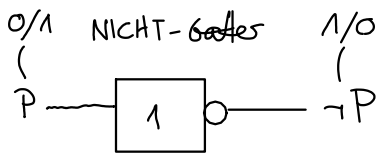
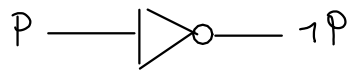
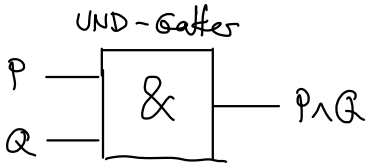
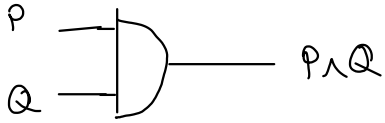
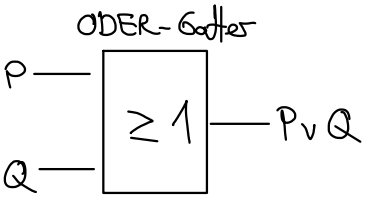
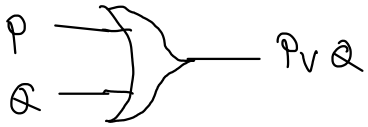
Prof. Dr. M. Helbig
Mathematik – Übung

Hochschule Rosenheim
University of Applied Sciences

Logik-Anwendungen

Anwendung: Entwurf von Schaltkreisen

Zu den logischen Verknüpfungen gibt es Schaltungselemente/Gatter:

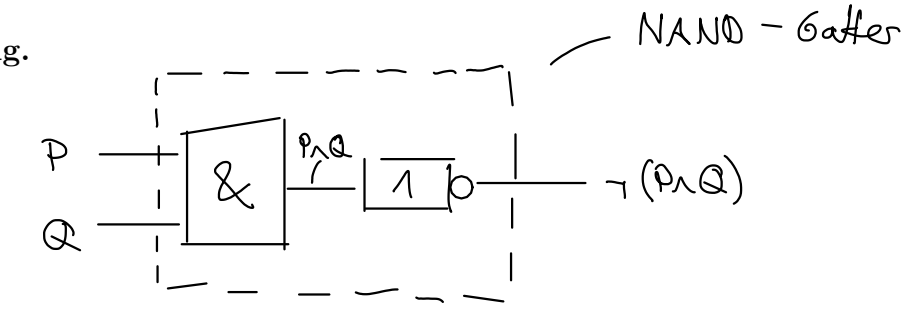
Logische Verknüpfung	Gatter (IEC)	Gatter (ANSI)
$\neg P$		
$P \wedge Q$		
$P \vee Q$		

Schaltkreise. Entwerfen Sie einen Schaltkreis für:

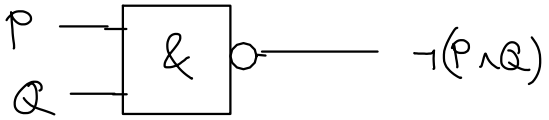
1. **NAND.** $\neg(P \wedge Q)$
2. $(P \wedge Q) \vee \neg P$

Lösung.

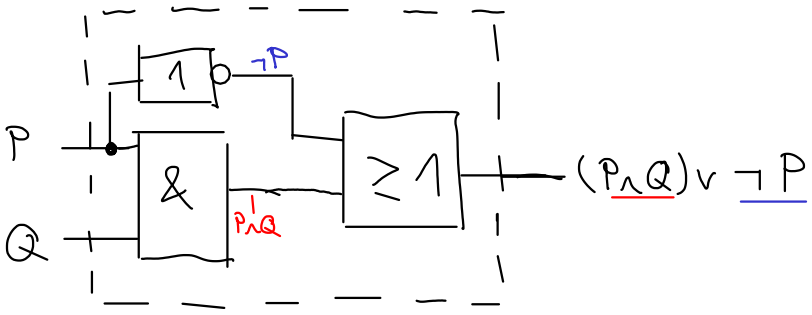
1.



Abkürzende Notation für NAND:



2.



Eigener Lösungsversuch.

Disjunktive Normalform (DNF)

Frage. Kann man zu jeder erdenklichen logischen Verknüpfung eine Schaltung bauen?
z.B. $P \oplus Q$, $P \Rightarrow Q$, ... z.B. $(P \oplus Q) \Rightarrow (P \wedge \neg Q)$ für $P \circ Q$

Antwort. Ja! Mittels der DNF

Satz (DNF). Man kann jede beliebige logische Verknüpfung $P \circ Q$ (mit zwei Parametern P, Q) äquivalent schreiben als:

P	Q	$P \circ Q$
0	0	$0 \circ 0$
0	1	$0 \circ 1$
1	0	$1 \circ 0$
1	1	$1 \circ 1$

Wahrheitswerte der Verknüpfung!

$$P \circ Q \Leftrightarrow (0 \circ 0 \wedge \neg P \wedge \neg Q) \vee (0 \circ 1 \wedge \neg P \wedge Q) \vee (1 \circ 0 \wedge P \wedge \neg Q) \vee (1 \circ 1 \wedge P \wedge Q)$$

← DNF

XOR-Gatter. Entwerfen Sie einen Schaltkreis von $P \oplus Q$.
Hinweis: Berechnen Sie dazu die DNF.

Lösung.

P	Q	$P \oplus Q$
0	0	0
0	1	1
1	0	1
1	1	0

$P \oplus Q \Leftrightarrow$ DNF

$$(0 \wedge \neg P \wedge \neg Q) \vee (1 \wedge \neg P \wedge Q) \vee (1 \wedge P \wedge \neg Q) \vee (0 \wedge P \wedge Q) \Leftrightarrow 0 \vee (\neg P \wedge Q) \vee (P \wedge \neg Q) \vee 0$$

$\Leftrightarrow (\neg P \wedge Q) \vee (P \wedge \neg Q)$ ← DNF

A	$0 \wedge A$
0	0
1	0

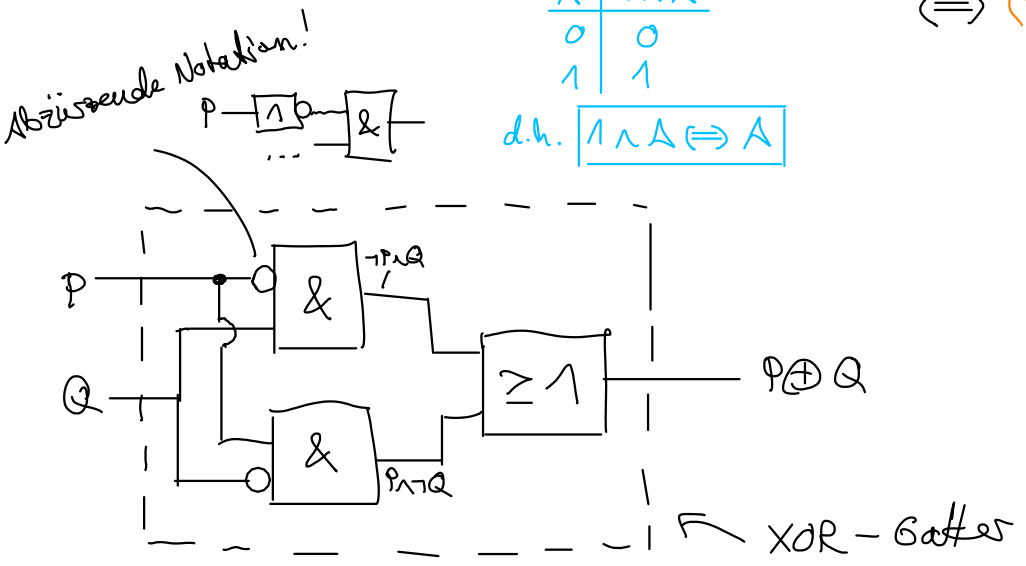
d.h. $0 \wedge A \Leftrightarrow 0$

A	$0 \vee A$
0	0
1	1

d.h. $0 \vee A \Leftrightarrow A$

A	$1 \wedge A$
0	0
1	1

d.h. $1 \wedge A \Leftrightarrow A$

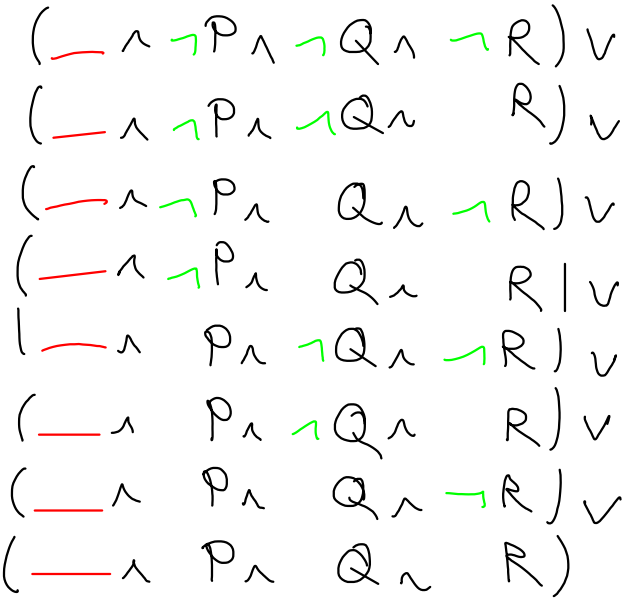


Eigener Lösungsversuch.

Dies geht auch für Verknüpfung mit beliebig vielen Parametern $P_1, P_2, \ldots P_n$ ($n >= 1$).

Satz (DNF) mit 3 Parametern. Man kann jede beliebige logische Verknüpfung $\circ(P, Q, R)$ (mit drei Parametern P, Q, R; oder analog auch beliebig Vielen) äquivalent schreiben als:

P	Q	R	$\circ(P,Q,R)$
0	0	0	$\circ(0,0,0)$
0	0	1	$\circ(0,0,1)$
0	1	0	$\circ(0,1,0)$
0	1	1	$\circ(0,1,1)$
1	0	0	$\circ(1,0,0)$
1	0	1	$\circ(1,0,1)$
1	1	0	$\circ(1,1,0)$
1	1	1	$\circ(1,1,1)$



Übung DNF mit 3 Parametern. → siehe Übungen.

Anwendung: Programmierung

Die Wahrheitswerte und die logischen Verknüpfungen gibt es auch in Programmiersprachen:

Wahrheitswert Logik	C	Java
falsch/false/0	0	false
wahr/true/1	$\neq 0$ z.B. 1, -2	true

In C wird der Datentyp int dazu missbraucht. Alle int $\neq 0$ werden als wahr interpretiert (manchmal hilfreich, aber meist gefährlich). In stdbool.h gibt es das Makro bool und die Konstanten true/false, um den Code besser lesbar zu gestalten (es wird damit klar, dass man die Zahl als Bool'schen Wert und nicht als Zahl benutzt).

In Java gibt es einen eigenen Datentyp boolean. Das ist gut so :-)

→ Demo in IDE

Logische Verknüpfung	C / Java
$\neg$	!
$\wedge$	&&
$\vee$	||

Beispiel Auswertung einer Aussage in C. Die Aussage $P \wedge Q$ mit $P : 5 = 4$ und $Q : 3 < 4$ wird in C wie folgt ausgewertet:

```
(5 == 4) && (3 < 4)
    0        1
    0
```

→ Demo in IDE

Nicht zu verwechseln mit den bitweisen Operationen `&` / `|`.

Beispiel Bitweises UND. Bei den Zahlen 1 und 2 als 32-bit `int` ergibt der Ausdruck `1&2` in C:

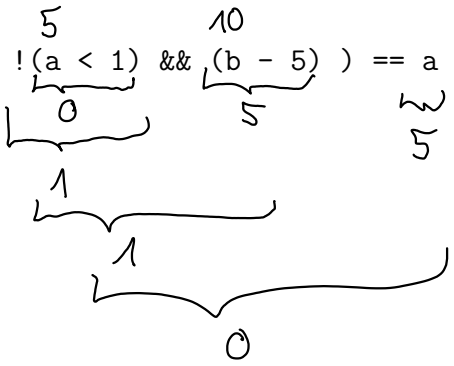
```
 1  = (0000 0000 0000 0000 0000 0000 0000 0001)
 2  = (0000 0000 0000 0000 0000 0000 0000 0010)
================================================
1&2 = (0000 0000 0000 0000 0000 0000 0000 0000) = 0
```

If-Bedingung auswerten. Was wird auf die Konsole ausgegeben?

```
int a = 5;
int b = 10;

if ( (!(a<1) && (b-5)) == a) {
   printf("A");
}
else {
   printf("B");
}
```

Lösung. Bedingung `( !(a < 1) && (b - 5) ) == a`



→ B wird ausgegeben!

Eigener Lösungsversuch. Bedingung `( !(a < 1) && (b - 5) ) == a`

If-Bedingung vereinfachen. Vereinfachen Sie die Aussage in der If-Bedingung:

```
int a;
int b;
...
if( a<10 && (a<10 || b==a-1) )
{
   ...
}
```

Lösung. Bedingung `a < 10 && (a < 10 || b == a-1)`

$P \wedge (P \vee Q)$ ($P \vee 0$)

$\Longleftrightarrow (P \vee 0) \wedge (P \vee Q) \overset{\text{Distr.}}{\Longleftrightarrow} P \vee \underbrace{(0 \wedge Q)}_{0} \Longleftrightarrow \underline{P}$ (Absorption)

$\rightsquigarrow$ `if(a<10) {...}`

Eigener Lösungsversuch. Bedingung `a < 10 && (a < 10 || b == a-1)`