Kapitel 4

Aussagenlogik

Booleschen Ausdrücke (zum Beispiel $B \Rightarrow \neg F$) ergeben eine einfache logische Sprache, die als Aussagenlogik bezeichnet wird. Mit Aussagenlogik können viele interessante Sachverhalte modelliert werden. Ein prominentes Beispiel ist die Beschreibung von Schaltkreisen.

Unser Hauptinteresse gilt Techniken für die Vereinfachung von Booleschen Ausdrücken. Beispielsweise kann der Ausdruck

$$(\neg B \Rightarrow F) \ \wedge \ (F \wedge B \Rightarrow \neg E) \ \wedge \ (E \vee \neg B \Rightarrow \neg F)$$

zu $B \wedge (\neg F \vee \neg E)$ vereinfacht werden.

4.1 Boolesche Funktionen

Wir beginnen mit der zweielementigen Menge

$$\mathbb{B} \stackrel{\text{def}}{=} \{0, 1\}$$

Unter einer *Booleschen Funktion* verstehen wir eine Funktion des Typs $\mathbb{B}^n \to \mathbb{B}$ ($n \geq 0$). Abbildung 4.1 zeigt fünf prominente Boolesche Funktionen, die als *Negation*, *Konjunktion*, *Disjunktion*, *Implikation* und *Äquivalenz* bekannt sind.

Da Boolesche Funktionen endlich sind, kann man sie durch Tabellen beschreiben. Wir zeigen das am Beispiel von Konjunktion, Disjunktion, Implikation und Äquivalenz:

x	y	$x \wedge y$	$x \vee y$	$x \Rightarrow y$	$x \Leftrightarrow y$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

$$\neg \in \mathbb{B} \to \mathbb{B} \qquad \textit{Negation}$$
$$\neg x = 1 - x$$

$$\wedge \in \mathbb{B} \times \mathbb{B} \to \mathbb{B} \qquad \textit{Konjunktion}$$
$$x \wedge y = \min\{x, y\}$$

$$\vee \in \mathbb{B} \times \mathbb{B} \to \mathbb{B} \qquad \textit{Disjunktion}$$
$$x \vee y = \max\{x, y\}$$

$$\Rightarrow \in \mathbb{B} \times \mathbb{B} \to \mathbb{B} \qquad \textit{Implikation}$$
$$x \Rightarrow y = \text{if } x = 1 \text{ then } y \text{ else } 1$$

$$\Leftrightarrow \in \mathbb{B} \times \mathbb{B} \to \mathbb{B} \qquad \textit{Äquivalenz}$$
$$x \Leftrightarrow y = \text{if } x = y \text{ then } 1 \text{ else } 0$$

Abbildung 4.1: Fünf prominente Boolesche Funktionen

Variablen, die nur die zwei Werte in $\mathbb{B}$ annehmen können, bezeichnen wir als *Boolesche Variablen*. Ausdrücke, die aus den Konstanten 0 und 1, Booleschen Variablen und Booleschen Funktionen gebildet sind, bezeichnen wir als *Boolesche Ausdrücke*. Gleichungen zwischen Booleschen Ausdrücken bezeichnen wir als *Boolesche Gleichungen*.

Abbildung 4.2 zeigt einige gültige Boolesche Gleichungen, die wichtige Eigenschaften von Konjunktion, Disjunktion und Negation beschreiben.[1] Das symmetrische Verhalten von Konjunktion und Disjunktion sowie 0 und 1 wird als *Dualitätseigenschaft* bezeichnet.

Die Gültigkeit von Booleschen Gleichungen kann man zeigen, indem man sie für alle Werte der in ihnen vorkommenden Variablen nachrechnet. Dieses Nachrechnen lässt sich mithilfe sogenannter *Wahrheitstafeln* übersichtlich darstellen. Wir zeigen dies am Beispiel der ersten de Morganschen Gleichung:

[1] Die Gleichungen sind mit Bedacht ausgewählt. Sie beschreiben die Klasse der Booleschen Algebren. Diese Klasse kann kompakter mit den Gleichungen für Kommutativität, Distributivität, Komplementierung und Identität beschrieben werden, da die anderen Gleichungen aus diesen folgen. Die Beweise dafür findet man zum Beispiel in: J. Eldon Whitesitt, *Boolean algebra and its applications*, Addison Wesley, 1961.

$$
\begin{aligned}
(x \wedge y) \wedge z &= x \wedge (y \wedge z) && \textit{(Assoziativität)}\\
(x \vee y) \vee z &= x \vee (y \vee z)\\[1ex]
x \wedge y &= y \wedge x && \textit{(Kommutativität)}\\
x \vee y &= y \vee x\\[1ex]
x \wedge x &= x && \textit{(Idempotenz)}\\
x \vee x &= x\\[1ex]
x \wedge (y \vee z) &= (x \wedge y) \vee (x \wedge z) && \textit{(Distributivität)}\\
x \vee (y \wedge z) &= (x \vee y) \wedge (x \vee z)\\[1ex]
x \wedge (x \vee y) &= x && \textit{(Absorption)}\\
x \vee (x \wedge y) &= x\\[1ex]
\neg(x \wedge y) &= \neg x \vee \neg y && \textit{(de Morgan)}\\
\neg(x \vee y) &= \neg x \wedge \neg y\\[1ex]
\neg\neg x &= x && \textit{(Doppelnegation)}\\[1ex]
x \wedge \neg x &= 0 && \textit{(Komplemente)}\\
x \vee \neg x &= 1\\[1ex]
x \wedge 1 &= x && \textit{(Identitäten)}\\
x \vee 0 &= x
\end{aligned}
$$

Abbildung 4.2: Eigenschaften von $\wedge$, $\vee$ und $\neg$ ($x, y, z \in \mathbb{B}$)

$$
\begin{aligned}
0 &= x \wedge \neg x\\
1 &= \neg(x \wedge \neg x)\\
x \vee y &= \neg(\neg x \wedge \neg y)\\
x \Rightarrow y &= \neg(x \wedge \neg y)\\
x \Leftrightarrow y &= \neg(x \wedge \neg y) \wedge \neg(y \wedge \neg x)
\end{aligned}
$$

Abbildung 4.3: Darstellung von 0, 1, $\vee$, $\Rightarrow$ und $\Leftrightarrow$ mit $\neg$ und $\wedge$ ($x, y \in \mathbb{B}$)

x	y	$\neg(x \wedge y)$	$\neg x \vee \neg y$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Die gültigen Gleichungen in Abbildung 4.3 zeigen, dass die Konstanten 0 und 1 sowie die Funktionen Disjunktion, Implikation und Äquivalenz mithilfe der Funktionen Konjunktion und Negation beschrieben werden können. Später werden wir zeigen, dass jede Boolesche Funktion mit Konjunktion und Negation beschrieben werden kann.

Damit wir nicht so viele Klammern schreiben müssen, legen wir eine Rangfolge für die Booleschen Infixoperatoren fest: $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ (aufsteigend). Damit können wir den Ausdruck

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

auch ohne Klammern beschreiben:

$$\neg x \wedge y \vee z \Rightarrow x \wedge y$$

Außerdem legen wir fest, dass die Booleschen Infixoperatoren linksassoziativ geklammert werden.

4.2 Modellierung mit Booleschen Ausdrücken

Um 1850 entdeckte George Boole, dass sich logische Aussagen (im philosophischen Sinne) durch Boolesche Ausdrücke beschreiben lassen. Dabei ist eine Aussage wahr, wenn der beschreibende Ausdruck den Wert 1 hat, und falsch, wenn der beschreibende Ausdruck den Wert 0 hat. Diese Entdeckung ist spektakulär, da sie etwas Kompliziertes (die Bestimmung des Wahrheitswertes einer logischen Aussagen) auf etwas Einfaches (das Rechnen mit 0 und 1) reduziert.

Am Beispiel einer sogenannten Denksportaufgabe zeigen wir, wie man logische Aussagen mit Booleschen Ausdrücken beschreibt.

> „Worin besteht das Geheimnis Ihres langen Lebens?“ wird ein Hundertjähriger gefragt. „Ich halte mich streng an drei Diätregeln: Wenn ich kein Bier zu einer Mahlzeit trinke, dann esse ich immer Fisch. Immer wenn ich sowohl Fisch und Bier zur selben Mahlzeit esse, verzichte ich auf Eiscreme. Wenn ich Eiscreme esse oder kein Bier trinke, dann esse ich keinen Fisch.“ Können Sie diese Regeln vereinfachen?

Der erste Schritt besteht darin, Mahlzeiten durch Tripel

$$(B, E, F) \in \mathbb{B}^3$$

darzustellen, die wie folgt zu verstehen sind:

- $B = 1$ genau dann, wenn zu der Mahlzeit Bier getrunken wird.
- $E = 1$ genau dann, wenn zu der Mahlzeit Eiscreme gegessen wird.
- $F = 1$ genau dann, wenn zu der Mahlzeit Fisch gegessen wird.

Diese Modellierung von Mahlzeiten stellt eine radikale Abstraktion dar. Es werden nur die Aspekte einer Mahlzeit dargestellt, die für die Regeln relevant sind. Insgesamt können damit nur $2^3 = 8$ verschiedene Typen von Mahlzeiten dargestellt werden.

Die drei Diätregeln können wir jetzt wie folgt durch Boolesche Ausdrücke beschreiben:

$$\begin{array}{ll} R_1: & \neg B \Rightarrow F \\ R_2: & F \wedge B \Rightarrow \neg E \\ R_3: & E \vee \neg B \Rightarrow \neg F \end{array}$$

Dabei handelt es sich bei B, E und F um Boolesche Variablen und bei R_1, R_2 und R_3 um Namen für Ausdrücke. Eine Mahlzeit erfüllt die Diätregeln, wenn der Ausdruck

$$D: \quad R_1 \wedge R_2 \wedge R_3$$

für die durch die Mahlzeit festgelegten Werte für die Variablen B, E und F zu 1 auswertet.

Man überzeigt sich leicht, dass der Ausdruck D zu 1 auswerten, wenn man die Werte der Variablen wie folgt wählt:

$$B = 1, \; F = 0, \; E = 1$$

Das bedeutet, das eine Mahlzeit, bei der Bier getrunken, kein Fisch gegessen, und Eiscreme genossen wird, die Diätregeln erfüllt.

Die Denksportaufgabe verlangt nach einer vereinfachten Darstellung der Diätregeln. Auf unser Modell übertragen bedeutet dies, dass ein möglichst einfacher Boolescher Ausdruck V gesucht ist, für den die Gleichung $D = V$ gilt. Mithilfe einer Wahrheitstabelle kann man sich davon überzeugen, dass

$$D = B \wedge (\neg F \vee \neg E)$$

gilt (für alle Werte der Variablen B, E und F). Also kann man die Diätregeln des Hundertjährigen auch wie folgt formulieren: „Trinke zu jeder Mahlzeit Bier und esse zu keiner Mahlzeit sowohl Fisch und Eiscreme“.

4.3 Ein Interpreter für Aussagenlogik

Die durch Boolesche Ausdrücke gegebene Modellierungssprache wird als *Aussagenlogik* bezeichnet. Wir skizzieren einen Interpreter für Aussagenlogik, der nach dem Vorbild eines Standard ML Interpreters entworfen ist.

Wir führen den Interpreter am Beispiel der gerade besprochenen Denksportaufgabe vor. Zunächst deklarieren wir drei Boolesche Variablen `B`, `E` und `F`:

```
var B E F
```

Danach deklarieren wir drei Boolesche Ausdrücke, die die Diätregeln beschreiben:

```
exp R1 = -B => F
exp R2 = F * B => -E
exp R3 = E + -B => -F
```

Dabei steht − für Negation, `*` für Konjunktion und + für Disjunktion. Die Bezeichner `R1`, `R2` und `R3` werden als Namen für Boolesche Ausdrücke verwendet. Schließlich deklarieren wir den Ausdruck

```
exp D = R1 * R2 * R3
```

der die Konjunktion der Diätregeln beschreibt. Damit ist die Modellierung der Denksportaufgabe abgeschlossen.

Der wesentliche Berechnungsdienst des Interpretierers besteht darin, Ausdrücke zu vereinfachen. Die Vereinfachung eines Ausdrucks liefert immer einen äquivalenten Ausdruck. Hier sind einige Beispiele für Vereinfachungen.

```
simplify D * -B
```

`0`

Diese Vereinfachung besagt, dass es keine Mahlzeit gibt, die die Diätregeln einhält und bei der kein Bier getrunken wird.

```
simplify D => B
```

`1`

Diese Vereinfachung besagt, dass bei jeder Mahlzeit, die die Diätregeln einhält, Bier getrunken wird.

```
simplify D
```

`B * (-E + -F)`

Diese Vereinfachung besagt, dass eine Mahlzeit genau dann die die Diätregeln einhält, wenn sie Bier, aber nicht gleichzeitig Eiscreme und Fisch, beinhaltet.

$$
\begin{array}{llll}
X, Y & \in & \mathit{Var} & \mathit{Variable} \\
A, B & \in & \mathit{For} = & \mathit{Formel} \\
 & & \quad X & \quad \mathit{Atom} \\
 & & \mid \neg A & \quad \mathit{Negation} \\
 & & \mid A_1 \wedge A_2 & \quad \mathit{Konjunktion}
\end{array}
$$

$$
\sigma \in \Sigma = \mathit{Var} \to \mathbb{B} \quad \mathit{Belegung}
$$

$$
\mathcal{F} \in \mathit{For} \to \Sigma \to \mathbb{B}
$$

$$
\begin{aligned}
\mathcal{F}[\![X]\!]\sigma &= \sigma X \\
\mathcal{F}[\![\neg A]\!]\sigma &= 1 - \mathcal{F}[\![A]\!]\sigma \\
\mathcal{F}[\![A_1 \wedge A_2]\!]\sigma &= \min\{\mathcal{F}[\![A_1]\!]\sigma,\ \mathcal{F}[\![A_2]\!]\sigma\}
\end{aligned}
$$

Abbildung 4.4: Syntax und Semantik der Sprache AL

Die Beschreibung des Interpreters soll Sie davon überzeugen, dass Aussagenlogik als Grundlage für ein praktisch relevantes Software-Werkzeug dienen kann. Im Folgenden werden Sie grundlegende Techniken für die Vereinfachung von Booleschen Ausdrücken kennenlernen, mit denen Sie den Berechnungsdienst des Interpreters realisieren können.

4.4 Die aussagenlogische Sprache AL

Sei *Var* eine nichtleere Menge.

Abbildung 4.4 definiert relativ zu *Var* eine logische Sprache AL. Die Formeln von AL formalisieren Boolesche Ausdrücke, die mit Variablen, Negation und Konjunktion gebildet werden können.

Auf den ersten Blick erscheint AL sehr karg, da nur für Negation und Konjunktion Syntax vorhanden ist. Die Gleichungen in Abbildung 4.2 zeigen jedoch, dass die Formeln von AL auch 0, 1, $\vee$, $\Rightarrow$ und $\Leftrightarrow$ beschreiben können. Abbildung 4.5 definiert entsprechende notationale Abkürzungen. Dabei ist X_0 eine festgewählte Variable in *Var*. Mit der Notation

$$0 \Rightarrow X \vee Y$$

$$\begin{array}{rcl}
0 & \mapsto & X_0 \wedge \neg X_0 \\
1 & \mapsto & \neg 0 \\
A_1 \vee A_2 & \mapsto & \neg(\neg A_1 \wedge \neg A_2) \\
A_1 \Rightarrow A_2 & \mapsto & \neg(A_1 \wedge \neg A_2) \\
A_1 \Leftrightarrow A_2 & \mapsto & \neg(A_1 \wedge \neg A_2) \wedge \neg(A_2 \wedge \neg A_1)
\end{array}$$

Abbildung 4.5: Abkürzungen für AL

wird also dieselbe Formel in *For* beschrieben wie mit

$$\neg((X_0 \wedge \neg X_0) \wedge \neg\neg(\neg X \wedge \neg Y))$$

Die *Äquivalenz von Formeln* und eine *Substitutionsfunktion* definieren wir erwartungsgemäß wie folgt:

$$A \models\!\!\!\dashv B \quad \stackrel{\text{def}}{\iff} \quad \mathcal{F}[\![A]\!] = \mathcal{F}[\![B]\!]$$

$$\_[\_/\_] \in \mathit{For} \times \mathit{For} \times \mathit{Var} \to \mathit{For}$$

$$\begin{aligned}
Y[B/X] &= \text{if } X = Y \text{ then } B \text{ else } Y \\
(\neg A)[B/X] &= \neg(A[B/X]) \\
(A_1 \wedge A_2)[B/X] &= A_1[B/X] \wedge A_2[B/X]
\end{aligned}$$

Wunschgemäß gelten das Substitutionslemma und der Ersetzungssatz. Die Beweise entsprechen denen in Abschnitt 3.4.

Lemma 4.4.1 (Substitution) *Seien $A, B \in \mathit{For}$, $X \in \mathit{Var}$ und $\sigma \in \Sigma$. Dann:*

$$\mathcal{F}[\![A[B/x]]\!]\sigma = \mathcal{F}[\![A]\!](\sigma[(\mathcal{F}[\![B]\!]\sigma)/X])$$

Satz 4.4.2 (Ersetzung) *Seien $A_1, A_2, B_1, B_2 \in \mathit{For}$ und $X \in \mathit{Var}$. Dann:*

$$A_1 \models\!\!\!\dashv A_2 \ \wedge\ B_1 \models\!\!\!\dashv B_2 \quad \Rightarrow \quad A_1[B_1/X] \models\!\!\!\dashv A_2[B_2/X]$$

Vorkommende Variablen

Wir definieren eine Funktion VV, die zu einer Formel die Menge der in der Formel *vorkommenden Variablen* liefert:

$$VV \in For \to \mathcal{P}_{fin}(Var)$$

$$\begin{aligned} VV(X) &= \{X\} \\ VV(\neg A) &= VV(A) \\ VV(A_1 \wedge A_2) &= VV(A_1) \cup VV(A_2) \end{aligned}$$

Proposition 4.4.3 *Sei $A \in For$ eine Formel und seien $\sigma, \sigma' \in \Sigma$ Belegungen, die auf $VV(A)$ übereinstimmen. Dann: $\mathcal{F}[\![A]\!]\sigma = \mathcal{F}[\![A]\!]\sigma'$.*

Beweis Durch strukturelle Induktion über $A \in For$. □

Die Sprache AL ist relativ zu einer vorgegebenen Variablenmenge Var definiert. Wenn wir zwei Mengen Var und Var' betrachten, bekommen wir zwei Sprachen AL_{Var} und $\mathrm{AL}_{Var'}$. Die Formelmengen und Äquivalenzen der beiden Sprachen bezeichnen wir mit For_{Var}, $For_{Var'}$, $\models\mid_{Var}$ und $\models\mid_{Var'}$. Erwartungsgemäß haben wir:

Proposition 4.4.4 (Variablenunabhängigkeit) *Sei $Var' \subseteq Var$. Dann:*

1. $For_{Var'} \subseteq For_{Var}$.
2. $\forall A, B \in For_{Var'}: \quad A \models\mid_{Var'} B \iff A \models\mid_{Var} B$.

Die Proposition hat eine wichtige praktische Konsequenz: Wenn wir wissen wollen, ob zwei Formeln A und B äquivalent sind, können wir uns immer auf die endliche Menge Var' der Variablen zurückziehen, die in A und B vorkommen. Es genügt dann, die Äquivalenz für die endlich vielen Belegungen in $Var' \to \mathbb{B}$ nachzurechnen. Diese Beobachtung rechtfertigt das Nachrechnen von Äquivalenzen mit Wahrheitstafeln.

Verifikation von Äquivalenzen mit Wahrheitstafeln

Machen Sie sich klar, dass alle Gleichungen in den Abbildungen 4.4 und 4.5 gültige Äquivalenzen von AL entsprechen. Beispielsweise gilt für alle $X, Y \in Var$ die Äquivalenz

$$\neg(X \wedge Y) \models\mid \neg X \vee \neg Y$$

Die Gültigkeit dieser Äquivalenz kann man mithilfe einer Wahrheitstafel zeigen:

X	Y	$\neg(X \wedge Y)$	$\neg X \vee \neg Y$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Diese Wahrheitstafel haben Sie bereits in Abschnitt 4.1 gesehen, dort allerdings als Beweis für die Boolesche Gleichung

$$\forall x, y \in \mathbb{B}: \ \neg(x \wedge y) = \neg x \vee \neg y$$

Gültigkeit und Erfüllbarkeit

Für AL definieren wir die folgenden Sprechweisen:

$$\begin{aligned}
A \textit{ gültig} &\overset{\text{def}}{\iff} A \models\!\!\!\!\dashv 1 \\
A \textit{ unerfüllbar} &\overset{\text{def}}{\iff} A \models\!\!\!\!\dashv 0 \\
A \textit{ erfüllbar} &\overset{\text{def}}{\iff} A \not\models\!\!\!\!\dashv 0 \\
\sigma \textit{ erfüllt } A &\overset{\text{def}}{\iff} \mathcal{F}[\![A]\!]\sigma = 1
\end{aligned}$$

Gültige aussagenlogische Formeln werden auch oft als *Tautologien* bezeichnet.

Proposition 4.4.5 *Für jede Formel $A \in For$ gilt:*

$$\begin{aligned}
A \textit{ gültig} &\iff \forall \sigma \in \Sigma: \ \mathcal{F}[\![A]\!]\sigma = 1 \\
A \textit{ unerfüllbar} &\iff \forall \sigma \in \Sigma: \ \mathcal{F}[\![A]\!]\sigma = 0 \\
A \textit{ erfüllbar} &\iff \exists \sigma \in \Sigma: \ \mathcal{F}[\![A]\!]\sigma = 1 \\
A \models\!\!\!\!\dashv B &\iff \forall \sigma \in \Sigma: \ \mathcal{F}[\![A]\!]\sigma = \mathcal{F}[\![B]\!]\sigma
\end{aligned}$$

Proposition 4.4.6 *Für jede Formel $A \in For$ gilt:*

$$\begin{aligned}
A \textit{ gültig} &\iff \neg A \textit{ unerfüllbar} \iff A \models\!\!\!\!\dashv 1 \\
A \textit{ unerfüllbar} &\iff \neg A \textit{ gültig} \iff A \models\!\!\!\!\dashv 0 \\
A \models\!\!\!\!\dashv B &\iff A \Leftrightarrow B \textit{ gültig}
\end{aligned}$$

Die drei Eigenschaften Gültigkeit, Unerfüllbarkeit und Äquivalenz können sich also wechselseitig ausdrücken.

Im Folgenden entwickeln wir Techniken, mit denen sich die obigen Eigenschaften in der Regel effizienter entscheiden lassen als durch systematisches Nachrechnen mit Wahrheitstafeln. Allerdings gibt es für jede dieser Techniken immer „worst case“ Beispiele, für die sie nicht effizienter sind als das Nachrechnen mit Wahrheitstafeln. Das liegt daran, dass die Erfüllbarkeit von Boolschen Ausdrücken das kanonische NP-vollständige Problem ist.

Proposition 4.4.7 *Das Problem „gegeben $A \in For$, entscheide ob A erfüllbar ist“ ist NP-vollständig.*

4.5 Die Mengeninterpretation für AL

Die Denotionsfunktion

$$\mathcal{F} \in For \to (\Sigma \to \mathbb{B})$$

interpretiert die Formeln von AL als Beschreibungen für Funktionen in $\Sigma \to \mathbb{B}$. Wir definieren jetzt eine zweite Denotationsfunktion

$$\mathcal{M} \in For \to \mathcal{P}(\Sigma)$$

die die Formeln von AL als Beschreibungen für Teilmengen von Σ interpretiert. Während $\mathcal{F}$ die syntaktischen Operationen $\neg$ und $\wedge$ als die entsprechenden Booleschen Operationen interpretiert, interpretiert $\mathcal{M}$ diese Operationen als Mengenkomplement und Mengendurchschnitt. Trotz dieser scheinbar erheblichen Unterschiede liefern beide Interpretationen dieselbe Äquivalenzrelation für Formeln. Die durch $\mathcal{F}$ realisierte Interpretation bezeichnen wir als *Boolesche Interpretation*, und die durch $\mathcal{M}$ realisierte Interpretation als *Mengeninterpretation*.

Wir definieren $\mathcal{M}$ durch strukturelle Rekursion über *For* wie folgt:

$$\begin{aligned}
&\mathcal{M} \in For \to \mathcal{P}(\Sigma) \\
&\qquad \mathcal{M}[\![X]\!] = \{\, \sigma \in \Sigma \mid \sigma X = 1 \,\} \\
&\qquad \mathcal{M}[\![\neg A]\!] = \Sigma - \mathcal{M}[\![A]\!] \\
&\qquad \mathcal{M}[\![A \wedge B]\!] = \mathcal{M}[\![A]\!] \cap \mathcal{M}[\![B]\!]
\end{aligned}$$

Mit den für AL definierten Abkürzungen (siehe Abbildung 4.5) bekommen wir die folgenden Gleichungen:

Proposition 4.5.1 *Für alle $A, B \in For$ und $X \in Var$:*

$$\begin{aligned}
\mathcal{M}[\![\neg A]\!] &= \Sigma - \mathcal{M}[\![A]\!] \\
\mathcal{M}[\![A \wedge B]\!] &= \mathcal{M}[\![A]\!] \cap \mathcal{M}[\![B]\!] \\
\mathcal{M}[\![A \vee B]\!] &= \mathcal{M}[\![A]\!] \cup \mathcal{M}[\![B]\!] \\
\mathcal{M}[\![A \wedge \neg B]\!] &= \mathcal{M}[\![A]\!] - \mathcal{M}[\![B]\!] \\
\mathcal{M}[\![A \Rightarrow B]\!] &= (\Sigma - \mathcal{M}[\![A]\!]) \cup \mathcal{M}[\![B]\!] \\
\mathcal{M}[\![1]\!] &= \Sigma \\
\mathcal{M}[\![0]\!] &= \emptyset \\
\mathcal{M}[\![X]\!] &= \{\, \sigma \in \Sigma \mid \sigma X = 1 \,\} \\
\mathcal{M}[\![\neg X]\!] &= \{\, \sigma \in \Sigma \mid \sigma X = 0 \,\}
\end{aligned}$$

Die durch $\mathcal{M}$ realisierte Interpretation der Formeln von AL ist oft sehr intuitiv. Wir zeigen das am Beispiel der Diätregeln. Dazu wählen wir $Var = \{B, E, F\}$. Eine Belegung $\sigma \in \Sigma$ hat dann die Form

$$\{B \mapsto a,\ E \mapsto b,\ F \mapsto c\}$$

mit $a, b, c \in \mathbb{B}$. Eine Belegung kann also als ein Tripel verstanden werden, dessen Komponenten durch die „Marken“ B, E, F identifiziert werden.[2] Die Belegungen stellen die möglichen Mahlzeiten dar. Eine Diätregel ist eine Formel, die die Menge aller Mahlzeiten darstellt, die die Regel einhalten.

Wir stellen jetzt die Verbindung zwischen $\mathcal{F}$ und $\mathcal{M}$ her.

Proposition 4.5.2 *Die Funktion*

$$\begin{aligned}
&\tau \in (\Sigma \to \mathbb{B}) \to \mathcal{P}(\Sigma) \\
&\tau(f) = \{\, \sigma \in \Sigma \mid f(\sigma) = 1 \,\}
\end{aligned}$$

ist eine Bijektion mit $\mathcal{M} = \tau \circ \mathcal{F}$. Also gilt für alle Formeln A und Belegungen σ:

$$\sigma \in \mathcal{M}[\![A]\!] \iff \mathcal{F}[\![A]\!]\sigma = 1$$

Beweis Man überzeugt sich leicht davon, dass τ eine Bijektion ist. Durch strukturelle Induktion über $A \in For$ zeigt man, dass

$$\forall A \in For\ \forall \sigma \in \Sigma\colon\ \sigma \in \mathcal{M}[\![A]\!] \iff \mathcal{F}[\![A]\!]\sigma = 1$$

gilt. Also $\mathcal{M} = \tau \circ \mathcal{F}$. □

[2] In Programmiersprachen bezeichnet man solche markierten Tupel oft als Verbunde (Records in Pascal und Standard ML, Structures in C).

Proposition 4.5.3 *Für alle* $A, B \in For$*:* $A \models\!\!\mid B \iff \mathcal{M}[\![A]\!] = \mathcal{M}[\![B]\!]$.

Beweis Folgt sofort aus Proposition 4.5.2. □

Proposition 4.5.4 *Seien A und B Formeln. Dann gilt:*

$$(A \Rightarrow B) \models\!\!\mid 1 \iff \mathcal{M}[\![A]\!] \subseteq \mathcal{M}[\![B]\!]$$

Die Lösungsmenge einer Gleichung $A = B$ ist die Menge $\mathcal{M}[\![A \Leftrightarrow B]\!]$:

Proposition 4.5.5 *Seien* $A, B \in For$*. Dann:*

$$\{\, \sigma \in \Sigma \mid \mathcal{F}[\![A]\!]\sigma = \mathcal{F}[\![B]\!]\sigma \,\} = \mathcal{M}[\![A \Leftrightarrow B]\!]$$

Damit genügen zum Lösen von Gleichungen Techniken, die zu einer Formel eine hinreichend explizite Darstellung ihrer $\mathcal{M}$-Denotation liefern.

4.6 Denotationen

Die Denotationsfunktion $\mathcal{M} \in For \to \mathcal{P}(\Sigma)$ ordnet jeder Formel eine Teilmenge von Σ zu. Wir können also die Formeln von AL als Darstellungen für Teilmengen von Σ auffassen. Diese Sichtweise legt die Frage nahe, welche Teilmengen von Σ sich durch die Formeln von AL beschreiben lassen.

Um diese Frage zu beantworten, definieren wir eine Menge $Den \subseteq \mathcal{P}(\Sigma)$ mithilfe von Inferenzregeln:

$$\frac{X \in Var}{\{\, \sigma \in \Sigma \mid \sigma X = 1 \,\} \in Den}$$

$$\frac{P \in Den}{\Sigma - P \in Den} \qquad \frac{P \in Den \quad Q \in Den}{P \cap Q \in Den}$$

Die Elemente von *Den* bezeichnen wir als *Denotationen*.

Zuerst zeigen wir, dass die Denotationen genau die Teilmengen von Σ sind, die sich durch die Formeln von AL darstellen lassen.

Proposition 4.6.1 $Den = \{\, \mathcal{M}[\![A]\!] \mid A \in For \,\}$.

Beweis Wir beweisen zwei Aussagen:

(1) $\forall P \in Den\ \exists A \in For\colon\ P = \mathcal{M}[\![A]\!]$

(2) $\forall A \in For\ \exists P \in Den\colon\ P = \mathcal{M}[\![A]\!]$

Die erste Aussage zeigt man durch Regelinduktion über *Den*. Die zweite Aussage zeigt man durch strukturelle Induktion über *For*. Für beide Induktionsbeweise sind die Gleichungen der Proposition 4.5.1 hilfreich. □

Eine Formel kann man als ein Konstruktionsprotokoll auffassen, das beschreibt, wie eine Denotation mithilfe der *Den* definierenden Inferenzregeln konstruiert wird. Mit dieser Sichtweise ist $\mathcal{M}$ die Funktion, die einem Konstruktionsprotokoll die konstruierte Menge zuordnet.

Mit *Den* haben wir eine Charakterisierung der mit den Formeln von AL beschreibaren Mengen, die unabhängig von der Syntax von AL ist.

Proposition 4.6.2 *Die Menge Den erfüllt die folgenden Eigenschaften:*

1. $\emptyset \in Den$ *und* $\Sigma \in Den$.
2. *Wenn* $P, Q \in Den$, *dann* $P \cap Q \in Den$, $P \cup Q \in Den$ *und* $P - Q \in Den$.
3. *Wenn* $X \in Var$, *dann* $\{\, \sigma \in \Sigma \mid \sigma X = 1 \,\} \in Den$ *und* $\{\, \sigma \in \Sigma \mid \sigma X = 0 \,\} \in Den$.

Beweis Folgt aus Proposition 4.5.1. □

Proposition 4.6.3 *Sei* $M \subseteq \Sigma$. *Dann:*

$$M \;=\; \bigcup_{\sigma \in M} \left(\bigcap_{X \in Var} \{\, \sigma' \in \Sigma \mid \sigma'(X) = \sigma(X) \,\} \right)$$

Beweis Die Behauptung der Proposition ist äquivalent zu:

$$\forall \sigma' \in \Sigma \colon \quad \sigma' \in M \iff \exists \sigma \in M \; \forall X \in Var \colon \; \sigma'(X) = \sigma(X)$$

Diese Äquivalenz gilt trivialerweise. □

Proposition 4.6.4 *Sein Var endlich. Dann* $Den = \mathcal{P}(\Sigma)$.

Beweis Sei $M \subseteq \Sigma$. Da *Var* endlich ist, ist Σ endlich. Also ist auch M endlich. Also benutzt die Darstellung von M in Proposition 4.6.3 nur endliche Schnitte und Vereinigungen. Da zudem für alle $\sigma \in \Sigma$ wegen Proposition 4.6.2 $\{\, \sigma' \in \Sigma \mid \sigma' X = \sigma X \,\} \in Den$ gilt, folgt wieder mit Proposition 4.6.2, dass $M \in Den$. □

Wenn es nur endlich viele Variablen gibt, kann man also jede Teilmenge von Σ mit einer Formeln von AL darstellen.

Signifikante Variablen

Wir definieren:

$$Ex \in Var \times \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$$
$$Ex(X, M) = \{\, \sigma[b/X] \mid \sigma \in M \wedge b \in \mathbb{B} \,\}$$

Wir nennen $Ex(X, M)$ die *X-Projektion von M*. Offensichtlich $M \subseteq Ex(X, M)$.

Eine Variable $X \in Var$ heißt *signifikante Variable einer Menge $M \in \Sigma$* genau dann, wenn $M \neq Ex(X, M)$. Die Menge aller signifikanten Variablen einer Menge $M \in \Sigma$ bezeichnen wir mit $SV(M)$. Offensichtlich gilt: $SV(\emptyset) = SV(\Sigma) = \emptyset$.

Proposition 4.6.5 *Sei $D \in Den$ und $X \in Var$. Dann:*

$$X \in SV(D) \iff \exists \sigma \in D\ \exists b \in \mathbb{B}\colon\ \sigma[b/X] \notin D$$

Proposition 4.6.6 *Sei A eine Formel und seien $\sigma, \sigma' \in \Sigma$ Belegungen, die auf $VV(A)$ übereinstimmen. Dann: $\sigma \in \mathcal{M}[\![A]\!] \iff \sigma' \in \mathcal{M}[\![A]\!]$.*

Proposition 4.6.7 *Sei $A \in For$. Dann kommt jede signifikante Variable von $\mathcal{M}[\![A]\!]$ in A vor.*

Beweis Durch Widerspruch. Sei $X \in SV(\mathcal{M}[\![A]\!]) - VV(A)$. Proposition 4.6.5 liefert ein $\sigma \in \mathcal{M}[\![A]\!]$ und ein $b \in \mathbb{B}$ mit $\sigma[b/X] \notin \mathcal{M}[\![A]\!]$. Da σ und $\sigma[b/X]$ auf $VV(A)$ übereinstimmen und $\sigma \in \mathcal{M}[\![A]\!]$, folgt $\sigma[b/X] \in \mathcal{M}[\![A]\!]$ (Proposition 4.6.6). Widerspruch. □

Proposition 4.6.8 *Jede Denotation $D \in Den$ hat nur endlich viele signifikante Variablen.*

Beweis Propositionen 4.6.1 und 4.6.7. □

Lemma 4.6.9 *Sei $A \in For$, $\sigma \in \mathcal{M}[\![A]\!]$ und $\sigma' \in \Sigma$. Wenn σ und σ' auf den signifikanten Variablen von $\mathcal{M}[\![A]\!]$ übereinstimmen, dann $\sigma' \in \mathcal{M}[\![A]\!]$.*

Beweis Sei $VV(A) - SV(\mathcal{M}[\![A]\!]) = \{X_1, \ldots, X_n\}$. Wir definieren:

$$\sigma'' \stackrel{\text{def}}{=} (\cdots(\sigma[\sigma'(X_1)/X_1])\cdots[\sigma'(X_n)/X_n])$$

Da $\sigma \in \mathcal{M}[\![A]\!]$ und $\mathcal{M}[\![A]\!] = Ex(X_i, \mathcal{M}[\![A]\!])$ für alle $i \in \{1, \ldots, n\}$, folgt $\sigma'' \in \mathcal{M}[\![A]\!]$. Da σ'' und σ' auf allen in A vorkommenden Variablen übereinstimmen, folgt $\sigma' \in \mathcal{M}[\![A]\!]$ mit Proposition 4.6.6. □

Proposition 4.6.10 *Sei $D \in Den$ und $\sigma, \sigma' \in \Sigma$. Dann:*

$$\sigma \in D \iff \exists \sigma' \in D\ \forall X \in SV(D)\colon\ \sigma(x) = \sigma'(X)$$

Beweis Die Richtung „⇒“ ist trivial. Die andere Richtung folgt aus Proposition 4.6.1 und Lemma 4.6.9. □

Satz 4.6.11 (Darstellung) *Sei $D \in Den$ eine Denotation. Dann gilt:*

1. $D = \bigcup_{\sigma \in D} \left(\bigcap_{X \in SV(D)} \{ \sigma' \in \Sigma \mid \sigma' X = \sigma X \} \right)$

2. $D = \bigcap_{\sigma \in \Sigma - D} \left(\bigcup_{X \in SV(D)} \{ \sigma' \in \Sigma \mid \sigma' X \neq \sigma X \} \right)$

Beweis Die erste Aussage ist äquivalent zu:

$$\forall \sigma' \in \Sigma : \quad \sigma' \in D \iff \exists \sigma \in D \; \forall X \in SV(D) : \; \sigma'(X) = \sigma(X)$$

Diese Aussage folgt aus Proposition 4.6.10.

Die zweite Aussage zeigen wir ebenfalls mit von Proposition 4.6.10, wobei wir für D das Komplement $\Sigma - D$ einsetzen. Sei $\sigma' \in \Sigma$. Dann:

$$\begin{aligned} \sigma' \in D &\iff \neg(\sigma' \in \Sigma - D) \\ &\iff \neg(\exists \sigma \in \Sigma - D \; \forall X \in SV(\Sigma - D) : \; \sigma'(X) = \sigma(X)) \quad \text{Prop. 4.6.10} \\ &\iff \forall \sigma \in \Sigma - D \; \exists X \in SV(\Sigma - D) : \; \sigma'(X) \neq \sigma(X) \\ &\iff \forall \sigma \in \Sigma - D \; \exists X \in SV(D) : \; \sigma'(X) \neq \sigma(X) \end{aligned}$$ □

Der Darstellungssatz hat weitreichende Konsequenzen. Insbesondere folgt aus dem Darstellungssatz die Existenz von konjunktiven und disjunktiven Normalformen (Satz 4.8.7).

4.7 Klauseldarstellung

Wir wenden uns jetzt der Vereinfachung von Formeln zu. Die Vereinfachung einer Formel A bedeutet, eine möglichst einfache Formel B zu finden, die dieselbe Menge wie A darstellt. Wenn A beispielsweise die leere Menge darstellt, sollte die Vereinfachung von A die Formel 0 liefern.

Wir müssen also eine Menge von „einfachsten“ Formeln definieren. Dabei soll jede Denotation durch genau eine „einfachste“ Formel darstellbar sein.

Die Formeln von AL sind keine geeignete Datenstruktur für dieses Vorhaben. Wir weichen daher auf ein besser geeignetes Darstellungssystem für die Denotationen

$$
\begin{array}{rcll}
C, D & \in & Cla = \mathcal{P}_{fin}(For) & \quad \textit{Klausel} \\
S & \in & \mathcal{P}(Cla) & \quad \textit{Klauselmenge}
\end{array}
$$

$$\mathcal{D} \in \mathcal{P}(Cla) \to \mathcal{P}(\Sigma)$$

$$\mathcal{D}[\![S]\!] = \{\, \sigma \in \Sigma \mid \exists C \in S \;\; \forall A \in C\colon \;\; \sigma \in \mathcal{M}[\![A]\!] \,\}$$

$$\mathcal{K} \in \mathcal{P}(Cla) \to \mathcal{P}(\Sigma)$$

$$\mathcal{K}[\![S]\!] = \{\, \sigma \in \Sigma \mid \forall C \in S \;\; \exists A \in C\colon \;\; \sigma \in \mathcal{M}[\![A]\!] \,\}$$

Abbildung 4.6: Disjunktive und konjunktive Klauseldarstellung

aus, das als *Klauseldarstellung* bekannt ist. Am Anfang einer Vereinfachung steht die Überführung einer Formel in Klauseldarstellung. Danach findet die eigentliche Vereinfachung im Rahmen der Klauseldarstellung statt. Zunächst ermittelt man eine sogenannte Normalform, danach geht man zu einer sogenannten Primform über. Primformen sind eindeutige und „einfachste" Darstellungen für Denotationen. Die als Ergebnis der Vereinfachung erhaltene Primform kann man dann wieder als Formel darstellen.

Die Klauseldarstellung orientiert sich an den durch Satz 4.6.11 formulierten Darstellungseigenschaften der Denotationen. Entsprechend betrachten wir zwei duale Darstellungssysteme, die wir als konjunktive und disjunktive Klauseldarstellung bezeichnen. Beim ersten Lesen sollten Sie sich nur auf eine dieser Varianten konzentrieren.

Abbildung 4.6 definiert die beiden Klauseldarstellungen. Eine *Klausel* ist eine endliche Menge von Formeln. Für Klauselnmengen geben wir zwei symmetrische Denotationsfunktionen an. Die *disjunktive Denotationsfunktion* $\mathcal{D}$ verknüpft die Klauseln einer Menge disjunktiv und die Formeln einer Klausel konjunktiv. Die *konjunktive Denotationsfunktion* $\mathcal{K}$ verknüpft die Klauseln einer Menge konjunktiv und die Formeln einer Klausel disjunktiv.

Proposition 4.7.1 *Sei S eine Klauselmenge. Dann:*

$$\mathcal{K}[\![S]\!] = \bigcap_{C \in S} \bigcup_{A \in C} \mathcal{M}[\![A]\!] \qquad \textit{und} \qquad \mathcal{D}[\![S]\!] = \bigcup_{C \in S} \bigcap_{A \in C} \mathcal{M}[\![A]\!]$$

Das Wechseln von einer Formeldarstellung zu einer Klauseldarstellung ist einfach:

Proposition 4.7.2 $\forall A \in For\colon\ \mathcal{M}[\![A]\!] = \mathcal{D}[\![\{\{A\}\}]\!] = \mathcal{K}[\![\{\{A\}\}]\!]$.

Die Rückkehr von einer endlichen Klauseldarstellung (nur solche werden bei der Vereinfachung benötigt) zu einer Formeldarstellung ist auch einfach. Bei der disjunktiven Klauseldarstellung werden die Formeln einer Klausel konjunktiv und die den Klauseln entsprechenden Formeln disjunktiv verknüpft. Bei der konjunktiven Klauseldarstellung werden die Formeln einer Klausel disjunktiv und die den Klauseln entsprechenden Formeln konjunktiv verknüpft.

Proposition 4.7.3 *Zu jeder endlichen Klauselmenge S existieren Formeln A und B mit $\mathcal{D}[\![S]\!] = \mathcal{M}[\![A]\!]$ und $\mathcal{K}[\![S]\!] = \mathcal{M}[\![B]\!]$.*

Proposition 4.7.4 (Leere Klauselmenge) $\mathcal{D}[\![\emptyset]\!] = \emptyset$ *und* $\mathcal{K}[\![\emptyset]\!] = \Sigma$.

Proposition 4.7.5 (Leere Klausel) *Sei $\emptyset \in S \subseteq Cla$. Dann $\mathcal{D}[\![S]\!] = \Sigma$ und $\mathcal{K}[\![S]\!] = \emptyset$.*

Zwei Klauselmengen S und S' heißen *äquivalent* genau dann, wenn $\mathcal{D}[\![S]\!] = \mathcal{D}[\![S]\!]'$ und $\mathcal{K}[\![S]\!] = \mathcal{K}[\![S]\!]'$. Wir schreiben $S \models\!\!\!\!\dashv S'$ genau dann, wenn S und S' äquivalent sind.

Bereinigung

Eine Klausel C heißt *trivial* genau dann, wenn es eine Formel A gibt, so dass $A \in C$ und $\neg A \in C$. Wenn man triviale Klauseln löscht, bleiben die Denotationen einer Klauselmenge unverändert:

Proposition 4.7.6 (Triviale Klauseln) *Sei S eine Klauselmenge und C eine triviale Klausel. Dann $S \cup \{C\} \models\!\!\!\!\dashv S$.*

Eine Klausel C *subsumiert* eine Klausel D genau dann, wenn $C \subseteq D$. Die Denotationen einer Klauselmenge bleiben unverändert, wenn man subsumierte Klauseln löscht:

Proposition 4.7.7 (Subsumierte Klauseln) *Sei S eine Klauselmenge und C eine Klausel, die von einer Klausel in S subsumiert wird. Dann $S \cup \{C\} \models\!\!\!\!\dashv S$.*

Sei S eine Klauselmenge. Wir definieren die *Bereinigung von S* wie folgt:

$$Ber(S) \stackrel{\text{def}}{=} \{\, C \in S \mid C \text{ nichttrivial und } \forall D \in S\colon\ D \subseteq C \Rightarrow D = C \,\}$$

Die Bereinigung einer Klauselmenge erhält man, indem man alle trivialen und subsumierten Klauseln löscht. Eine Klauselmenge S heißt *bereinigt* genau dann, wenn $Ber(S) = S$. Offensichtlich ist die Bereinigung einer Klauselmenge S eine bereinigte Teilmenge von S.

Proposition 4.7.8 (Bereinigung) *Für jede bereinigte Klauselmenge S gilt:*

$$S = Ber(S) = Ber(\{ C \in Cla \mid \exists C' \in S\colon\ C' \subseteq C \})$$

Aus den Propositionen 4.7.6 und 4.7.7 folgt, dass Bereinigung die Denotationen einer Klauselmenge invariant lässt:

Proposition 4.7.9 (Bereinigung) *Für jede Klauselmenge S gilt: $S \models\!\!\!\dashv Ber(S)$.*

4.8 Normalformen

Ein *Literal* ist eine Formel $A \in For$, so dass eine Variable $X \in Var$ existiert mit $A = X$ oder $A = \neg X$. Ein Literal kann man sich also als eine mit einem Vorzeichen versehene Variable vorstellen. Mit *Lit* bezeichnen wir die Menge aller Literale.

Mit $|L|$ bezeichnen wir die Variable eines Literals L.

Zwei Literale L_1 und L_2 heißen *komplementär* genau dann, wenn $L_1 = \neg L_2$ oder $L_2 = \neg L_1$ gilt. Mit $\overline{L}$ bezeichnen wir das zu L komplementäre Literal. Für jedes Literal L gilt $\overline{\overline{L}} = L$.

Eine Klausel heißt *literal* genau dann, wenn sie nur Literale enthält. Eine Klauselmenge heißt *literal* genau dann, wenn sie nur literale Klauseln enthält.

Mit endlich vielen Variablen kann man offensichtlich nur endlich viele literale Klauseln bilden. Also gilt:

Proposition 4.8.1 *Sei $V \subseteq Var$ endlich. Dann:*

1. *Es gibt nur endlich viele literale Klauseln, die nur Variablen aus V enthalten.*
2. *Es gibt nur endlich viele literale Klauselmengen, die nur Variablen aus V enthalten.*

Folglich ist eine literale Klauselmenge unendlich genau dann, wenn sie unendlich viele Variablen enthält.

Eine Klausel heißt *normal* genau dann, wenn sie literal und nichttrivial ist. Eine Klauselmenge heißt *normal* genau dann, wenn sie nur normale Klauseln enthält. Die Menge aller normalen Klauseln bezeichnen wir mit *NCla*.

Proposition 4.8.2 (Normale Klauseln) *Sei C eine normale Klausel. Dann ist*

$$\sigma_1 \stackrel{\text{def}}{=} \lambda\, X \in \mathit{Var}.\ \text{if } X \in C \text{ then } 1 \text{ else } 0$$

eine Belegung mit $\sigma_1 \in \mathcal{D}[\![\{C\}]\!]$. Weiter ist

$$\sigma_2 \stackrel{\text{def}}{=} \lambda\, X \in \mathit{Var}.\ \text{if } X \in C \text{ then } 0 \text{ else } 1$$

eine Belegung mit $\sigma_2 \notin \mathcal{K}[\![\{C\}]\!]$.

Proposition 4.8.3 *Für jede normale Klauselmenge S gilt:*

$$\mathcal{D}[\![S]\!] = \emptyset \iff \mathcal{K}[\![S]\!] = \Sigma \iff S = \emptyset$$

Eine *Normalform* ist eine endliche und normale Klauselmenge. Die Menge aller Normalformen bezeichnen wir mit *NF*.

Eine *disjunktive Normalform* für eine Denotation $D \in \mathit{Den}$ [Formel $A \in \mathit{For}$] ist eine Normalform S mit $\mathcal{D}[\![S]\!] = D$ [$\mathcal{D}[\![S]\!] = \mathcal{M}[\![A]\!]$]. Eine *konjunktive Normalform* für eine Denotation $D \in \mathit{Den}$ [Formel $A \in \mathit{For}$] ist eine Normalform S mit $\mathcal{K}[\![S]\!] = D$ [$\mathcal{D}[\![S]\!] = \mathcal{M}[\![A]\!]$]. Offensichtlich gilt:

Proposition 4.8.4 *Sei S eine Normalform. Dann ist S eine disjunktive Normalform für $\mathcal{D}[\![S]\!]$ und eine konjunktive Normalform für $\mathcal{K}[\![S]\!]$.*

Proposition 4.8.5 *Die leere Menge ist die einzige disjunktive Normalform für unerfüllbare Formeln und die einzige konjunktive Normalform für gültige Formeln.*

Beweis Folgt aus den Propositionen 4.8.3 und 4.8.4. □

Proposition 4.8.6 *Sei S_1 eine konjunktive Normalform für eine Formel A_1 und S_2 eine konjunktive Normalform für eine Formel A_2. Dann ist $S_1 \cup S_2$ eine konjunktive Normalform für $A_1 \wedge A_2$.*

Wir zeigen jetzt, dass jede Denotation durch eine disjunktive und durch eine konjunktive Normalform darstellen lässt. Der Schlüssel dazu ist die Tatsache, dass jede Denotation nur endlich viele signifikante Variablen hat und sich daher gemäß dem Darstellungssatz 4.6.11 mit endlichen Schnitten und Vereinigungen darstellen lässt.

Satz 4.8.7 (Existenz von Normalformen) *Für jede Denotation $D \in Den$ gilt:*

1. $\{\, \{\, L \in Lit \mid \sigma \in \mathcal{M}[\![L]\!]$ *und* $|L| \in SV(D) \,\} \mid \sigma \in D \,\}$ *ist eine disjunktive Normalform für D, die nur signifikante Variablen von D enthält.*
2. $\{\, \{\, L \in Lit \mid \sigma \notin \mathcal{M}[\![L]\!]$ *und* $|L| \in SV(D) \,\} \mid \sigma \in \Sigma - D \,\}$ *ist eine konjunktive Normalform für D, die nur signifikante Variablen von D enthält.*

Beweis Da jede Denotation nur endlich viele signifikante Variablen hat (Proposition 4.6.8), sind die angegebenen normalen Klauselmengen endlich (Proposition 4.8.1). Es handelt sich also um Normalformen. Das es sich jeweils um Darstellungen der Denotation D handelt folgt aus dem Darstellungssatz 4.6.11. □

Korollar 4.8.8 *Sei $V \subseteq Var$ endlich. Dann existieren nur endlich viele Denotation $D \in Den$ mit $SV(D) \subseteq V$.*

Beweis Satz 4.8.7 sagt, dass sich jede Denotation $D \in Den$ mit $SV(D) \subseteq V$ durch eine Normalform darstellen lässt, die nur Variablen aus V enthält. Da es nur endlich viele solcher Normalformen gibt (Proposition 4.8.1), gibt es auch nur endlich viele darstellbare Denotationen. □

Lemma 4.8.9 (Tangenten) *Sei $D \in Den$. Dann:*

1. $D = \mathcal{K}[\![\{\, C \in NCla \mid D \subseteq \mathcal{K}[\![\{C\}]\!] \,\}]\!]$.
2. $D = \mathcal{D}[\![\{\, C \in NCla \mid \mathcal{D}[\![\{C\}]\!] \subseteq D \,\}]\!]$.

Beweis Wir zeigen nur die erste Behauptung. Die zweite Behauptung folgt mit einem dualen Argument.

„$\subseteq$“. Sei $\sigma \in D$. Sei $C \in NCla$ mit $D \subseteq \mathcal{K}[\![\{C\}]\!]$. Dann existiert $L \in C$ mit $\sigma \in \mathcal{M}[\![L]\!]$. Also $\sigma \in \mathcal{K}[\![\{\, C \in NCla \mid D \subseteq \mathcal{K}[\![\{C\}]\!] \,\}]\!]$.

„$\supseteq$“. Sei S eine konjunktive Normalform für D (existiert nach Satz 4.8.7). Dann $S \subseteq \{\, C \in NCla \mid D \subseteq \mathcal{K}[\![\{C\}]\!] \,\}$. Also

$$\mathcal{K}[\![\{\, C \in NCla \mid D \subseteq \mathcal{K}[\![\{C\}]\!] \,\}]\!] \subseteq \mathcal{K}[\![S]\!] = D$$ □

4.9 Dualität

Betrachten Sie nochmal die gültigen Booleschen Gleichungen in Abbildung 4.2. Das symmetrische Verhalten von Konjunktion und Disjunktion sowie 0 und 1 wird als *Dualität* bezeichnet. Im Folgenden zeigen wir einige Dualitätseigenschaften.

Wir definieren:

$$\widehat{\ } \in \Sigma \to \Sigma$$
$$\widehat{\sigma}(X) = 1 - \sigma(X)$$

Wir bezeichnen $\widehat{\sigma}$ als *die zu σ duale Belegung*.

Proposition 4.9.1 $\forall \sigma \in \Sigma\colon\ \widehat{\widehat{\sigma}} = \sigma$. *Also ist* $\widehat{\ } \in \Sigma \to \Sigma$ *eine Bijektion.*

Lemma 4.9.2 *Sei S eine literale Klauselmenge. Dann:*

$$\forall \sigma \in \Sigma\colon\quad \sigma \in \mathcal{K}[\![S]\!] \iff \widehat{\sigma} \notin \mathcal{D}[\![S]\!]$$

Beweis Sei $\sigma \in \Sigma$. Dann:

$$\begin{aligned}
\sigma \in \mathcal{K}[\![S]\!] &\iff \forall C \in S\ \exists L \in C\colon\ \mathcal{F}[\![L]\!]\sigma = 1 \\
&\iff \neg\, \exists C \in S\ \forall L \in C\colon\ \mathcal{F}[\![L]\!]\sigma \neq 1 \\
&\iff \neg\, \exists C \in S\ \forall L \in C\colon\ \mathcal{F}[\![L]\!]\widehat{\sigma} = 1 \\
&\iff \widehat{\sigma} \notin \mathcal{D}[\![S]\!]
\end{aligned}$$

□

Satz 4.9.3 *Seien S und S′ literale Klauselmengen. Dann:*

$$\mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![S']\!] \iff \mathcal{D}[\![S']\!] \subseteq \mathcal{D}[\![S]\!]$$

Beweis

$$\begin{aligned}
\mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![S']\!] &\iff \forall \sigma \in \Sigma\colon\ \sigma \in \mathcal{K}[\![S]\!] \Rightarrow \sigma \in \mathcal{K}[\![S']\!] \\
&\iff \forall \sigma \in \Sigma\colon\ \widehat{\sigma} \notin \mathcal{D}[\![S]\!] \Rightarrow \widehat{\sigma} \notin \mathcal{D}[\![S']\!] && \text{Lemma 4.9.2} \\
&\iff \forall \sigma \in \Sigma\colon\ \widehat{\sigma} \in \mathcal{D}[\![S']\!] \Rightarrow \widehat{\sigma} \in \mathcal{D}[\![S]\!] \\
&\iff \forall \sigma \in \Sigma\colon\ \sigma \in \mathcal{D}[\![S']\!] \Rightarrow \sigma \in \mathcal{D}[\![S]\!] && \widehat{\ }\ \text{bijektiv} \\
&\iff \mathcal{D}[\![S']\!] \subseteq \mathcal{D}[\![S]\!]
\end{aligned}$$

□

Die durch den Satz formulierte Dualitätseigenschaft gilt nicht für beliebige Klauselmengen. Betrachten Sie dazu die Klauselmengen $S = \{\{0\}, \{X\}\}$ und $S' = \{\emptyset\}$. Dann $\mathcal{K}[\![S]\!] = \mathcal{K}[\![S']\!]$ und $\mathcal{D}[\![S]\!] \neq \mathcal{D}[\![S']\!]$.

Korollar 4.9.4 *Seien S und S′ literale Klauselmengen. Dann:*

$$\mathcal{K}[\![S]\!] = \mathcal{K}[\![S']\!] \iff \mathcal{D}[\![S]\!] = \mathcal{D}[\![S']\!] \iff S \models\!\!\!\!\!\dashv S'$$

Wir betrachten jetzt Dualitätseigenschaften von Formeln. Obwohl unsere Formeln ohne Disjunktion gebildet sind, können wir die Essenz dieser Eigenschaften formulieren.

Wir definieren:

$$
\begin{aligned}
\widehat{\ } \in \mathit{For} &\to \mathit{For} \\
\widehat{X} &= X \\
\widehat{\neg A} &= \neg \widehat{A} \\
\widehat{A \wedge B} &= \widehat{A} \vee \widehat{B}
\end{aligned}
$$

Wir bezeichnen $\widehat{A}$ als *die zu A duale Formel*.

Proposition 4.9.5 *Seien $A, B \in For$. Dann:*

$$
\begin{aligned}
\widehat{0} &\models\!\!\!\!\dashv 1 \\
\widehat{1} &\models\!\!\!\!\dashv 0 \\
\widehat{A \vee B} &\models\!\!\!\!\dashv \widehat{A} \wedge \widehat{B} \\
\widehat{A \Rightarrow B} &\models\!\!\!\!\dashv \neg(\widehat{B} \Rightarrow \widehat{A}) \\
\widehat{A \Leftrightarrow B} &\models\!\!\!\!\dashv \neg(\widehat{A} \Leftrightarrow \widehat{B})
\end{aligned}
$$

Beweis Folgt mit dem Ersetzungssatz und den bekannten Äquivalenzen. □

Lemma 4.9.6 $\forall A \in \mathit{For}\ \forall \sigma \in \Sigma\colon\ \mathcal{F}[\![\widehat{A}]\!]\widehat{\sigma} = 1 - \mathcal{F}[\![A]\!]\sigma$.

Beweis Durch strukturelle Induktion über $A \in \mathit{For}$. □

Proposition 4.9.7 $\forall A \in \mathit{For}\colon\ \widehat{\widehat{A}} \models\!\!\!\!\dashv A$.

Beweis Sei $A \in \mathit{For}$ und $\sigma \in \Sigma$. Dann folgt mit Lemma 4.9.6:

$$\mathcal{F}[\![\widehat{\widehat{A}}]\!]\sigma = 1 - \mathcal{F}[\![\widehat{A}]\!]\widehat{\sigma} = 1 - (1 - \mathcal{F}[\![A]\!]\widehat{\widehat{\sigma}})) = \mathcal{F}[\![A]\!]\sigma$$ □

Satz 4.9.8 $\forall A, B \in \mathit{For}\colon\ A \models\!\!\!\!\dashv B \iff \widehat{A} \models\!\!\!\!\dashv \widehat{B}$.

Beweis Seien $A, B \in \mathit{For}$. Dann:

$$
\begin{aligned}
A \models\!\!\!\!\dashv B &\iff \forall \sigma \in \Sigma\colon\ \mathcal{F}[\![A]\!]\sigma = \mathcal{F}[\![B]\!]\sigma \\
&\iff \forall \sigma \in \Sigma\colon\ \mathcal{F}[\![\widehat{A}]\!]\widehat{\sigma} = \mathcal{F}[\![\widehat{B}]\!]\widehat{\sigma} && \text{Lemma 4.9.6} \\
&\iff \forall \sigma \in \Sigma\colon\ \mathcal{F}[\![\widehat{A}]\!]\sigma = \mathcal{F}[\![\widehat{B}]\!]\sigma && \widehat{\ }\ \text{bijektiv} \\
&\iff \widehat{A} \models\!\!\!\!\dashv \widehat{B}
\end{aligned}
$$ □

Korollar 4.9.9 $\forall A \in \mathit{For}\colon$ *A gültig* $\iff$ $\neg\widehat{A}$ *gültig.*

4.10 Primformen

Wir definieren jetzt eine Klasse von Normalformen, die wir als Primformen bezeichnen. Wir werden zeigen, dass zu jeder Normalform genau eine äquivalente Primform existiert. Damit haben wir eindeutige Darstellungen für die Denotationen von AL. Der in Abschnitt 4.3 skizzierte Interpreter berechnet konjunktive Primformen, die er als Formeln darstellt.

Primformen haben wichtige Anwendungen beim Schaltkreisentwurf und in der Künstlichen Intelligenz. Bisher gibt es allerdings noch kein Lehrbuch, das Primformen systematisch behandelt.[3]

Proposition 4.10.1 *Sei $D \in$ Den und C eine literale Klausel. Weiter sei $C' = \{\, L \in C \mid |L| \in SV(D) \,\}$. Dann:*

1. $D \subseteq \mathcal{K}[\![\{C\}]\!] \;\Rightarrow\; D \subseteq \mathcal{K}[\![\{C'\}]\!]$.
2. $\mathcal{D}[\![\{C\}]\!] \subseteq D \;\Rightarrow\; \mathcal{D}[\![\{C'\}]\!] \subseteq D$.

Beweis Wir zeigen nur die erste Behauptung. Die zweite Behauptung folgt mit einem dualen Argument.

Sei $D \subseteq \mathcal{K}[\![\{C\}]\!]$ und $\sigma \in D$. Wir müssen zeigen, dass $\sigma \in \mathcal{K}[\![\{C'\}]\!]$. Wir wählen ein $\sigma' \in \Sigma$ wie folgt:

- $\forall X \in SV(D)\colon\; \sigma'(X) = \sigma(X)$.
- $\forall L \in C\colon\; |L| \notin SV(D) \Rightarrow \mathcal{F}[\![L]\!]\sigma' = 0$.

Da $\sigma \in D$, gilt $\sigma' \in D$. Also $\sigma' \in \mathcal{K}[\![\{C\}]\!]$, da $D \subseteq \mathcal{K}[\![\{C\}]\!]$. Also existiert ein $L \in C$ mit $\mathcal{F}[\![L]\!]\sigma = 1$. Aus der Konstruktion von σ' folgt, dass $L \in C'$ und $\mathcal{F}[\![L]\!]\sigma = \mathcal{F}[\![L]\!]\sigma' = 1$. Also $\sigma \in \mathcal{K}[\![\{C'\}]\!]$. □

Eine Normalform S heißt *Primform* genau dann, wenn

$$S = Ber(\{\, C \in NCla \mid \mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![\{C\}]\!] \,\})$$

gilt. Die Menge aller Primformen bezeichnen wir mit PF.

Proposition 4.10.2 (Eindeutigkeit von Primformen) *Seien S und S' Primformen. Dann gilt:*

$$S = S' \iff \mathcal{K}[\![S]\!] = \mathcal{K}[\![S']\!] \iff \mathcal{D}[\![S]\!] = \mathcal{D}[\![S']\!] \iff S \models\!\!\!\!\!= S'$$

[3] Eine interessante Arbeit über die Berechnung von Primformen ist: Alex Kean und George Tsiknis, An Incremental Method for Generating Prime Implicants/Implicates. Journal of Symbolic Computation (1990) 9, 185–206.

Beweis Wegen Korollar 4.9.4 genügt es zu zeigen, dass

$$\mathcal{K}[\![S]\!] = \mathcal{K}[\![S']\!] \;\Rightarrow\; S = S'$$

gilt. Sei $\mathcal{K}[\![S]\!] = \mathcal{K}[\![S']\!]$. Dann:

$$\begin{aligned} S &= Ber(\{\, C \in NCla \mid \mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![\{C\}]\!] \,\}) && S \text{ Primform} \\ &= Ber(\{\, C \in NCla \mid \mathcal{K}[\![S']\!] \subseteq \mathcal{K}[\![\{C\}]\!] \,\}) && \mathcal{K}[\![S]\!] = \mathcal{K}[\![S']\!] \\ &= S' && S' \text{ Primform} \end{aligned}$$ □

Eine *disjunktive Primform* für eine Denotation $D \in Den$ [Formel $A \in For$] ist eine Primform S mit $\mathcal{D}[\![S]\!] = D$ [$\mathcal{D}[\![S]\!] = \mathcal{M}[\![A]\!]$]. Eine *konjunktive Primform* für eine Denotation $D \in Den$ [Formel $A \in For$] ist eine Primform S mit $\mathcal{K}[\![S]\!] = D$ [$\mathcal{D}[\![S]\!] = \mathcal{M}[\![A]\!]$]. Aus der obigen Proposition folgt, dass jede Denotation höchstens eine disjunktive und höchstens eine konjunktive Primform hat.[4]

Proposition 4.10.3 *Sei S eine Primform. Dann ist S die konjunktive Primform für $\mathcal{K}[\![S]\!]$ und die disjunktive Primform für $\mathcal{D}[\![S]\!]$.*

Beweis Trivial. Vergleiche Propostion 4.8.4. □

Satz 4.10.4 (Existenz von Primformen) *Sei $D \in Den$. Dann:*

1. *$Ber(\{\, C \in NCla \mid D \subseteq \mathcal{K}[\![\{C\}]\!] \,\})$ ist die konjunktive Primform für D.*
2. *$Ber(\{\, C \in NCla \mid \mathcal{D}[\![\{C\}]\!] \subseteq D \,\})$ ist die disjunktive Primform für D.*

Beweis Wir zeigen nur die erste Behauptung. Die zweite Behauptung folgt mit einem dualen Argument.

Sei $S = Ber(\{\, C \in NCla \mid D \subseteq \mathcal{K}[\![\{C\}]\!] \,\})$. Mit Proposition 4.10.1 folgt, dass S nur signifikante Variablen von D enthält. Also ist S endlich und folglich eine Normalform. Mit dem Tangentenlemma 4.8.9 und Proposition 4.7.9 folgt $\mathcal{K}[\![S]\!] = D$. Also ist S die konjunktive Primform für D. □

Zu jeder Normalform existiert also genau eine äquivalente Primform. Wir sprechen von der *Primform einer Normalform*.

Die nächste Proposition sagt, dass eine Primform eine normale Klausel logisch impliziert genau dann, wenn sie die Klausel subsumiert.

Proposition 4.10.5 *Sei S eine Primform und C eine normale Klausel. Dann:*

$$\mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![\{C\}]\!] \iff \exists C' \in S\colon\; C' \subseteq C \iff \mathcal{D}[\![\{C\}]\!] \subseteq \mathcal{D}[\![S]\!]$$

[4] Die Klauseln in der konjunktiven Primform einer Formel A werden als die *Primimplikate* von A bezeichnet. Die Klauseln in der disjunktiven Primform einer Formel A werden als die *Primimplikanten* von A bezeichnet.

Beweis Es genügt die erste Äquivalenz zu zeigen. Die zweite Äquivalenz folgt dann mit Proposition 4.9.3.

„$\Rightarrow$“. Sei $\mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![C]\!]$. Da S eine Primform ist, gilt $S = Ber(\{\, C \in NCla \mid \mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![\{C\}]\!] \,\})$. Also existiert $C' \in S$ mit $C' \subseteq C$.

„$\Leftarrow$“. Sei $C' \in S$ mit $C' \subseteq C$. Sei $\sigma \in \mathcal{K}[\![S]\!]$. Wir müssen zeigen, dass $\sigma \in \mathcal{K}[\![\{C\}]\!]$. Da $C' \in S$, existiert ein Literal $L' \in C'$ mit $\sigma \in \mathcal{M}[\![L]\!]$. Da $C' \subseteq C$, folgt $\sigma \in \mathcal{K}[\![\{C\}]\!]$. □

Lemma 4.10.6 *Sei S eine bereinigte Normalform mit*

$$\forall C \in NCla\colon \quad \mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![C]\!] \;\Rightarrow\; \exists C' \in S\colon\; C' \subseteq C$$

Dann ist S eine Primform.

Beweis Sei S eine bereinigte Normalform wie verlangt. Dann gilt

$$(*) \quad \mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![C]\!] \iff \exists C' \in S\colon\; C' \subseteq C$$

Also folgt:

$$\begin{aligned} S &= Ber(S) && S \text{ bereinigt} \\ &= Ber(\{\, C \in NCla \mid \exists C' \in S\colon\; C' \subseteq C \,\}) \\ &= Ber(\{\, C \in NCla \mid \mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![\{C\}]\!] \,\}) && (*) \end{aligned}$$

□

Proposition 4.10.7 *Sei S eine Primform und X eine Variable. Dann sind die folgen Aussagen äquivalent:*

1. *X kommt in S vor.*
2. *X ist eine signifikante Variable von* $\mathcal{K}[\![S]\!]$.
3. *X ist eine signifikante Variable von* $\mathcal{D}[\![S]\!]$.

Wir wenden uns jetzt der Berechnung von Primformen zu. Abbildung 4.7 gibt einen Überblick. Zunächst haben wir Formeln, Normalformen und Primformen als mögliche Darstellungen für die Denotationen von AL. Die Primformen sind eindeutige Darstellungen, das heißt, jede Denotation kann nur durch genau eine konjunktive und genau eine disjunktive Primform dargestellt werden. Zunächst werden wir zwei als Vereinfachung bezeichnete Verfahren kennenlernen, mit denen wir zu einer Formel disjunktive und konjunktive Normalformen berechnen können. Danach führen wir ein als Resolution bezeichnetes Verfahren ein, mit dem wir Normalformen in äquivalente Primformen überführen können.

$$
\begin{tikzcd}
{} \arrow[rr, "\text{Vereinfachung}"] & & {} \arrow[rr, "\text{Resolution}"] & & {} \\
\mathit{For} \arrow[drr, "\mathcal{M}"] & & \mathit{NF} \arrow[d, "\mathcal{D},\ \mathcal{K}"] & \supseteq & \mathit{PF} \arrow[dll, leftrightarrow] \\
& & \mathit{Den} & &
\end{tikzcd}
$$

Abbildung 4.7: Berechnung von Normalformen und Primformen

4.11 Disjunktive Vereinfachung

Für Klauselmengen existieren Vereinfachungsregeln, mit denen man jede Klauselmenge auf eine normale Klauselmenge mit derselben Denotation überführen kann. Je nachdem, ob man die disjunktive oder die konjunktive Denotation invariant halten will, muss man verschiedene (aber symmetrische) Vereinfachungsregeln benutzen.

Wir unterscheiden zwischen *interner und externer Syntax*. Die interne Syntax ist durch die Formeln in *For* gegeben. Die externe Syntax ist durch Klauseln und Klauselmengen gegeben. Die Vereinfachungsregeln ersetzen interne Syntax schrittweise durch externe Syntax. Bei normalen Klauselmengen kommt die interne Syntax nur noch in der Form von negierten Variablen vor.

Abbildung 4.8 zeigt die disjunktiven Vereinfachungsregeln für Klauselmengen in schematischer Darstellung. Dabei ist S, C als Abkürzung für die Klauselmenge $S \cup \{C\}$ und C, A als Abkürzung für die Klausel $C \cup \{A\}$ zu verstehen. Wenn die Notationen S, C beziehungsweise C, A auf der linken Seite einer Vereinfachungsregel benutzt werden, soll zudem $C \notin S$ beziehungsweise $A \notin C$ gelten.

Die erste Regel eliminiert Doppelnegationen, die bei Formeln in Klauseln ganz oben auftreten. Die zweite Regel ersetzt eine konjunktive Formel in einer Klausel durch ihre zwei Konstituenten. Die dritte Regel eliminiert triviale Klauseln. Die vierte Regel unterscheidet sich von den bisherigen, da sie eine Klausel durch zwei Klauseln ersetzt. Sie basiert auf der Äquivalenz

$$A \wedge \neg(A_1 \wedge A_2) \models\!\!\!\models (A \wedge \neg A_1) \vee (A \wedge \neg A_2)$$

Ohne die in Abbildung 4.8 verwendeten Abkürzungen können wir die disjunktiven Vereinfachungsregeln wie folgt definieren: Eine Klauselmenge S kann zu einer Klauselmenge

$$
\begin{array}{lrcl}
(1) & S,\ (C,\ \neg\neg A) & \stackrel{\mathrm{d}}{\to} & S,\ (C,\ A) \\
(2) & S,\ (C,\ A_1 \wedge A_2) & \stackrel{\mathrm{d}}{\to} & S,\ (C,\ A_1,\ A_2) \\
(3) & S,\ (C,\ A,\ \neg A) & \stackrel{\mathrm{d}}{\to} & S \\
(4) & S,\ (C,\ \neg(A_1 \wedge A_2)) & \stackrel{\mathrm{d}}{\to} & S,\ (C,\ \neg A_1),\ (C,\ \neg A_2)
\end{array}
$$

Abbildung 4.8: Die disjunktiven Vereinfachungsregeln

1. $(S - \{C\}) \cup \{(C - \{\neg\neg A\}) \cup \{A\}\}$ vereinfacht werden, wenn $C \in S$ und $\neg\neg A \in C$.
2. $(S - \{C\}) \cup \{(C - \{A_1 \wedge A_2\}) \cup \{A_1, A_2\}\}$ vereinfacht werden, wenn $C \in S$ und $A_1 \wedge A_2 \in C$.
3. $S - C$ vereinfacht werden, wenn $C \in S$ und C trivial ist.
4. $(S - \{C\}) \cup \{(C - \{\neg(A_1 \wedge A_2)\}) \cup \{\neg A_1\},\ (C - \{\neg(A_1 \wedge A_2)\}) \cup \{\neg A_2\}\}$ vereinfacht werden, wenn $C \in S$ und $\neg(A_1 \wedge A_2) \in C$.

Wir schreiben $S \stackrel{\mathrm{d}}{\to} S'$ genau dann, wenn die Klauselmenge S' aus der Klauselmenge S durch die Anwendung einer disjunktiven Vereinfachungsregel erhalten werden kann. Die wesentlichen Eigenschaften der disjunktiven Vereinfachungsregeln werden durch den folgenden Satz formuliert.

Satz 4.11.1 (Disjunktive Vereinfachung) *Sei S eine Klauselmenge. Dann:*

1. *Wenn $S \stackrel{\mathrm{d}}{\to} S'$, dann $\mathcal{D}[\![S]\!] = \mathcal{D}[\![S']\!]$.*
2. *S ist normal genau dann, wenn es kein S' mit $S \stackrel{\mathrm{d}}{\to} S'$ gibt.*
3. *Wenn S endlich ist, dann gibt es keine unendliche Kette*

$$S \stackrel{\mathrm{d}}{\to} S_1 \stackrel{\mathrm{d}}{\to} S_2 \stackrel{\mathrm{d}}{\to} S_3 \stackrel{\mathrm{d}}{\to} \cdots$$

Beweis Die ersten zwei Behauptungen des Satzes sind leicht zu verifizieren. Für die dritte Behauptung machen wir uns klar, dass die Vereinfachungsregeln eine Klausel entweder eliminieren oder durch ein oder zwei echt kleinere Klauseln ersetzen. Als Größe einer Klausel wählen wir

$$|C| = \sum_{A \in C} |A|$$

wobei die Größe einer Formel rekursiv wie folgt definiert ist:

$$\begin{aligned} |X| &= 1 \\ |\neg A| &= 1 + |A| \\ |A_1 \wedge A_2| &= 1 + |A_1| + |A_2| \end{aligned}$$

□

Für die Terminierung der Vereinfachungsregeln (Teil (3) des Satzes) haben wir ein Argument verwendet, das eine explizite Formulierung verdient:

Lemma 4.11.2 (Terminierung) *Sei M eine Menge, $\gamma \in M \to \mathbb{N}$ und $\succ \subseteq \mathcal{P}_{fin}(M) \times \mathcal{P}_{fin}(M)$. Wenn für alle $U, V \in \mathcal{P}_{fin}(M)$*

$$U \succ V \;\Rightarrow\; \exists u \in U - V \;\forall v \in V - U : \;\gamma(u) > \gamma(v)$$

gilt, dann gibt es keine unendliche Kette $U_1 \succ U_2 \succ U_3 \succ \cdots$.

Beweis Es erfordert etwas Aufwand, das Lemma ausgehend von einfacheren Tatsachen zu beweisen. Das Lemma folgt aus einem Ergebnis für wohlfundierte Ordnungen auf Multimengen (siehe z.B. [Baader/Nipkow, Kapitel 2.5]). □

Die disjunktiven Vereinfachungsregeln liefern ein Berechnungsverfahren für disjunktive Normalformen und ein Entscheidungsverfahren für die Unerfüllbarkeit von Formeln. Gegeben eine Formel A, wenden wir auf $\{\{A\}\}$ solange disjunktive Vereinfachungsregeln an, bis keine mehr anwendbar ist. Satz 4.11.1 sagt uns, dass wir dann eine disjunktive Normalform für A erreicht haben. Proposition 4.8.3 sagt uns, dass A genau dann unerfüllbar ist, wenn die disjunktive Normalform leer ist.

Korollar 4.11.3 *Zu jeder endlichen Klauselmenge kann man mithilfe der disjunktiven Vereinfachungsregeln eine disjunktive Normalform berechnen.*

Wir schreiben $S \stackrel{\mathrm{d}}{\to}^* S'$ genau dann, wenn es eine Kette wie folgt gibt ($n \geq 1$):

$$S = S_1 \stackrel{\mathrm{d}}{\to} S_2 \stackrel{\mathrm{d}}{\to} \ldots \stackrel{\mathrm{d}}{\to} S_n = S'$$

Korollar 4.11.4 *Sei $A \in For$. Dann: A unerfüllbar $\iff \{\{A\}\} \stackrel{\mathrm{d}}{\to}^* \emptyset$.*

Abbildung 4.9 zeigt zusätzliche Vereinfachungsregeln, die die Berechnung der disjunktiven Normalform beschleunigen, aber nichts an den in Satz 4.11.1 formulierten Eigenschaften ändern.

$$
\begin{array}{rcll}
S,\ C,\ C' & \stackrel{\mathrm{d}}{\to} & S,\ C & \text{falls } C \subseteq C' \\
S,\ (C,\ 0) & \stackrel{\mathrm{d}}{\to} & S & \\
S,\ (C,\ 1) & \stackrel{\mathrm{d}}{\to} & S,\ C & \\
S,\ (C,\ A_1 \vee A_2) & \stackrel{\mathrm{d}}{\to} & S,\ (C,\ A_1),\ (C,\ A_2) & \\
S,\ (C,\ \neg(A_1 \vee A_2)) & \stackrel{\mathrm{d}}{\to} & S,\ (C,\ \neg A_1,\ \neg A_2) & \\
S,\ (C,\ A_1 \Rightarrow A_2) & \stackrel{\mathrm{d}}{\to} & S,\ (C,\ \neg A_1),\ (C,\ A_2) & \\
S,\ (C,\ \neg(A_1 \Rightarrow A_2)) & \stackrel{\mathrm{d}}{\to} & S,\ (C,\ A_1,\ \neg A_2) &
\end{array}
$$

Abbildung 4.9: Zusätzliche disjunktive Vereinfachungsregeln

4.12 Tableau-Notation

Wir wollen jetzt genauer verstehen, wie die disjunktiven Vereinfachungsregeln arbeiten. Zunächst unterscheiden wir zwischen den *einfachen Regeln* (die ersten drei) und der *verzweigenden Regel* (die vierte). Die verzweigende Regel ist die einzige Regel, die die Anzahl der Klauseln erhöhen kann. Für die meisten Klauselmengen führt die verzweigende Regel zu einem explosiven Anwachsen der Klauseln. Wenn man disjunktive Vereinfachung effizient realisieren will, wird man die verzweigende Regel nur dann anwenden, wenn keine einfache Regel mehr anwendbar ist.

Ohne notationale Tricks ist es mühsam, eine konkrete disjunktive Vereinfachungskette aufzuschreiben. Die sogenannte Tableau-Notation erweist sich auch für größere Beispiele als handhabbar.[5]

Als Beispiel betrachten wir eine Formel A wie folgt

$$(X \vee Z) \wedge ((\neg Z \vee \neg X) \vee \neg Y) \wedge ((\neg Y \wedge X) \vee \neg Z) \wedge (\neg X \vee (Z \wedge Y))$$

Das Tableau in Abbildung 4.10 zeigt die Essenz einer disjunktiven Vereinfachungskette $\{\{A\}\} \stackrel{\mathrm{d}}{\to}^* \emptyset$. Das Tableau ist als Baum organisiert. Jede Verzweigung des Baumes repräsentiert eine Anwendung der verzweigenden Regel. Die Anwendung der einfachen Regeln erfolgt stillschweigend und ist im Tableau nicht

[5] Die Tableau-Notation geht auf Hintikka (1955) zurück und wird auch von Schütte (1956), Beth (1959) und Smullyan (1968) benutzt.

$$\begin{array}{ll}
(1) & X \vee Z \\
(2) & (\neg Z \vee \neg X) \vee \neg Y \\
(3) & (\neg Y \wedge X) \vee \neg Z \\
(4) & \neg X \vee (Z \wedge Y)
\end{array}$$

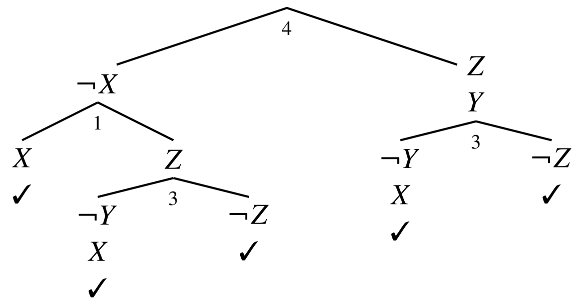

Abbildung 4.10: Ein geschlossenes Tableau

vermerkt. Die Anwendung der Regel (3) zur Elimination von trivialen Klauseln ist mit dem Symbol ✓ markiert.

Die Wurzel des Baumes ist mit den Formeln markiert, die man aus $\{\{A\}\}$ durch Anwendung der einfachen Regeln erhält. Die erste Anwendung der verzweigenden Regel eliminiert die disjunktive Formel (4). Durch die Baumrepräsentation ist es möglich, den gemeinsamen Teil der zwei neuen Klauseln nur einmal zu repräsentieren (das ist die entscheidende Idee der Tableau-Notation). Auf der rechten Seite wurde die Konjunktion sofort durch die Anwendung der entsprechenden einfachen Regel eliminiert.

Die maximalen Pfade des durch das Tableau dargestellen Baumes (das sind die Pfade von der Wurzel zu den Blättern) repräsentieren Klauseln. Offensichtlich können wir die entsprechende Klauselmenge von $\{\{A\}\}$ ausgehend mithilfe der disjunktiven Vereinfachungsregeln erreichen. Da auf jedem Pfad ein komplementäres Paar von Formeln liegt, enthält diese Klauselmenge nur triviale Klauseln. Also können wir $\{\{A\}\}$ sogar zu der leeren Klauselmenge vereinfachen. Folglich ist A unerfüllbar.

Ein Tableau heißt *vollständig* genau dann, wenn es eine Vereinfachungskette darstellt, die mit einer Normalform endet. Ein Tableau heißt *geschlossen* genau dann, wenn es eine Vereinfachungskette darstellt, die mit der leeren Normalform $\emptyset$ endet. Das ist der Fall, wenn auf jedem maximalen Pfad ein komplementäres Paar von Formeln liegt (so wie im Beispiel). Ein geschlossenes Tableau für eine Ausgangsformel A repräsentiert einen Beweis für die Unerfüllbarkeit von A.

Abbildung 4.11 zeigt ein weiteres vollständiges Tableau, das eine disjunktive Ver-

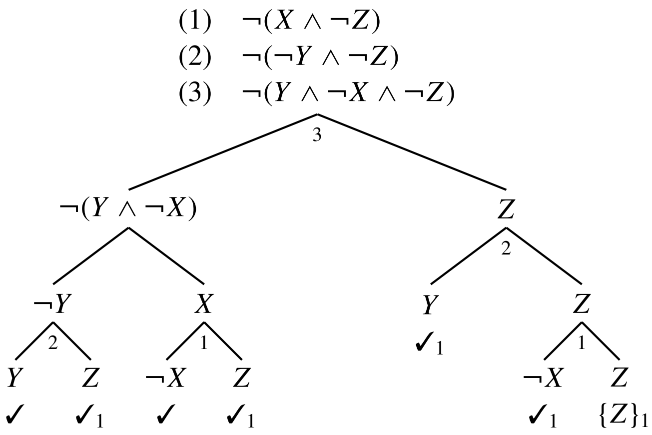

Abbildung 4.11: Ein vollständiges Tableau mit subsumierten Pfaden

einfachungskette darstellt, die ausgehend von der Klauselmenge

$$\neg(X \wedge \neg Z) \;\wedge\; \neg(\neg Y \wedge \neg Z) \;\wedge\; \neg(Y \wedge \neg X \wedge \neg Z)$$

die Normalform $\{\{Z\}\}$ liefert. Diesmal wird eine zusätzliche Vereinfachungsregel benutzt, mit der subsumierte Klauseln gelöscht werden können (die erste Regel in Abbildung 4.9). Die maximalen Pfade, die den durch die Subsumptionsregel gelöschten Klauseln entsprechen, sind durch $\checkmark_1$ markiert, wobei die subsumierende Klausel (in der Abbildung ganz rechts) durch den Index 1 identifiziert wird.

4.13 Konjunktive Vereinfachung

Die konjunktiven Vereinfachungsregeln sind symmetrisch zu den disjunktiven Vereinfachungsregeln. Sie sind in Abbildung 4.12 angegeben.

Wir schreiben $S \overset{\text{k}}{\to} S'$ genau dann, wenn die Klauselmenge S' aus der Klauselmenge S durch die Anwendung einer konjunktiven Vereinfachungsregel erhalten werden kann. Die wesentlichen Eigenschaften der konjunktiven Vereinfachungsregeln werden durch den folgenden Satz formuliert.

Satz 4.13.1 (Konjunktive Vereinfachung) *Sei S eine Klauselmenge. Dann:*

1. *Wenn $S \overset{\text{k}}{\to} S'$, dann $\mathcal{K}[\![S]\!] = \mathcal{K}[\![S']\!]$.*
2. *S ist normal genau dann, wenn es kein S' mit $S \overset{\text{k}}{\to} S'$ gibt.*

(1)	$S,\ (C,\ \neg\neg A)$	$\stackrel{\text{k}}{\to}$	$S,\ (C,\ A)$
(2)	$S,\ (C,\ \neg(A_1 \wedge A_2))$	$\stackrel{\text{k}}{\to}$	$S,\ (C,\ \neg A_1,\ \neg A_2)$
(3)	$S,\ (C,\ A,\ \neg A)$	$\stackrel{\text{k}}{\to}$	S
(4)	$S,\ (C,\ A_1 \wedge A_2)$	$\stackrel{\text{k}}{\to}$	$S,\ (C,\ A_1),\ (C,\ A_2)$

Abbildung 4.12: Die konjunktiven Vereinfachungsregeln

3. *Wenn S endlich ist, dann gibt es keine unendliche Kette*

$$S \stackrel{\text{k}}{\to} S_1 \stackrel{\text{k}}{\to} S_2 \stackrel{\text{k}}{\to} S_3 \stackrel{\text{k}}{\to} \cdots$$

Beweis Analog zu dem Satz über disjunktive Vereinfachung. □

Die konjunktiven Vereinfachungsregeln liefern ein Berechnungsverfahren für konjunktive Normalformen und ein Entscheidungsverfahren für die Gültigkeit von Formeln. Gegeben eine Formel A, wenden wir auf $\{\{A\}\}$ solange konjunktive Vereinfachungsregeln an, bis keine mehr anwendbar ist. Satz 4.13.1 sagt uns dann, dass wir eine konjunktive Normalform für A erreicht haben. Proposition 4.8.3 sagt uns, dass A genau dann gültig ist, wenn die konjunktive Normalform leer ist.

Korollar 4.13.2 *Zu jeder endlichen Klauselmenge kann man mithilfe der konjunktiven Vereinfachungsregeln eine konjunktive Normalform berechnen.*

Wir schreiben $S \stackrel{\text{k}}{\to}^* S'$ genau dann, wenn es eine Kette wie folgt gibt ($n \geq 1$):

$$S = S_1 \ \stackrel{\text{k}}{\to}\ S_2 \ \stackrel{\text{k}}{\to}\ \ldots \ \stackrel{\text{k}}{\to}\ S_n = S'$$

Korollar 4.13.3 *Sei $A \in For$. Dann: A gültig $\iff \{\{A\}\} \stackrel{\text{k}}{\to}^* \emptyset$.*

Abbildung 4.13 zeigt zusätzliche Vereinfachungsregeln, die die Berechnung der konjunktiven Normalform beschleunigen, aber nichts an den in Satz 4.13.1 formulierten Eigenschaften ändern.

Die Tableau-Notation kann natürlich auch für konjunktive Vereinfachungsketten verwendet werden.

$$
\begin{array}{rcll}
S,\ C,\ C' & \stackrel{\text{k}}{\rightarrow} & S,\ C & \text{falls } C \subseteq C' \\
S,\ (C,\ 1) & \stackrel{\text{k}}{\rightarrow} & S & \\
S,\ (C,\ 0) & \stackrel{\text{k}}{\rightarrow} & S,\ C & \\
S,\ (C,\ A_1 \vee A_2) & \stackrel{\text{k}}{\rightarrow} & S,\ (C,\ A_1,\ A_2) & \\
S,\ (C,\ (\neg(A_1 \vee A_2)) & \stackrel{\text{k}}{\rightarrow} & S,\ (C,\ \neg A_1),\ (C,\ \neg A_2) & \\
S,\ (C,\ A_1 \Rightarrow A_2) & \stackrel{\text{k}}{\rightarrow} & S,\ (C,\ \neg A_1,\ A_2) & \\
S,\ (C,\ \neg(A_1 \Rightarrow A_2)) & \stackrel{\text{k}}{\rightarrow} & S,\ (C,\ A_1),\ (C,\ \neg A_2) &
\end{array}
$$

Abbildung 4.13: Zusätzliche konjunktive Vereinfachungsregeln

4.14 Sequenten

Mit den Vereinfachungsregeln für Klauseln kann man bis auf Negation alle interne Syntax eliminieren. Wenn wir jede Klausel in einen postiven und negativen Teil aufteilen und die Formel im negativen Teil implizit mit einer Negation versehen, können wir auch interne Negationen elimieren. Diese Idee geht auf Gentzen zurück und liefert sehr elegante Vereinfachungsregeln.

Zweigeteilte Klauseln nennt man *Sequenten*. Sequenten können wir als ein Paar $\langle C, D\rangle$ aus zwei Klauseln formalisieren.

Für Sequenten betrachten wir hier nur die konjunktiven Vereinfachungsregeln.[6] Das bedeutet, dass die Formeln eines Sequenten disjunktiv verknüpft sind. Als den negativen Teil eines Sequenten wählen wir die linke Klausel. Die Denotation eines Sequenten

$$\langle\{A_1, \dots, A_m\},\ \{B_1, \dots, B_n\}\rangle$$

ist dann gleich der Denotation der Formel

$$\neg A_1 \vee \cdots \vee \neg A_m \vee B_1 \vee \cdots \vee B_n$$

Wir können die Denotation des Sequenten also auch durch die Formel

$$A_1 \wedge \cdots \wedge A_m \ \Rightarrow\ B_1 \vee \cdots \vee B_n$$

[6] Natürlich kann man genausogut die disjunktive Vereinfachungsregeln betrachten.

$$\begin{array}{rcll} C, D & \in & Cla = \mathcal{P}_{\mathit{fin}}(For) & \quad \mathit{Klausel} \\ (C \Rightarrow D) & \in & Cla \times Cla & \quad \mathit{Sequent} \\ G & \in & \mathcal{P}(Cla \times Cla) & \quad \mathit{Sequentenmenge} \end{array}$$

$$\mathcal{S} \in \mathcal{P}(Cla \times Cla) \to \mathcal{P}(\Sigma)$$

$$\mathcal{S}[\![G]\!] = \{ \sigma \in \Sigma \mid \forall (C \Rightarrow D) \in G : \\ (\forall A \in C\colon\ \sigma \in \mathcal{M}[\![A]\!]) \ \Rightarrow\ (\exists A \in D\colon\ \sigma \in \mathcal{M}[\![A]\!]) \}$$

Abbildung 4.14: Syntax und Semantik von Sequenten

beschreiben. Dieser Darstellung entsprechend schreibt man für einen Sequenten $\langle C, D\rangle$ meistens $(C \Rightarrow D)$. Diese Notation macht explizit, dass der negative Teil des Sequenten links steht.

Abbildung 4.14 definiert die Syntax und Semantik von Sequenten entsprechend den obigen Ausführungen.

Proposition 4.14.1 $\forall A \in For\colon\ \mathcal{M}[\![A]\!] = \mathcal{S}[\![\{(\emptyset \Rightarrow \{A\})\}]\!]$.

Proposition 4.14.2 (Leere Sequentenmenge) $\mathcal{S}[\![\emptyset]\!] = \Sigma$.

Proposition 4.14.3 (Leerer Sequent) *Sei* $(\emptyset \Rightarrow \emptyset) \in G \subseteq Cla \times Cla$. *Dann* $\mathcal{S}[\![G]\!] = \emptyset$.

Ein Sequent $(C \Rightarrow D)$ heißt *trivial* genau dann, wenn $C \cap D \neq \emptyset$.

Proposition 4.14.4 (Triviale Sequenten) *Sei* G *eine Sequentenmenge und* $(C \Rightarrow D)$ *ein trivialer Sequent. Dann* $\mathcal{S}[\![G - \{(C \Rightarrow D)\}]\!] = \mathcal{S}[\![G]\!]$.

Ein nichttrivialer Sequent $(C \Rightarrow D)$ heißt *normal* genau dann, wenn jede Formel in $C \cup D$ eine Variable ist. Eine Sequentenmenge G heißt *normal* genau dann, wenn jeder Sequent in G normal ist.

Proposition 4.14.5 *Für jede normale Sequentenmenge* G *gilt:*

$$\mathcal{S}[\![G]\!] = \Sigma \iff G = \emptyset$$

$$
\begin{array}{llcl}
(T) & G,\ (C, A \Rightarrow D, A) & \overset{s}{\rightarrow} & G \\
(L\neg) & G,\ (C, \neg A \Rightarrow D) & \overset{s}{\rightarrow} & G,\ (C \Rightarrow D, A) \\
(R\neg) & G,\ (C \Rightarrow D, \neg A) & \overset{s}{\rightarrow} & G,\ (C, A \Rightarrow D) \\
(L\wedge) & G,\ (C, A_1 \wedge A_2 \Rightarrow D) & \overset{s}{\rightarrow} & G,\ (C, A_1, A_2 \Rightarrow D) \\
(R\wedge) & G,\ (C \Rightarrow D, A_1 \wedge A_2) & \overset{s}{\rightarrow} & G,\ (C \Rightarrow D, A_1),\ (C \Rightarrow D, A_2)
\end{array}
$$

Abbildung 4.15: Vereinfachungsregeln für Sequenten

Abbildung 4.15 zeigt die Vereinfachungsregeln für Sequenten. Beachten Sie, dass pro Boolescher Verknüpfung immer genau zwei Regeln benötigt werden, die die Verknüpfung links beziehungsweise rechts eliminieren.

Wir schreiben $G \overset{s}{\rightarrow} G'$ genau dann, wenn die Sequentenmenge G' aus der Sequentenmenge G durch die Anwendung einer Vereinfachungsregel erhalten werden kann. Die wesentlichen Eigenschaften der Vereinfachungsregeln für Sequenten werden durch den folgenden Satz formuliert.

Satz 4.14.6 (Vereinfachung von Sequenten) *Sei G eine Sequentenmenge. Dann:*

1. *Wenn $G \overset{s}{\rightarrow} G'$, dann $\mathcal{S}[\![G]\!] = \mathcal{S}[\![G']\!]$.*
2. *G ist normal genau dann, wenn es kein G' mit $G \overset{s}{\rightarrow} G'$ gibt.*
3. *Wenn G endlich ist, dann gibt es keine unendliche Kette*

$$G \overset{s}{\rightarrow} G_1 \overset{s}{\rightarrow} G_2 \overset{s}{\rightarrow} G_3 \overset{s}{\rightarrow} \cdots$$

Beweis Analog zu dem Satz über disjunktive Vereinfachung von Klauseln. □

Der obige Satz liefert ein Entscheidungsverfahren für die Gültigkeit von Formeln. Gegeben eine Formel A, wenden wir auf $\{(\emptyset \Rightarrow \{A\})\}$ solange Vereinfachungsregeln an, bis wir eine normale Sequentenmenge G erreichen. Dann ist A gültig genau dann wenn G leer ist.

Wir schreiben $S \overset{s}{\rightarrow}^* S'$ genau dann, wenn es eine Kette wie folgt gibt ($n \geq 1$):

$$S = S_1 \overset{s}{\rightarrow} S_2 \overset{s}{\rightarrow} \ldots \overset{s}{\rightarrow} S_n = S'$$

Korollar 4.14.7 *Sei $A \in For$. Dann: A gültig $\iff \{(\emptyset \Rightarrow \{A\})\} \overset{s}{\rightarrow}^* \emptyset$.*

$$\frac{C \cap D \neq \emptyset}{\vdash (C \Rightarrow D)}$$

$$\frac{\vdash (C \Rightarrow D, A)}{\vdash (C, \neg A \Rightarrow D)} \qquad \frac{\vdash (C, A \Rightarrow D)}{\vdash (C \Rightarrow D, \neg A)}$$

$$\frac{\vdash (C, A_1, A_2 \Rightarrow D)}{\vdash (C, A_1 \wedge A_2 \Rightarrow D)} \qquad \frac{\vdash (C \Rightarrow D, A_1) \qquad \vdash (C \Rightarrow D, A_2)}{\vdash (C \Rightarrow D, A_1 \wedge A_2)}$$

Abbildung 4.16: Beweisregeln für Sequenten

Die Inferenzregeln in Abbildung 4.16 heißen Beweisregeln für Sequenten und definieren eine Menge von Sequenten. Die Beweisregeln entsprechen genau den Vereinfachungsregeln für Sequenten, wobei die linke Seite einer Vereinfachungsregel der Konklusion und die rechte Seite den Prämissen der Beweisregel entspricht. Die Beweisregeln sind also umgedrehte Vereinfachungsregeln, die alle gültigen Sequenten generieren.

Proposition 4.14.8 (Korrektheit) *Sei* $\vdash (C \Rightarrow D)$*. Dann* $\mathcal{S}[\![\{(C \Rightarrow D)\}]\!] = \Sigma$.

Beweis Durch Regelinduktion. □

Lemma 4.14.9 *Sei* $G_1 \xrightarrow{s} \ldots \xrightarrow{s} G_n$ *und* $G_n = \emptyset$*. Dann gilt* $\vdash (C \Rightarrow D)$ *für alle* $(C \Rightarrow D) \in G_1$.

Beweis Durch Induktion über n. □

Satz 4.14.10 (Vollständigkeit) *Sei* $\mathcal{S}[\![\{(C \Rightarrow D)\}]\!] = \Sigma$*. Dann* $\vdash (C \Rightarrow D)$.

Beweis Korollar 4.14.7 liefert eine Kette

$$\{(C \Rightarrow D)\} = G_1 \xrightarrow{s} \ldots \xrightarrow{s} G_n = \emptyset$$

Damit folgt $\vdash (C \Rightarrow D)$ mit dem obigen Lemma. □

4.15 Resolution

Resolution ist eine Regel, mit der zu zwei Klauseln eine logisch implizierte Klausel gebildet werden kann, die als Resolvente bezeichnet wird. Mit Resolution

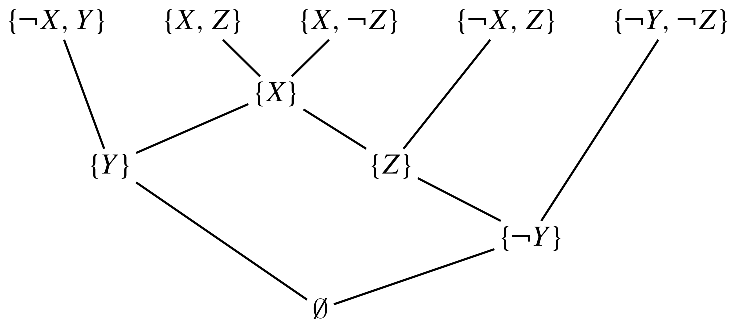

Abbildung 4.17: Ein Resolutionsgraph

bekommen wir ein Verfahren für die Berechnung von Primformen. Dieses Verfahren überführt Normalformen in Primformen, indem es subsumierte Klauseln löscht und nicht subsumierte Resolventen hinzufügt.

Eine Klausel D heißt *Resolvente von zwei Klauseln* C_1 *und* C_2 genau dann, wenn eine Formel A existiert mit:

1. $A \in C_1$ und $\neg A \in C_2$.
2. $D = (C_1 - \{A\}) \cup (C_2 - \{\neg A\})$.

Eine Klausel D heißt *Resolvente einer Klauselmenge* S genau dann, wenn C Resolvente von zwei Klauseln in S ist.

Das Hinzufügen von Resolventen lässt die Denotationen einer Klauselmenge invariant:

Proposition 4.15.1 *Sei C Resolvente einer Klauselmenge S. Dann $S \models\!\!\!\mid S \cup \{C\}$.*

Beweis Die Behauptung folgt aus der Gültigkeit der Formeln:

$$\begin{array}{ll} (1) & (A \wedge B_1) \;\vee\; (\neg A \wedge B_2) \;\Leftarrow\; B_1 \wedge B_2 \\ (2) & (A \vee B_1) \;\wedge\; (\neg A \vee B_2) \;\Rightarrow\; B_1 \vee B_2 \end{array}$$

□

Abbildung 4.17 zeigt mithilfe eines sogenannten *Resolutionsgraphen*, wie eine Klauselmenge

$$S \;=\; \{\{\neg X, Y\},\ \{X, Z\},\ \{X, \neg Z\},\ \{\neg X, Z\},\ \{\neg Y, \neg Z\}\}$$

schrittweise um insgesamt fünf Resolventen erweitert wird. Da schließlich die leere Klausel hinzugefügt wird, die alle Klausel subsumiert, haben wir mit den Propositionen 4.15.1 und 4.7.9: $S \models\!\!\!\mid \{\emptyset\}$. Also ist $\{\emptyset\}$ die Primform von S.

Eine Klausel D heißt *echte Resolvente einer Klauselmenge S* genau dann, wenn gilt:

1. D ist Resolvente von S.
2. D ist nicht trivial.
3. $D \notin S$ und D wird von keiner Klausel in S subsumiert.

Machen Sie sich klar, dass der Resolutionsgraph in Abbildung 4.17 nur echte Resolventen hinzufügt.

Eine Klauselmenge heißt *saturiert* genau dann, wenn sie keine echte Resolvente hat.

Satz 4.15.2 (Resolution) *Eine Normalform ist genau dann eine Primform, wenn sie bereinigt und saturiert ist.*

Beweis Sei S eine Primform. Dann $S = Ber(\{C \in NCla \mid \mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![\{C\}]\!]\})$. Also ist S bereinigt. Die Saturiertheit von S zeigen wir durch Widerspruch. Sei C eine echte Resolvente von S. Dann $\mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![\{C\}]\!]$. Also existiert $C' \in S$ mit $C' \subseteq C$. Widerspruch.

Sei S eine bereinigte und saturierte Normalform. Dann folgt mit Lemma 4.17.3 und Lemma 4.10.6, dass S eine Primform ist. □

4.16 Berechnung von Primformen

Seien S und S' zwei Klauselmengen. Wir schreiben $S \xrightarrow{\text{r}} S'$ genau dann, wenn eine der zwei folgenden Bedingungen gilt:

1. $S' = S \cup \{C\}$ und C ist echte Resolvente von S.
2. $S' = S - \{C\}$ und $C \in S$ und C wird von einer Klausel in S' subsumiert.

$S \xrightarrow{\text{r}} S'$ gilt also genau dann, wenn man S' aus S durch das Hinzufügen einer echten Resolvente oder durch das Löschen einer subsumierten Klausel erhalten kann.

Satz 4.16.1 (Saturation) *Sei S eine Klauselmenge. Dann:*

1. *Wenn $S \xrightarrow{\text{r}} S'$, dann $S \models\!\!\!\dashv S'$. Zudem ist S' normal, wenn S normal ist.*
2. *Eine Normalform S ist eine Primform genau dann, wenn es kein S' mit $S \xrightarrow{\text{r}} S'$ gibt.*

3. *Wenn S endlich ist, dann gibt es keine unendliche Kette*

$$S \xrightarrow{\mathrm{r}} S_1 \xrightarrow{\mathrm{r}} S_2 \xrightarrow{\mathrm{r}} S_3 \xrightarrow{\mathrm{r}} \cdots$$

Beweis Die erste Behauptung folgt aus den Propositionen 4.15.1 und 4.7.7. Die zweite Behauptung folgt aus dem Resolutionssatz.

Wir zeigen jetzt die dritte Behauptung durch Widerspruch. Sei S eine endliche Klauselmenge, für die eine unendliche Kette

$$S \xrightarrow{\mathrm{r}} S_1 \xrightarrow{\mathrm{r}} S_2 \xrightarrow{\mathrm{r}} S_3 \xrightarrow{\mathrm{r}} \cdots$$

existiert. Wir definieren

$$Cla_S \overset{\mathrm{def}}{=} \mathcal{P}(\{\, A \in For \mid \exists C \in S:\ A \in C\,\})$$

Da S endlich ist, ist auch Cla_S endlich. Cla_S ist die Menge aller Klauseln, die man mit den in den Klauseln von S enthalten Formeln bilden kann. Aus der Definition von $\xrightarrow{\mathrm{r}}$ folgt:

$$\forall i \in \mathbb{N}^+:\ S_i \subseteq Cla_S$$

Für jede Klauselmenge S_i der Kette definieren wir

$$Sub_i \overset{\mathrm{def}}{=} \{\, C \in Cla_S \mid \exists D \in S_i:\ D \subseteq C\,\}$$

Wir haben:

- $Sub_i = Sub_{i+1}$, wenn S_{i+1} aus S_i durch Löschen einer subsumierten Klausel erhalten wird.
- $Sub_i \subsetneq Sub_{i+1}$, wenn S_{i+1} aus S_i durch Hinzufügen einer echten Resolvente erhalten wird.
- $Sub_1 \subseteq Sub_2 \subseteq Sub_3 \subseteq \ \cdots\ \subseteq Cla_S$.

Da Cla_S endlich ist, kann die Kette nur endlich viele Resolventen hinzufügen. Also werden ab einem bestimmten S_k nur noch subsumierte Klauseln gelöscht. Da S_k endlich ist, können aber nur endlich viele Klauseln gelöscht werden. Damit haben wir einen Widerspruch zu der Annahme, dass die Kette unendlich ist. □

Korollar 4.16.2 *Für jede Normalform kann eine äquivalente Primform durch Hinzufügen von echten Resolventen und durch Löschen von subsumierten Klauseln berechnet werden.*

Aus dem obigen Korollar und der Berechenbarkeit von konjunktiven und disjunktiven Normalformen folgt:

Korollar 4.16.3 (Berechnung von Primformen) *Zu jeder Formel A kann die konjunktive und die disjunktive Primform berechnet werden.*

Ein Beispiel

Als Beispiel berechnen wir eine konjunktive und disjunktive Primform der Formel

$$(\neg X \Rightarrow Z) \ \wedge \ (Z \wedge X \Rightarrow \neg Y) \ \wedge \ (Y \vee \neg X \Rightarrow \neg Z) \tag{4.1}$$

Diese Formel entspricht der Konjunktion der Diätregeln in Abschnitt 4.2, wobei wir jetzt die Buchstaben X, Y und Z anstelle von B, E und F für die Variablen benutzen.

Zunächst berechnen wir eine konjunktive Normalform der Formeln. Dazu genügt es, eine konjunktive Normalform für jede der drei Teilformeln zu bestimmen und diese dann zu vereinigen. Da die Teilformeln sehr einfach sind, verzichten wir auf den Einsatz von Tableaus und arbeiten mit dem Ersetzungssatz 4.4.2:

$$\begin{aligned}
\neg X \Rightarrow Z \ &\models\mid \ \neg(\neg X \wedge \neg Z) \ \models\mid \ X \vee Z \\
Z \wedge X \Rightarrow \neg Y \ &\models\mid \ \neg(Z \wedge X \wedge Y) \ \models\mid \ \neg X \vee \neg Y \vee \neg Z \\
Y \vee \neg X \Rightarrow \neg Z \ &\models\mid \ \neg((Y \vee \neg X) \wedge Z) \ \models\mid \ (\neg Y \wedge X) \vee \neg Z \\
&\models\mid \ (\neg Y \vee \neg Z) \wedge (X \vee \neg Z)
\end{aligned}$$

Damit haben wir die konjunktive Normalform

$$\{\{X, Z\}, \ \{\neg X, \neg Y, \neg Z\}, \ \{\neg Y, \neg Z\}, \ \{X, \neg Z\}\}$$

für die Formel 4.1. Wir löschen die zweite Klausel, da sie von der dritten subsumiert wird, und fügen die Resolvente $\{X\}$ der ersten und letzten Klausel hinzu:

$$\{\{X, Z\}, \ \{\neg Y, \neg Z\}, \ \{X, \neg Z\}, \ \{X\}\}$$

Jetzt löschen wir die von $\{X\}$ subsumierten Klauseln und bekommen die folgende konjunktive Primform für die Formel 4.1:

$$\{\{X\}, \ \{\neg Y, \neg Z\}\}$$

Die gerade vorgeführte Berechnung der konjunktiven Primform aus der anfänglichen konjunktiven Normalform wird durch den Resolutionsgraphen in Abbildung 4.18 übersichtlich dargestellt. Subsumierte Klauseln werden dabei durch das Symbol ✓ markiert.

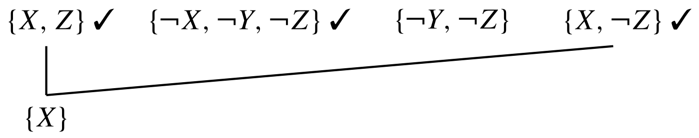

Abbildung 4.18: Ein Resolutionsgraph

Die Berechnung einer disjunktiven Primform für die Fomel 4.1 ist besonders einfach, wenn wir von der Formel ausgehen, die der konjunktiven Primform entspricht. Mit dem Distributivitätsgesetz bekommen wir:

$$X \wedge (\neg Y \vee \neg Z) \models\!\mid (X \wedge \neg Y) \vee (X \wedge \neg Z)$$

Also ist

$$\{\{X, \neg Y\},\ \{X, \neg Z\}\}$$

eine disjunktive Normalform für die Fomel 4.1. Da diese Klauselmenge bereinigt und saturiert ist, handelt es sich um eine disjunktiven Primform für die Fomel 4.1.

Minimalformen

Eine Normalform heißt *Minimalform* genau dann, wenn das Löschen einer Klausel oder das Löschen eines Literals immer zu einer nicht-äquivalenten Normalform führt.

Proposition 4.16.4 *Sei S eine Minimalform. Dann ist S eine Teilmenge der Primform von S.*

Beweis Übungsaufgabe. □

Als Bespiel betrachten wir die Primform:

$$S = \{\{\neg X, Z\},\ \{\neg Y, Z\},\ \{\neg X, Y\},\ \{Y, \neg Z\},\ \{X, \neg Z\},\ \{X, \neg Y\}\}$$

Proposition 4.16.4 schränkt den Suchraum für zu S äquivalente Minimalformen auf die Teilmengen von S ein (2^6 viele). Man überzeugt sich leicht davon, dass

$$S_1 = \{\{\neg Y, Z\},\ \{\neg X, Y\},\ \{X, \neg Z\}\}$$
$$S_2 = \{\{\neg X, Z\},\ \{Y, \neg Z\},\ \{X, \neg Y\}\}$$

zwei zu S äquivalente Minimalformen sind (Hinzufügen von Resolventen führt zur Primform zurück, Löschen von Klauseln eliminiert diese Möglichkeit). Es gibt weitere zu S äquivalente Minimalformen, diese enthalten allerdings 4 Klauseln. Hier ist eine Bespiel:

$$S_3 = \{\{\neg Y, Z\},\ \{\neg X, Y\},\ \{Y, \neg Z\},\ \{X, \neg Y\}\}$$

4.17 Beweis des Resolutionssatzes

Wir beweisen jetzt das für den Beweis des Resolutionssatzes erforderliche Lemma 4.17.3.

Lemma 4.17.1 *Sei $S \subseteq NCla$, $C \in NCla$ und*

$$S' = \{\, D - C \mid D \in S \text{ und } \forall L \in C\colon\ \overline{L} \notin D \,\}$$

Dann:

1. *S' und C haben keine Variablen gemeinsam.*
2. *$\mathcal{K}[\![S']\!] \neq \emptyset \;\Rightarrow\; \mathcal{K}[\![S]\!] \neq \emptyset$.*
3. *S' nicht saturiert $\;\Rightarrow\;$ S nicht saturiert.*

Beweis Die erste Behauptung ist offensichtlich.

Für die zweite Behauptung nehmen wir an, dass $\sigma' \in \mathcal{K}[\![S']\!]$ gilt. Da S' und C keine Variablen gemeinsam haben, gibt es eine Belegung σ, die für die Variablen in S' mit σ' übereinstimmt und $\forall L \in C\colon\ \mathcal{F}[\![\overline{L}]\!]\sigma = 1$ erfüllt. Also gilt $\sigma \in \mathcal{K}[\![S]\!]$.

Für die dritte Behauptung nehmen wir an, dass S' eine echte Resolvente hat. Wir gehen in zwei Schritten von S' zu S zurück und zeigen jeweils, dass die Existenz einer echten Resolvente erhalten bleibt.

Zuerst fügen wir die beim Übergang von S zu S' gelöschten Literale aus C wieder zu den Klauseln von S' hinzu. Nach dem Hinzufügen der Literale kann die Resolvente immer noch gebildet werden. Sie kann jetzt nicht trivial sein, da die neuen Literale nur bisher nicht vorkommende Variablen enthalten und diese nur mit einem Vorzeichen einführen. Die Wiedereinführung der Literale kann auch zu keiner Subsumtion der Resolvente führen (da Löschen von Literalen in allen Klauseln Subsumption erhält).

Im zweiten Schritt fügen wir wieder die Klauseln hinzu, die Komplemente von Literalen von C enthalten. Die neuen Klauseln können die Resolvente nicht subsumieren, da sie jeweils mindestens ein Literal enthalten, dass bisher nicht vorkam. Also ist S nicht saturiert. □

Lemma 4.17.2 *Sei S eine saturierte Normalform mit $\mathcal{K}[\![S]\!] = \emptyset$. Dann $\emptyset \in S$.*

Beweis Durch Induktion über die Anzahl n der in S vorkommenden Variablen.

Sei $n = 0$. Die leere Klausel ist die einzige Klausel, die keine Variablen enthält. Da $\mathcal{K}[\![\emptyset]\!] = \Sigma$, folgt $S \neq \emptyset$. Also $\emptyset \in S$.

Sei $n > 0$. Sei X eine Variable, die in S vorkommt. Wir definieren S_1 und S_2 wie folgt:

$$S_1 = \{\, C - \{X\} \mid C \in S,\ \neg X \notin C \,\}$$
$$S_2 = \{\, C - \{\neg X\} \mid C \in S,\ X \notin C \,\}$$

Mit Lemma 4.17.1 folgt, dass $\mathcal{K}[\![S_1]\!] = \mathcal{K}[\![S_2]\!] = \emptyset$, und dass S_1 und S_2 beide saturiert sind. Also gilt nach Induktionsannahme $\emptyset \in S_1$ und $\emptyset \in S_2$.

Jetzt zeigen wir durch Widerspruch, dass $\emptyset \in S$. Sei $\emptyset \notin S$. Da $\emptyset \in S_1$ und $\emptyset \in S_2$, folgt, dass die Klauseln $\{X\}$ und $\{\neg X\}$ in S sind. Da S saturiert ist, muss die Resolvente $\emptyset$ dieser Klauseln in S sein. Widerspruch. □

Lemma 4.17.3 *Sei S eine saturierte Normalform, $C \in NCla$ und $\mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![\{C\}]\!]$. Dann existiert eine Klausel $C' \in S$ mit $C' \subseteq C$.*

Beweis Sei $S' = \{\, D - C \mid D \in S \text{ und } \forall L \in C\colon\ \overline{L} \notin D \,\}$. Mit Lemma 4.17.1 folgt, dass S' saturiert ist. Sei $S'' = S \cup \{\, \{\overline{L}\} \mid L \in C \,\}$. Da $\mathcal{K}[\![S]\!] \subseteq \mathcal{K}[\![\{C\}]\!]$, folgt $\mathcal{K}[\![S'']\!] = \emptyset$. Da $S' = \{\, D - C \mid D \in S'' \text{ und } \forall L \in C\colon\ \overline{L} \notin D \,\}$ folgt mit Lemma 4.17.1, dass $\mathcal{K}[\![S']\!] = \emptyset$. Also $\emptyset \in S'$ mit Lemma 4.17.2. Also existiert eine Klausel $C' \in S$ mit $C' \subseteq C$. □

4.18 Kompaktheit

Sei $\sigma \in \Sigma$ und $M \subseteq For$. Wir definieren:

$$\sigma \text{ erfüllt } M \ \overset{\text{def}}{\Longleftrightarrow}\ \forall A \in M\colon\ \sigma \in \mathcal{M}[\![A]\!]$$

Eine Menge $M \subseteq For$ heißt *erfüllbar* genau dann, wenn es ein $\sigma \in \Sigma$ gibt, das M erfüllt. Eine Menge $M \subseteq For$ heißt *unerfüllbar* genau dann, wenn sie nicht erfüllbar ist.

Wir werden jetzt zeigen, dass es zu einer unerfüllbaren Menge $M \subseteq For$ stets eine endliche Teilmenge gibt, die unerfüllbar ist. Dieses Ergebnis ist als Kompaktheitseigenschaft der Aussagenlogik bekannt und wird für die Beweise einiger wichtiger prädikatenlogischer Sätze benötigt.

Annahme: Für den Rest des Kapitels nehmen wir an, dass *Var* abzählbar ist.

Satz 4.18.1 (Kompaktheit) *Eine Formelmenge ist erfüllbar genau dann, wenn jede ihrer endlichen Teilmengen erfüllbar ist.*

Beweis Die Richtung von links nach rechts ist trivial.

Sei M eine Formelmenge, für die jede endliche Teilmenge erfülbar ist. Wir werden eine Belegung σ konstruieren, die M erfüllt.

Ohne Beschränkung der Allgemeinheit können wir annehmen, dass keine zwei Formeln in M äquivalent sind (Hinzufügen von äquivalenten Formel erhält Erfüllbarkeit).

Sei $X_0, X_1, X_2, \cdots$ eine Aufzählung der Variablen von *Var*. Sei $M_n \subseteq M$ die Menge aller Formeln von M, in denen nur die Variablen $X_1, \ldots, X_n$ vorkommen. Dann:

$$M_0 \subseteq M_1 \subseteq M_2 \subseteq \cdots \subseteq M = \bigcup_{n \in \mathbb{N}} M_n$$

Wegen Korollar 4.8.8 sind alle M_n endlich. Also existiert für alle $n \in \mathbb{N}$ eine Belegung σ_n, die M_n erfüllt.

Wir werden eine Belegung σ konstruieren, für die gilt: für alle $n \in \mathbb{N}$ gibt es unendlich viele $k \in \mathbb{N}$, so dass σ mit σ_k auf $\{X_0, \ldots, X_n\}$ übereinstimmt.

Bevor wir σ konstruieren, zeigen wir, dass σ die Menge M erfüllt. Sei $A \in M$. Dann existiert ein $n \in \mathbb{N}$ mit $A \in M_n$. Also existiert ein $k \geq n$, so dass σ mit σ_k auf $X_0, \ldots, X_n$ übereinstimmt. Da $A \in M_k$ und σ_k die Menge M_k erfüllt, gilt $\sigma_k \in \mathcal{M}[\![A]\!]$. Da in A höchstens die Variablen $X_0, \ldots, X_n$ vorkommen, gilt auch $\sigma \in \mathcal{M}[\![A]\!]$ (Proposition 4.6.6). Also erfüllt σ die Menge M.

Wir konstruieren jetzt σ. Dazu wählen wir eine Folge

$$s_0 \subseteq s_1 \subseteq s_2 \subseteq \cdots$$

von Funktionen $Var \overset{fin}{\rightharpoonup} \mathbb{B}$ wie folgt

1. $Dom\ s_n = \{X_0, \ldots, X_n\}$.
2. Für alle $n \in \mathbb{N}$ existieren unendlich viele $k \in \mathbb{N}$ mit $s_n \subseteq \sigma_k$.

und setzen $\sigma = \bigcup_{n \in \mathbb{N}} s_n$. □

Korollar 4.18.2 *Eine Formelmenge ist unerfüllbar genau dann, wenn sie eine unerfüllbare endlichen Teilmenge hat.*