impliziert. Hieraus ergibt sich aber

$$\{y \mid y \sim x\} \subseteq \{y \mid y \sim_R x\}.$$

Jede Äquivalenzklasse von $\sim$ ist also in einer Äquivalenzklasse von $\sim_R$ enthalten. Damit gilt $Ind(\sim_R) \leq Ind(\sim)$. Da $Ind(\sim)$ endlich ist, hat auch $\sim_R$ endlichen Index.

iii) $\Longrightarrow$ i) Mit $[x]_R$ bezeichnen wir die Äquivalenzklasse von $x \in X^*$ bez. $\sim_R$. Wir betrachten den endlichen Automaten

$$\mathcal{A} = (X, \{[x]_R \mid x \in X^*\}, [\lambda]_R, \delta, \{[y]_R \mid y \in R\})$$

mit

$$\delta([x]_R, a) = [xa]_R.$$

Wir bemerken zuerst, dass aufgrund der Voraussetzung, dass $\sim_R$ eine R-Relation ist, die Definition von $\mathcal{A}$ zulässig ist. (Die Endlichkeit der Zustandsmenge von $\mathcal{A}$ folgt aus der Endlichkeit des Index von $\sim_R$. Falls $[x]_R = [y]_R$, so ist auch $[xa]_R = [ya]_R$, denn $\sim_R$ ist eine Rechtskongruenz und daher gilt mit $x \sim_R y$, d. h. $[x]_R = [y]_R$, auch $xa \sim_R ya$.) Ferner beweist man mittels vollständiger Induktion über die Länge von x leicht, dass $\delta^*([\lambda]_R, x) = [x]_R$ für alle $x \in X^*$ gilt. Damit folgt

$$T(\mathcal{A}) = \{x \mid \delta^*([\lambda]_R, x) \in \{[y]_R \mid y \in R\}\} = \{x \mid [x]_R \in \{[y]_R \mid y \in R\}\} = R.$$

Damit ist R als regulär nachgewiesen. □

Im Beweisteil ii) $\Longrightarrow$ iii) wurde eigentlich die folgende Aussage bewiesen.

Folgerung 4.4 *Für jede R-Relation $\sim$ einer beliebigen regulären Sprache R gilt*

$$\mathit{Ind}(\sim) \geq \mathit{Ind}(\sim_R).$$ □

Übungsaufgaben

1. Man entscheide für jede der folgenden Relationen auf der Menge $\{a, b\}^*$, ob es sich um eine Rechtskongruenz handelt und begründe die Entscheidung:
   (a) $R_1 = \{ (aa, ba) \}$.
   (b) $R_2 = \{ (x^n, y^n) \mid n \geq 1,\ x \in \{a, b\},\ y \in \{a, b\} \}$.
   (c) $R_3 = \{ (w, w) \mid w \in \{a, b\}^* \}$.
   (d) $R_4 = \{ (aw, bw) \mid w \in \{a, b\}^* \}$.
   (e) $R_5 = \{ (a^n w, a^m w) \mid w \in \{a, b\}^*,\ n \geq 1,\ m \geq 1 \}$.
   (f) $R_6 = \{ (wa^n, wa^m) \mid w \in \{a, b\}^*,\ n \geq 1,\ m \geq 1 \}$.
   (g) $R_7 = \{ (wx^n, wx^m) \mid w \in \{a, b\}^*,\ x \in \{a, b\},\ n \geq 1,\ m \geq 1 \}$.
   (h) $R_8 = \emptyset$.
2. Man entscheide für jede der obigen Aquivalenzrelationen, ob es sich um eine Verfeinerung der folgenden Mengen handelt:
   (a) $S_1 = \{ w \mid w \in \{a, b\}^*,\ |w| = 20 \}$.
   (b) $S_2 = \{ x^n \mid x \in \{a, b\},\ n \geq 1 \}$.
   (c) $S_3 = \{a, b\}^* \setminus \{ x^n \mid x \in \{a, b\},\ n \geq 1 \}$.
   (d) $S_4 = \{ wba \mid w \in \{a, b\}^* \}$.
3. Welchen Index haben die obigen Äquivalenzrelationen?
4. Für welche natürlichen Zahlen i mit $1 \leq i \leq 4$ ist die Relation R_7 eine S_i-Relation?

4.2. Minimierung endlicher Automaten

In diesem Abschnitt diskutieren wir als Komplexitätsmaß die Anzahl der Zustände eines endlichen Automaten. Dazu setzen wir für einen endlichen Automaten $\mathcal{A} = (X, Z, z_0, \delta, F)$ und eine reguläre Sprache R

$$z(\mathcal{A}) = \#(Z),$$
$$z(R) = \min\{z(\mathcal{A}) \mid T(\mathcal{A}) = R\}.$$

Wir sagen, dass $\mathcal{A}$ *minimaler* Automat für R ist, falls $T(\mathcal{A}) = R$ und $z(\mathcal{A}) = z(R)$ gelten.

Als erstes geben wir ein Resultat an, dass die Größe $z(R)$ zu einer Sprache R bestimmt.

Satz 4.5 *Für jede reguläre Sprache R gilt $z(R) = Ind(\sim_R)$.*

Beweis: Wir bemerken zuerst, dass aus dem Teil iii) $\Longrightarrow$ i) des Beweises von Satz 4.3 folgt, dass es einen Automaten $\mathcal{A}$ mit $z(\mathcal{A}) = Ind(\sim_R)$ gibt.

Es sei nun $\mathcal{A} = (X, Z, z_0, \delta, F)$ ein beliebiger endlicher Automat mit $T(\mathcal{A}) = R$. Dann ist nach Beispiel 4.1 die Relation $\sim_{\mathcal{A}}$ eine R-Relation, für die $z(\mathcal{A}) = Ind(\sim_{\mathcal{A}})$ gilt. Wegen Folgerung 4.4 erhalten wir daraus sofort $z(\mathcal{A}) \geq Ind(\sim_R)$.

Die Kombination dieser beiden Erkenntnisse besagt gerade die Behauptung. □

Wir beschreiben nun für einen Automaten $\mathcal{A}$ einen Automaten, der minimal für $T(\mathcal{A})$ ist.

Es sei $\mathcal{A} = (X, Z, z_0, \delta, F)$ ein endlicher Automat. Für $z \in Z$ und $z' \in Z$ setzen wir $z \approx_{\mathcal{A}} z'$ genau dann, wenn für alle $x \in X^*$ genau dann $\delta^*(z, x)$ in F liegt, wenn auch $\delta^*(z', x)$ in F liegt. Falls der Automat $\mathcal{A}$ aus dem Kontext klar ist, schreiben wir einfach $\approx$ anstelle von $\approx_{\mathcal{A}}$

Lemma 4.6 *Die Relation $\approx_{\mathcal{A}}$ ist eine Äquivalenzrelation auf der Menge der Zustände des Automaten $\mathcal{A}$.*

Beweis: Wir verzichten auf den einfachen Standardbeweis. □

Mit $[z]_\approx$ bezeichnen wir die Äquivalenzklasse von $z \in Z$ bezüglich $\approx$ $(= \approx_{\mathcal{A}})$. Wegen $\lambda \in X^*$ impliziert $z \approx z'$ für zwei Zustände z und z', dass $z \in F$ genau dann gilt, wenn auch $z' \in F$ gilt. Daraus folgt $[z]_\approx \subseteq F$.

Wir setzen

$$Z_\approx = \{[z]_\approx \mid z \in Z\} \quad \text{und} \quad F_\approx = \{[z]_\approx \mid z \in F\}$$

und konstruieren den Automaten

$$\mathcal{A}_\approx = (X, Z_\approx, [z_0]_\approx, \delta_\approx, F_\approx)$$

mit

$$\delta_\approx([z]_\approx, a) = [\delta(z, a)]_\approx.$$

Wir zeigen, dass die Definition von $\mathcal{A}_\approx$ zulässig ist. Dazu haben wir nachzuweisen, dass die Definition von $\delta_\approx$ unabhängig von der Auswahl des Repräsentanten der Äquivalenzklasse ist. Es seien daher z und z' zwei Zustände mit $[z]_\approx = [z']_\approx$ und a ein beliebiges Element aus X. Dann muss $z \approx z'$ gelten. Wir betrachten ein beliebiges Wort $w \in X^*$. Dann liegt $\delta^*(z, aw)$ in F genau dann wenn $\delta^*(z', aw)$ in F liegt. Wegen $\delta^*(z, aw) = \delta^*(\delta(z, a), w)$ und $\delta^*(z', aw) = \delta^*(\delta(z', a), w)$ folgt $\delta(z, a) \approx \delta(z', a)$ und damit auch $[\delta(z, a)]_\approx = [\delta(z', a)]_\approx$, was zu zeigen war.

Wir zeigen nun, dass $\mathcal{A}_\approx$ minimal für die von $\mathcal{A}$ akzeptierte reguläre Sprache ist.

Satz 4.7 *Für jeden Automaten $\mathcal{A}$ ist $\mathcal{A}_\approx$ minimaler Automat für $T(\mathcal{A})$.*

Beweis: Wir zeigen zuerst mittels vollständiger Induktion über die Wortlänge, dass sich die Definition von $\delta_\approx : Z_\approx \times X \to Z_\approx$ auch auf die Erweiterung $\delta^*_\approx : Z_\approx \times X^* \to Z_\approx$ übertragen lässt, d. h. dass $\delta^*_\approx([z]_\approx, y) = [\delta^*(z, y)]_\approx$ für alle $y \in X^*$ gilt. Nach (genereller) Definition der Erweiterung haben wir

$$\begin{aligned}
&\delta^*_\approx([z]_\approx, \lambda) = [z]_\approx = [\delta^*(z, \lambda)]_\approx,\\
&\delta^*_\approx([z]_\approx, a) = \delta_\approx([z]_\approx, a) = [\delta(z, a)]_\approx = [\delta^*(z, a)]_\approx,
\end{aligned}$$

womit der Induktionsanfang gesichert ist. Der Induktionsschluss ist durch

$$\begin{aligned}
\delta^*_\approx([z]_\approx, ya) &= \delta_\approx(\delta^*_\approx([z]_\approx, y), a)\\
&= \delta_\approx([\delta^*(z, y)]_\approx, a)\\
&= [\delta(\delta^*(z, y), a)]_\approx\\
&= [\delta^*(z, ya)]_\approx
\end{aligned}$$

gegeben.

Hieraus erhalten wir sofort

$$\begin{aligned}
T(\mathcal{A}_\approx) &= \{x \mid \delta^*_\approx([z_0]_\approx, x) \in F_\approx\}\\
&= \{x \mid [\delta^*(z_0, x)]_\approx \in F_\approx\}\\
&= \{x \mid \delta^*(z_0, x) \in F\}\\
&= T(\mathcal{A}).
\end{aligned}$$

Damit bleibt nur noch $z(\mathcal{A}_\approx) = z(T(\mathcal{A}))$ zu zeigen. Wegen Satz 4.5 reicht es nachzuweisen, dass $z(\mathcal{A}_\approx) = \mathit{Ind}(\sim_{T(\mathcal{A})})$ erfüllt ist. Dazu betrachten wir die Relation $\sim_{\mathcal{A}_\approx}$ entsprechend Beispiel 4.1, für die $z(\mathcal{A}_\approx) = \mathit{Ind}(\sim_{\mathcal{A}_\approx})$ gilt, und beweisen $\sim_{\mathcal{A}_\approx} = \sim_{T(\mathcal{A})}$, womit der Beweis vollständig ist.

Es seien zuerst $x \in X^*$ und $y \in X^*$ zwei Wörter mit $x \not\sim_{\mathcal{A}_\approx} y$. Dann gilt

$$\delta^*_\approx([z_0]_\approx, x) \neq \delta^*_\approx([z_0]_\approx, y).$$

Hieraus erhalten wir $[\delta^*(z_0, x)]_\approx \neq [\delta^*(z_0, y)]_\approx$ und deshalb $\delta^*(z_0, x) \not\approx \delta^*(z_0, y)$. Nach der Definition von $\approx$ heißt dies, dass es ein Wort $w \in X^*$ gibt, für das entweder

$$\delta^*(\delta^*(z_0, x), w) \in F \quad \text{und} \quad \delta^*(\delta^*(z_0, y), w) \notin F$$

oder

$$\delta^*(\delta^*(z_0, x), w) \notin F \quad \text{und} \quad \delta^*(\delta^*(z_0, y), w) \in F$$

erfüllt ist. Daher gilt entweder $\delta^*(z_0, xw) \in F$ und $\delta^*(z_0, yw) \notin F$ oder $\delta^*(z_0, xw) \notin F$ und $\delta^*(z_0, yw) \in F$ und folglich entweder $xw \in T(\mathcal{A})$ und $yw \notin T(\mathcal{A})$ oder $xw \notin T(\mathcal{A})$ und $yw \in T(\mathcal{A})$. Letzteres bedeutet aber gerade $x \not\sim_{T(\mathcal{A})} y$.

Analog beweist man, dass $x \sim_{\mathcal{A}_\approx} y$ für zwei Wörter $x \in X^*$ und $y \in X^*$ auch $x \sim_{T(\mathcal{A})} y$ zur Folge hat.

Somit stimmen die Äquivalenzrelationen $\sim_{\mathcal{A}_\approx}$ und $\sim_{T(\mathcal{A})}$ überein, was zu zeigen war.□

Leider geben die bisherigen Betrachtungen keine Hinweis, wie der Index von $\sim_R$ zu gegebenem R zu ermitteln ist, denn nach Definition von $\sim_R$ sind unendlich viele Wörter zu untersuchen, um $x \sim_R y$ festzustellen. Analog ist die Konstruktion von $\mathcal{A}_\approx$ nicht algorithmisch, denn auch die Feststellung, ob $x \approx y$ gilt, erfordert die Betrachtung von unendlich vielen Wörtern.

Deshalb beschreiben wir jetzt einen Algorithmus zum Bestimmen der Relation $\approx = \approx_{\mathcal{A}}$ für einen endlichen Automaten $\mathcal{A}$. Dies versetzt uns dann in die Lage, zu $\mathcal{A}$ den minimalen Automaten $\mathcal{A}_\approx$ zu konstruieren. Dazu beginnen wir mit einer Liste aller Paare von Zuständen und markieren jeweils ein Paar, wenn sich aus dem Verhalten der beiden Zustände des Paares entsprechend der Überführungsfunktion oder der Menge der akzeptierenden Zustände und der aktuellen Liste ergibt, dass die beiden Zustände des Paares nicht äquivalent sind.

Der *Reduktionsalgorithmus* besteht aus den folgenden Schritten:

1. Erstelle eine Liste aller Paare (z, z') von Zuständen $z \in Z$ und $z' \in Z$.

2. Streiche ein Paar (z, z') falls entweder $z \in F$ und $z' \notin F$ oder $z \notin F$ und $z' \in F$ gilt.

3. Führe die folgende Aktion solange aus, bis die Liste nicht mehr geändert wird: Falls für ein Paar (z, z') und ein $a \in X$ das Paar $(\delta(z, a), \delta(z', a))$ nicht in der Liste ist, so streiche (z, z').

Beispiel 4.8 *Wir betrachten den endlichen Automaten*

$$\mathcal{A} = (\{a, b\}, \{0, 1, 2, 3, 4, 5\}, 0, \delta, \{1, 2, 5\}),$$

dessen Überführungsfunktion δ dem Zustandsgraphen aus Abbildung 4.1 zu entnehmen ist.
Wir erhalten entsprechend Schritt 1 des Algorithmus zuerst die Liste

$$\begin{array}{llllll}
(0,0), & (0,1), & (0,2), & (0,3), & (0,4), & (0,5), \\
(1,0), & (1,1), & (1,2), & (1,3), & (1,4), & (1,5), \\
(2,0), & (2,1), & (2,2), & (2,3), & (2,4), & (2,5), \\
(3,0), & (3,1), & (3,2), & (3,3), & (3,4), & (3,5), \\
(4,0), & (4,1), & (4,2), & (4,3), & (4,4), & (4,5), \\
(5,0), & (5,1), & (5,2), & (5,3), & (5,4), & (5,5).
\end{array}$$

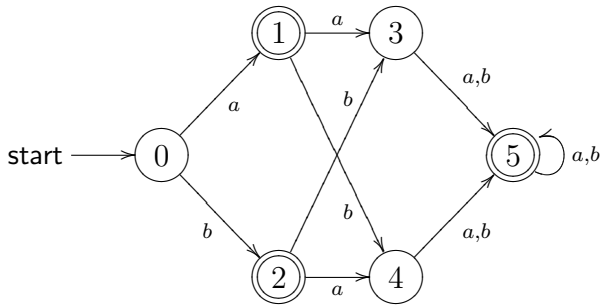


Abbildung 4.1: Zustandsgraph des Automaten $\mathcal{A}$ aus Beispiel 4.8

Hieraus entsteht nach Schritt 2 die Liste

$$(0,0),\ (0,3),\ (0,4),\ (1,1),\ (1,2),\ (1,5),$$
$$(2,1),\ (2,2),\ (2,5),\ (3,0),\ (3,3),\ (3,4),$$
$$(4,0),\ (4,3),\ (4,4),\ (5,1),\ (5,2),\ (5,5).$$

Wir haben nun solange wie möglich Schritt 3 anzuwenden. Hierzu gehen wir stets die Liste durch und streichen die entsprechenden Paare und wiederholen diesen Prozess. Nach dem ersten Durchlauf streichen wir nur die Paare $(1,5)$ *(da* $(\delta(1,a),\delta(5,a)) = (3,5)$ *nicht mehr in der Liste ist),* $(2,5)$, $(5,1)$ *und* $(5,2)$. *Beim zweiten Durchlauf werden die Paare* $(0,3)$ *(da das Paar* $(\delta(0,a),\delta(3,a)) = (1,5)$ *beim ersten Durchlauf gerade gestrichen wurde),* $(0,4)$, $(3,0)$ *und* $(4,0)$ *gestrichen. Beim dritten Durchlauf wird nichts mehr gestrichen. Damit ergibt sich abschließend die Liste*

$$(0,0),\ (1,1),\ (1,2),\ (2,1),\ (2,2),\ (3,3),\ (3,4),\ (4,3),\ (4,4),\ (5,5).$$

Wir zeigen nun, dass mittels des oben angegebenen Algorithmus wirklich die Relation $\approx_{\mathcal{A}}$ berechnet wird. Dies ist im Wesentlichen die Aussage des folgenden Lemmas.

Lemma 4.9 *Es seien* $\mathcal{A} = (X, Z, z_0, \delta, F)$ *ein endlicher Automat sowie* $z \in Z$ *und* $z' \in Z$ *zwei Zustände des Automaten. Dann ist das Paar* (z, z') *genau dann in der durch den Reduktionsalgorithmus erzeugten Liste enthalten, wenn* $z \approx_{\mathcal{A}} z'$ *gilt.*

Beweis: Wir beweisen mittels vollständiger Induktion über die Anzahl der Schritte des Reduktionsalgorithmus die folgende äquivalente Aussage: Das Paar (z, z') wird genau dann durch den Reduktionsalgorithmus aus der Liste gestrichen, wenn es ein Wort $x \in X^*$ gibt, für das entweder $\delta^*(z,x) \in F$ und $\delta^*(z',x) \notin F$ oder $\delta^*(z,x) \notin F$ und $\delta^*(z',x) \in F$ gelten. Im Folgenden nehmen wir stets ohne Beschränkung der Allgemeinheit an, dass der erste dieser beiden Fälle eintritt.

Erfolgt ein Streichen des Paares (z, z') im zweiten Schritt, so hat wegen $\delta^*(z,\lambda) = z$ und $\delta^*(z',\lambda) = z'$ das leere Wort die gewünschte Eigenschaft. Hat umgekehrt das Leerwort die Eigenschaft, so wird das Paar im zweiten Schritt gestrichen.

Das Paar (z, z') werde nun im dritten Schritt gestrichen. Dann gibt es ein Element $a \in X$ so, dass das Paar $(\delta(z,a),\delta(z',a))$ bereits früher gestrichen wurde. Nach Induktionsannahme gibt es ein Wort $x \in X^*$ mit $\delta^*(\delta(z,a),x) \in F$ und $\delta^*(\delta(z',a),x) \notin F$. Damit haben wir auch $\delta^*(z,ax) \in F$ und $\delta^*(z',ax) \notin F$, womit die Behauptung bewiesen

ist. Durch Umkehrung der Schlüsse zeigt man, dass aus der Existenz eines Wortes ax mit $\delta^*(z, ax) \in F$ und $\delta^*(z', ax) \notin F$ folgt, dass (z, z') gestrichen wird. □

Beispiel 4.8 (Fortsetzung) *Wir erhalten aus der bereits oben erzeugten Liste die folgenden Äquivalenzklassen bez.* $\approx$*:*

$$[0]_\approx = \{0\},\ [1]_\approx = [2]_\approx = \{1, 2\},\ [3]_\approx = [4]_\approx = \{3, 4\},\ [5]_\approx = \{5\}.$$

Hieraus resultiert nun entsprechend obiger Konstruktion der minimale Automat

$$\mathcal{A}_\approx = (X, \{[0]_\approx, [1]_\approx, [3]_\approx, [5]_\approx\}, [0]_\approx, \delta_\approx, \{[1]_\approx, [5]_\approx\}),$$

dessen Überführungsfunktion aus Abbildung 4.2, in der wir $[i]$ *anstelle von* $[i]_\approx$, $0 \le i \le 5$, *schreiben, entnommen werden kann.*

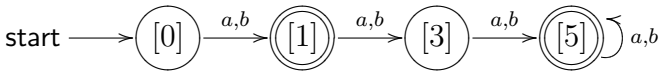


Abbildung 4.2: Zustandsgraph des minimalen Automaten zu $\mathcal{A}$ aus Beispiel 4.8

Wir bemerken, dass aus der Darstellung des minimalen Automaten sofort

$$T(\mathcal{A}) = T(\mathcal{A}_\approx) = X \cup X^3X^*$$

zu sehen ist.

Wir untersuchen nun die Komplexität des Reduktionsalgorithmus. Es gibt n^2 Paare von Zuständen. Bei Schritt 2 oder jedem Durchlauf entsprechend Schritt 3 streichen wir ein Paar oder stoppen. Das Durchmustern einer Liste der Länge r erfordert $O(r)$ Schritte. Damit ist durch

$$\sum_{i=1}^{n^2} O(i) = O\Big(\sum_{i=1}^{n^2} i\Big) = O(n^4)$$

eine obere Schranke für die Komplexität gegeben.

Wir merken hier ohne Beweis an, dass es einen erheblich effizienteren Algorithmus zum Minimieren von endlichen Automaten gibt.

Satz 4.10 *Die Konstruktion des minimalen Automaten zu einem gegebenem endlichen Automaten mit n Zuständen ist in* $O(n \cdot \log(n))$ *Schritten möglich.* □

Wir wollen nun beweisen, dass der minimale Automat im Wesentlichen eindeutig bestimmt ist. Um das „im Wesentlichen“ exakt zu fassen, geben wir die folgende Definition.

Definition 4.11 *Zwei Automaten* $\mathcal{A} = (X, Z, z_0, \delta, F)$ *und* $\mathcal{A}' = (X, Z', z_0', \delta', F')$ *heißen zu einander* isomorph*, wenn es eine eineindeutige Funktion* φ *von* Z *auf* Z' *mit folgenden Eigenschaften gibt:*

i) $\varphi(z_0) = z_0'$.

ii) *Für $z \in Z$ gilt $z \in F$ genau dann, wenn auch $\varphi(z) \in F'$ gilt.*

iii) *Für alle $z \in Z$ und $a \in X$ gilt $\delta'(\varphi(z), a) = \varphi(\delta(z, a))$.*

Intuitiv liegt Isomorphie zwischen zwei Automaten vor, wenn diese sich nur in der Bezeichnung der Zustände unterscheiden (und ansonsten das gleiche Verhalten zeigen).

Satz 4.12 *Sind $\mathcal{A}$ und $\mathcal{A}'$ zwei minimale Automaten für eine reguläre Menge R, so sind $\mathcal{A}$ und $\mathcal{A}'$ zu einander isomorph.*

Beweis: Es seien $\mathcal{A} = (X, Z, z_0, \delta, F)$ und $\mathcal{A}' = (X, Z', z_0', \delta', F')$ zwei minimale Automaten für R.

Für zwei Zustände $z \in Z$ und $z' \in Z$ von $\mathcal{A}$ ist $z \not\approx_{\mathcal{A}} z'$. Dies folgt daraus, dass aufgrund der Minimalität von $\mathcal{A}$ der Automat $\mathcal{A}_{\approx}$ genauso viel Zustände wie $\mathcal{A}$ hat. Also können keine zwei Zustände von $\mathcal{A}$ in einer Äquivalenzklasse bez. $\approx_{\mathcal{A}}$ liegen. Nun beweist man analog zum letzten Teil des Beweises von Satz 4.7, dass die Relationen $\sim_{\mathcal{A}}$ und $\sim_R$ übereinstimmen.

Entsprechend ergibt sich auch $\sim_{\mathcal{A}'} = \sim_R$ und damit $\sim_{\mathcal{A}} = \sim_{\mathcal{A}'}$.

Es sei nun $z \in Z$. Dann gibt es ein Wort $x \in X^*$ mit $\delta^*(z_0, x) = z$. Wir setzen

$$\varphi(z) = (\delta')^*(z_0', x).$$

Dies liefert eine zulässige Definition von φ, denn der Wert von $\varphi(z)$ hängt nicht von der Wahl von x ab. Sind nämlich $x \in X^*$ und $y \in X^*$ mit $\delta^*(z_0, x) = \delta^*(z_0, y)$, so ergibt sich zuerst $x \sim_{\mathcal{A}} y$ und damit auch $x \sim_{\mathcal{A}'} y$, woraus $(\delta')^*(z_0', x) = (\delta')^*(z_0', y)$ folgt.

Wir zeigen, dass φ ein Isomorphismus ist.
Wegen $z_0 = \delta^*(z_0, \lambda)$ ergibt sich $\varphi(z_0) = (\delta')^*(z_0', \lambda) = z_0'$.
Es sei $z \in F$. Dann gibt es ein Wort $x \in R$ mit $z = \delta^*(z_0, x)$ und es gilt $\varphi(z) = (\delta')^*(z_0', x)$. Wegen $x \in R$ und $R = T(\mathcal{A}')$ erhalten wir $\varphi(z) \in F'$. Es sei umgekehrt $\varphi(z) \in F'$. Dann liegt x mit $\varphi(z) = (\delta')^*(z_0', x)$ in R und somit ist $z = \delta(z_0, x) \in F$.

Außerdem gilt

$$\begin{aligned}\delta'(\varphi(z), a) &= \delta'((\delta')^*(z_0', x), a) = (\delta')^*(z_0', xa) \\ &= \varphi(\delta^*(z_0, xa)) = \varphi(\delta(\delta^*(z_0, x), a)) \\ &= \varphi(\delta(z, a)).\end{aligned}$$

□

Minimale deterministische endliche Automaten sind also bis auf „Umbenennen“ der Zustände eindeutig bestimmt. Dies gestattet uns, von *dem minimalen Automaten* für eine Sprache zu sprechen.

Abschließend merken wir an, dass bei nichtdeterministischen endlichen Automaten ein minimaler Automat nicht eindeutig bestimmt ist.

Übungsaufgaben

1. Es sei $R = \{aab, ab\}$. Man zeige, dass die Wörter folgender Paare nicht zur gleichen Äquivalenzklasse bezüglich der Relation $\sim_R$ gehören:

(a) a und b.

(b) a und aa.

(c) aa und b.

(d) λ und a.

Geben Sie einen deterministischen endlichen Automaten für die Sprache R an.

2. Gegeben sei der deterministische endliche Automat

$$\mathcal{A} = (\{a, b\}, \{z_0, z_1, z_2, z_3, z_4, z_5\}, z_0, \{z_0, z_2\}, \delta)$$

mit der Überführungsfunktion δ entsprechend der folgenden Tabelle:

δ	z_0	z_1	z_2	z_3	z_4	z_5
a	z_0	z_3	z_0	z_3	z_5	z_5
b	z_1	z_2	z_3	z_0	z_5	z_4

Konstruieren Sie einen zu $\mathcal{A}$ äaquivalenten minimalen deterministischen endlichen Automaten.

3. Gegeben sei der deterministische endliche Automat

$$\mathcal{A} = (\{a, b\}, \{z_0, z_1, z_2, z_3, z_4, z_5\}, z_0, \{z_0, z_2\}, \delta)$$

mit der Überführungsfunktion δ entsprechend der folgenden Tabelle:

δ	z_0	z_1	z_2	z_3	z_4	z_5
a	z_1	z_4	z_1	z_0	z_4	z_2
b	z_3	z_2	z_5	z_4	z_4	z_4

Konstruieren Sie einen zu $\mathcal{A}$ äaquivalenten minimalen deterministischen endlichen Automaten.

Literaturverzeichnis

[1] J. Dassow, Theoretische Informatik. Vorlesungsskript, 2000.

[2] W. Dörfler, *Mathematik für Informatiker*. Hanser-Verlag München, 1977.

[3] M. R. Garey und D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[4] S. B. Gaskov, The depth of Boolean functions. *Probl. Kibernetiki* **34** (1978) 265–268.

[5] J. Gruska, On a classification of context-free languages. ßem Kybernetika **3** (1967) 22–29.

[6] J. Gruska, Some classifications of context-free languages. *Information and Control* **14** (1969) 152–179.

[7] J. Gruska, Complexity and unambiguity of context-free grammars and languages. *Information and Control* **18** (1971) 502–519.

[8] J. Gruska, Descriptional complexity (of languages) – a short survey. In: *Mathematical Foundations of Computer Science 1976* (Hrsg. A. Mazurkiewicz), Lecture Notes in Computer Science 45, Springer-Verlag, 1976, 65–80.

[9] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Adddison Wesley, Reading, Massachusetts, 1979.

[10] J. E. Hopcroft and J. D. Ullman, *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. Addison Wesley, Reading, Bonn, 1990.

[11] S. W. Jablonski, Einführung in die Theorie der Funktionen der k-wertigen Logik. In: *Diskrete Mathematik und mathematische Fragen der Kybernetik* (Hrsg. S. W. Jablonski und O. B. Lupanov), Akademie-Verlag, 1980.

[12] D. C. Kozen, *Automata and Computability*. Springer-Verlag, 1997.

[13] O. B. Lupanov, Complexity of formulae realisation of functions of logical algebra. *Probl. Kibernetiki* **3** (1962) 782–811.

[14] O. B. Lupanov, A method of synthesis of control systems. *Probl. Kibernetiki* **23** (1970) 31–110.

[15] W. F. McColl und M. Paterson, The depth of all Boolean functions. *SIAM J. Comp.* **6** (1977) 373–380.

[16] T. Ottmann und P. Widmayer, *Algorithmen und Datenstrukturen.* B.I.-Wissenschaftsverlag, Mannheim, 1993.

[17] B. Reichel, Some classifications of Indian parallel languages. *J. Inf. Process. Cybern.* (EIK) **26** (1990) 85–99.

[18] U. Schöning, *Theoretische Informatik kurz gefaßt.* B.I.-Wissenschaftsverlag, 1992.

[19] R. Sedgewick, *Algorithmen.* Addison-Wesley, 1990.

[20] K. Wagner, *Einführung in die Theoretische Informatik.* Springer-Verlag, 1994.

[21] I. Wegener, *The Complexity of Boolean Functions.* Teubner, Stuttgart und Wiley, New York, 1987.

[22] I. Wegener, *Theoretische Informatik.* Teubner-Verlag, 1993.