6.7 Schaltnetze

Eine formale Beschreibung von „Algorithmen“ kann man wie gesehen mit Hilfe von TURING-Maschinen geben. Ein einfacher zu definierendes und handzuhabendes Konzept ist das BOOLEsche **Schaltnetz**, das die in einem Algorithmus auszuführenden Bitoperationen in Form eines Ablaufdiagramms darstellt. Es wird auf zwei Weisen verallgemeinert:

probabilistisches Schaltnetz – damit werden probabilistische Algorithmen formalisiert,

polynomiale Schaltnetzfamilie – mit deren Hilfe kann die Komplexität eines Algorithmus bei wachsender Größe der Eingabedaten ausgedrückt werden.

Man kommt so zum Begriff der Schaltnetzkomplexität, der 1949 von SHANNON eingeführt wurde und für die Anwendung in der Kryptologie besonders gut geeignet ist.

Ein **Schaltnetz** ist ein azyklischer gerichteter markierter Graph, dessen Knoten sämtlich den Innengrad 0 oder 2 haben; das heisst, in jedem Knoten enden 0 oder 2 Pfeile. Ein **Eingang** ist ein Knoten mit Innengrad 0. Ein **Ausgang** ist ein Knoten mit Außengrad 0; von ihm gehen also keine weiteren Pfeile aus. Alle Knoten sind mit einem der Symbole $\oplus$ oder $\otimes$ markiert. Einige Eingänge sind als konstant gekennzeichnet; an ihnen liegt stets ein festes Bit 0 oder 1 an. Eine **Eingabe** ist eine Belegung der r nichtkonstanten Eingänge mit einer Bitfolge $x = (x_1, \ldots, x_r) \in \mathbb{F}_2^r$. An den inneren Knoten (“Gattern“) werden die anliegenden Bits dann jeweils $\bmod 2$ addiert oder multipliziert, je nach Markierung des Knotens. Am Ausgang ensteht nach Durchlaufen des gesamten Schaltnetzes die **Ausgabe** $y \in \mathbb{F}_2^s$. Das Schaltnetz definiert also eine Funktion

$$C \colon \mathbb{F}_2^r \longrightarrow \mathbb{F}_2^s$$

und gibt deren algorithmische Zerlegung in Bitoperationen wieder. Durch ein Schaltnetz ausdrücken läßt sich jede solche Funktion, die nur durch Addition und Multiplikation gebildet werden kann, also jedes Polynom, also jede Funktion $\mathbb{F}_2^r \longrightarrow \mathbb{F}_2^s$. Im Fall $r = 2$, $s = 1$ gibt es

$$(\#\mathbb{F}_2)^{\#(\mathbb{F}_2 \times \mathbb{F}_2)} = 2^4 = 16$$

derartige Funktionen. Sie sind alle in Tabelle 1 aufgezählt, aus der man auch leicht die entsprechenden Schaltnetze entnehmen kann; sie werden als Polynome $a+bX+cY+dXY \in \mathbb{F}_2[X, Y]$ beschrieben (algebraische Normalform). Allgemein werden Schaltnetz und zugehörige Funktion, da Verwirrung nicht zu befürchten ist, mit dem gleichen Buchstaben bezeichnet.

Bemerkungen und Beispiele.

a	b	c	d	Polynom	logische Operation	
0	0	0	0	0	FALSE	Konstante
1	0	0	0	1	TRUE	Konstante
0	1	0	0	X	X	Projektion
1	1	0	0	$1+X$	$\neg X$	negierte Projektion
0	0	1	0	Y	Y	Projektion
1	0	1	0	$1+Y$	$\neg Y$	negierte Projektion
0	1	1	0	$X+Y$	$X \oplus Y$	XOR
1	1	1	0	$1+X+Y$	$X \Longleftrightarrow Y$	Äquivalenz
0	0	0	1	XY	$X \wedge Y$	AND
1	0	0	1	$1+XY$	$\neg(X \wedge Y)$	NAND
0	1	0	1	$X+XY$	$X \wedge (\neg Y)$	
1	1	0	1	$1+X+XY$	$X \Longrightarrow Y$	Implikation
0	0	1	1	$Y+XY$	$(\neg X) \wedge Y$	
1	0	1	1	$1+Y+XY$	$X \Longleftarrow Y$	Implikation
0	1	1	1	$X+Y+XY$	$X \vee Y$	OR
1	1	1	1	$1+X+Y+XY$	$\neg(X \vee Y)$	NOR

Tabelle 1: Die 16 zweistelligen Bitoperationen

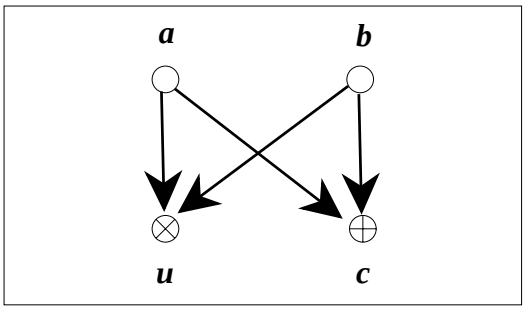

Abbildung 1: Schaltnetz zur Addition zweier Einbit-Zahlen

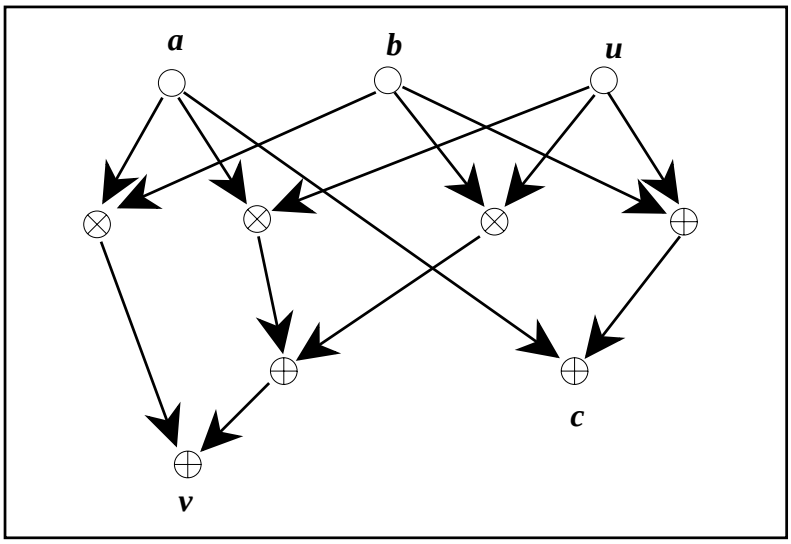

Abbildung 2: Schaltnetz zur Addition dreier Einbit-Zahlen

1. Die Addition zweier Bits a, b mit mod 2-Summe c und Übertrag u, also die „primitive“ Addition von Ziffern zur Basis 2, ist eine Funktion $C : \mathbb{F}_2^2 \longrightarrow \mathbb{F}_2^2$. Ein passendes Schaltnetz ist in Abbildung 1 wiedergegeben.

2. Die analoge Addition dreier Bits a, b, u mit mod 2-Summe c und Übertrag v ist der Grundbaustein, mit dem die Addition beliebiger Ganzzahlen zur Basis 2 aufgebaut wird. Sie wird durch eine Funktion $C : \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2^2$ mit dem Schaltnetz aus Abbildung 2 beschrieben. Eine Formel für v ist nämlich:

$$v = (a \otimes b) \oplus (a \otimes u) \oplus (b \otimes u) = \begin{cases} 1, & \text{wenn mindestens 2 Eingaben 1 sind,} \\ 0 & \text{sonst.} \end{cases}$$

3. Die Addition einer Einbit-Zahl zu einer s-Bit-Zahl verläuft nach einem ähnlichen, umfangreicheren Schema und lässt sich durch ein Schaltnetz mit $3s + 1$ Knoten, davon $s + 1$ Ausgängen, erledigen.

4. Die Multiplikation zweier Bits ist trivial. Interessanter ist es, die Multiplikation zweier Zweibit-Zahlen $2a_1 + a_0$ und $2b_1 + b_0$ mit Vierbit-Ergebnis $8c_3 + 4c_2 + 2c_1 + c_0$ durch ein Schaltnetz zu beschreiben. Der klassische Algorithmus ergibt die Formeln

$$\begin{aligned} &c_0 = a_0 \otimes b_0, \\ &t_1 = a_0 \otimes b_1, \qquad t_2 = a_1 \otimes b_0, \qquad c_1 = t_2 \oplus t_1, \qquad u_1 = t_2 \otimes t_1, \\ &t_3 = a_1 \otimes b_1, \qquad c_2 = t_3 \oplus u_1, \qquad c_3 = t_3 \otimes u_1 \end{aligned}$$

mit Hilfsbits t_1, t_2, t_3 und u_1. Sie lassen sich in das Schaltnetz aus Abbildung 3 übersetzen.

5. Auch logische Operationen und Verzweigungen kann man auf Additionen und Multiplikationen in $\mathbb{F}_2$ reduzieren. Für die logischen Operationen sieht man das an der Tabelle 1. Eine Verzweigung wird etwa so beschrieben: Gegeben seien drei Schaltnetze $C_0, C_1 : \mathbb{F}_2^r \longrightarrow \mathbb{F}_2^s$ und $E : \mathbb{F}_2^r \longrightarrow \mathbb{F}_2$. Gesucht ist ein Schaltnetz $C : \mathbb{F}_2^r \longrightarrow \mathbb{F}_2^s$ mit

$$C(x) = \begin{cases} C_0(x), & \text{falls } E(x) = 0, \\ C_1(x), & \text{falls } E(x) = 1. \end{cases}$$

Dieses konstruiert man mit Hilfe der Formeln

$$y \otimes v = \begin{cases} 0, & \text{falls } y = 0, \\ v, & \text{falls } y = 1. \end{cases}$$

$$(\neg y) \otimes u = \begin{cases} u, & \text{falls } y = 0, \\ 0, & \text{falls } y = 1. \end{cases}$$

$$[y \otimes v] \oplus [(1 \oplus y) \otimes u] = \begin{cases} u, & \text{falls } y = 0, \\ v, & \text{falls } y = 1. \end{cases}$$

Setzt man hier C_0, C_1 und E ein, so erhält man als Lösung

$$C(x) = [E(x) \otimes C_1(x)] \oplus [(1 \oplus E(x)) \otimes C_0(x)].$$

Dieses Schaltnetz ist in Abbildung 4 dargestellt. Es hat auch einen konstanten Eingang.

6. Der Vergleich $x \geq y$ zweier Bits ist eine der 16 zweistelligen Bitoperationen, nämlich $1 \oplus y \oplus x \otimes y$. Allgemeiner lässt sich ein Vergleich von $n+1$-Bit-Zahlen $x = (x_n \dots x_0)$ und $y = (y_n \dots y_0)$ so zusammenbasteln:

$$C(x, y) = \begin{cases} 1, & \text{wenn } x_n > y_n \\ & \text{oder } x_n = y_n \text{ und } x' \geq y', \\ 0, & \text{wenn } x_n = y_n \text{ und } x' < y' \\ & \text{oder } x_n < y_n \end{cases}$$

mit $x' = (x_{n-1} \dots x_0)$ und $y' = (y_{n-1} \dots y_0)$.

Es ist intuitiv klar und auch leicht zu zeigen, daß sich jeder Algorithmus durch ein Schaltnetz beschreiben läßt; genau nach diesem Prinzip der bitweisen Operationen arbeiten ja Computer auf der untersten Ebene. Klar ist auch, daß man mit einem solchen simplen Modell keine genauen Aufwandsberechnungen für reale Computer durchführen kann; wohl aber ist

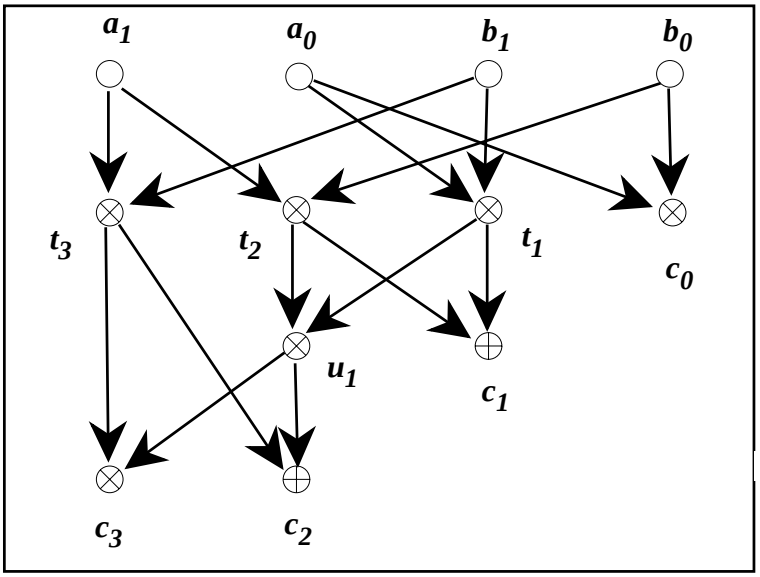

Abbildung 3: Schaltnetz zur Multiplikation zweier Zweibit-Zahlen

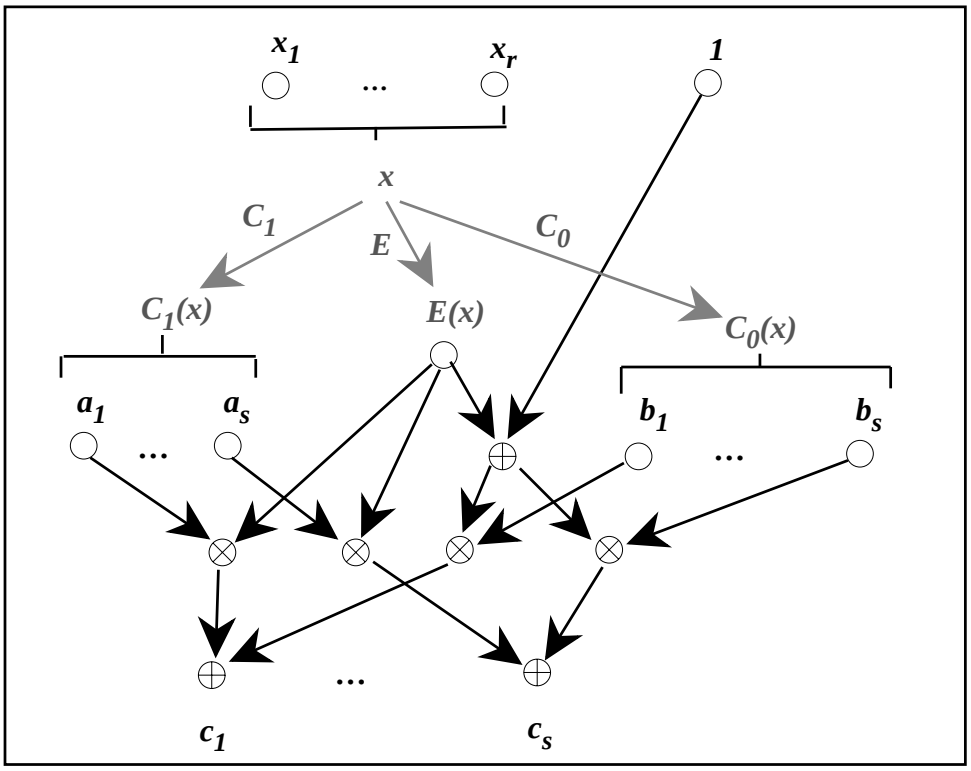

Abbildung 4: Schaltnetz für eine Verzweigung

die Unterscheidung, ob ein Algorithmus effizient, also mit polynomialem Zeitaufwand, arbeitet oder nicht, damit sehr gut möglich. Dazu betrachtet man die beiden Werte $\#C$, die **Größe** des Schaltnetzes, also die Anzahl der Knoten, und $t(C)$, die **Tiefe**, also die größte Weglänge innerhalb des Schaltnetzes. Die Größe ist ein Komplexitätsmaß für die serielle Abarbeitung, die Tiefe eines für die unbeschränkt parallele Abarbeitung des Algorithmus. Das Schaltnetz für die Addition zweier Bits, Abbildung 1 hat die Größe 4 und die Tiefe 1. Das Schaltnetz für die Addition dreier Bits, Abbildung 2 hat die Größe 10 und die Tiefe 3. Aus dem obigen Beispiel 3 erhält man ein einfaches (längst nicht optimales) Schaltnetz zur Addition von s Einbit-Zahlen mit $A(s)$ Knoten, darunter $a(s)$ Ausgängen. Nach Beispiel 1 ist $A(2) = 4$, $a(2) = 2$, nach Beispiel 3 gilt die Rekursion $a(s) = a(s-1)+1$ und $A(s) = A(s-1) + 2a(s-1) + 1$. Daraus folgt $a(s) = s$ und $A(s) = s^2$ durch Induktion. Das Schaltnetz für die Multiplikation zweier Zweibitzahlen, Abbildung 3 hat die Größe 12 und die Tiefe 3. Beim Schaltnetz für die Verzweigung, Abbildung 4 gelten die Formeln

$$\begin{aligned} \#C &= \#C_1 + \#C_2 + \#E - 2r + 3s + 2, \\ t(C) &= \max\{t(C_0) + 2, t(C_1) + 2, t(E) + 3\}. \end{aligned}$$

Für die Größe $V(n)$ des Schaltnetzes zum Vergleich in Beispiel 6 gilt die Rekursionsformel $V(n) = V(n-1)+11$ mit V(2) = 6, also $V(n) = 11n-5$. Die Aufwandsabschätzungen für die Ganzzahl-Operationen mit der Basis $B = 2$ des Zahlensystems besagen, dass Paare von n-Bitzahlen mit Schaltnetzen der Größe $\mathrm{O}(n)$ addiert und subtrahiert und mit Schaltnetzen der Größe $\mathrm{O}(n^2)$ multipliziert und dividiert werden können.

Bei der Definition von Schaltnetzen wird manchmal der Innengrad nicht durch 2 beschränkt. Diese Beschränkung ist aber durchaus sinnvoll, da reale Maschinen in jedem Schritt nur eine bestimmte Anzahl von Bits verarbeiten können. Ob man die Schranke auf 2 festsetzt oder höher, spielt für die folgenden Komplexitätsüberlegungen allerdings keine Rolle.

Zwischenergebnisse können in einem Schaltnetz wegen des unbeschränkten Außengrades beliebig oft verwendet werden; das entspricht einem unbeschränktem Speicher auf einem realen Computer. Ein engerer Begriff, bei dem dies vermieden wird, ist die BOOLEsche Formel: Eine solche ist ein Schaltnetz, dessen Graph ein Baum ist. Das bedeutet, dass der Außengrad jedes Knotens 1 ist (außer den Ausgängen mit Außengrad 0). Ein solches Schaltnetz entspricht genau dem Aufbau einer ausgeschriebenen Formel aus zweistelligen Operationen, wo ja auch jedes Zwischenergebnis nur einmal verwendet werden kann. Insbesondere muss jede Eingabe so oft wiederholt werden, wie sie gebraucht wird.

Bei einer weiteren allgemeineren Definition von Schaltnetzen werden auch beliebige zweistellige Bitoperationen statt nur $\oplus$ (XOR) und $\otimes$ (AND) zugelassen. Tabelle 1 zeigt auch, wie sich alle solchen durch $\oplus$ und $\otimes$ ausdrücken lassen. Diese allgemeinere Definition gestattet in der Regel etwas

kürzere Schaltnetze, hat aber keine lohnenden Auswirkungen auf die Komplexitätsüberlegungen.

Schaltnetze (oder Algorithmen) dienen nicht nur zur Berechnung von Funktionen, sondern auch zum Finden von Lösungen, formalisiert durch das Erfüllen einer Relation. Seien $X \subseteq \mathbb{F}_2^r$ und $Y \subseteq \mathbb{F}_2^s$ zwei Mengen und E eine Relation auf $X \times Y$, beschrieben durch eine Funktion

$$E : X \times Y \longrightarrow \mathbb{F}_2.$$

Gefragt sind Algorithmen, die zu gegebenem $x \in X$ ein $y \in Y$ finden mit $E(x, y) = 1$. Den bisher behandelten Spezialfall einer auszuwertenden Funktion $f : X \longrightarrow Y$ findet man als rechtseindeutige Relation wieder – $E(x, y) = 1 \Longleftrightarrow y = f(x)$. Man erfasst aber auch das Lösen von Gleichungen, etwa von linearen Gleichungssystemen: $X = \mathbf{M}_{m,n}(\mathbb{F}_2)$ = die Menge der $m \times n$-Matrizen, $Y = \mathbb{F}_2^m \times \mathbb{F}_2^n$, $E(x, y) = 1 \Longleftrightarrow xy_1 = y_2$ (als Produkt Matrix $\times$ Vektor).

Ein Schaltnetz $C : \mathbb{F}_2^r \longrightarrow \mathbb{F}_2^s$ erfüllt E, wenn $C(Y) \subseteq Y$ und $E(x, C(x)) = 1$ für alle $x \in X$.