**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 zur Lösung verwendeten Algorithmus ist. Ist die Beschreibung durch ein Programm gegeben, so kann als Größe des Programms die Anzahl der Befehle im Programm (Anzahl der "line of codes") angesehen werden; wird der Algorithmus durch eine TURING-Maschine beschrieben, zu 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 der 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], [15], [17], [19], [3], [13], [16].

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 und Herrn Ivo Rössling für die sorgfältige Durchsicht älterer Versionen des Skriptes, wodurch sich die Qualität deutlich verbessert hat.

Jürgen Dassow / Bianca Truthe April – Juli 2008

# Inhaltsverzeichnis

**Vorwort** **1**

**1 Komplexität Boolescher Funktionen** **5**

1.1 Boolesche Funktionen . . . . . 5
1.2 Schaltkreise und Komplexitätsmaße . . . . . 10
1.3 Das Vollständigkeitskriterium von Post . . . . . 15
1.4 Beziehungen zwischen den Maßen . . . . . 24
1.5 Asymptotische Komplexität – untere und obere Schranken . . . . . 30
1.6 Minimierung von Schaltkreisen . . . . . 42
1.7 Verzweigungsprogramme und ihre Komplexität . . . . . 52
1.8 Schaltkreise versus Turing-Maschinen . . . . . 56

**Literaturverzeichnis** **65**

# 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 Methode zur Darstellung 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

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

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. Abbildung 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

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

als abkürzende Bezeichnungen ein.

| | |
|---|---|
| 0 0 0 ... 0 0 0 | $f(0,0,0\ldots0,0,0)$ |
| 0 0 0 ... 0 0 1 | $f(0,0,0\ldots0,0,1)$ |
| 0 0 0 ... 0 1 0 | $f(0,0,0\ldots0,1,0)$ |
| 0 0 0 ... 0 1 1 | $f(0,0,0\ldots0,1,1)$ |
| 0 0 0 ... 1 0 0 | $f(0,0,0\ldots1,0,0)$ |
| ... | ... |
| 0 1 1 ... 1 1 1 | $f(0,1,1\ldots1,1,1)$ |
| 1 0 0 ... 0 0 0 | $f(1,0,0\ldots0,0,0)$ |
| ... | ... |
| 1 1 1 ... 1 1 0 | $f(1,1,1\ldots1,1,0)$ |
| 1 1 1 ... 1 1 1 | $f(1,1,1\ldots1,1,1)$ |

Abbildung 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 $2^{2^n}$. □

Abbildung 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 |

Abbildung 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.

Abbildung 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 und diese wird gerade durch den Wert von $x_1 \wedge x_2$ angegeben. Daher verwenden wir für die Konjunktion auch die Bezeichnung $x_1 \cdot x_2$.

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

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

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

Abbildung 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.

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

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

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

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

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

und

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

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

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

und

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

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

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

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

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

*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

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

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

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

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) bzw. b) sind spezielle *disjunktive* bzw. *konjunktive Normalformen* von $f$. 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.

**Beispiel 1.1** Wir betrachten die durch die Tabelle aus Abbildung 1.4 gegebene dreistellige Boolesche Funktion $f$.

| $x_1$ | $x_2$ | $x_3$ | $f(x_1, x_2, x_3)$ |
|---|---|---|---|
| 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 |

Abbildung 1.4:

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 (1.2), (1.3), (1.4) 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 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 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 1.1** *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 $k$ 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 1.2** *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 1.2** Wir betrachten die Menge

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

und die Graphen $S_1$ und $S_2$ aus Abbildung 1.5.

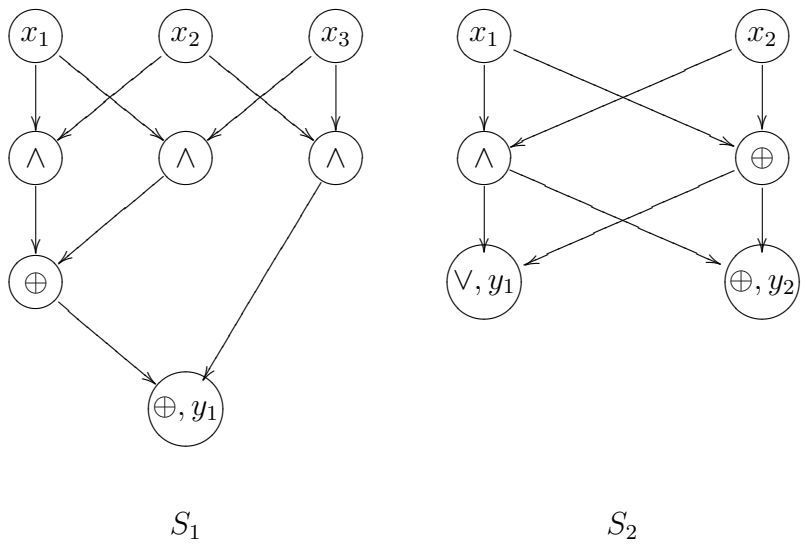



Abbildung 1.5: Beispiele für Schaltkreise

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

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

und

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

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 1.3** *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 1.3** Wir setzen Beispiel 1.2 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:

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

und 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

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

und in den Knoten der Tiefe 2 die Funktionen

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

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 1.4** 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.

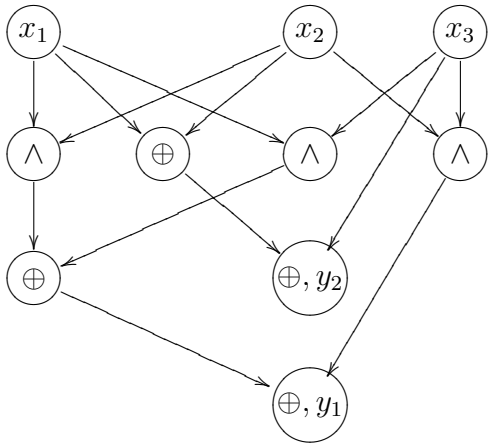



Abbildung 1.6: Schaltkreis $S_3$ zur Addition von drei Dualziffern

Somit gelten

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

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 1.6 der Größe 7 und der Tiefe 3 das Gewünschte leistet.

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

**Definition 1.4** *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 1.5** Wir betrachten die Funktionen $f_1$ und $f_2$, die durch die Schaltkreise aus Beispiel 1.2 über $\{\vee, \wedge, \oplus\}$ berechnet werden. Wegen der Minimierung bei der Komplexitätsbestimmung erhalten wir direkt

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

sowie

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

Aufgrund von (1.6) berechnet aber der Schaltkreis $S_4$ aus Abbildung 1.7 ebenfalls die Funktion $f_2$. Damit gilt

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

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

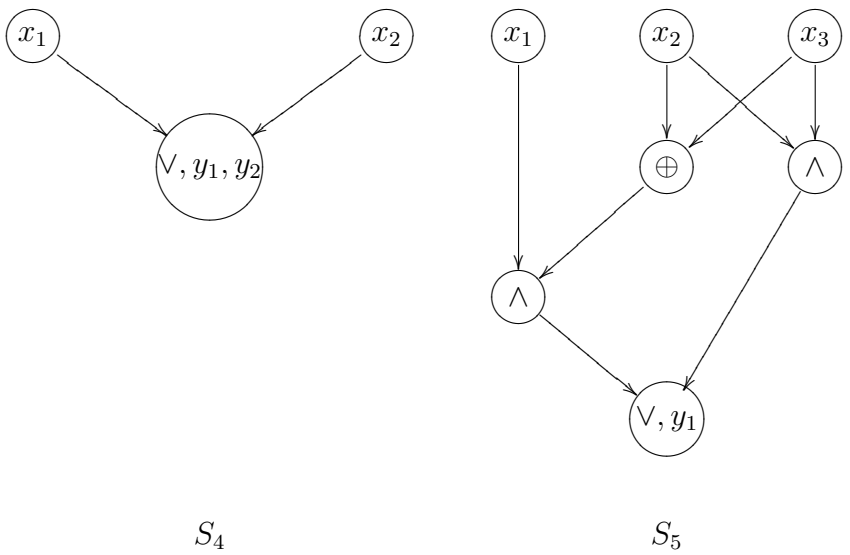



Abbildung 1.7: Schaltkreise $S_4$ und $S_5$

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

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

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

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

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

*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.*

[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

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 bzw. Ausdrücke 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 mittels 1. und 2.

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. Die Menge $\mathcal{S}$ bestehe aus den Funktion Negation, Konjunktion, Disjunktion und Parity-Funktion. Der Schaltkreis $S$ über $\mathcal{S}$ sei durch Abbildung 1.8 gegeben.

   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 $\mathcal{S}$ 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 $\mathcal{S}$ 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 besteht.

## 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.

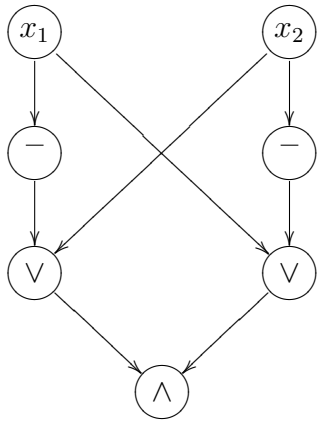



Abbildung 1.8: Schaltkreis für Aufgabe 1 (zu Abschnitt 1.1)

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

$$\{\overline{\phantom{x}}, \vee, \wedge\} \quad \text{bzw.} \quad \{\wedge, \oplus, k_1\}$$

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 1.6** *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.

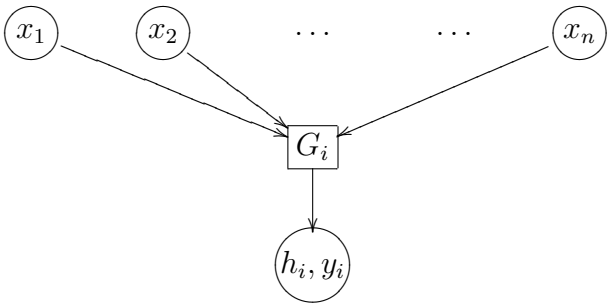
**Lemma 1.3** *Eine Menge $\mathcal{S} \subseteq B$ ist genau dann vollständig, wenn jede Boolesche Funktion aus $B$ durch einen Schaltkreis über $\mathcal{S}$ berechnet werden kann.*

*Beweis.* i) 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.

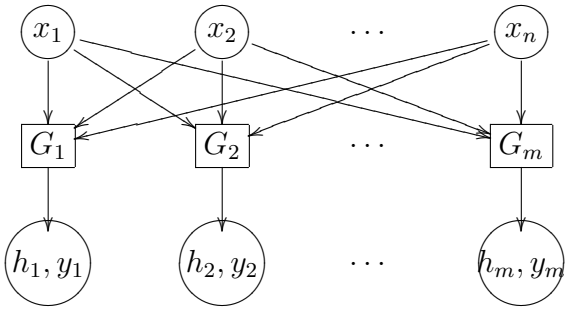
ii) Es sei daher nun jede Funktion aus $B$ berechenbar. Ferner sei $f$ eine Boolesche Funktion aus $B_{n,m}$ für gewisse natürliche Zahlen $n \geq 1$ und $m \geq 1$. Dann gibt es offenbar $m$ Funktionen $f_1, f_2, \ldots, f_m$ aus $B$ derart, dass

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

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 1.4** *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 $f$ berechnet. Für eine Funktion $f \in \mathcal{S}_2$ 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 $\mathcal{S}_2$ 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

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

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

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

$$f_d(x_1, x_2, \ldots, x_n) = \overline{f(\overline{a_1}, \overline{a_2}, \ldots, \overline{a_n})}.$$

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 1.5** *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 1.3 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 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 $< D$ auf $D$.* 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 $Q$.

$Q = T_0$. 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.

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

$Q = Mon$. 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.

$Q = Lin$. 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

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

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

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

gilt, woraus

$$\begin{aligned} f(a_1, a_2, \ldots, a_n) &= a_0 \oplus \bigoplus_{i=1}^{k} a_i(a_{i,0} \oplus \bigoplus_{j=1}^{n} a_{i,j}x_j) \\ &= (a_0 \oplus \bigoplus_{i=1}^{k} a_i a_{i,0}) \oplus (\bigoplus_{i=1}^{k} a_i a_{i,1})x_1 \oplus (\bigoplus_{i=1}^{k} a_i a_{i,2})x_2 \oplus \ldots \oplus (\bigoplus_{i=1}^{k} a_i a_{i,n})x_n \\ &= b_0 \oplus b_1 x_1 \oplus b_2 x_2 \oplus \ldots \oplus b_n x_n \end{aligned}$$

mit

$$\begin{aligned} b_0 &= a_0 \oplus (\bigoplus_{i=1}^{k} a_0 a_{i,0}), \\ b_j &= \bigoplus_{i=1}^{k} a_i a_{i,j} \quad \text{für} \quad 1 \leq j \leq n \end{aligned}$$

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

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

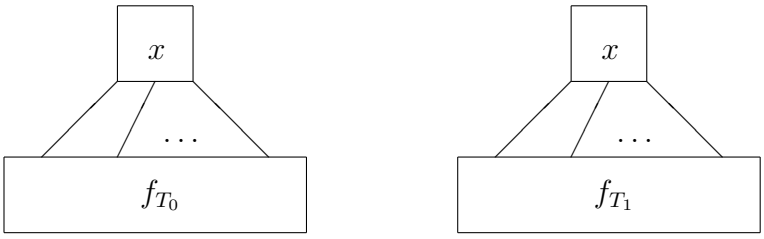
Damit haben wir bewiesen, dass Schaltkreise über einer Teilmenge $\mathcal{S}$ von $Q$ nur Funktionen aus $Q$ berechnen können. Da $Q \subset B$ gilt, kann $\mathcal{S}$ nicht vollständig sein.

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 $\{\vee, \wedge, ^{-}\}$ und Lemma 1.4 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 1.9 berechnet.

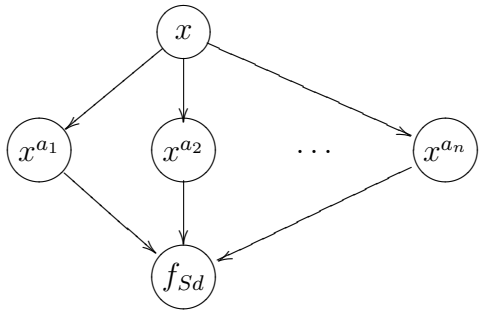



Abbildung 1.9: Schaltkreis, der im Knoten $f_{Sd}$ eine Konstante berechnet ($x^1$ ist die Identität, und wir verbinden den Eingangsknoten direkt mit $f_{Sd}$; $x^0$ ist die Negation)

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 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

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

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 1.10 die Negation. Ersetzen wir die Konstanten durch die Schaltkreise aus (a), so erhalten wir einen Schaltkreis über $\mathcal{S}$, der die Negation berechnet.

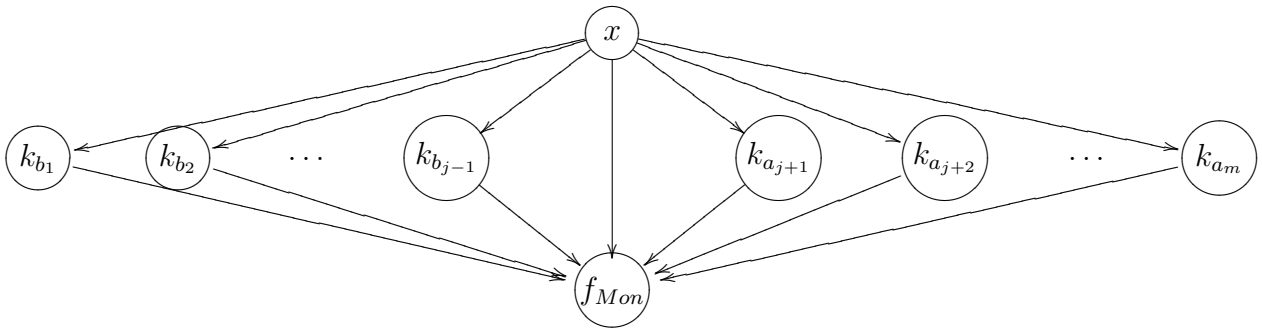



Abbildung 1.10: Schaltkreis zur Berechnung der Negation

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

$$\begin{aligned} f_{Lin}(x_1, x_2, \ldots, x_k) &= x_1 x_2 \cdot f_1(x_3, x_4, \ldots, x_k) \oplus x_1 \cdot f_2(x_3, x_4, \ldots, x_k) \\ &\quad \oplus x_2 \cdot f_3(x_3, x_4, \ldots, x_k) \oplus f_4(x_3, x_4, \ldots, x_k) \end{aligned}$$

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 1.11 a). Offenbar berechnet $S$ die Funktion $f'_{Lin}$ mit

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

für gewisse $\alpha, \beta, \gamma \in \{0,1\}$. Daher berechnet der Schaltkreis aus Abbildung 1.11 b) wegen

$$\begin{aligned} x_1x_2 &= ((x_1 \oplus \beta)(x_2 \oplus \alpha) + \alpha(x_1 \oplus \beta) \oplus \beta(x_2 \oplus \alpha) \oplus \gamma) \\ &\quad \oplus (\alpha\beta \oplus \gamma) \end{aligned}$$

die Konjunktion. Da in Abhängigkeit von $\alpha, \beta, \gamma$ 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 1.11 über $\mathcal{S}$ realisiert werden. Somit ist die Konjunktion durch einen Schaltkreis über $\mathcal{S}$ berechenbar.

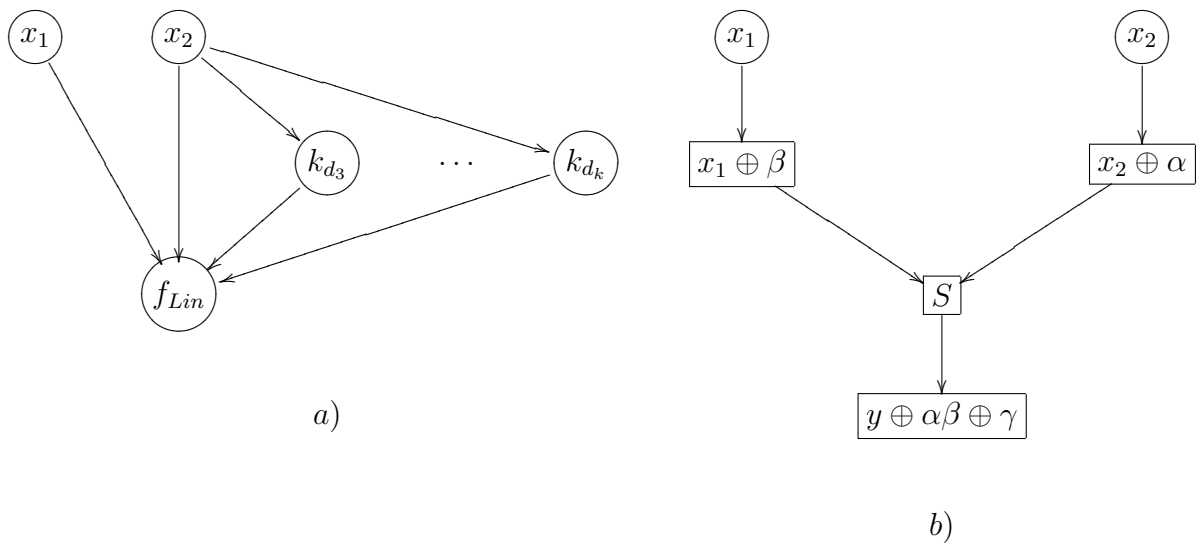



Abbildung 1.11: Schaltkreise zur Berechnung der Konjunktion

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

## 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) $\{\overline{\phantom{x}}, \wedge\}$, 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

   $$f_i(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & x_1x_2\ldots x_n \text{ ist die Dualdarstellung von } i \\ 0 & \text{sonst} \end{cases}$$

   definiert sind.

3. Beweisen Sie, dass für eine Funktion $f \notin T_0$ (or $F \notin T_1$ 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: Benutzen 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 anzugeben.

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

**Satz 1.6** *Sind $\mathcal{S}_1$ und $\mathcal{S}_2$ zwei vollständige Mengen, so gibt es (von $\mathcal{S}_1$ und $\mathcal{S}_2$ abhängige) Konstanten $c_1, c_2, d_1, d_2$ und $c_{k,1}, c_{k,2}$, $k \geq 1$ so, dass für jede Boolesche Funktion $f$*

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

*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

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

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

**Satz 1.7** *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 gilt $C_{\mathcal{S}}(f) = C(S)$. 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,... $C(S)-1$. $r_i$ sei 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

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

  und ersetzen das Gatter $i$ durch den Schaltkreis aus Abbildung 1.4.

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

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

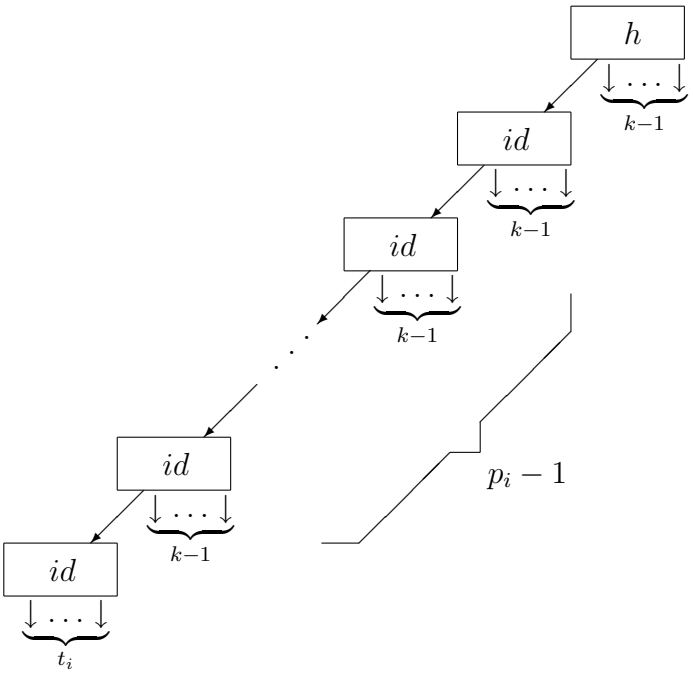



Abbildung 1.12: Reduktion des Ausgangsgrades auf $k$

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 $p_i$

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

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

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

$\sum_{i=1}^{C(S)-1} r_i$ ist 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 1.7 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 1.8** *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 $\lceil \frac{C(T_2)}{3} \rceil - 1$ 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

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

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 1.9** *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 1.8 i). □

Ohne Beweis geben wir noch die Abschätzung in der anderen Richtung. Für den technisch aufwendigen Beweis verweisen wir auf [18].

**Satz 1.10** *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 1.10 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 1.9 $\log(\log(C_{\mathcal{S}}(f))) \leq \log(D_{\mathcal{S}}(f))$ 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 1.8 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 $C_{\{\wedge,\vee,\overline{\phantom{x}}\}}(f) \leq 4C_{\{\oplus,\wedge,k_1\}}(f)$ für alle Booleschen Funktionen $f$ gilt.

   b) Geben Sie Boolesche Funktionen $f_1, f_2$ und $f_3$ an, für die

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

   gilt.

2. Beweisen Sie, dass die Abschätzung aus Satz 1.8 i) nicht verbessern lässt (d.h., zeigen Sie, dass es eine Funktion gibt, bei der in Satz 1.8 i) die Gleichheit gilt).

3. Berechnen $k(\mathcal{S})$ für $\mathcal{S} = \{\wedge, \vee, \overline{\phantom{x}}\}$ 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 1.11** *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 1.11 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 1.7** *Für ein Komplexitätsmaß $K \in \{C, L, D\}$ und eine Menge $\mathcal{S} \subseteq B$ von Basisfunktionen definieren wir die Funktion $K_{\mathcal{S}} : \mathbf{N} \to \mathbf{N}$ 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 1.7 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 1.12** *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 $2^{n-k}$ Tupel $\alpha_1, \alpha_2, \ldots, \alpha_{2^{n-k}}$ 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 Abb. 1.13 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

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

und die Funktion $f_{i,w}$, $f_{i,w}^{(1)}$ und $f_{i,w}^{(2)}$ durch

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

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

$$\begin{aligned} f_{i,w}(x_1, x_2, \ldots, x_n) &= f_{i,w}^{(1)}(x_1, x_2, \ldots, x_n) \wedge f_{i,w}^{(2)}(x_1, x_2, \ldots, x_n), \\ f(x_1, x_2, \ldots, x_n) &= \bigvee_{i=1}^{p} \bigvee_{w \in \{0,1\}^s} f_{i,w}(x_1, x_2, \ldots, x_n), \end{aligned}$$

| | $x_{n-k+1} \dots x_n$ | $\alpha_1 \quad \cdots \quad \alpha_{2^{n-k}}$ | $x_1$ $\vdots$ $x_{n-k}$ |
|---|---|---|---|
| $A_1$ | $\beta_1$ $\vdots$ $\beta_s$ | | $s$ |
| $\dots$ | $\dots$ | | |
| $A_i$ | $\beta_{(i-1)s+1}$ $\vdots$ $\beta_{is}$ | $\vert\ \vert$ $\vert w \vert$ $\vert\ \vert$ $\vert\ \vert$ $\vert w \vert$ $\vert\ \vert$ | $s$ |
| $\dots$ | $\dots$ | | |
| $A_{p-1}$ | $\beta_{(p-2)s+1}$ $\vdots$ $\beta_{(p-1)s}$ | | $s$ |
| $A_p$ | $\beta_{(p-1)s+1}$ $\vdots$ $\beta_{2^k}$ | | $s'$ |

Abbildung 1.13: $(k, s)$-Darstellung einer Funktion (hier haben wir den allgemeinen Fall mit einem kleineren letzten Block notiert)

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

Wir benutzen die letzte Darstellung zur Konstruktion eines Schaltkreises $S$ für $f$. $S$ ist in Abb. 1.14 gegeben. Dabei bedeuten $D_1$ und $D_2$ Schaltkreise zur Berechnung aller Konjunktionen vom Typ $x_1^{a_1} x_2^{a_2} \dots x_{n-k}^{a_{n-k}}$ bzw. $x_{n-k+1}^{a_{n-k+1}} x_{n-k+2}^{a_{n-k+2}} \dots x_n^{a_n}$. $E_1$ und $E_2$ sind Schaltkreise zur Berechnung aller Alternativen dieser Konjunktionen, die zur Berechnung von $f_{i,w}^{(1)}$ bzw. $f_{i,w}^{(2)}$ durch konjunktive Normalformen erforderlich sind. $G$ berechnet alle Konjunktionen $f_{i,w} = f_{i,w}^{(1)} \wedge f_{i,w}^{(2)}$ für $1 \leq i \leq p$ und $w \in \{0,1\}^s$. $F$ 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 Abb. 1.14 aber stets nur eine Ausgabe und/oder Eingabe angegeben.

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 $x_1^{a_1} x_2^{a_2} \dots x_{n-k}^{a_{n-k}}$ unter Verwendung von $(n-k-1)$ $\wedge$-Gattern. Da wir insgesamt $2^{n-k}$ Konjunktionen berechnen müssen, ergibt sich insgesamt eine Komplexität von

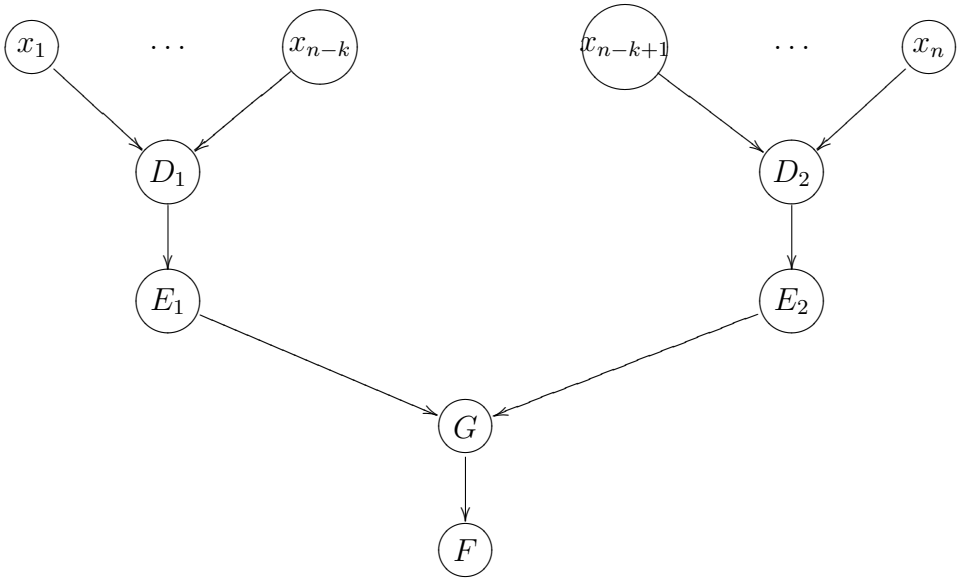



Abbildung 1.14: Schaltkreis entsprechend der $(k, s)$-Darstellung

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.

- $C(E_1) \leq (\frac{2^k}{s} + 1) \cdot 2^{n-k}$
  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 $p \leq (\frac{2^k}{s} + 1)$

- $C(E_2) \leq s \cdot 2^s \cdot (\frac{2^k}{s} + 1)$
  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.

- $C(G) \leq (\frac{2^k}{s} + 1) \cdot 2^s$
  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.

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

Insgesamt erhalten wir daher

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

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

$$\log(\frac{s}{2^k}) = \log(s) - k \leq \log(n - 5\log(n)) - 3log(n) \leq \log(n) - 3\log(n) = -2\log(n) = \log(\frac{1}{n^2})$$

ergibt sich

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

Wegen

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

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

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

Hieraus folgt sofort

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

und daher die obere Schranke

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

□

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 zur Konstruktion von längenoptimalen Schaltkreisen verwendet werden. Das Auffächern der Knoten in mehreren 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 dem russischen Mathematiker Oleg B. Lupanov der Beweis des folgenden Satzes, auf den wir hier verzichten.

**Satz 1.13** *Es gibt eine natürliche Zahl $n_0 \geq 1$ derart, dass für jede natürliche Zahl $n \geq n_0$*

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

*gilt.*

**Satz 1.14** *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 jede aus $n$ Elementen $x_i$ oder $\overline{x_j}$, $1 \leq$

$i, j \leq n$. 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 ([12]) haben Satz 1.14 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 1.15** *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 1.16** *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, höchsten 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 1.16 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 1.16 ist.

**Folgerung 1.17** *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:

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

Damit folgt aus Lemma 1.16

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

□

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

**Satz 1.18** *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.

$L(n,b) < 2^{2^n}$ ist gleichwertig zu $\log(L(n,b)) < 2^n$. Nach Lemma 1.16 und unserer Wahl von $b$ gilt für hinreichend großes $n$

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

□

**Satz 1.19** *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 1.8 und Satz 1.18 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 erfüllt die Tiefe von $f$ die Beziehung

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

□

**Satz 1.20** *Für jede (beliebig kleine) Zahl $\delta$ 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 1.18 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 1.17 die Ungleichung (1.22) gilt. Beachten wir noch $b \leq 2^n$, so erhalten wir

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

Da für große $n$ offenbar $16e(n+\frac{2^n}{n}(1-\delta)) \leq 2^n$ 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 1.12, 1.13, 1.15, 1.18, 1.19 und 1.20, so ergeben sich die Relationen

$$\begin{aligned} \frac{2^n}{n}(1-\delta) &\leq C(n) \leq \frac{2^n}{n} + o\Big(\frac{2^n}{n}\Big) , \\ \frac{2^n}{\log(n)}(1-\delta(n)) &\leq L(n) \leq \frac{2^{n+1}}{\log(n)} + o\Big(\frac{2^n}{\log(n)}\Big) , \\ n - \log(\log(n)) - 2 &\leq D(n) \leq n - \log(\log(n)) + 2 + o(1) , \end{aligned}$$

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

**Satz 1.21** *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 1.15 nur die von uns gezeigte schwächere Relation aus Satz 1.14 verwenden, so ergibt sich nur $D(n) = O(n)$.

In den Beweisen von Satz 1.18 und Satz 1.20 haben wir nur die Existenz von Funktionen mit großer Länge bzw. großer Komplexität nachgewiesen. Der Beweis sagt aber zum einen nicht aus, wieviel 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 1.20 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) $\delta$ 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 $\frac{2^n}{n}(1-\delta)$ besitzt, 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.

Die 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 Dualdarstellung durch $\underline{a}$, $\underline{b}$ und $\underline{c}$ gegeben sind, wenn wir die Tupel als Dualdarstellung interpretieren, und setzen

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

**Satz 1.22** *Es gilt* $C(f) \geq 3 \cdot 2^k - 3$.

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

Dieser Satz gibt nur eine lineares Wachstum (mit dem Faktor 3) in der Anzahl $2^k + 3k + 3$ der Variablen. Man beachte, dass auf der andere Seite Funktionen, die von allen Variablen wesentlich abhängen, lineares Wachstum aufweisen (siehe Satz 1.11).

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 1.8.

### Übungsaufgaben

1. Zeigen Sie, dass die Methode zum Beweis von Satz 1.14 angewandt auf das Maß $C$ eine zu große obere Schranke verglichen mit Satz 1.12 liefert.

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

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

   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 bzw. 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, den 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 1.8** *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.*

*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.*

*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 1.9** *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

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

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 1.10** *i) Ein K-Ausdruck $U$ heißt Implikant der Funktion $f$, falls für jede Belegung der Variablen $U$ 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.*

Offenbar ist K-Ausdruck $U$ genau dann ein Implikant der Funktion $f$, wenn $U^{-1}(1) \subseteq f^{-1}(1)$.

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

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

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

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

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

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

für den wegen (1.24) und (1.25)

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

gilt. Daraus ergibt sich die Gleichheit $f^{-1}(1) = W_1^{-1}(1) \cup W_2^{-1}(1) \cup \ldots \cup W_r^{-1}(1)$. 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 1.23** *Ein hinsichtlich der Kosten minimaler KA-Ausdruck für eine Boolesche Funktion $f$ ist die Alternative von Primimplikanten von $f$.* □

Wegen Lemma 1.23 sind für die Gewinnung kostengünstiger KA-Ausdrücke die Primimplikanten von besonderen Interesse. Daher suchen wir einen Algorithmus zur Bestimmung aller Primimplikanten einer Funktion $f$. Ein solcher Algorithmus stammt von Quine und McCluskey. Er ist in Abb. 1.15 angegeben.

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

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

*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 $f$

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

Abbildung 1.15: Algorithmus zur Bestimmung der Primimplikanten (von Quine und McCluskey)

- $Q_i$ 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 $m_{\underline{a}}$ mit $\underline{a} \in f^{-1}(1)$. Wegen $m_{\underline{a}}^{-1}(1) = \{\underline{a}\} \subseteq f^{-1}(1)$ ist $m_{\underline{a}}$ 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 $U = m_{\underline{a}}$. Ist $U$ noch Implikant von $f$ und 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 1.24** *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.*

*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 1.6** Wir betrachten die 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 ist. 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 1.6

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

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 Abb. 1.16 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 kostengünstige KA-Ausdruck $V$ begleitet.

*Eingabe*: PI-Tafel der Funktion $f$

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

$R_0 := \emptyset$ ;
$T_0 :=$ 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 $M_i := \{\underline{a} :$ zu $\underline{a}$ gehörige Spalte enthält genau eine $1\}$ ;
        $N_i := \{U : U$ ist Primimplikant in $T_i, U(\underline{a}) = 1$ für ein $a \in M_i\}$ ;
        $R := R \cup N_i$ ;
        $T_i'$ 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 1.16: Algorithmus zur Reduktion der 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 1.6** (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 zur Reduktion 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 |

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 Durchführen einer Streichung 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 $2^n$ bzw. $3^n$ 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 1.6** (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 1.6 zeigt, dass der kostenminimale KA-Ausdruck nicht eindeutig bestimmt ist.

Jedoch war die Bestimmung eines minimalen KA-Ausdrucks in Beispiel 1.6 aus der reduzierten PI-Tafel sehr einfach möglich. Wir wollen nun zeigen, dass dies im allgemeinen eine schwierige Aufgabe ist.

**Satz 1.25** *Das Problem des Ermittelns eines minimalen KA-Ausdrucks für die 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 vollzogen 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 zur Bestimmung 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 Überdeckunssproblem 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

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

Die Menge der Primimplikanten von $f$ ist dann durch

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

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 $x_i\overline{x_{n+1}}\overline{x_{n+2}}$ 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

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

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 1.7** Wir betrachten die Funktion

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

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 1.6 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 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 $x_1\overline{x_2}x_3 \vee x_1x_2\overline{x_3} \vee \overline{x_1}\,\overline{x_2}x_3$, die minimal bez. Größe bzw. Tiefe sind.

## 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 1.11** *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*

- *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 durch Markierung der erreichten Senke gegeben.*

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

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 1.8** Wir betrachten den Graphen $G$ aus Abb. 1.17, der offensichtlich ein Verzweigungsprogramm ist.

Betrachten wir das Tupel $(0,1,0)$, so beginnt der zugehörige Weg in der Wurzel $x_1$, geht entlang der mit 0 markierten Kanten 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

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

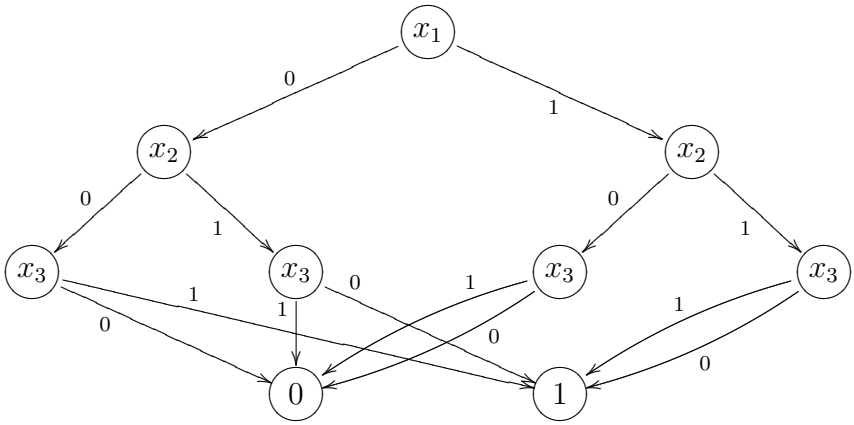



Abbildung 1.17: Verzweigungsprogramm $G$

| | | | |
|---|---|---|---|
| 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 1.12** *i) Für ein Verzweigungsprogramm $G$ sind die Größe $VC(G)$ bzw. 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*

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

$VC(G)$ ist die Anzahl der Knoten, die mit einer Variablen markiert sind.

**Beispiel 1.8** (Fortsetzung) Für das Verzweigungsprogramm $G$ aus Abb. 1.17 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. 1.17 nicht optimal für $f_G$, da das Verzweigungsprogramm $G'$ aus Abb. 1.18 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 $G$.)

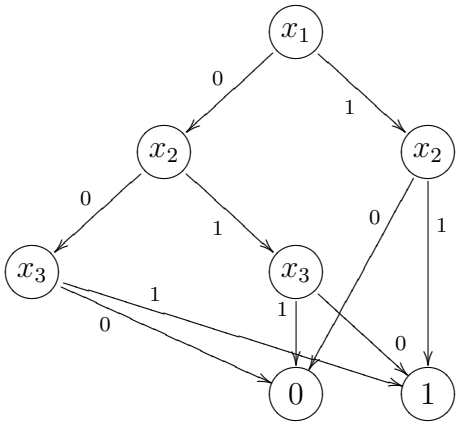



Abbildung 1.18: Verzweigungsprogramm $G'$

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

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

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 Schranken für die Komplexität Boolescher Funktionen bei Realisierung durch Verzweigungsprogramme. Im Wesentlichen gewinnen wir die Schranken durch eine Verallgemeinerung der Konstruktion aus Abb. 1.17, bei der zuerst nach der Belegung von $x_1$, dann nach der von $x_2$ usw. gefragt wird.

**Satz 1.26** *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 Knoten, 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 $f$ berechnet. Ferner gelten

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

womit der Satz bewiesen ist. □

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

**Satz 1.27** *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 1.28** *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 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. Bestimmen Sie einen kostenminimaler 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 $f(x_1, x_2, x_3, 1) = \overline{f(x_1, x_2, x_3, 0)}$.

## 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 [15], [17] und [19] 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, $z_0$ der Anfangszustand, $Q$ die Menge der

Endstände und $\delta$ die Überführungsfunktion sind. $\delta$ 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 die Turing-Maschine $M$ das Wort $w = a_1 a_2 \ldots a_n$, $a_i \in X$ 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, und der Kopf befindet sich erneut über dieser Zelle.

$w$ 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 zur Unterscheidung 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 $M$ die Sprache $L \subset X^+$ entscheidet, wenn $M$ jedes Wort aus $L$ akzeptiert und jedes Wort aus $X^* \setminus L$ ablehnt.

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

Dies 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 $t'(n) = 5 \cdot t(n) + 4$ entschieden wird. 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 Definition nicht hinsichtlich der Komplexitätsresultate nicht, wenn man nur am asymptotischen Verhalten der 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 die 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 $\mathcal{F}$ die 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 $g : \mathbf{N} \to \mathbf{N}$ eine Funktion. Wir sagen, dass $\mathcal{F}$ die 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 $h : \mathbf{N} \to \{0,1\}$ 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

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

und setzen

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

Offensichtlich ist $L$ unentscheidbar, denn wäre $L$ entscheidbar, so kann $h$ 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

$$f_n(\underbrace{0,0,\ldots,0}_{n\ mal}) = \begin{cases} 1 & h(n) = 0 \\ 0 & h(n) = 1 \end{cases},$$
$$f_n(\underbrace{0,0,\ldots,0}_{n-1\ mal},1) = \begin{cases} 1 & h(n) = 1 \\ 0 & h(n) = 0 \end{cases},$$
$$f_n(x_1, x_2, \ldots, x_n) = 0 \text{ in allen anderen Fällen.}$$

Hieraus folgt sofort

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

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

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

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

$x_i$ durch $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ mit $\sum_{j=0}^{r} \alpha_j \cdot 2^j = i$,
$z_i$ durch $(\beta_0, \beta_1, \ldots, \beta_r)$ mit $\sum_{j=0}^{s} \beta_j \cdot 2^j = i$.

Ferner kodieren wir $*$ durch das Tupel $(0,0,\ldots,0)$, das der Zahl 0 entspricht, d.h. wir betrachten $*$ als $x_0$. Fassen wir nun die Überführungsfunktion $\delta$ 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

$$\begin{aligned} \delta_1' &: \{0,1\}^{s+1} \times \{0,1\}^{r+1} = \{0,1\}^{r+s+2} \to \{0,1\}^{r+1} \\ \delta_2' &: \{0,1\}^{s+1} \times \{0,1\}^{r+1} = \{0,1\}^{r+s+2} \to \{0,1\}^{s+1} \end{aligned}$$

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 $L$ dual 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 $\delta_1'$ und $\delta_2'$ vornehmen. Dann wird in jedem Takt der Zustand entsprechend der Funktion $\delta_1'$ 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 $\delta_2'$, 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 1.13** *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 $\delta_1'$ und $\delta_2'$ 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 $\delta_1'$ stets aktualisieren und durch $\delta_2'$ 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 $\delta$ durch folgende Tabelle gegeben ist:

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

$M$ geht bei Lesen eines $a$ in den Zustand $r$ ($r$ und seine gestrichenen Versionen steht für *richtig*) und läuft nach links 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. $M$ ist also bewegungsuniform.

Die zugehörige Schaltung für das Wort $ab$ oder für ein beliebiges Eingabewort der Länge 2 ist in Abbildung 1.19 zu sehen, wobei wir auf die Kodierungen durch Tupel über $\{0, 1\}$ verzichten und daher mit $\delta_1$ und $\delta_2$ operieren. Neben den beiden Eingabesymbole sind auch die Zellen davor und dahinter zu berücksichtigen. Die Ausgabe erfolgt beim untersten $\delta_2$.

Somit wird jeder Schritt der Turing-Maschine durch zwei Schaltkreise, die $\delta_1'$ und $\delta_2'$ realisieren, simuliert. Gehen wir von einer Turing-Maschine mit einer Zeitkomplexität $t$ aus, so benötigen wir $t(n)$ Schalkreise für $\delta_1'$ und $t(n)$ Schaltkreise für $\delta_2'$ 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 1.29** *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 [18].

**Satz 1.30** *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 1.31** *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 bewegungsunabhängigen 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,

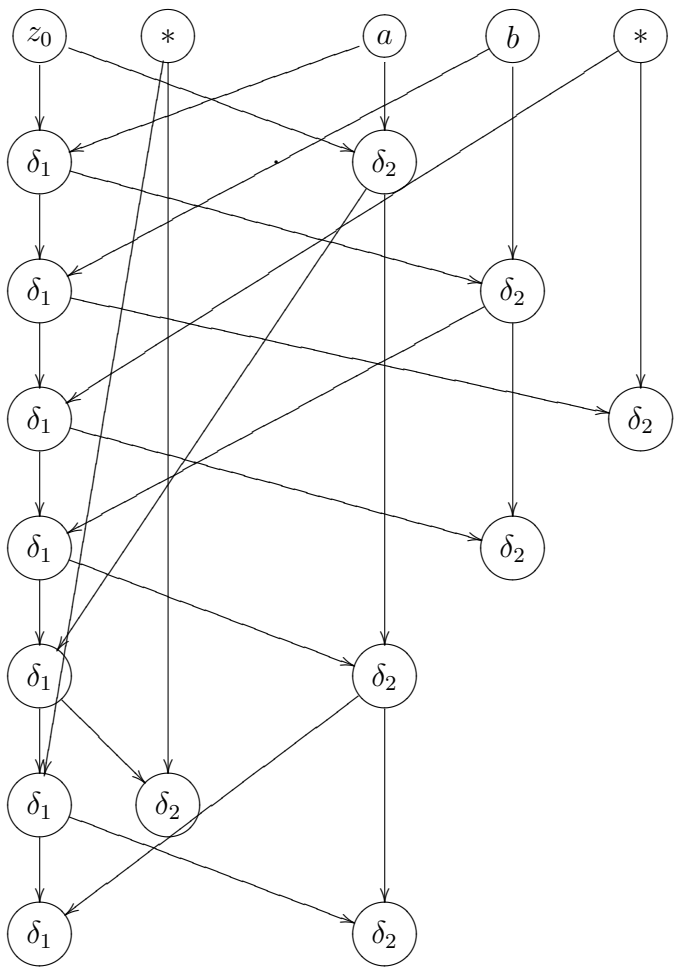



Abbildung 1.19: Schaltung zur Abarbeitung eines Wortes der Länge 2

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

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 1.32** *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 die quadratische $(p(w), p(w))$-Matrix $A(w)$ durch $a_{k,k'} = 1$, wenn 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 $a_{k,k'}^f = 1$, 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

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

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

$$O(1+\log(t(n)) \log(p(w))+\log(\#(K_a))) \leq O(\log(t(n)) \cdot \log(k(n))+\log(k(n))) \leq O(\log(t(n)) \cdot l(n)).$$

□

# 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] 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* (Ed. A. Mazurkiewicz), Lecture Notes in Computer Science 45, Springer-Verlag, 1976, 65–80.

[9] S. W. Jablonski, Einführung in die Theorie der Funktionen der $k$-wertigen Logik. In: *Diskrete Mathematik und mathematische Fragen der Kybernetik* (Herausg. S. W. Jablonski und O. B. Lupanov), Akademie-Verlag, 1980.

[10] O. B. Lupanov, Complexity of formulae realisation of functions of logical algebra. *Probl. Kibernetiki* **3** (1962) 782–811.

[11] O. B. Lupanov, A method of synthesis of control systems. *Probl. Kibernetiki* **23** (1970) 31–110.

[12] McColl und M. Paterson, The depth of all Boolean functions. *SIAM J. Comp.* **6** (1977) 373–380.

[13] T. Ottmann und P. Widmayer, *Algorithmen und Datenstrukturen*. BI-Wissenschaftsverlag, Mannheim, 1993.

[14] B. Reichel, Some classifications of Indian parallel languages. *J. Inf. Process. Cybern.* (EIK) **26** (1990) 85–99.

[15] U. Schöning, *Theoretische Informatik kurz gefaßt.* B.I.Wissenschaftsverlag, 1992.

[16] R. Sedgewick, *Algorithmen.* Addison-Wesley, 1990.

[17] K. Wagner, *Einführung in die Theoretische Informatik.* Springer-Verlag, 1994.

[18] I. Wegener, *The Complexity of Boolean Functions.* Teubner, Stuttgart und Wiley, New York, 1987.

[19] I. Wegener, *Theoretische Informatik.* Teubner-Verlag, 1993.