2.7 Belegungstabellen, Äquivalenz und disjunktive Normalform

Leitidee

Ein Term beschreibt eine Schaltung. Aber welche Werte liefert sie eigentlich? Das zeigt die **Belegungstabelle**. Spannend wird es, wenn man zwei Terme vergleicht: Oft sehen zwei Terme ganz verschieden aus und liefern trotzdem in jeder Zeile dasselbe. Dann beschreiben sie Schaltungen, die genau dasselbe tun – und man kann die billigere bauen. Umgekehrt hat man beim Entwurf oft nur eine Tabelle: „In diesen Fällen soll der Ausgang 1 sein." Mit der **disjunktiven Normalform** lernst du ein Verfahren, das zu jeder solchen Tabelle zuverlässig einen passenden Term liefert.

Lernziele

Nach diesem Abschnitt kannst du

- zu einem Term systematisch die Belegungstabelle erstellen,
- den Begriff der Äquivalenz zweier Terme erläutern,
- prüfen, ob zwei Terme äquivalent sind – von Hand und mit Python,
- aus einer Belegungstabelle die disjunktive Normalform (DNF) ablesen,
- die DNF zu einem gegebenen Term erstellen.

2.7.1 Belegungstabellen zu Termen

Eine Belegungstabelle enthält für jede mögliche Belegung der Variablen eine Zeile. Bei n Variablen gibt es 2^n Belegungen: bei zwei Variablen 4, bei drei Variablen 8, bei vier Variablen 16.

Damit keine Zeile fehlt, schreibst du die Belegungen immer in derselben Reihenfolge auf: Du **zählst im Binärsystem** von 000 bis 111. Dann wechselt die letzte Variable in jeder Zeile, die mittlere alle zwei Zeilen und die erste alle vier Zeilen. Die erste Spalte besteht also aus vier Nullen und vier Einsen, die zweite aus zwei Nullen, zwei Einsen, zwei Nullen, zwei Einsen.

Bei größeren Termen rechnet man nicht alles auf einmal aus. Man legt **Hilfsspalten** für die Teilterme an – und zwar in der Reihenfolge, in der man sie im Termbaum **von unten nach oben** findet. Für den Term $(A \wedge \neg B) \vee C$ sind das zuerst $\neg B$, dann $A \wedge \neg B$ und zuletzt der ganze Term:

A	B	C	$\neg B$	$A \wedge \neg B$	$(A \wedge \neg B) \vee C$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	1

Jede Hilfsspalte entsteht aus Spalten weiter links. Die Spalte $A \wedge \neg B$ zum Beispiel ist nur dort 1, wo **sowohl** in der Spalte A **als auch** in der Spalte $\neg B$ eine 1 steht. Die letzte Spalte ist 1, wo in der Spalte $A \wedge \neg B$ **oder** in der Spalte C eine 1 steht.

Tipp

Lies eine Zeile der Belegungstabelle immer als ganzen Satz. Die fünfte Zeile besagt: „Ist A mit 1, B mit 0 und C mit 0 belegt, dann hat der Term den Wert 1."

Arbeitsauftrag 77: Vom Term zur Belegungstabelle

Erstelle **von Hand** die vollständige Belegungstabelle zum Term

$$(A \vee B) \wedge \neg C.$$

Überprüfe dein Ergebnis anschließend mit einem Python-Programm.

A	B	C	$A \vee B$	$\neg C$	$(A \vee B) \wedge \neg C$
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

2.7.2 Äquivalenz: verschiedene Terme, gleiche Wirkung

Vergleiche die beiden Terme $A \vee (A \wedge B)$ und A. Sie sehen ganz verschieden aus. Schau dir aber ihre Belegungstabellen an:

A	B	$A \wedge B$	$A \vee (A \wedge B)$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Die letzte Spalte stimmt Zeile für Zeile mit der Spalte A überein! Egal, wie man A und B belegt: Beide Terme liefern immer denselben Wert. Man sagt: Die beiden Terme sind **äquivalent** (gleichwertig).

Für Schaltungen ist das sehr nützlich. Die Schaltung zu $A \vee (A \wedge B)$ braucht ein UND- und ein ODER-Gatter. Die Schaltung zu A braucht gar kein Gatter – der Ausgang ist einfach mit dem Eingang A verbunden. Beide Schaltungen verhalten sich genau gleich. Natürlich baut man dann die einfachere.

Definition: Äquivalenz

Zwei aussagenlogische Terme heißen **äquivalent**, wenn sie für **jede** mögliche Belegung der Variablen denselben Wert liefern, wenn also ihre Belegungstabellen übereinstimmen.

Beachte: „äquivalent“ ist ein Wort, mit dem wir **über** zwei Terme sprechen – so wie man über zwei Wörter sagen kann, dass sie gleich lang sind. Es ist kein neues Zeichen, das man in einen Term hineinschreibt wie $\wedge$ oder $\vee$. Wir schreiben deshalb einfach: „Die Terme $A \vee (A \wedge B)$ und A sind äquivalent.“

Merke

Zwei aussagenlogische Terme heißen **äquivalent**, wenn sie für **jede** mögliche Belegung denselben Wahrheitswert liefern – also **dieselbe Belegungstabelle** besitzen. Äquivalenz ist ein Begriff **über** Terme, kein eigener Operator!

Nicht äquivalent: Ein Gegenbeispiel genügt

Um zu zeigen, dass zwei Terme äquivalent sind, musst du **alle** Zeilen prüfen. Um zu zeigen, dass sie **nicht** äquivalent sind, reicht **eine einzige** Zeile, in der sie sich unterscheiden. Eine solche Zeile heißt **Gegenbeispiel**.

A	B	$A \vee B$	$A \oplus B$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

ODER und XOR stimmen in drei Zeilen überein, aber in der letzten Zeile nicht. Für $A = 1, B = 1$ ist $A \vee B = 1$, aber $A \oplus B = 0$. Also sind die beiden nicht äquivalent.

Äquivalenz mit Python prüfen

Bei drei oder vier Variablen ist der Vergleich von Hand mühsam. Python kann das schnell erledigen. Die Idee: Man geht alle Belegungen durch und vergleicht die Werte beider Terme. Am Anfang nimmt man an, dass die Terme äquivalent sind. Findet man eine Belegung, in der sie sich unterscheiden, merkt man sich das.

```python
def term_links(a, b):
    return a or (a and b)

def term_rechts(a, b):
    return a

aequivalent = True
for a in [0, 1]:
    for b in [0, 1]:
        if term_links(a, b) != term_rechts(a, b):
            aequivalent = False
            print(f"Unterschied bei A={a} B={b}")

print("Äquivalent:", aequivalent)
```

AUSGABE

```
Äquivalent: True
```

- **Zeilen 1–5:** Jeder Term wird als eigene Funktion geschrieben.
- **Zeile 7:** Die Variable `aequivalent` startet mit `True`: Solange kein Unterschied gefunden ist, gehen wir von Äquivalenz aus.
- **Zeilen 8–9:** Zwei geschachtelte Schleifen erzeugen alle vier Belegungen.
- **Zeilen 10–12:** Liefern die beiden Funktionen verschiedene Werte, wird `aequivalent` auf `False` gesetzt und das Gegenbeispiel ausgegeben.
- **Zeile 14:** Am Ende steht fest, ob es irgendeine Belegung mit einem Unterschied gab.

Hier wurde kein Unterschied gefunden, also gibt das Programm nur `True` aus.

Versuch: Probiere es aus

Ersetze in `term_links` das `or` durch `and`. Welche Gegenbeispiele findet das Programm jetzt? Prüfe auch: Ist `a and (a or b)` äquivalent zu `a`?

Arbeitsauftrag 78: Äquivalenz prüfen

Prüfe **mit Python**, ob die beiden folgenden Terme äquivalent sind. Vergleiche dazu für **alle** Belegungen, ob beide Funktionen stets denselben Wert liefern.

Term A: $\neg(A \wedge B)$ Term B: $\neg A \vee \neg B$

Ergänze das Programm:

```python
def term_a(a, b):
    return not (a and b)

def term_b(a, b):
    return (not a) or (not b)

aequivalent = True
for a in [0, 1]:
    for b in [0, 1]:
        # Vergleich einbauen und ggf. aequivalent = False setzen
        ...

print("Äquivalent:", aequivalent)
```

Erstelle zusätzlich die Belegungstabelle beider Terme von Hand.

Meine Tabelle:

Die Regeln von De Morgan

Die Äquivalenz aus dem letzten Auftrag ist so wichtig, dass sie einen eigenen Namen hat. Der englische Mathematiker **Augustus De Morgan** hat im 19. Jahrhundert zwei Regeln aufgeschrieben, die zeigen, wie man eine verneinte Klammer auflöst:

Merke

Regeln von De Morgan: Für alle Belegungen von A und B gilt

- $\neg(A \wedge B)$ ist äquivalent zu $\neg A \vee \neg B$ („nicht beide“ = „mindestens einer nicht“),
- $\neg(A \vee B)$ ist äquivalent zu $\neg A \wedge \neg B$ („keiner von beiden“ = „der eine nicht und der andere auch nicht“).

Merkregel: Zieht man das $\neg$ in die Klammer, wird **jede** Variable verneint, und aus $\wedge$ wird $\vee$ bzw. aus $\vee$ wird $\wedge$.

Ein Beispiel aus dem Alltag: „Es ist **nicht** so, dass es regnet **oder** windet“ bedeutet dasselbe wie „Es regnet **nicht und** es windet **nicht**“. Das ist die zweite Regel.

Für Schaltungen heißt das: Ein NAND-Gatter (UND mit verneintem Ausgang) leistet dasselbe wie ein ODER-Gatter, dessen beide Eingänge vorher verneint werden. Ein NOR-Gatter leistet dasselbe wie ein UND-Gatter mit zwei verneinten Eingängen. Mit den Regeln von De Morgan kann man also Gatter gegeneinander austauschen, ohne dass sich das Verhalten der Schaltung ändert.

Arbeitsauftrag 79: Die zweite Regel von De Morgan

Zeige mit einer Belegungstabelle, dass $\neg(A \vee B)$ und $\neg A \wedge \neg B$ äquivalent sind. Fülle dazu die Tabelle aus und begründe dein Ergebnis in einem Satz.

A	B	$A \vee B$	$\neg(A \vee B)$	$\neg A$	$\neg B$	$\neg A \wedge \neg B$
0	0					
0	1					
1	0					
1	1					

Begründung:

Stolperstein

Ein sehr häufiger Fehler ist, das $\neg$ einfach in die Klammer „hineinzuziehen“: $\neg(A \wedge B)$ ist **nicht** äquivalent zu $\neg A \wedge \neg B$. Gegenbeispiel: Für $A = 1$, $B = 0$ ist $\neg(1 \wedge 0) = \neg 0 = 1$, aber $\neg 1 \wedge \neg 0 = 0 \wedge 1 = 0$. Wer das $\neg$ in die Klammer zieht, muss das $\wedge$ in ein $\vee$ umwandeln (und umgekehrt).

2.7.3 Die disjunktive Normalform (DNF)

Beim Schaltungsentwurf steht am Anfang oft eine Belegungstabelle: In diesen Fällen soll der Ausgang 1 sein, in den anderen 0. Wie kommt man von der Tabelle zu einem Term? Dafür gibt es ein Verfahren, das **immer** funktioniert.

Die Idee: ein Steckbrief für jede 1-Zeile

Schau dir diese Tabelle an:

A	B	**Ausgang**
0	0	0
0	1	1
1	0	1
1	1	0

Der Ausgang ist in zwei Zeilen 1. Für jede dieser Zeilen schreiben wir einen „Steckbrief“: eine Konjunktion, die **genau in dieser einen Zeile** den Wert 1 hat und sonst überall 0.

- Zeile $A = 0$, $B = 1$: Wir brauchen „A ist 0 und B ist 1“. „A ist 0“ heißt dasselbe wie „$\neg A$ ist 1“. Der Steckbrief lautet also $\neg A \wedge B$.
- Zeile $A = 1$, $B = 0$: Der Steckbrief lautet $A \wedge \neg B$.

Der Ausgang soll 1 sein, wenn wir in der **einen oder** in der **anderen** Zeile sind. Also verbinden wir die Steckbriefe mit $\vee$:

$$(\neg A \wedge B) \vee (A \wedge \neg B)$$

Das ist die **disjunktive Normalform** der Tabelle. Übrigens ist die Tabelle genau die von XOR. Wir haben also nebenbei herausgefunden, wie man XOR mit $\wedge$, $\vee$ und $\neg$ schreibt.

Definition: Minterm und disjunktive Normalform

Ein **Minterm** ist eine Konjunktion, in der **jede** Variable genau einmal vorkommt, entweder ohne oder mit $\neg$. Er hat genau in einer Zeile der Belegungstabelle den Wert 1.

Die **disjunktive Normalform (DNF)** einer Belegungstabelle ist die Disjunktion ($\vee$) der Minterme aller Zeilen, in denen das Ergebnis 1 ist.

Der Name erklärt sich so: **disjunktiv**, weil die Minterme mit $\vee$ (Disjunktion) verbunden werden; **Normalform**, weil jeder Term auf diese feste, „normale“ Form gebracht werden kann.

Merke

Die **disjunktive Normalform (DNF)** ist eine standardisierte Darstellung eines Terms:

- Für **jede** Zeile der Belegungstabelle, in der der Ausgang 1 ist, bildet man einen **Minterm**: die Konjunktion aller Variablen, wobei Variablen mit Wert 0 negiert werden.
- Alle Minterme werden anschließend **disjunktiv** (mit $\vee$) verknüpft.

Beispiel: Ist der Ausgang bei $A = 1, B = 0$ und bei $A = 0, B = 1$ gleich 1 (sonst 0), lautet die DNF $(A \wedge \neg B) \vee (\neg A \wedge B)$ – das ist genau XOR!

Ob man die Minterme in der Reihenfolge $(\neg A \wedge B) \vee (A \wedge \neg B)$ oder $(A \wedge \neg B) \vee (\neg A \wedge B)$ aufschreibt, ist egal. Bei $\vee$ kommt es auf die Reihenfolge nicht an.

Mit Python lässt sich prüfen, dass die DNF wirklich die Tabelle von oben liefert:

```python
def dnf_beispiel(a, b):
    return int((not a and b) or (a and not b))

for a in [0, 1]:
    for b in [0, 1]:
        print(f"A={a} B={b} -> {dnf_beispiel(a, b)}")
```

AUSGABE

```
A=0 B=0 -> 0
A=0 B=1 -> 1
A=1 B=0 -> 1
A=1 B=1 -> 0
```

Die Klammern um `not a and b` sind in Python eigentlich nicht nötig, weil `and` stärker bindet als `or`. Sie machen die Funktion aber leichter lesbar.

Warum funktioniert das?

Jeder Minterm ist in **genau einer** Zeile 1 – nämlich in der Zeile, aus der er gebaut wurde. In jeder anderen Zeile stimmt mindestens eine Variable nicht, und die Konjunktion wird 0. Die Disjunktion aller Minterme ist deshalb genau in den Zeilen 1, aus denen ein Minterm gebaut wurde – und das sind genau die 1-Zeilen

der Tabelle. Die DNF hat also dieselbe Belegungstabelle wie der Ausgang. Wenn die Tabelle zu einem Term gehört, ist die DNF zu diesem Term **äquivalent**.

Das Verfahren Schritt für Schritt

1. Suche in der Belegungstabelle alle Zeilen, in denen das Ergebnis 1 ist. Die Zeilen mit 0 brauchst du nicht.
2. Schreibe für jede dieser Zeilen den Minterm auf: alle Variablen mit $\wedge$ verbunden; eine Variable, die in der Zeile 0 ist, bekommt ein $\neg$.
3. Setze um jeden Minterm Klammern und verbinde alle Minterme mit $\vee$.

Beispiel

DNF der Alarmanlage. Wir kürzen ab: T für Tür, S für Scharf, R für Rauch. Der Term lautet $(T \wedge S) \vee R$. Zuerst die Belegungstabelle, dann für jede 1-Zeile der Minterm:

T	S	R	$(T \wedge S) \vee R$	**Minterm**
0	0	0	0	
0	0	1	1	$\neg T \wedge \neg S \wedge R$
0	1	0	0	
0	1	1	1	$\neg T \wedge S \wedge R$
1	0	0	0	
1	0	1	1	$T \wedge \neg S \wedge R$
1	1	0	1	$T \wedge S \wedge \neg R$
1	1	1	1	$T \wedge S \wedge R$

Die DNF lautet:

$$(\neg T \wedge \neg S \wedge R) \vee (\neg T \wedge S \wedge R) \vee (T \wedge \neg S \wedge R) \vee (T \wedge S \wedge \neg R) \vee (T \wedge S \wedge R)$$

Sie ist äquivalent zu $(T \wedge S) \vee R$, aber viel länger. Im nächsten Abschnitt lernst du, wie man eine DNF wieder vereinfacht.

Stolperstein

- **Negation vertauscht:** Eine Variable bekommt ein $\neg$, wenn sie in der Zeile **0** ist – nicht, wenn sie 1 ist.
- **Variable vergessen:** In jedem Minterm müssen **alle** Variablen vorkommen. $\neg T \wedge R$ ist kein Minterm, wenn es drei Variablen gibt.
- **Falsche Zeilen:** Nur die Zeilen mit Ergebnis **1** werden verwendet.
- **Falsche Zeichen:** Innerhalb eines Minterms steht $\wedge$, zwischen den Mintermen $\vee$ – nicht umgekehrt.

So kannst du es sagen

So sprichst du über Belegungstabellen und Terme – vom einfachen zum genauen Satz:

- *„Die Tabelle passt zum Term."* → *„Die Belegungstabelle gehört zum Term."*
- *„In Zeile 5 kommt 1 raus."* → *„Die fünfte Zeile der Belegungstabelle besagt: Ist A mit 1 und sind B und C mit 0 belegt, hat der Term den Wert 1."*
- *„Die beiden Terme sind gleich."* → *„Die beiden Terme sind äquivalent, denn ihre Belegungstabellen stimmen überein."*

Arbeitsauftrag 80: DNF ablesen

Gegeben ist folgende Belegungstabelle:

A	B	C	**Ausgang**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

1. Bestimme die DNF auf Papier.
2. Schreibe die DNF als Python-Funktion `meine_dnf(a, b, c)` und überprüfe sie mit drei geschachtelten Schleifen gegen die Tabelle.

Meine DNF:

Arbeitsauftrag 81: DNF zu einem Term

Bestimme die DNF zum Term $(A \vee B) \wedge \neg C$ aus dem Auftrag *Vom Term zur Belegungstabelle*. Benutze deine Belegungstabelle von dort.

Meine DNF:

2.7.4 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Arbeitsauftrag 82: Belegungstabelle mit zwei Variablen ●○○

Erstelle die Belegungstabelle zum Term $\neg A \vee B$. Benutze die Hilfsspalte.

A	B	$\neg A$	$\neg A \vee B$
0	0		
0	1		
1	0		
1	1		

Arbeitsauftrag 83: Minterme aufschreiben ●○○

1. Schreibe zu jeder Belegung den Minterm auf (drei Variablen A, B, C).

   a) $A = 1, B = 0, C = 1$ b) $A = 0, B = 0, C = 0$ c) $A = 0, B = 1, C = 1$

2. Jetzt umgekehrt: Bei welcher Belegung hat der Minterm $\neg A \wedge B \wedge \neg C$ den Wert 1?

Deine Antworten:

Arbeitsauftrag 84: DNF mit zwei Variablen ●○○

Bestimme die DNF zu dieser Belegungstabelle. Beschreibe in Worten, wann der Ausgang 1 ist.

A	B	**Ausgang**
0	0	1
0	1	0
1	0	0
1	1	1

DNF und Beschreibung:

Gymnasium Ottweiler

Arbeitsauftrag 85: DNF mit drei Variablen ●○○

Bestimme die DNF zu dieser Belegungstabelle.

A	B	C	**Ausgang**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

DNF:

Arbeitsauftrag 86: Äquivalent oder nicht? ●●○

Prüfe mit Belegungstabellen, ob die beiden Terme äquivalent sind. Wenn nicht, gib eine Belegung an, in der sie sich unterscheiden.

a) $A \vee (\neg A \wedge B)$ und $A \vee B$
b) $\neg A \wedge \neg B$ und $\neg(A \wedge B)$

Deine Tabellen und deine Antwort:

Gymnasium Ottweiler

Arbeitsauftrag 87: Vom Term zur DNF ●●○

Gegeben ist der Term $(A \wedge \neg B) \vee (B \wedge C)$.

1. Erstelle die Belegungstabelle mit Hilfsspalten.
2. Bestimme die DNF.

A	B	C	$\neg B$	$A \wedge \neg B$	$B \wedge C$	**Term**
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

DNF:

Arbeitsauftrag 88: Äquivalenz mit Python prüfen ●●○

Prüfe mit Python, ob die Terme äquivalent sind. Wenn nicht, soll das Programm alle Belegungen ausgeben, in denen sie sich unterscheiden.

a) $A \wedge (B \vee C)$ und $(A \wedge B) \vee (A \wedge C)$
b) $A \vee (B \wedge C)$ und $(A \vee B) \wedge C$

Tipp

Schreibe jeden Term als eigene Funktion. Gehe dann vor wie im Abschnitt *Äquivalenz mit Python prüfen*, nur mit drei geschachtelten Schleifen.

Arbeitsauftrag 89: Warnleuchte im Auto ●●●

Eine Warnleuchte im Auto ($L = 1$) leuchtet, wenn der Motor läuft ($M = 1$) und der Gurt nicht angelegt ist ($G = 0$). Sie leuchtet außerdem, wenn der Motor läuft und eine Tür offen ist ($T = 1$).

1. Stelle die Belegungstabelle für M, G, T auf.
2. Bestimme die DNF.
3. Mia hat den kürzeren Term $M \wedge (\neg G \vee T)$ gefunden. Prüfe mit Python, ob er zu deiner DNF äquivalent ist.

Tabelle und DNF:

2.7.5 Das Wichtigste in Kürze

- Eine **Belegungstabelle** hat bei n Variablen 2^n Zeilen. Die Belegungen werden wie Binärzahlen von $0 \ldots 0$ bis $1 \ldots 1$ aufgeschrieben. Hilfsspalten für Teilterme erleichtern das Ausrechnen.
- Zwei Terme sind **äquivalent**, wenn ihre Belegungstabellen übereinstimmen. Äquivalenz ist ein Begriff über Terme, kein Operator.
- Für den Nachweis der Äquivalenz prüft man alle Zeilen. Für den Nachweis, dass Terme nicht äquivalent sind, genügt ein **Gegenbeispiel**.
- **Regeln von De Morgan:** $\neg(A \wedge B)$ ist äquivalent zu $\neg A \vee \neg B$, und $\neg(A \vee B)$ ist äquivalent zu $\neg A \wedge \neg B$.
- Ein **Minterm** ist eine Konjunktion aller Variablen (Variablen mit Wert 0 negiert). Er ist genau in einer Zeile 1.
- Die **DNF** ist die Disjunktion der Minterme aller 1-Zeilen. Sie ist zum ursprünglichen Term äquivalent, aber oft lang.

Selbstcheck

- ☐ Ich kann zu einem Term mit Hilfsspalten die Belegungstabelle erstellen.
- ☐ Ich kann erklären, wann zwei Terme äquivalent sind.
- ☐ Ich kann mit einem Gegenbeispiel zeigen, dass zwei Terme nicht äquivalent sind.
- ☐ Ich kann die Regeln von De Morgan nennen und mit einer Belegungstabelle begründen.
- ☐ Ich kann mit Python prüfen, ob zwei Terme äquivalent sind.
- ☐ Ich kann aus einer Belegungstabelle die DNF ablesen.
- ☐ Ich kann die DNF zu einem gegebenen Term bestimmen.