

**Länge von Formeln.** Die Kostenfunktionen für Algorithmen, die als Eingabe eine aussagenlogische Formel haben, werden üblicherweise in Abhängigkeit der Formellänge erstellt, oftmals auch als zweistellige Funktion  $T(n, m)$ , wobei  $n$  für die Anzahl an Atomen in der Eingabeformel und  $m$  für die Länge der Eingabeformel steht.

Ist  $\alpha$  eine aussagenlogische Formel, so bezeichnet  $|\alpha|$  die *Länge* von  $\alpha$ . Diese ist durch die Anzahl an in  $\alpha$  vorkommenden Operatoren definiert. Für unsere Wahl der Syntax der Aussagenlogik mit den Basisoperatoren  $\neg$  und  $\wedge$  bezeichnet  $|\alpha|$  die Anzahl an Vorkommen von  $\neg$  und  $\wedge$  in  $\alpha$ . Dies entspricht folgenden Rekursionsformeln für die Länge von aussagenlogischen Formeln. Wir setzen  $|true| = |x| = 0$  für alle  $x \in AP$  und

$$\begin{aligned} |\neg\alpha| &= |\alpha| + 1 \\ |\alpha \wedge \beta| &= |\alpha| + |\beta| + 1 \end{aligned}$$

Die abgeleitete Konstante *false* (die durch  $\neg true$  definiert wurde) hat also die Länge 1, während  $\neg(x \wedge \neg y)$  die Länge 3 hat (2 Negationen, 1 Konjunktion). Man beachte, dass die Länge einer Formel des Typs  $\bigwedge_{1 \leq i \leq n} \alpha_i$  gleich  $n-1$  ist (und nicht etwa 1). Für den Disjunktionsoperator ergibt sich

$$|\alpha \vee \beta| = |\neg(\neg\alpha \wedge \neg\beta)| = |\alpha| + |\beta| + 4,$$

wobei die Konstante “4” für “3 Negationen plus 1 Konjunktion” steht. Damit ergibt sich für den Implikationsoperator:

$$|\alpha \rightarrow \beta| = |\neg\alpha \vee \beta| = (|\alpha| + 1) + |\beta| + 4 = |\alpha| + |\beta| + 5$$

Die so definierte Länge ist nützlich für induktive Argumente. Das Prinzip der strukturellen Induktion für aussagenlogische Formeln entspricht genau einer gewöhnlichen Induktion nach der Länge der Formeln. Für algorithmische Betrachtungen sind wir jedoch nur an der *asymptotischen Länge* von Formeln interessiert. Die konkreten Werte der additiven Konstanten (z.B. “1” für die Konjunktion, “4” für Disjunktion, “5” für die Implikation) spielen keine Rolle. Daher ist es für die asymptotische Länge irrelevant, ob man eine Darstellung von  $\alpha$  mit den Basisoperatoren  $\wedge$  und  $\neg$  fordert und den beiden Basisoperatoren jeweils eine Längeneinheit zuordnet und für die abgeleiteten Operatoren  $\vee$  und  $\rightarrow$  4 bzw. 5 Längeneinheiten veranschlagt werden, *oder* ob man Darstellungen mit den Operatoren  $\wedge, \vee, \neg, \rightarrow$  zulässt und jedem dieser Operatoren nur eine Längeneinheit zuordnet. Ebenso ist es für die asymptotische Länge unerheblich, ob man der Konstanten *true* und den atomaren Formeln  $x \in AP$  null Längeneinheiten oder eine Längeneinheit zuweist. Vorsicht ist jedoch mit anderen abgeleiteten Operatoren wie  $\leftrightarrow$  oder  $\oplus$  geboten. Wir betrachten hier exemplarisch den Paritätsoperator. Für diesen gilt:

$$|\alpha \oplus \beta| = |(\neg\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta)| = 2|\alpha| + 2|\beta| + \text{const},$$

wobei “const” gemäß der oben angegebenen Definition der Formellänge für die Zahl 8 steht (jeweils 1 Kosteneinheit für die beiden Negationen und die beiden Konjunktionen und 4 Kosteneinheiten für die Disjunktion). Veranschlagt man eine Längeneinheit für jeden vorkommenden Operator, so ergibt sich der Wert 5 für “const” (2 Negationen, 2 Konjunktionen, 1 Disjunktion). In beiden Fällen gilt:

$$|\alpha \oplus \beta| > 2|\alpha| + 2|\beta|$$

Betrachte nun die Formeln  $\alpha_n = x_1 \oplus x_2 \oplus \dots \oplus x_n$  für  $n \geq 2$ .

<sup>14</sup> Für die Formellänge der  $\alpha_n$ 's gilt:

$$|\alpha_n| \geq 2|\alpha_{n-1}| \geq 4|\alpha_{n-2}| \geq \dots \geq 2^{n-2}|\alpha_2| \geq 2^{n-2}$$

Die Länge von  $\alpha_n$  wächst also exponentiell, obwohl mit der Schreibweise  $x_1 \oplus x_2 \oplus \dots \oplus x_n$  nur  $2n-1$  Zeichen verwendet werden und man daher lineare Formellänge vermuten könnte.

#### 4.1.2 Normalformen

Für jede Art von Logik spielen Normalformen eine wichtige Rolle. Diese machen gewisse syntaktische Einschränkungen und dienen oftmals als Ausgangspunkt für Algorithmen oder können auch Beweise vereinfachen. Wir betrachten hier die positive Normalform, die nur eine eingeschränkte Nutzung der Negation erlaubt sowie zweistufige Normalformen (konjunktive und disjunktive Normalform). Im Kontext dieser Normalformen sind die Konjunktionen und Disjunktionen "gleichberechtigt", d.h., wir betrachten beide  $\wedge$  und  $\vee$  als Basisoperatoren.

**Literal.** Ein *Literal* ist eine Formel der Art  $x$  oder  $\neg x$ , wobei  $x$  ein Atom ist. Literale der Form  $x$  heißen *positiv*; Literale der Form  $\neg x$  werden *negativ* genannt. Die Literale  $x$  und  $\neg x$  sind *komplementär* zueinander. Ist  $L$  ein Literal, so schreiben wir  $\bar{L}$  für die Komplementierung des Literals:

$$\bar{L} \stackrel{\text{def}}{=} \begin{cases} \neg x & : \text{ falls } L = x \text{ ein positives Literal ist} \\ x & : \text{ falls } L = \neg x \text{ ein negatives Literal ist.} \end{cases}$$

Offenbar gilt  $\bar{\bar{L}} \equiv L$ .

#### Positive Normalform (PNF)

In der positiven Normalform (kurz PNF), manchmal auch *Negationsnormalform* genannt, ist nur eine eingeschränkte Nutzung des Negationsoperators zulässig ist. Formeln in PNF verwenden den Negationsoperator nur auf der Ebene der Literale. Um dieselbe Ausdrucksstärke wie die Aussagenlogik zu garantieren, wird neben dem Konjunktionsoperator  $\wedge$  der hierzu duale Operator  $\vee$  (der Disjunktionsoperator) benötigt. D.h., aussagenlogische Formeln in positiver Normalform (PNF) über der Aussagenmenge  $AP$  werden durch Konjunktionen und Disjunktionen aus den Konstanten *true* und *false*, sowie den Literalen  $x$  und  $\neg x$  für  $x \in AP$  gebildet. Die abstrakte Syntax von PNF-Formeln ist wie folgt:

$$\alpha ::= \text{true} \mid \text{false} \mid x \mid \neg x \mid \alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2 \quad (\text{Formeln in PNF})$$

Beispielsweise ist  $z \wedge (\neg x \vee (\neg y \wedge z)) \wedge (x \vee \neg z)$  in PNF, während  $\neg(x \vee \neg y)$  oder  $\neg x \vee \neg \neg y$  nicht in PNF sind. Man beachte, dass abgeleitete Operatoren, wie die Implikation, Parität oder Äquivalenz, nicht ohne weiteres in PNF-Formeln eingesetzt werden können. Z.B. ist  $(x \wedge z) \rightarrow y$  nicht in PNF, obwohl auf den ersten Blick nur positive Literale "sichtbar" sind. Hierzu muss man sich klarmachen, dass  $(x \wedge z) \rightarrow y$  lediglich eine abkürzende Schreibweise für  $\neg(x \wedge z) \vee$

<sup>14</sup>Der Paritätsoperator ist assoziativ, was die klammerlose Schreibweise rechtfertigt. Formal können die Formeln  $\alpha_n$  induktiv definiert werden, etwa  $\alpha_2 = x_1 \oplus x_2$  und  $\alpha_n = \alpha_{n-1} \oplus x_n$  für  $n \geq 3$ .

$y$  ist; eine Formel, die offenbar nicht in PNF ist. Für die asymptotische Länge einer PNF-Formel reicht es in  $\alpha$  die Anzahl an Vorkommen der Operatoren  $\wedge$  und  $\vee$  zu zählen.

Jede aussagenlogische Formel  $\alpha$  kann in eine äquivalente PNF-Formel derselben asymptotischen Länge transformiert werden. Hierzu sind lediglich die Dualität zwischen Konjunktion und Disjunktion (die De Morgan'schen Regeln), die Dualität der Konstanten *true* und *false* sowie das Gesetz der doppelten Verneinung auszunutzen, um alle Negationen "nach innen" zu ziehen. Geht man von einer aussagenlogischen Formel  $\alpha$  aus, die nur die in der primären Syntax angegebenen Operatoren  $\neg$  und  $\wedge$  (aber keine abgeleiteten Operatoren) verwendet, so erhält man durch sukzessives Anwenden der nachstehenden Transformationsregeln ("von außen nach innen") eine zu  $\alpha$  äquivalente PNF-Formel:

$$\begin{aligned} \neg true & \rightsquigarrow false \\ \neg\neg\gamma & \rightsquigarrow \gamma \\ \neg(\gamma_1 \wedge \gamma_2) & \rightsquigarrow \neg\gamma_1 \vee \neg\gamma_2 \end{aligned}$$

Die Hinzunahme des Disjunktionsoperators als Basisoperator ist problemlos, da dann lediglich  $\neg(\gamma_1 \vee \gamma_2) \rightsquigarrow \neg\gamma_1 \wedge \neg\gamma_2$  als zusätzliche Transformationsregel hinzugefügt werden muss. Werden obige Regeln solange wie möglich eingesetzt, so ist klar, dass der Negationsoperator nur noch auf Ebene der Literale vorkommt und die resultierende Formel  $\beta$  somit in PNF ist. Wendet man die obigen Transformationsregeln stets "von außen nach innen" an, also nur auf solche Teilformeln  $\neg\gamma$ , die nicht Teilformel einer anderen Teilformel  $\neg\delta$  sind, so kann die Anzahl an Transformationsschritten durch die Anzahl an Teilformeln von  $\alpha$  nach oben abgeschätzt werden. (Genauer gilt: die Anzahl an vorgenommenen Transformationsschritten ist höchstens  $|\alpha|$  plus Anzahl an Vorkommen der Konstanten *true* in  $\alpha$ .) Mit der Anzahl an Transformationsschritten ist auch die Länge von  $\beta$  in  $\mathcal{O}(|\alpha|)$ . Dies ist wie folgt einsichtig. Wie oben erwähnt reicht es für die asymptotische Länge einer PNF-Formel die Anzahl an Disjunktionen und Konjunktionen zu zählen. Die ersten beiden Transformationsregeln entfernen eine bzw. zwei Negationen, die dritte Regel ersetzt eine Negation und eine Konjunktion durch zwei Negationen und eine Disjunktion. In allen drei Fällen bleibt die Anzahl an Konjunktionen und Disjunktionen unverändert. Also ist die asymptotische Länge von  $\beta$  gleich der Anzahl an Konjunktionen plus Disjunktionen von  $\alpha$ . Diese ist durch  $|\alpha|$  beschränkt. Somit gilt  $|\beta| = \mathcal{O}(|\alpha|)$ .

Die Erstellung einer äquivalenten PNF-Formel durch sukzessives Anwenden der Transformationsregeln illustrieren wir am Beispiel der aussagenlogischen Formel

$$\alpha = \neg((x \vee \neg z) \wedge \neg(x \wedge \neg y)).$$

Wir wenden sukzessive die De Morganschen Regeln und die Regel zur doppelten Verneinung an, um die Negationen vollständig nach innen zu ziehen und redundante Negationen aufzulösen:

$$\begin{aligned} \alpha &= \neg((x \vee \neg z) \wedge \neg(x \wedge \neg y)) \\ &\equiv \neg(x \vee \neg z) \vee \neg\neg(x \wedge \neg y) \\ &\equiv \neg(x \vee \neg z) \vee (x \wedge \neg y) \\ &\equiv (\neg x \wedge \neg\neg z) \vee (x \wedge \neg y) \\ &\equiv (\neg x \wedge z) \vee (x \wedge \neg y) \end{aligned}$$

Mit dem angegebenen Verfahren und den obigen Überlegungen zur Länge der konstruierten PNF-Formel erhalten wir folgenden Satz:

**Satz 4.11 (Universalität der PNF).** *Zu jeder aussagenlogischen Formel  $\alpha$  gibt es eine PNF-Formel  $\beta$  mit  $\alpha \equiv \beta$  und  $|\beta| = \mathcal{O}(|\alpha|)$ .*

### Zweistufige Normalformen (KNF, DNF)

Traditionell spielen zweistufige Normalformen, in denen die Formeln aus Konjunktionen über Disjunktionen (konjunktive Normalform), oder umgekehrt, Disjunktionen über Konjunktionen (disjunktive Normalform), aufgebaut sind und auf unterster Ebene nur Literale zugelassen sind, eine wichtige Rolle in der Informatik. In der technischen Informatik sind sie für die Logiksynthese zentral, da dort zweistufige Normalformen (insbesondere die disjunktive Normalform) als Ausgangspunkt für den Entwurf von PLA (programmable logic arrays) dienen. In der Algorithmik und Komplexitätstheorie kommt der konjunktiven Normalform eine herausragende Bedeutung zu.

**Einstufige Formeln (Klauseln, Monome).** Zunächst klären wir einige Grundbegriffe. Beginnen wir zunächst mit einstufigen Formeln, welche Literale  $L_1, \dots, L_k$  mit Junktoren desselben Typs (entweder Konjunktion oder Disjunktion) verknüpfen. Ein *Monom* oder *Konjunktionsterm* ist eine Formel des Typs  $L_1 \wedge L_2 \wedge \dots \wedge L_k$ , wobei  $L_1, \dots, L_k$  Literale sind. Formeln des Typs  $L_1 \vee L_2 \vee \dots \vee L_k$  werden *Klauseln* oder auch *Disjunktionsterme* genannt. Ein Monom heißt *widersprüchlich*, falls es zueinander komplementäre Literale  $x$  und  $\neg x$  enthält. Widersprüchliche Monome sind offenbar unerfüllbar. Klauseln mit zueinander komplementären Literalen sind dagegen tautologisch:

$$\dots \vee x \vee \neg x \vee \dots \equiv \text{true}.$$

Zur Bezeichnung von Literalen verwenden wir üblicherweise den Buchstaben  $L$ , während wir für Klauseln griechische Kleinbuchstaben wie  $\kappa$  (“kappa”),  $\lambda$  (“lambda”),  $\sigma$  (“sigma”) oder  $\tau$  (“tau”) benutzen, und  $\mu$  (“mu”) oder  $\nu$  (“nu”) für Monome; jeweils mit oder ohne Index.

Für eine Klausel  $\kappa$  ist  $\kappa = \bigvee_{i \in \emptyset} \dots = \text{false}$  die Klausel bestehend aus 0 Literalen. In diesem Fall spricht man auch von der *leeren Klausel*. Sie wird auch mit dem Symbol  $\square$  bezeichnet. Bei einer Klausel, die aus genau einem Literal besteht, spricht man auch von einer *Einheitsklausel*. Z.B. ist  $\kappa = \neg x \vee y \vee z$  eine Klausel mit drei Literalen und  $\kappa' = \neg x$  eine Einheitsklausel. Eine *negative Klausel* bezeichnet eine Klausel, die keine positiven Literale enthält, aber mindestens ein negatives Literal; also eine Klausel der Form  $\neg x_1 \vee \dots \vee \neg x_k$  mit  $k \geq 1$ . Entsprechend wird eine nicht-leere Klausel *positiv* genannt, wenn sie keine negativen Literale enthält; also wenn sie die Gestalt  $x_1 \vee \dots \vee x_k$  mit  $k \geq 1$  hat.

**Konjunktive Normalform (KNF).** Eine Formel in konjunktiver Normalform, kurz KNF-Formel genannt, ist eine Konjunktion von Klauseln; also von der Form

$$\alpha = \bigwedge_{1 \leq i \leq m} \underbrace{(L_{i,1} \vee L_{i,2} \vee \dots \vee L_{i,k_i})}_{i\text{-te Klausel}},$$

wobei die  $L_{i,j}$  Literale sind und  $m \geq 0$ ,  $k_1, \dots, k_i \geq 0$ . Z.B. ist  $\alpha = (\neg x \vee y) \wedge (y \vee z)$  eine zu  $\beta = y \vee (\neg x \wedge z)$  äquivalente Formel in KNF. Offenbar ist eine KNF-Formel  $\alpha$  genau

dann unter einer gegebenen Belegung  $I$  wahr, wenn es in jeder Klausel von  $\alpha$  wenigstens ein Literal gibt, das unter  $I$  wahr ist. Ist also  $\alpha$  wie oben, dann gilt  $\alpha^I = 1$  genau dann, wenn es zu jedem  $i \in \{1, \dots, m\}$  einen Index  $j \in \{1, \dots, k_i\}$  mit  $L_{i,j}^I = 1$  gibt. Im Sonderfall  $m = 0$  ist  $\alpha$  eine KNF-Formel bestehend aus 0 Klauseln. In diesem Fall ist

$$\alpha = \bigwedge_{i \in \emptyset} \dots = true.$$

Falls eine der Klauseln von  $\alpha$  leer ist, dann hat  $\alpha$  die Form  $\dots \wedge \square \wedge \dots = \dots \wedge false \wedge \dots$ , und ist somit unerfüllbar. Die asymptotische Länge einer KNF-Formel  $\alpha$  ist durch die totale Anzahl an Literalen in  $\alpha$  gegeben. Liegt z.B. die KNF-Formel

$$\alpha = \bigwedge_{1 \leq i \leq m} \bigvee_{1 \leq j \leq k_i} L_{i,j}$$

vor, so ist die asymptotische Länge von  $|\alpha|$  gleich  $k_1 + \dots + k_m$ .

**Disjunktive Normalform (DNF).** Formeln in disjunktiver Normalform, auch DNF-Formeln oder Polynome genannt, sind Disjunktionen von 0 oder mehreren Monomen. Beispielsweise ist  $(x \wedge \neg y \wedge \neg z) \vee (\neg y) \vee (x \wedge z)$  eine Formel in DNF. Die allgemeine Gestalt von DNF-Formeln ist also

$$\alpha = \bigvee_{1 \leq i \leq m} \bigwedge_{1 \leq j \leq k_i} L_{i,j},$$

wobei  $L_{i,j}$ 's Literale sind und  $m \geq 0$ ,  $k_1, \dots, k_m \geq 0$ . Auch hier sind die Sonderfälle  $m = 0$  oder  $k_i = 0$  möglich. Für den Sonderfall  $m = 0$  ist  $\alpha$  eine Disjunktion von 0 Monomen, also  $\alpha = \bigvee_{i \in \emptyset} \dots = false$ . Ist  $k_i = 0$ , so ist das  $i$ -te Monom leer (d.h., eine Konjunktion von 0 Literalen). Das  $i$ -te Monom steht somit für die Konstante  $\bigwedge_{j \in \emptyset} \dots = true$ . Wie im Falle von KNF-Formeln ist die asymptotische Länge einer DNF-Formel  $\alpha$  durch die totale Anzahl an Literalen in  $\alpha$  gegeben.

Die beiden zweistufigen Normalformen sind dual zueinander, da das Vertauschen von Konjunktionen und Disjunktionen und das Ersetzen jedes Literals  $L$  durch das komplementäre Literal  $\bar{L}$  jede KNF-Formel  $\alpha$  in eine DNF-Formel  $\beta$  mit  $\beta \equiv \neg \alpha$  überführt. Ist also

$$\alpha = \bigvee_{1 \leq i \leq m} \bigwedge_{1 \leq j \leq k_i} L_{i,j}$$

eine DNF-Formel, so ist

$$\beta = \bigwedge_{1 \leq i \leq m} \bigvee_{1 \leq j \leq k_i} \bar{L}_{i,j}$$

eine zu  $\neg \alpha$  äquivalente KNF-Formel.

**Satz 4.12 (Universalität von KNF und DNF).** *Zu jeder aussagenlogischen Formel  $\alpha$  gibt es eine äquivalente KNF-Formel und eine äquivalente DNF-Formel.*

Für den Beweis von Satz 4.12 geben wir zwei alternative Verfahren zur Konstruktion äquivalenter KNF- bzw. DNF-Formeln an.

**KNF/DNF via Wertetafel.** Das einfachste Verfahren zur Erstellung einer DNF oder KNF geht von einer Wertetafel aus und verknüpft die sogenannten *Minterme* disjunktiv (DNF) bzw. *Maxterme* konjunktiv (KNF).

**Definition 4.13 (Min-/Maxterme).** Ist  $AP = \{x_1, \dots, x_n\}$ , so wird jedes Monom der Form  $L_1 \wedge L_2 \wedge \dots \wedge L_n$  mit  $L_i \in \{x_i, \neg x_i\}$  *Minterm* und jede Klausel der Form  $L_1 \vee L_2 \vee \dots \vee L_n$  mit  $L_i \in \{x_i, \neg x_i\}$  *Maxterm* genannt. ■

Min- und Maxterme stehen in Eins-zu-Eins-Beziehung zu den Belegungen für  $AP$ . Hierzu wird jedem Minterm  $\mu = L_1 \wedge L_2 \wedge \dots \wedge L_n$  genau diejenige Belegung  $I$  zugeordnet, unter der alle Literale  $L_i$  von  $\mu$  wahr sind, also  $x_i^I = 1$ , falls  $L_i = x_i$  und  $x_i^I = 0$ , falls  $L_i = \neg x_i$ . Dual hierzu wird jedem Maxterm  $\kappa = L_1 \vee L_2 \vee \dots \vee L_n$  diejenige Belegung  $I$  zugeordnet, unter der alle Literale in  $\kappa$  falsch sind, also  $x_i^I = 1$ , falls  $L_i = \neg x_i$  und  $x_i^I = 0$ , falls  $L_i = x_i$ . Ein Minterm für eine aussagenlogische Formel  $\alpha$  ist ein Minterm, für den die zugeordnete Belegung ein Modell für  $\alpha$  ist. Offenbar ist  $\alpha$  dann äquivalent zu derjenigen DNF-Formel, die man durch Disjunktion aller Minterme für  $\alpha$  erhält. Ein Maxterm für  $\alpha$  ist ein Maxterm, für den die induzierte Belegung kein Modell für  $\alpha$  ist. Eine zu  $\alpha$  äquivalente KNF-Formel erhält man durch Konjunktion aller Maxterme für  $\alpha$ . Durch Inspektion der Wertetafeln für eine aussagenlogische Formel  $\alpha$  erhält man die Min- bzw. Maxterme und kann daraus wie eben beschrieben eine äquivalente DNF- bzw. KNF-Formel generieren. Die hier vorgestellte Methode zur Generierung einer DNF- bzw. KNF-Formel zeigt auch, dass es zu jeder Formel eine äquivalente Formel in DNF oder KNF gibt. Jedoch führt diese stets zu einer exponentiellen Laufzeit. Ferner sind die resultierenden Normalformen oftmals sehr lang.

Wir betrachten nun als Beispiel die aussagenlogische Formel

$$\alpha = \neg((x \vee \neg z) \wedge \neg(x \wedge \neg y)),$$

welche wir schon im vorigen Abschnitt über die PNF verwendet haben. In der folgenden Tabelle ist für jede mögliche Belegung  $I$  der Wahrheitswert von  $\alpha$  aufgelistet. Auf der rechten Seite sind die korrespondierenden Maxterme angegeben:

$x^I$	$y^I$	$z^I$	$\alpha^I$	
0	0	0	0	← Maxterm $x \vee y \vee z$
0	0	1	1	
0	1	0	0	← Maxterm $x \vee \neg y \vee z$
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	0	← Maxterm $\neg x \vee \neg y \vee z$
1	1	1	0	← Maxterm $\neg x \vee \neg y \vee \neg z$

Verknüpft man die Maxterme konjunktiv, so erhält man folgende zu  $\alpha$  äquivalente KNF-Formel:

$$(x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$$

In analoger Weise erhalten wir eine zu  $\alpha$  äquivalente DNF-Formel. Hierzu benötigen wir die Minterme von  $\alpha$ , die ebenfalls aus der Wertetafel abgelesen werden können:

$x^I$	$y^I$	$z^I$	$\alpha^I$	
0	0	0	0	
0	0	1	1	← Minterm $\neg x \wedge \neg y \wedge z$
0	1	0	0	
0	1	1	1	← Minterm $\neg x \wedge y \wedge z$
1	0	0	1	← Minterm $x \wedge \neg y \wedge \neg z$
1	0	1	1	← Minterm $x \wedge \neg y \wedge z$
1	1	0	0	
1	1	1	0	

Eine zu  $\alpha$  äquivalente DNF-Formel erhält man durch Disjunktion der Minterme:

$$(\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge z) \vee (x \wedge \neg y \wedge \neg z) \vee (x \wedge \neg y \wedge z)$$

**KNF/DNF via PNF und Äquivalenzgesetze.** Ein alternatives Verfahren, zu einer aussagenlogischen Formel eine äquivalente Formel in KNF oder DNF zu generieren, besteht darin, zunächst eine äquivalente PNF-Formel zu erstellen, die dann mit Hilfe der Distributivgesetze für die Disjunktion und Konjunktion in die gewünschte zweistufige Normalform gebracht wird. Sei  $\alpha$  eine beliebige aussagenlogische Formel, dann kann diese in eine äquivalente Formel  $\alpha'$  in PNF transformiert werden, indem man vorkommende Negationen “nach innen zieht” (wie oben im Abschnitt über PNF beschrieben). Durch sukzessives Anwenden der Distributivgesetze kann  $\alpha'$  in eine äquivalente aussagenlogische KNF- oder DNF-Formel umgewandelt werden. Wir betrachten hier nur exemplarisch die Konstruktion einer äquivalenten KNF-Formel, bei der folgende Transformationsregeln auf die PNF-Formel  $\alpha'$  angewendet werden, um die Disjunktionen nach innen zu ziehen:

$$\begin{aligned} \beta \vee (\gamma_1 \wedge \gamma_2) &\rightsquigarrow (\beta \vee \gamma_1) \wedge (\beta \vee \gamma_2) \\ (\gamma_1 \wedge \gamma_2) \vee \beta &\rightsquigarrow (\gamma_1 \vee \beta) \wedge (\gamma_2 \vee \beta) \end{aligned}$$

Die Transformation zur DNF erfolgt analog mit entsprechend dualen Distributivgesetzen.

Als Beispiel nehmen wir die aussagenlogische Formel  $\alpha$ , die schon im letzten Abschnitt betrachtet wurde:

$$\alpha = \neg((x \vee \neg z) \wedge \neg(x \wedge \neg y))$$

Eine zu  $\alpha$  äquivalente PNF-Formel  $\alpha'$  erhält man wie im Abschnitt über die PNF mit dem gleichen Beispiel für  $\alpha$  beschrieben:

$$\alpha' = (\neg x \wedge z) \vee (x \wedge \neg y)$$

Die Transformation zu einer äquivalenten KNF-Formel mittels der oben aufgelisteten Distributivgesetze erfolgt dann schrittweise wie folgt:

$$\begin{aligned} \alpha &\equiv \alpha' = (\neg x \wedge z) \vee (x \wedge \neg y) \\ &\equiv ((\neg x \wedge z) \vee x) \wedge ((\neg x \wedge z) \vee \neg y) \\ &\equiv ((\neg x \vee x) \wedge (z \vee x)) \wedge ((\neg x \vee \neg y) \wedge (z \vee \neg y)) \\ &\equiv (z \vee x) \wedge (\neg x \vee \neg y) \wedge (z \vee \neg y) \end{aligned}$$

Offenbar ist die resultierende Formel in der Tat eine KNF-Formel. Das Verfahren beruhend auf den Äquivalenzregeln ist effizienter als das Verfahren via Wertetabellen, welches wir im letzten Abschnitt betrachtet haben (vergleiche z.B. die Länge der entstandenen KNF-Formel). Jedoch ist ein exponentielles Blowup möglich und für manche Formeln sogar unvermeidbar, wie der folgende Satz belegt.

**Satz 4.14 (Exponentielles Blowup durch den Übergang zu KNF und DNF).**

- (a) Es gibt DNF-Formeln  $\alpha_n$  der Länge  $|\alpha_n| = \mathcal{O}(n)$ , so dass jede zu  $\alpha_n$  äquivalente KNF-Formel mindestens  $2^n$  Klauseln hat.
- (b) Es gibt KNF-Formeln  $\gamma_n$  der Länge  $|\gamma_n| = \mathcal{O}(n)$ , so dass jede zu  $\gamma_n$  äquivalente DNF-Formel mindestens  $2^n$  Monome hat.

*Beweis.* Zunächst beweisen wir Aussage (a). Die DNF-Formeln

$$\alpha_n \stackrel{\text{def}}{=} (x_1 \wedge y_1) \vee \dots \vee (x_n \wedge y_n)$$

haben lineare Länge, da die Anzahl an Konjunktionen und Disjunktionen in  $\alpha_n$  gleich  $2n-1$  ist und somit  $|\alpha_n| = \mathcal{O}(n)$ . Äquivalente KNF-Formeln ergeben sich durch Anwenden des Distributivgesetzes. Man erhält folgende KNF-Formel mit genau  $2^n$  Klauseln:

$$\alpha'_n \stackrel{\text{def}}{=} \bigwedge_{\substack{(z_1, \dots, z_n) \\ z_j \in \{x_j, y_j\} \text{ für } 1 \leq j \leq n}} (z_1 \vee \dots \vee z_n).$$

Beispielsweise gilt für  $n = 2$ :

$$\alpha'_2 = (x_1 \vee x_2) \wedge (x_1 \vee y_2) \wedge (y_1 \vee x_2) \wedge (y_1 \vee y_2).$$

Formal kann die Äquivalenz von  $\alpha_n$  und  $\alpha'_n$  entweder durch Argumentation mit den Distributivgesetzen nachgewiesen werden oder auch explizit durch den Nachweis, dass  $\alpha_n$  und  $\alpha'_n$  unter allen Belegungen denselben Wahrheitswert haben:

- Ist  $I$  eine erfüllende Belegung für  $\alpha_n$ , so gibt es ein  $j \in \{1, \dots, n\}$ , so dass  $x_j^I = y_j^I = 1$ . Da jede Klausel von  $\alpha'_n$  entweder  $x_j$  oder  $y_j$  enthält, ist  $I$  zugleich erfüllend für  $\alpha'_n$ .
- Ist  $I$  eine Belegung mit  $\alpha_n^I = 0$ , so gilt für jedes  $j \in \{1, \dots, n\}$ , dass wenigstens eines der Atome  $x_j$  oder  $y_j$  mit 0 belegt ist. Sei nun

$$z_j \stackrel{\text{def}}{=} \begin{cases} x_j & : \text{ falls } x_j^I = 0 \\ y_j & : \text{ sonst} \end{cases}$$

für  $1 \leq j \leq n$ . Dann ist  $z_j^I = 0$  für  $1 \leq j \leq n$  und somit  $z_1 \vee \dots \vee z_n$  eine Klausel von  $\alpha'_n$ , welche unter  $I$  den Wahrheitswert 0 hat. Also ist  $I$  auch für  $\alpha'_n$  nicht erfüllend.

Wir zeigen nun, dass jede zu  $\alpha_n$  äquivalente KNF-Formel  $2^n$  oder mehr Klauseln hat. Hierzu betrachten wir eine *kürzeste* zu  $\alpha_n$  äquivalente KNF-Formel

$$\beta_n = \lambda_1 \wedge \dots \wedge \lambda_k.$$

und zeigen, dass  $k \geq 2^n$ . Es ist klar, dass in  $\beta_n$  höchstens die Atome  $x_1, \dots, x_n, y_1, \dots, y_n$  vorkommen und dass kein Literal mehrfach in einer der Klauseln von  $\beta_n$  vorkommen kann. Andernfalls könnte  $\beta_n$  verkürzt werden.

*Behauptung 1.*  $\beta_n$  enthält keine negativen Literale.

Intuitiv ist diese Aussage klar, da sonst  $\beta_n$  verkürzt werden kann. Eine formale Beweismöglichkeit besteht darin, anzunehmen, dass  $\beta_n$  ein negatives Literal  $\neg z$  mit  $z \in \{x_1, \dots, x_n, y_1, \dots, y_n\}$  enthält. Man betrachtet dann diejenige Formel, welche aus  $\beta_n$  entsteht, indem ein (oder alle) Vorkommen von  $\neg z$  in  $\beta_n$  gestrichen werden und zeigt dann, dass die resultierende Formel immer noch zu  $\alpha_n$  äquivalent ist. Dies widerspricht dann der Annahme, dass  $\beta_n$  eine *kürzeste* zu  $\alpha_n$  äquivalente KNF-Formel ist.

Aus der ersten Behauptung folgt, dass alle Klauseln  $\lambda_i$  von  $\beta_n$  Disjunktionen von Atomen in  $\{x_1, \dots, x_n, y_1, \dots, y_n\}$  sind.

*Behauptung 2.* Für jede Klausel  $\lambda_i$  von  $\beta_n$  und jedes  $j \in \{1, \dots, n\}$  gilt: in  $\lambda_i$  kommt wenigstens eines der Literale  $x_j$  oder  $y_j$  vor.

*Beweis von Behauptung 2.* Wir nehmen an, dass weder  $x_j$  noch  $y_j$  in  $\lambda_i$  vorkommt, und führen diese Annahme zu einem Widerspruch. Hierzu betrachten wir die Belegung I, welche genau die in  $\lambda_i$  vorkommenden Atome mit 0 belegt, alle anderen mit 1. Da  $x_j$  und  $y_j$  nicht in  $\lambda_i$  vorkommen, gilt:

$$x_j^I = y_j^I = 1$$

Offenbar gilt  $\alpha_n^I = 1$ . Wegen  $\alpha_n \equiv \alpha'_n \equiv \beta_n$  gilt daher:

$$\beta_n^I = \alpha_n^I = 1$$

Andererseits ist  $\lambda_i^I = 0$ , da  $\lambda_i$  aufgrund der ersten Behauptung nur aus positiven Literalen besteht und diese nach Wahl von I mit 0 belegt sind. Hieraus folgt  $\beta_n^I = 0$ . Widerspruch. ]

In der dritten Behauptung zeigen wir nun, dass tatsächlich jede der  $2^n$  Klauseln  $\kappa = z_1 \vee \dots \vee z_n$  mit  $z_j \in \{x_j, y_j\}$  für  $1 \leq j \leq n$  der oben angegebenen zu  $\alpha_n$  äquivalenten KNF-Formel  $\alpha'_n$  in  $\beta_n$  vorkommen muss.

*Behauptung 3.* Jede der Klauseln  $\kappa$  von  $\alpha'_n$  kommt (modulo Permutation der Literale) in  $\beta_n = \lambda_1 \wedge \dots \wedge \lambda_k$  vor, d.h., es gibt einen Index  $i \in \{1, \dots, k\}$ , so dass  $\lambda_i \equiv \kappa$ .

*Beweis von Behauptung 3.* Sei  $\kappa$  eine Klausel von  $\alpha'_n$ . Sei I diejenige Belegung, unter der genau solche Atome, welche in  $\kappa$  vorkommen, zu 0 evaluieren. D.h.,

$$z^I = \begin{cases} 0 & : \text{ falls } z \text{ in } \kappa \text{ vorkommt} \\ 1 & : \text{ sonst} \end{cases}$$

Ist z.B.  $\kappa = x_1 \vee \dots \vee x_m \vee y_{m+1} \vee \dots \vee y_n$ , so betrachten wir die Belegung I mit

$$x_1^I = \dots x_m^I = y_{m+1}^I = \dots = y_n^I = 0 \quad \text{und} \quad y_1^I = \dots y_m^I = x_{m+1}^I = \dots = x_n^I = 1.$$

Offenbar ist  $\alpha'_n$  unter I falsch, d.h.,  $(\alpha'_n)^I = 0$ . Da  $\alpha'_n$  zu  $\alpha_n$  und  $\beta_n$  äquivalent ist, erhalten wir:

$$\beta_n^I = \alpha_n^I = 0$$

Also gibt es eine Klausel  $\lambda_i$  von  $\beta_n$ , so dass  $\lambda_i^I = 0$ . Da aber alle Atome, die nicht in  $\kappa$  vorkommen, mit 1 belegt sind, muss  $\lambda_i$  aus Atomen bestehen, die auch in  $\kappa$  vorkommen. Da aufgrund

der zweiten Behauptung jede Klausel von  $\beta_n$  aus mindestens  $n$  Literale besteht (nämlich je eines der Literale  $x_j$  oder  $y_j$  enthält) folgt  $\lambda_i \equiv \kappa$ .  $\lrcorner$

Aus der dritten Behauptung folgt, dass  $\beta_n$  aus  $2^n$  Klauseln besteht. Damit ist Aussage (a) bewiesen. Aussage (b) ergibt sich nun aus der Dualität von KNF und DNF. Die KNF-Formeln

$$\gamma_n \stackrel{\text{def}}{=} (x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \dots \wedge (x_n \vee y_n),$$

haben lineare Länge, aber jede äquivalente DNF-Formel hat  $2^n$  oder mehr Monome.  $\square$

In den folgenden Abschnitten befassen wir uns mit dem *Erfüllbarkeitsproblem* der Aussagenlogik. Dieses ist auch unter der Abkürzung SAT für die englische Bezeichnung satisfiability problem bekannt. Dieses zählt zu den algorithmisch schwierigen Problemen, für die es vermutlich keine effizienten Lösungsverfahren gibt. (Mehr hierzu in dem Modul “Theoretische Informatik und Logik”.)

Zunächst stellen wir fest, dass es Formeltypen gibt, für das Erfüllbarkeitsproblem sehr einfach zu lösen ist. Hierzu zählen DNF-Formeln. Eine DNF-Formel ist nämlich genau dann erfüllbar, wenn wenigstens eines ihrer Monome erfüllbar ist. Und Monome, also Konjunktionen von Literalen, sind genau dann erfüllbar, wenn sie keine zueinander komplementären Literale enthalten. Mit dieser Beobachtung ergibt sich ein DNF-SAT-Beweiser (damit ist ein Verfahren für den Erfüllbarkeitstest für DNF-Formeln gemeint) wie folgt. Sei  $\alpha$  die Eingabeformel, also eine DNF-Formel, für welche die Erfüllbarkeit zu prüfen ist. In einem vorbereitenden Schritt können zunächst alle widersprüchlichen Monome (d.h. Monome mit zueinander komplementären Literalen) gestrichen werden. Es entsteht eine DNF-Formel, die genau dann unerfüllbar ist, wenn sie kein Monom enthält, also wenn  $\alpha = \text{false}$ . Damit ergibt sich ein Verfahren für DNF-SAT, welches die Laufzeit  $\mathcal{O}(|\alpha|)$  hat. Dennoch ist die Erstellung einer äquivalenten DNF kein Allheilmittel, da kürzeste äquivalente DNF-Formeln exponentiell länger als die ursprüngliche Formel sein können.

Eine weitere Klasse, für die ein einfacher und effizienter Erfüllbarkeitstest möglich ist, ist die Klasse der Hornformeln, die in dem folgenden Abschnitt betrachtet wird.