1. Grundlagen der Informatik
Boolesche Algebra / Aussagenlogik

Inhalt

- Grundlagen digitaler Systeme
- **Boolesche Algebra / Aussagenlogik**
- Organisation und Architektur von Rechnern
- Algorithmen, Darstellung von Algorithmen mit Struktogrammen und Programmablaufplänen
- Zahlensysteme und interne Informationsdarstellung

Boolesche Algebra

Logikkalkül (George Boole, 1847)

Definition:

Eine Menge B von Elementen, über der zwei Operationen (+ und *) erklärt sind, ist genau dann eine Boolesche Algebra (B; +, *), wenn für beliebige Elemente a, b, c ϵ B folgende Axiome gelten:

(1) a+b = b+a
a*b = b*a

Kommutativität

(2) 0+a=a
1*a=a

Nullelement 0 bzgl. + und Einselement 1 bzgl. * existiert

(3) (a+b)*c = (a*c)+(b*c)
(a*b)+c = (a+c)*(b+c)

Distributivität einer Op. gegenüber der anderen Op.

(4) a+k(a)=1
a*k(a)=0

Zu jedem Element a ϵ B existiert ein komplementäres Element k(a)

Boolesche Algebra

Dualitätsprinzip:

Zu jeder Aussage, die sich aus den vier Axiomen der Booleschen Algebra herleiten lässt, existiert eine duale Aussage, die dadurch entsteht, dass man die Operationen + und * und gleichzeitig die neutralen Elemente 0 und 1 vertauscht.

Es gelten weiterhin:

(5) a+1 = 1
a*0 = 0

(6) a+a=a
a*a=a

Idempotenzgesetz

(7) a+(a*b) = a
a*(a+b) = a

Absorptionsgesetz

(8) k(a) + (a+b)=1
k(a)*(a*b) = 0

(9) a+(b+c) = (a+b)+c
a*(b*c) = (a*b)*c

Assoziativgesetz

Boolesche Algebra

Es gelten weiterhin (Fortsetzung):

(10) Für jedes a aus B existiert genau ein $\overline{a}$ aus B.
Wenn b = $\overline{a}$, $\overline{b}$ = a.

Negation der Negation

(11) $\overline{1} = 0$
$\overline{0} = 1$

(12) $\overline{(a+b)} = \overline{a} * \overline{b}$
$\overline{(a*b)} = \overline{a} + \overline{b}$

De Morgansches Gesetz

Die Boolesche Algebra legt noch keinen speziellen Anwendungsfall fest.

Alles, was aus Elementen und Operationen besteht, kann eine Boolesche Algebra sein, sofern die oben genannten Eigenschaften gelten.

Boolesche Algebra

Anwendung als:

- Mengenalgebra
- Schaltalgebra
- Aussagenlogik

Die Schaltalgebra und Aussagenlogik entsprechen einander bezüglich ihrer Variablen, Operationen und Gesetzmäßigkeiten.

Anwendung als Schaltalgebra

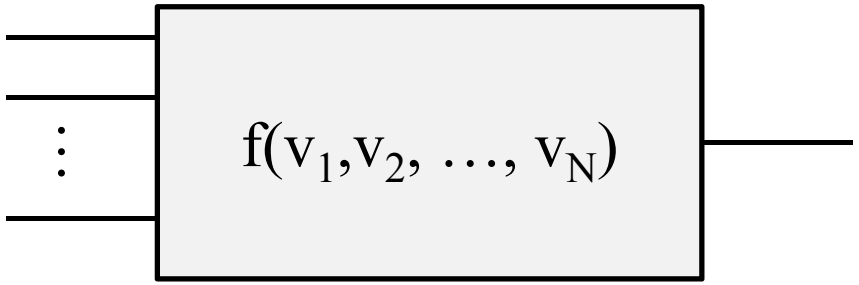
Schaltvariable

$v \in \{0,1\}$

Binäre Schaltfunktion:

$f: \{0,1\}^N \rightarrow \{0,1\}$

$z = f(v_1, v_2, \ldots, v_N); \quad z, v_i \in \{0,1\}$



Anwendung als Schaltalgebra

Definition einer Schaltfunktion durch eine Wahrheitstafel

V_N	V_{N-1}	…	V_2	V_1	f
0	0		0	0	0
0	0		0	1	1
0	0		1	0	0
…	…	…	…	…	
1	1		1	0	1
1	1		1	1	1

Hier nur eine spezielle Schaltfunktion gezeigt

N-stellige Funktion

$m=2^N$ Eingabemuster

2^m verschiedene N-stellige Schaltfunktionen

Anwendung als Schaltalgebra

Einstellige Schaltfunktionen durch eine Wahrheitstafel

a	Nullfunktion	Identität	Negation	Einsfunktion
0	0	0	1	1
1	0	1	0	1

1-stellige Funktion

$m=2^{1=2}$ Eingabemuster

$2^m = 2^2 = 4$ verschiedene einstellige Schaltfunktionen

Anwendung als Schaltalgebra

Zweistellige Schaltfunktionen durch eine Wahrheitstafel

a	b	f_0	UND	f_2	...	XOR	ODER	...	NOR	...	NAND	...
0	0	0	0	0		0	0	…	1	…	1	…
0	1	0	0	0		1	1		0		1	
1	0	0	0	1		1	1		0		1	
1	1	0	1	0		0	1	…	0	…	0	…

2-stellige Funktion

$m=2^2=4$ Eingabemuster

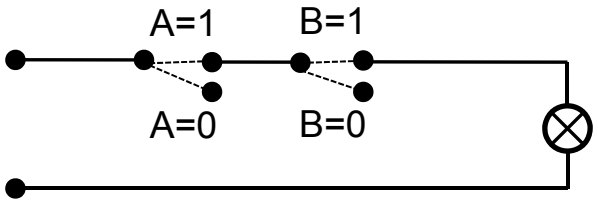
$2^m = 2^4 = 16$ verschiedene einstellige Schaltfunktionen

5 Funktionen hier als sinnvolle herausgestellt

Anwendung als Schaltalgebra

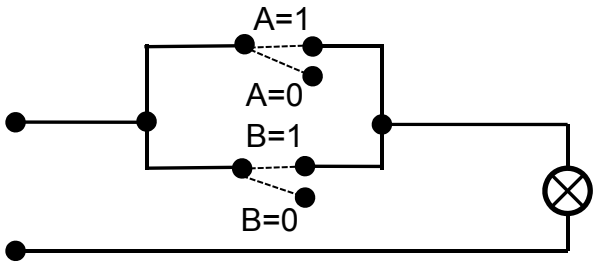
Ausgewählte Funktionen:

UND



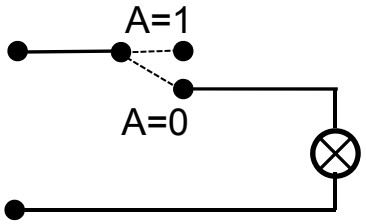
L=A UND B

ODER



L=A ODER B

NEGATION

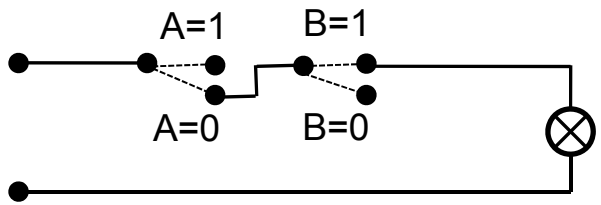


L= NICHT (A)

Anwendung als Schaltalgebra

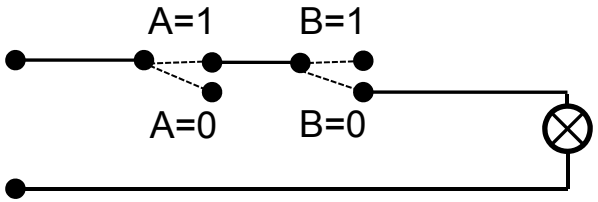
Ausgewählte Funktionen:

UND mit negiertem Operanden A



L=NICHT(A) UND B

UND mit negiertem Operanden B

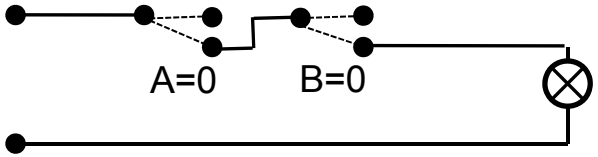


L=A UND NICHT(B)

Anwendung als Schaltalgebra

Ausgewählte Funktionen:

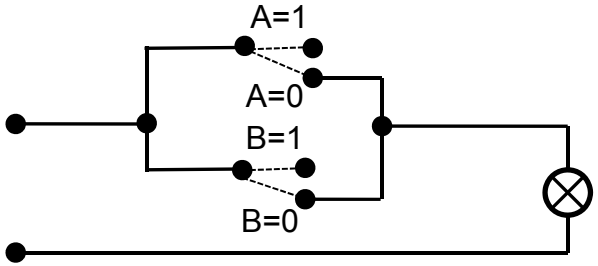
UND mit negierten Operanden A und B



L=NICHT(A) UND NICHT(B)
DE MORGAN:
L = NICHT(A ODER B)

NOR-Funktion (ODER, das am Ausgang negiert ist)

ODER mit negierten Operanden A und B



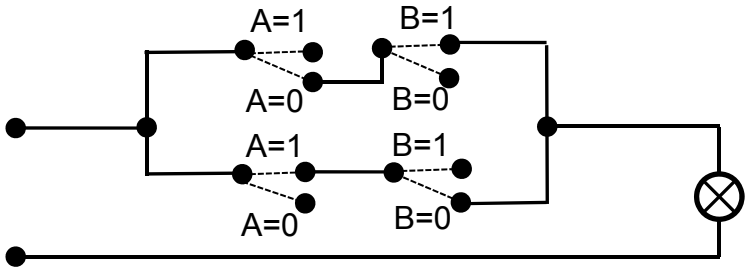
L=NICHT(A) ODER NICHT(B)
DE MORGAN:
L = NICHT (A UND B)

NAND-Funktion (UND, das am Ausgang negiert ist)

Anwendung als Schaltalgebra

Ausgewählte Funktionen:

XOR-Funktion (exklusives ODER)



L=(NICHT(A) UND B) ODER (A UND NICHT(B))
L= A XOR B

Schaltalgebra

Die Schaltalgebra ist eine spezielle Boolesche Algebra

B:= {0,1}

Nullelement: 0 … bedeutet Schalter geöffnet

Einselement: 1 … bedeutet Schalter geschlossen

Operation +: ODER

Operation *: UND

Negation : NICHT( )

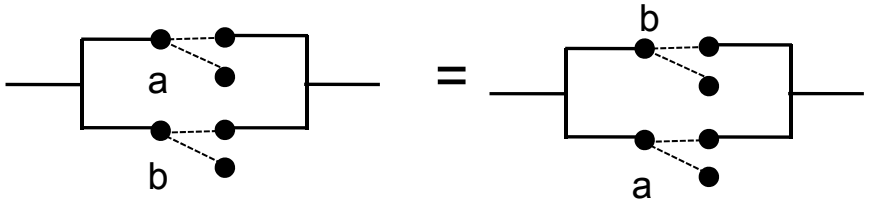
Es gelten die Axiome 1 bis 4 der Booleschen Algebra.

Die restlichen Regeln können hergeleitet werden.

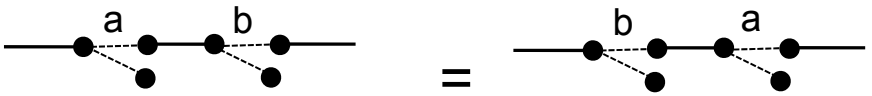
Schaltalgebra

Die Axiome der Booleschen Algebra gelten.

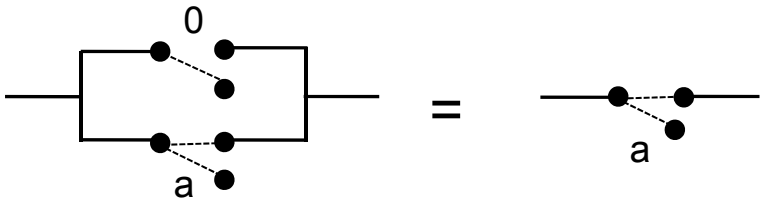
(1) a+b = b+a



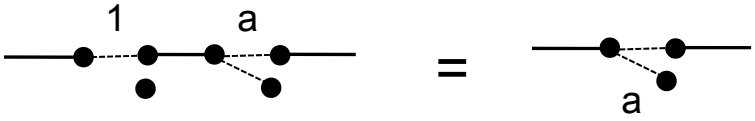
a*b = b*a



(2) 0+a=a



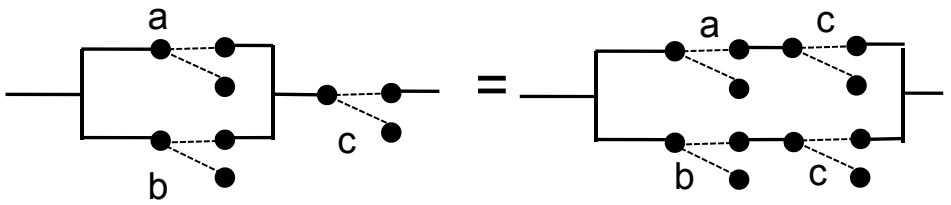
1*a=a



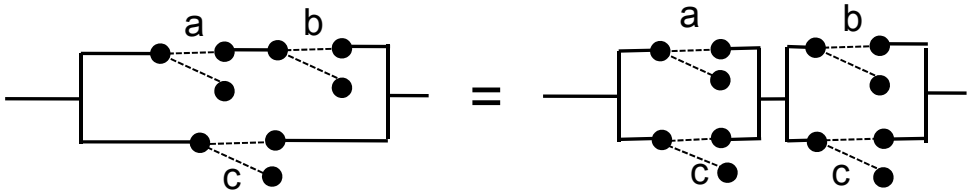
Schaltalgebra

Die Axiome der Booleschen Algebra gelten.

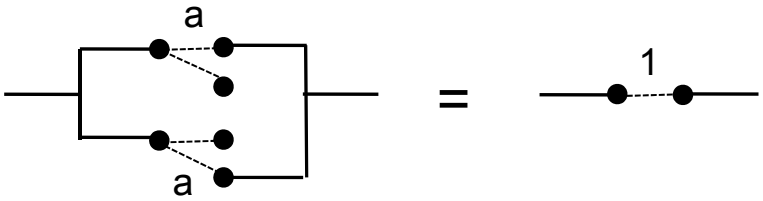
(3) (a+b)*c = (a*c)+(b*c)



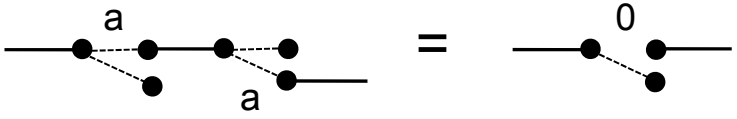
(a*b)+c = (a+c)*(b+c)



(4) a+k(a)=1



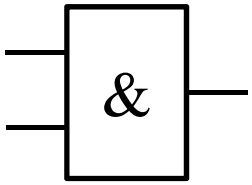
a*k(a)=0



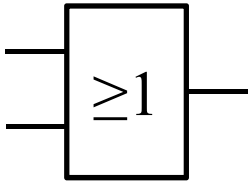
Schaltalgebra

Schaltsymbole für die gebräuchlichsten Funktionen:

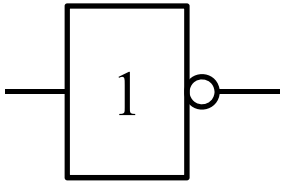
UND



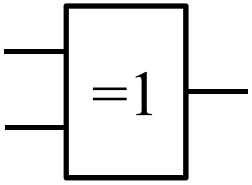
ODER



NEGATION



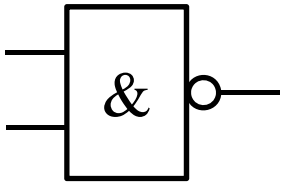
XOR



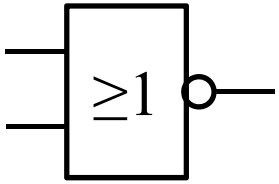
Schaltalgebra

Schaltsymbole

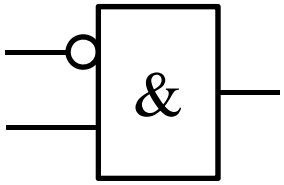
NAND



NOR



Negierte Eingänge,
Am Beispiel NICHT(a) UND b

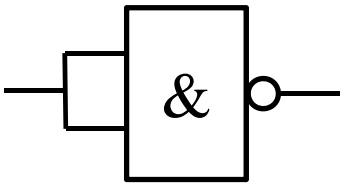


Schaltalgebra

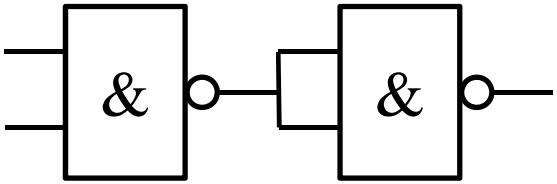
Vollständige Verknüpfungsbasis

Mit NAND lässt sich jede beliebige Schaltfunktion realisieren.

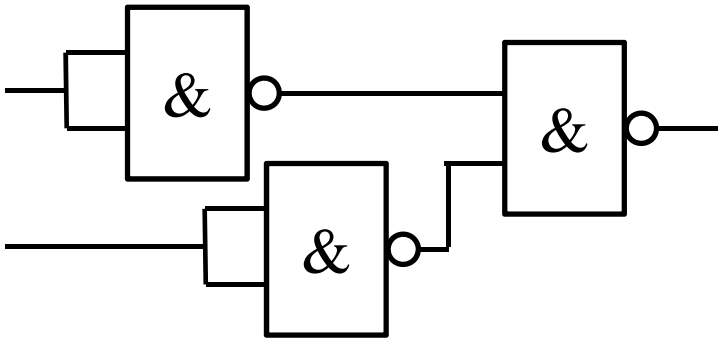
Negation



UND



ODER: A ODER B = NICHT (NICHT(A) UND NICHT(B))



Anwendung als Aussagenlogik

Aussagen sind formulierte

- Feststellungen, zum Beispiel „Tür geschlossen“
- Bedingungen, zum Beispiel x<5
- Relationen, wie a(i)<a(i+1)
- ‚berechnete‘ Aussagen, wie z.B. ‚2000 ist ein Schaltjahr‘

Aussagen …

- können den Ablauf (Steuerfluss) eines Programms beeinflussen
- können logisch verknüpft werden und ergeben neue Aussagen
- sind möglicherweise Eingabe oder Ausgabe von Programmen und werden durch Programme berechnet

Aussagenlogik

Die Aussagenlogik ist eine Boolesche Algebra.

Es gelten damit die Gesetze der Booleschen Algebra.

Spezialisierung zur Aussagenlogik:

- Werte: **falsch und wahr** als Entsprechung für 0 und 1
- Operationen:

Bezeich-nung			**Operator-Symbol**	**C-Operator**
Konjunktion	UND	AND	∧	&&
Disjunktion	ODER	OR	∨	||
Negation	NICHT	NOT	¯ (Überstrich)	!

Aussagenlogik

Definition der Operationen UND, ODER, NICHT
über Wahrheitstabellen

a UND b

a	b	a UND b
falsch	falsch	falsch
falsch	wahr	falsch
wahr	falsch	falsch
wahr	wahr	wahr

a ODER b

a	b	a ODER b
falsch	falsch	falsch
falsch	wahr	wahr
wahr	falsch	wahr
wahr	wahr	wahr

NICHT a

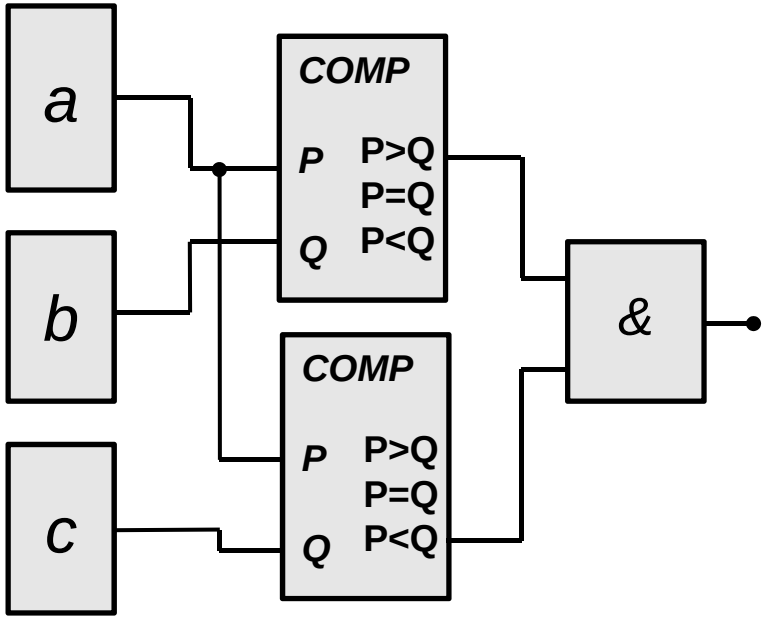
a	NICHT a
falsch	wahr
wahr	falsch

Vergleich zw. Aussagenlogik und Schaltalgebra

Aussagenlogik für Programme (Software)

```
…
if (a>b && a<c) {
  …
}
else {
  …
}
```

Schaltalgebra zur Realisierung von Schaltungen (Hardware)



Die Unterscheidung verschwindet, wenn Hardware direkt programmierbar wird, z.B. FPGAs

Bedingungs-Ausdrücke

Einzelne Aussagen können Bedingungsausdrücke sein

Ausage kalt: temperatur <10

Aussage zu_schnell: V >100

Bedingungsausdrücke werden durch aussagenlogische Operationen verknüpft.

Beispiele:

NICHT (zahl < 0) UND NICHT (zahl > 64)

zahl >= 0 UND zahl <= 64

NICHT (zahl <0 ODER zahl >64)

Alle drei Beispiele bedeuten das gleiche –sie drücken aus, dass zahl im Intervall zwischen 0 und 64 liegt

Eigenschaften (1)

a UND b = b UND a	Kommutativität bezüglich UND
a ODER b = b ODER a	Kommutativität bezüglich ODER
a UND wahr = a	Neutrales Element (wahr) bezüglich UND
a ODER falsch = a	Neutrales Element (falsch) bezüglich ODER
a UND k(a) = falsch	Komplementäres Element zu k(a): k(a) = NICHT a
a ODER k(a) = wahr	

Eigenschaften (2)

Verkettung von Operationen gleicher Art:

(a UND b) UND c = a UND (b UND c) = a UND b UND c

(a ODER b) ODER c = a ODER (b ODER c) = a ODER b ODER c

Verkettung unterschiedlicher Operationen:

(a UND b) ODER c = (a ODER c) UND (b ODER c)

a UND (b ODER c) = (a UND b) ODER (a UND c)

(a ODER b) UND c = (a UND c) ODER (b UND c)

a ODER (b UND c) = (a ODER b) UND (a ODER c)

Distributivität einer Operation gegenüber der anderen

Eigenschaften (3)

Verkettung unterschiedlicher Operationen:

(a UND b) ODER c ist nicht gleich a UND (b ODER c)

… bedeutet, dass sich für ausgewählte Werte von a,b und c unterschiedliche Ergebnisse der Ausdrücke ergeben können.

Beispiel:

(Autofahrt UND Stau) ODER Bahnverspätung

ist nicht gleich

Autofahrt UND (Stau ODER Bahnverspätung)

Der Ausdruck unten, würde zum Beispiel bei Bahnverspätung=wahr und Autofahrt=falsch einen anderen Wert als der Ausdruck oben ergeben.

Ein Gegenbeispiel reicht als Beweis für Ungleichheit im Allgemeinen

Eigenschaften (4)

Verkettung unterschiedlicher Operationen:

Wenn mehrere Operationen UND und ODER in einer aussagenlogischen Formel benutzt werden, müssen Klammern den Wirkungsbereich der Operatoren eindeutig festlegen.

Eigenschaften (5)

Weitere Regeln:

NICHT (a UND b) = (NICHT a) ODER (NICHT b)

De Morgansches Gesetz

NICHT (a ODER b) = (NICHT a) UND (NICHT b)

NICHT (NICHT a) = a

Negation der Negation

Weitere Operationen

Definition einer weiteren Operation mit Hilfe der anderen Operationen:

XOR (EXCLUSIVES ODER, deutsch: entweder oder)

a XOR b = (a UND (NICHT b)) ODER ((NICHT a) UND b)

FOLGERUNG a →b: aus a folgt b

a → b = (a UND b) ODER ((NICHT a) UND (NICHT b))
ODER ((NICHT a) UND b)

Regeln:

- wenn a wahr ist, dann muss auch b wahr sein.
- ist a falsch, dann darf b auch falsch sein
- es ist aber auch erlaubt, dass b wahr ist, wenn a falsch ist.

Aussagen und Bedingungen

Negation bezüglich Vergleichsoperatoren:

a:	x<5
NICHT a:	x>=5
b:	r>s
NICHT b:	r<=s
c:	i = j
NICHT c:	i ≠ j

Anwendung des De Morganschen Gesetzes:

a UND b UND c in einer Form als x<5 UND r>s UND i=j

kann umgeformt werden zu

NICHT (NICHT a ODER NICHT b ODER NICHT c) , das entspricht

NICHT(x>=5 ODER r<=s ODER i≠ j)

Minimierung aussagenlogischer Ausdrücke

Manchmal sind Ausdrücke lang und unübersichtlich …

Beispiel:
c = (a UND b) ODER (a UND (NICHT b)) ODER ((NICHT a) UND b)

Minimierung (Vereinfachung) durch Absorptionsgesetze

x UND (x ODER y) = x

x ODER (x UND y) = x

(x UND y) ODER (x UND (NICHT y) = x

(x ODER y) UND (x ODER (NICHT y)) = x

Das Beispiel oben ergibt eine einfachere Form:

c = a ODER ((NICHT a) UND b) =

(a ODER (NICHT a)) UND (a ODER b) = a ODER b