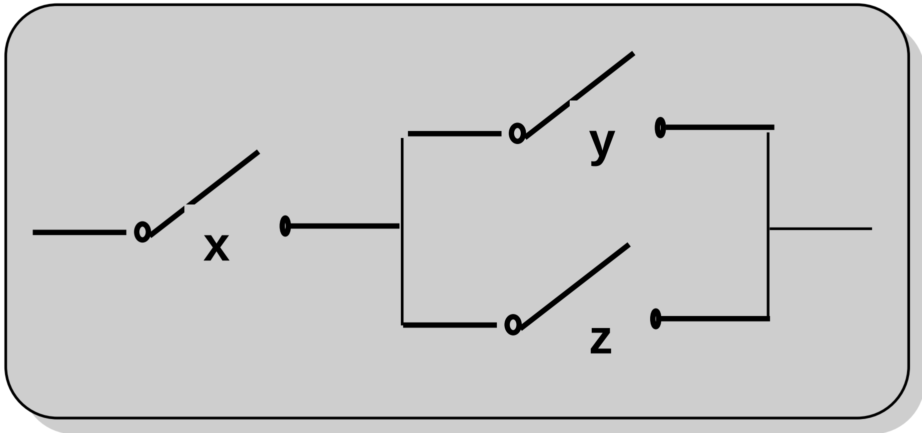
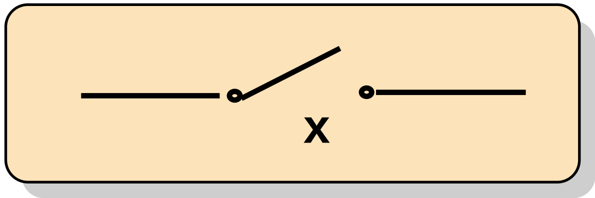
2. Schaltfunktionen und ihre Darstellung



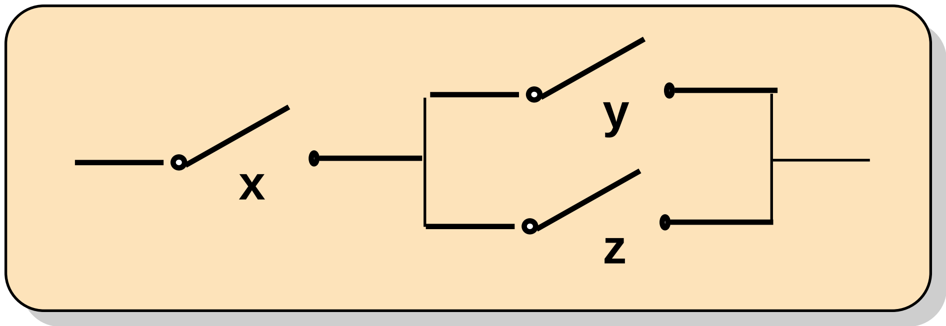
Schaltalgebra
Schaltkreise und -terme
Schaltfunktionen
Dualitätsprinzip
Boolesche Algebra
Darstellung von Schaltfunktionen

Schaltalgebra

Wir untersuchen das Verhalten von Schaltkreisen, die aus elementaren Schaltern zusammengebaut werden.



Ein-Aus-Schalter



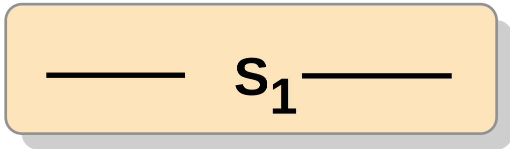
aus elementaren Schaltern gebauter Schaltkreis

Bei welchen Stellungen der Elementarschalter ist der Gesamtschalter geschlossen ?

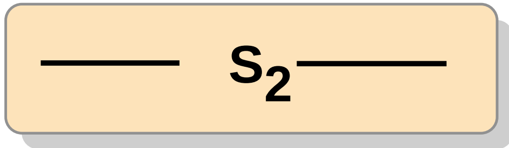
Serien/Parallele Schaltkreise

Beginnend mit den Elementarschaltern bauen wir neue Schaltkreise durch Serienschaltung oder Parallelschaltung

Sind

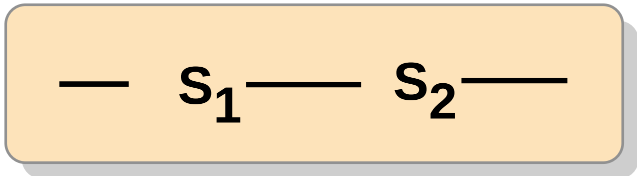


und

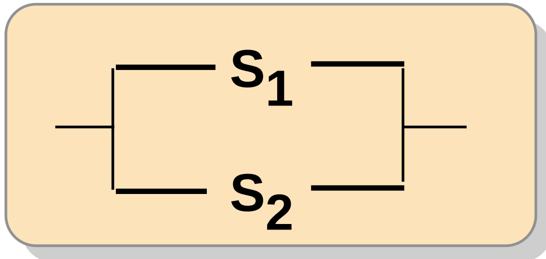


Schaltkreise,

so auch



und



S1 ∗ S2

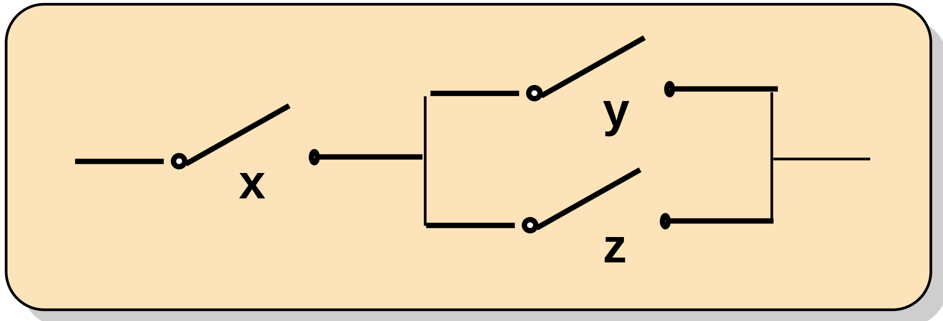
S1 + S2

Serien/Parallel-Terme

Jede Serien/Parallelschaltung läßt sich eindeutig durch einen Serien/Parallel-Schaltterm (S/P-Term) beschreiben, wobei die Elementarschalter Variablen entsprechen.

1. **Jede Variable ist ein S/P-Term.**
2. **Sind t_1 und t_2 S/P-Terme, so auch $t_1 + t_2$ und $t_1 * t_2$.**

Für die nebenstehende Schaltung ergibt sich :



$$S = x * (y + z)$$

Operationstafeln

Das Verhalten zusammengesetzter Schaltungen kann man tabellieren. Für jede Kombination der Komponentenschalter geben wir den Wert des zusammengesetzten Schalters an.

Ein = 1 Aus = 0

$S_1 \backslash S_2$	0	1
0	0	1
1	1	1

S1 + S2

$S_1 \backslash S_2$	0	1
0	0	0
1	0	1

S1 * S2

Das Verhalten zusammengesetzter Schaltkreise

Aus dem Verhalten der elementaren Operatoren
+ (Parallelschaltung) und ∗ (Serienschaltung) lässt sich das Verhalten komplexerer Schaltungen ermitteln :

x	y	z	y+z	x ∗ (y+z)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$$S = x * (y + z)$$

Schaltfunktionen

Eine Schaltfunktion ist eine n-stellige Operation auf der Menge { 0, 1}, das heißt also eine Abbildung

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

Jeder Schaltkreis realisiert eine Schaltfunktion.

$$\begin{aligned}
(0, 0, 0) &\longmapsto 0\\
(0, 0, 1) &\longmapsto 0\\
(0, 1, 0) &\longmapsto 0\\
(0, 1, 1) &\longmapsto 0\\
(1, 0, 0) &\longmapsto 0\\
(1, 0, 1) &\longmapsto 1\\
(1, 1, 0) &\longmapsto 1\\
(1, 1, 1) &\longmapsto 1
\end{aligned}$$

Die durch den Term x∗(y+z) realisierte Schaltfunktion

$f : \{ 0, 1\}^3 \longrightarrow \{ 0, 1\}$
$f(x,y,z) = x*(y+z)$

Abkürzung : $\underline{2} := \{ 0, 1 \}$

Mehrwertige Schaltfunktionen

Mehrwertige Schaltfunktionen

$$f : \underline{2}^n \longrightarrow \underline{2}^m$$

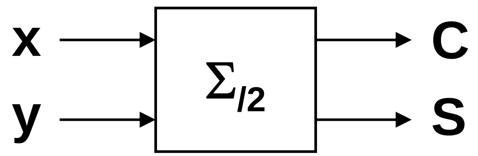
kann man aus den Komponentenfunktionen $f_1, \ldots, f_m$ zusammensetzen:

$$f(x_1,\ldots,x_n) = (f_1\,(x_1,\ldots,x_n)\,,\;.\;.\;.\;,\; f_m\,(x_1,\ldots,x_n)\,)$$

Beispiel: Halbaddierer

Ein Halbaddierer kann zwei Binärziffern addieren.
An den Eingängen liegen die Ziffern x und y.
An dem Ausgang S liegt die letzte Stelle der Summe,
an dem Ausgang C (für Carry) liegt der Übertrag (0 oder 1).

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

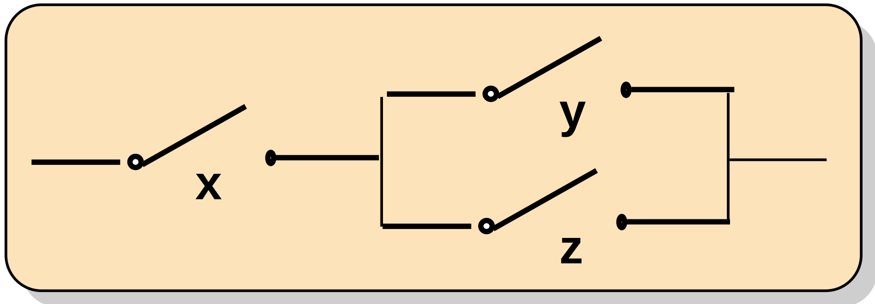


Äquivalenz

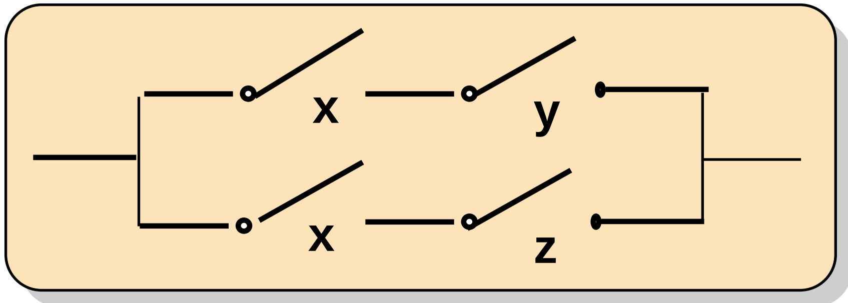
Jeder Schaltkreis beschreibt eindeutig eine Schaltfunktion. Verschiedene Schaltkreise können aber dieselbe Schaltfunktion beschreiben.

Zwei Schaltkreise heißen äquivalent, falls sie dieselbe Schaltfunktion beschreiben.

Die folgenden Schaltkreise sind äquivalent, wie man durch Vergleich ihrer Schaltfunktionen feststellen kann :



$x * (y + z)$



$(x * y) + (x * z)$

Gleichungen

Sind zwei Schaltkreise äquivalent, so identifiziert man die zugehörigen S/P-Terme und spricht von einer Gleichung.

Man schreibt $t_1 = t_2$, falls die durch t_1 und t_2 beschriebenen Schaltkreise dieselbe Schaltfunktion besitzen.

Beispiel für eine Gleichung :

$x + (y * z) = (x + y) * (x + z)$ **(Distributivität von + über *)**

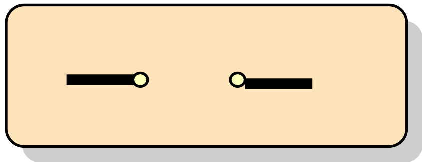
und ihre Herleitung :

x	y	z	y * z	x +(y*z)	x+y	x + z	(x + y) * (x + z)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

$\underline{0}$ und $\underline{1}$

Aus formalen Gründen führt man noch zwei spezielle Schaltkreise ein:

**Der immer geschlossene Schaltkreis.
Der zugehörige Term heißt $\underline{1}$.**



**Der immer offene Schaltkreis.
Der zugehörige Term heißt $\underline{0}$.**

Diese Schaltkreise werden auch durch die folgenden Gleichungen charakterisiert :

$$x + \underline{0} = x$$
$$x + \underline{1} = \underline{1}$$

$$x * \underline{0} = \underline{0}$$
$$x * \underline{1} = x$$

Gleichungen für Serien/Parallel Schaltkreise

$x + x = x$	Idempotenz	$x * x = x$
$x + y = y + x$	Kommutativität	$x * y = y * x$
$x + (y + z) = (x + y) + z$	Assoziativität	$x * (y * z) = (x * y) * z$
$x * (x + y) = x$	Absorption	$x + (x * y) = x$
$x * (y + z) = x * y + x * z$	Distributivität	$x + (y * z) = (x + y) * (x + z)$
$x + \underline{0} = x$		$x * \underline{1} = x$
$x + \underline{1} = \underline{1}$		$x * \underline{0} = \underline{0}$

Sei M eine nicht-leere Menge, auf der zwei Verknüpfungen + und * definiert sind und in der zwei Elemente 0 und 1 ausgezeichnet sind. Gelten für beliebige x, y, z $\in$ M die obigen Gleichungen, so heißt

(M; *, +, 0, 1) distributiver Verband mit 0 und 1.

Dualitätsprinzip

Zu einem Term t erhält man den dualen Term t^d, indem man

- **jede 0 durch 1 und jede 1 durch 0 ersetzt**
- **jedes + durch $*$ und jedes $*$ durch + ersetzt**

Definition:

Induktiv über den Aufbau der S/P Terme definieren wir den dualen Term t^d zu einem Term t:

- **Für die Konstanten sei:** $0^d = 1$ $\quad$ $1^d = 0$
- **Für Variablen x sei:** $x^d = x$
- **Für beliebige Terme u und v sei:** $(u * v)^d = u^d + v^d$ **und** $(u+v)^d = u^d * v^d$

Folgerungen:

- **Für jedem Term t gilt** $t = t^{d\,d}$.
- **Gilt eine Gleichung** $t_1 = t_2$, **so auch** $t_1^d = t_2^d$.

Monotone Schaltfunktionen

Jeder S/P-Schaltkreis realisiert eine Schaltfunktion.

Ist umgekehrt jede Schaltfunktion durch einen S/P-Schaltkreis realisierbar?

Definition: Eine Schaltfunktion $f : \underline{2}^n \to \underline{2}$ heißt ***monoton***, falls gilt :

$$(x_1,\ldots,x_n) \leq (y_1,\ldots,y_n) \;\Rightarrow\; f(x_1,\ldots,x_n) \leq f(y_1,\ldots,y_n)$$

komponenten-
weise

Satz: Jede durch einen S/P-Term realisierbare Schaltfunktion ist monoton.

Dieser Satz wird durch Induktion über den Aufbau von S/P-Termen gezeigt. Wie wir später sehen werden, gilt auch die Umkehrung.

Das Beispiel zeigt eine Schaltfunktion, die nicht durch eine S/P-Schaltung realisierbar ist:

⇨ **Wechselschaltung**

Beispiel:

x	y	f(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

Es gilt $(0,1) \leq (1,1)$ aber $f(0,1) > f(1,1)$.

Negation

Ist S ein Schaltkreis, so sei S' derjenige Schaltkreis, welcher genau dann offen ist, wenn S geschlossen ist.

S' heißt die Negation von S
(oder Komplement bzw. Inverses).

Die Negation ist die einfachste nicht-monotone Schaltfunktion.

Für die zweifache Negation gilt : $S'' = S$

Boolesche Schaltungen

Ein Schaltkreis, in dem neben Serien- und Parallel-Schaltung auch noch die Negation verwendet werden darf, heißt Boolesche Schaltung.

Der einer Booleschen Schaltung entsprechende Term heißt Boolescher Term. Die Gleichheit Boolescher Terme definiert man analog zu der Gleichheit von S/P-Termen.

Beispiel für eine Boolesche Gleichung :

$$(x + y)' = x' * y'$$

(-> de Morgan'sche Regeln)

und ihre Herleitung :

x	y	x+y	(x + y)'	x'	y'	x' * y'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Mit den de Morgan'schen Regeln zeigt man leicht folgenden Zusammenhang zwischen
Dualität und Negation:

Kommen in t die Variablen $x_1, ..., x_n$ vor, so gilt:

$$t^d(x_1,...,x_n) = (t(x_1',...,x_n'))'$$

Boolesche Gleichungen

Zu den bereits erwähnten Gleichungen eines distributiven Verbandes kommen noch die folgenden Gleichungen, in denen die Negation auftaucht:

$(x + y)' = x' * y'$	**de Morgan´sche Regeln**	$(x * y)' = x' + y'$
$x + x' = 1$	**Komplementregeln**	$x * x' = 0$

$$x'' = x$$

Ein distributiver Verband mit 0 und 1, in dem ein Komplement so definiert ist, dass die obigen Gleichungen gelten, heißt

Boolesche Algebra.

Boolesche Algebra

Die Boolesche Algebra ist ein algebraischer Kalkül, der vielfältige Gebiete beschreibt, darunter insbesondere

- **die Algebra der Aussagenlogik:**

  ({wahr, falsch}; $\wedge$, $\vee$, $\neg$, falsch, wahr)

- **die Mengenalgebra**

  ($\wp$(M); $\cap$, $\cup$, $\overline{}$, $\emptyset$, M) mit beliebiger nicht-leerer Menge M

- **die Schaltalgebra**

  ($\underline{2}$, *, +, ’, 0, 1)

George Boole (1815 -1864) :
An Investigation into the Laws of Thought

Aussagenlogik

Die Aussagenlogik ist ein zur Schaltalgebra isomorphes Modell einer booleschen Algebra. Man geht von einer Menge E von Elementaraussagen aus. Es ist gefordert, dass man zu jedem $e \in E$ feststellen kann, ob e wahr oder falsch ist (tertium non datur).

Beispiele von Elementaraussagen : **"2+2 = 5"**
"Microsoft ist eine Biersorte"
"Blei ist schwerer als Wasser"

1. **Jede Elementaraussage ist eine Aussage.**
2. **Sind A1 und A2 Aussagen, so auch**

   A1 $\wedge$ A2 (*Konjunktion*), A1 $\vee$ A2 (*Disjunktion*) und $\neg$ A1 (*Negation*).

Der Wahrheitswert einer zusammengesetzten Aussage berechnet sich aus den Wahrheitswerten der Teilaussagen anhand der Wahrheits-tabellen der logischen Operatoren.

Wahrheitstabellen

Die Wahrheitstabellen für $\vee$, $\wedge$ und $\neg$ entstehen aus den Tabellen für +, $*$ und ´, indem man jeweils 1 durch W und 0 durch F ersetzt ($\rightarrow$ Isomorphie).

$A_1 \backslash A_2$	F	W
F	F	W
W	W	W

$A_1 \vee A_2$

$A_1 \backslash A_2$	F	W
F	F	F
W	F	W

$A_1 \wedge A_2$

A	$\neg$A
F	W
W	F

$\neg A$

Für die Äquivalenz von Aussagen gelten genau die Gleichungen der Booleschen Algebra. Man muss lediglich +, $*$ und ´ durch $\vee$, $\wedge$ und $\neg$ sowie 0 und 1 durch F und W ersetzen.

Beispiel : $A \vee (A \wedge B) = A$

Mengenalgebra

Ausgehend von einer festen Grundmenge M betrachten wir $\wp$(M), die Menge aller Teilmengen von M. Auf $\wp$(M) sind die Operationen $\cup$, $\cap$ und $\overline{\ }$ definiert. Zwei Elemente von $\wp$(M) spielen eine Sonderrolle : $\emptyset$ und M.

Die Gleichungen der Booleschen Algebra gelten, wenn man +, * und ‘ durch $\cup$, $\cap$ und $\overline{\ }$ sowie 0 und 1 durch $\emptyset$ und M ersetzt.

<u>Beispiel</u> :
Für beliebige U, V, W aus $\wp$(M) gilt: $U \cup (V \cap W) = (U \cup V) \cap (U \cup W)$

<u>Satz von Stone:</u>
Jede Boolesche Algebra $\langle A, op_1, op_2, op_3; 0,1 \rangle$ mit endlichem A ist isomorph zu einer Mengenalgebra $\langle \wp(M), \cup, \cap, \overline{\ }; \emptyset, M \rangle$, d.h. es gibt eine Bijektion $\Psi : A \to \wp(M)$ mit:

$$\Psi(x\ op_1\ y) = \Psi(x) \cup \Psi(y)$$
$$\Psi(x\ op_2\ y) = \Psi(x) \cap \Psi(y)$$
$$\Psi(op_3\ x) = \overline{\Psi(x)}$$

$\rightarrow$ **Jede endliche Boolesche Algebra hat eine Elementzahl, die eine Zweierpotenz ist.**

Vereinfachende Schreibweisen

Um sich Schreibarbeit und Klammern zu sparen, werden folgende Vereinbarungen getroffen:

- **$*$ bindet stärker als +**
- **Negation ' bindet stärker als $*$**
- **Klammern in Summen bzw. Produkten mit mehreren Operanden entfallen (zulässig wegen der Assoziativgesetze)**
- **das Multiplikationszeichen für die Serienschaltung * kann entfallen**

<u>Beispiel</u>:

Statt ((x * (y')) * ((x + y) + z)) + (x') schreibe x y' (x+y+z)+x'

Termvereinfachungen

Eine Schaltfunktion kann durch mehrere Boolesche Terme dargestellt werden. Meist sucht man eine möglichst einfache Darstellung. Mit Hilfe der Gleichungen kann man einen Booleschen Term meist auf eine einfache Form bringen. (Wir lassen hier offen, was genau *einfach* heißt.)

Gegeben sei der Term

$$x'\,y'\,z' + x'\,y'\,z + x\,y'z' + x\,y\,z$$

Durch Anwendung der Gleichungen erhalten wir :

$= x'\,y'\,(z' + z) + x\,y'\,z' + x\,y\,z$	(Distributivität)
$= x'\,y'\,(z + z') + x\,y'\,z' + x\,y\,z$	(Kommutativität)
$= x'\,y'\,\underline{1} + x\,y'\,z' + x\,y\,z$	(Komplement)
$= x'\,y' + x\,y'\,z' + x\,y\,z$	(x*1=x)
$= x'\,y' + x'\,y'\,z' + x\,y'\,z' + x\,y\,z$	(Absorption)
$= x'\,y' + (x' + x)\,y'\,z' + x\,y\,z$	(Distributivgesetz)
$= x'\,y' + \underline{1}\,y'\,z' + x\,y\,z$	(Komplement)
$= x'\,y' + y'\,z' + x\,y\,z$	(x*1=x)

Darstellung von Schaltfunktionen

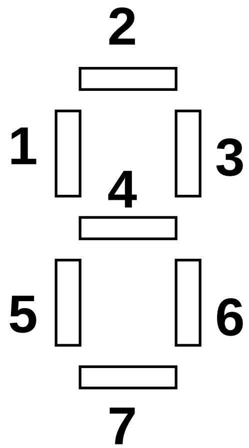
<u>Problem</u>: Finde Termdarstellung zu gegebener Schaltfunktion.

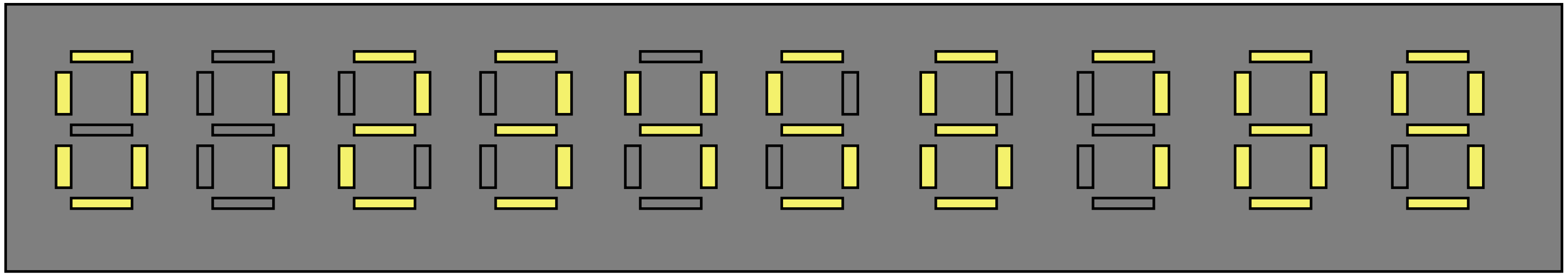
Beispiel: Gegeben sei folgende Schaltfunktion g:

i	x	y	z	g(x,y,z)
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

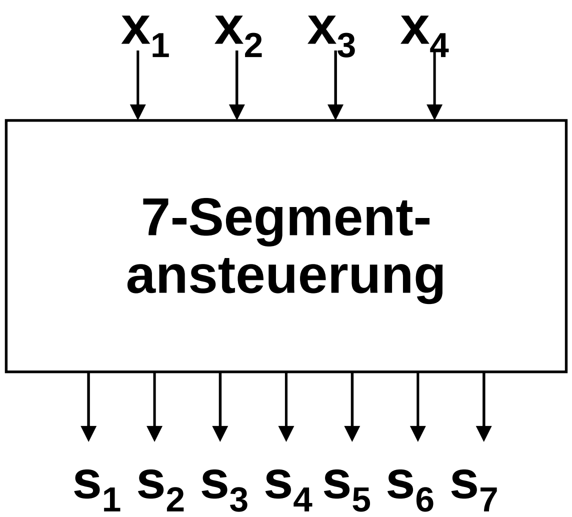
Wie kann ein Boolescher Term zur Darstellung von g bestimmt werden?

Beispiel: 7-Segmentanzeige





Darstellung der Ziffern 0 .. 9
durch 4 Eingabebits



1 ≈ Segment leuchtet
0 ≈ kein Leuchten

Zugehörige Schaltfunktion

x_1	x_2	x_3	x_4	s_1	s_2	s_3	s_4	s_5	s_6	s_7
0	0	0	0	1	1	1	0	1	1	1
0	0	0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	1	1	1	0	1
0	0	1	1	0	1	1	1	0	1	1
0	1	0	0	1	0	1	1	0	1	0
0	1	0	1	1	1	0	1	0	1	1
0	1	1	0	1	1	0	1	1	1	1
0	1	1	1	0	1	1	0	0	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Grundbegriffe

Sei $n \geq 1$ die Stelligkeit von Schaltfunktionen. Wir identifizieren im folgenden Schaltfunktionen und ihre Termdarstellung:

- Ein **Literal** ist eine Variable x_i oder ihre Negation x_i'.
  Bezeichnung: $x_i^1 := x_i$, $x_i^0 := x_i'$.

- Ein **Monom** ist ein Produkt von Literalen: $x_{i_1}^{\varepsilon_1} \ldots x_{i_k}^{\varepsilon_k}$ mit $k \leq n$
- Eine **Klausel** ist eine Summe von Literalen: $x_{i_1}^{\varepsilon_1} + \ldots + x_{i_k}^{\varepsilon_k}$ mit $k \leq n$

- Monome mit voller Länge $k = n$ werden als **Minterme** bezeichnet:
  $$x_{i_1}^{\varepsilon_1} \ldots x_{i_n}^{\varepsilon_n}.$$
- Klauseln voller Länge heißen **Maxterme** $x_{i_1}^{\varepsilon_1} + \ldots + x_{i_n}^{\varepsilon_n}$.

- Ein Monom a über $x_1, \ldots, x_n$ heißt **Implikant** einer Schaltfunktion $f: \underline{2}^n \to \underline{2}$, falls für alle $b_1, \ldots b_n$ gilt: $f_a(b_1,\ldots,b_n) <= f(b_1,\ldots,b_n)$.
  Dabei sei $f_a(x_1,\ldots,x_n) = a$.

Minterme

Ein Minterm ist eine atomare Schaltfunktion, die für genau eine Eingabe den Wert 1 liefert.

Die Schaltfunktion hat dann die Form

$$f(x_1,\ldots,x_n) = x_1^{\varepsilon_1} * x_2^{\varepsilon_2} \ldots x_n^{\varepsilon_n} \qquad \text{mit } x_i^{\varepsilon_i} = \begin{cases} x_i, \text{ falls } \varepsilon_i = 1 \\ x_i', \text{ sonst} \end{cases}$$

f heißt j-ter Minterm m_j, wobei $j = (\varepsilon_1 \ldots \varepsilon_n)_2$.

Beispiele:

i	x	y	z	$m_0(x,y,z) = x'\,y'\,z'$	$m_1(x,y,z) = x'\,y'\,z$	$m_4(x,y,z) = x\,y'\,z'$	$m_7(x,y,z) = x\,y\,z$
0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0
2	0	1	0	0	0	0	0
3	0	1	1	0	0	0	0
4	1	0	0	0	0	1	0
5	1	0	1	0	0	0	0
6	1	1	0	0	0	0	0
7	1	1	1	0	0	0	1

Atome

Sei < A, *, + , ´ ; 0, 1> eine Boolesche Algebra.

$a \in A$ mit $a \neq 0$ heißt **Atom** der Booleschen Algebra, falls

$a * b = a$ oder $a * b = 0$ für alle $b \in A$.

Atome einer Booleschen Algebra sind die "kleinsten Elemente", die multipliziert mit einem anderen entweder das Element selbst oder Null ergeben.

In Mengenalgebren $\wp(M)$ (mit Multiplikation $\cap$) sind die Atome gerade die einelementigen Mengen.

In der Algebra der Schaltfunktionen sind die Atome die Minterme.

Disjunktive Normalform

Sei f : $\underline{2}^n$ --> $\underline{2}$ eine Schaltfunktion und 0<= i <= 2^n-1 ein Zeilenindex der Funktionstabelle mit i = $(i_1...i_n)_2$.

i heißt einschlägiger Index zu f, falls $f(i_1,...,i_n) = 1$ ist.

Satz:

Jede Schaltfunktion f: $\underline{2}^n$ -> $\underline{2}$ ist eindeutig darstellbar als Summe der Minterme ihrer einschlägigen Indices, d.h. ist $I \subseteq \{0,...,2^n-1\}$ die Menge der einschlägigen Indices von f, so gilt:

$$f = \sum_{i \in I} m_i$$

und keine andere Minterm-Summe stellt f dar.

Diese Darstellung heißt disjunktive Normalform (DNF) von f.

Beweis der Existenz und Eindeutigkeit der DNF

Beispiel-Fortsetzung

Wir betrachten nochmals die Schaltfunktion g:

i	x	y	z	g(x,y,z)
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Die disjunktive Normalform von g ergibt sich zu:

$$
\begin{aligned}
&g(x,y,z)\\
&= m_0(x,y,z) + m_1(x,y,z) +\\
&\qquad m_4(x,y,z) + m_7(x,y,z)\\
&= x'y'z' + x'y'z + xy'z' + xyz
\end{aligned}
$$

Darstellung von Schaltfunktionen mit Maxtermen

Das vorgestellte Verfahren liefert für eine Schaltfunktion einen Booleschen Term, welcher umso komplizierter ist, je mehr 1-en die Schaltfunktion hat. Das Dualitätsprinzip eröffnet eine zweite Möglichkeit der Vorgehensweise, die immer dann sinnvoll ist, wenn mehr 1-en als 0-en vorhanden sind. Der endgültige Term ergibt sich dann als Produkt von Elementarsummen, sogenannten Maxtermen.

1. **Spezifikation:**

x	y	z	g(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

2. **Zerlegung in Maxterme:**

x	y	z	e_1	e_2	e_3
0	0	0	0	1	1
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	1	1	1

3. **Ergebnis:**

$$g(x,y,z) = e_1 * e_2 * e_3$$
$$= (x + y + z) * (x + y' + z') * (x' + y + z')$$

Maxterme und konjunktive Normalform

Sei $0 <= i <= 2^n-1$ mit $i = (i_1 ... i_n)_2$ und m_i der i-te Minterm.

Dann heißt die Funktion M_i: $\underline{2}^n \rightarrow \underline{2}$ mit

$$M_i(x_1,...,x_n) := m_i(x_1,...,x_n)' = (x_1^{i_1})' + ... + (x_n^{i_n})'$$

i-ter Maxterm.

Satz:

Jede Schaltfunktion f: $\underline{2}^n \rightarrow \underline{2}$ ist eindeutig darstellbar als Produkt der Maxterme ihrer nicht einschlägigen Indices.

Diese Darstellung heißt konjunktive Normalform (KNF) von f.

Funktionale Vollständigkeit

Ein System B = $\{f_1,\ldots,f_n\}$ von Schaltfunktionen heißt (funktional) vollständig, wenn sich jede Schaltfunktion allein durch Einsetzungen bzw. Kompositionen von Funktionen aus B darstellen läßt.

Es gilt somit: {+, ∗, ´ } ist vollständig.

Mit den de Morganschen Regeln folgt, dass auch {+, ´ } bzw. {∗, ´ } vollständig sind.

Man erkennt leicht, dass alle drei Systeme ohne ´ nicht mehr vollständig sind. Auch {´} ist nicht vollständig. Es existieren aber einelementige vollständige Systeme.

Einelementige vollständige Systeme

Das einelementige System {NAND} ist funktional vollständig, denn

- **{+,’ } ist funktional vollständig und**
- **es existieren NAND-Darstellungen dieser beiden Funktionen:**

x+y = (x NAND x) NAND (y NAND y)

x’ = (x NAND x)

Analog folgt, daß {NOR} funktional vollständig ist.

Bestimmung einer NAND-Darstellung

- **Ausgangspunkt: DNF oder disjunktive Form**
- **Doppelte Negation anwenden und innere Negation mit De Morgan nach innen ziehen**

Beispiel:

x	y	z	f(x,y,z)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

DNF: x‘y‘z‘ + x‘y‘z + x‘yz‘ + x‘yz + xy‘z‘
= (x‘y‘z‘ + x‘y‘z + x‘yz‘ + x‘yz + xy‘z‘)‘‘
= [(x‘y‘z‘)‘ · (x‘y‘z)‘ · (x‘yz‘)‘ · (x‘yz)‘ · (xy‘z‘)‘]‘

mehrstellige NAND-ANwendungen