2 Aussagenlogik (propositional calculus)

Bevor wir im nächsten Kapitel den Prädikatenkalkül und die zugehörigen Deduktionsmethoden vorstellen, wollen wir zunächst die Methoden und Konzepte in der etwas einfacheren Variante der Logik, nämlich in Aussagenlogik, uns genauer anschauen. Die Verallgemeinerung auf Prädikatenlogik ist dann nicht mehr ein so großer Schritt.

Dieser Abschnitt geht zwar detailliert auf Aussagenlogik, Kalküle und Eigenschaften ein, aber das Ziel ist es nicht, optimale und effiziente Verfahren für die Aussagenlogik vorzustellen.

Definition 2.1. Syntax

Die Syntax ist gegeben durch die Grammatik:

$$A ::= X \mid (A \wedge A) \mid (A \vee A) \mid (\neg\, A) \mid (A \Rightarrow A) \mid (A \Leftrightarrow A) \mid 0 \mid 1$$

Hierbei ist X ein Nichtterminal für aussagenlogische Variablen und A ein Nichtterminal für Aussagen. Die Konstanten $0, 1$ entsprechen falsch bzw. wahr. Üblicherweise werden bei der Notation von Aussagen Klammern weggelassen. wobei man die Klammerung aus den Prioritäten der Operatoren wieder rekonstruieren kann: Die Prioritätsreihenfolge ist: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$.

Die Zeichen $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$ nennt man Junktoren. *Aussagen der Form $0, 1$ oder x nennen wir* Atome. Literal *sind entweder Atome oder Aussagen der Form $\neg A$, wobei A ein Atom ist.*

$A \wedge B$:	*Konjunktion (Verundung).*
$A \vee B$:	*Disjunktion (Veroderung).*
$A \Rightarrow B$:	*Implikation .*
$A \Leftrightarrow B$:	*Äquivalenz.*
$\neg A$:	*negierte Formel.*
A	*Atom, falls A eine Variable ist.*
A	*Literal, falls A Atom oder negiertes Atom.*

Wir lassen auch teilweise eine erweiterte Syntax zu, bei der auch noch andere Operatoren zulässig sind wie: NOR, XOR, NAND.

Dies erweitert nicht die Fähigkeit zum Hinschreiben von logischen Ausdrücken, denn man kann diese Operatoren simulieren. Auch könnte man $0, 1$ weglassen, wie wir noch sehen werden, denn man kann 1 als Abkürzung von $A \vee \neg A$, und 0 als Abkürzung von $A \wedge \neg A$ auffassen.

Definition 2.2. Semantik *Zunächst definiert man für jede Operation $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ Funktionen $\{0,1\} \to \{0,1\}$ bzw. $\{0,1\}^2 \to \{0,1\}$. Die Funktion $f_\neg$ ist definiert als $f_\neg(0) = 1, f_\neg(1) = 0$. Ebenso definiert man $f_0 := 0, f_1 := 1$. Die anderen Funktionen f_{op} sind definiert gemäß folgender Tabelle.*

		$\wedge$	$\vee$	$\Rightarrow$	$\Leftarrow$	NOR	$NAND$	$\Leftrightarrow$	XOR
1	1	1	1	1	1	0	0	1	0
1	0	0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1	0	1
0	0	0	0	1	1	1	1	1	0

Der Einfachheit halber werden oft die syntaktischen Symbole auch als Funktionen gedeutet.

Definition 2.3. *Eine* Interpretation *I ist eine Funktion:* $I : \{aussagenlogische\ Variablen\} \to \{0, 1\}$.

Eine Interpretation I definiert für jede Aussage einen Wahrheitswert, wenn man sie auf Aussagen fortsetzt:

- $I(0) := 0, I(1) := 1$
- $I(\neg A) := f_{\neg}(I(A))$
- $I(A\ op\ B) := f_{op}(I(A), I(B))$, *wobei* $op \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow \ldots\}$

Wenn für eine Aussage F und eine Interpretation I gilt: $I(F) = 1$, dann schreiben wir auch: $I \models F$. Sprechweisen: I ist ein Modell für F, F gilt in I, I macht F wahr.

Definition 2.4. *Sei A ein Aussage.*

- *A ist eine* Tautologie (Satz, allgemeingültig) *gdw. für alle Interpretationen I gilt: $I \models A$.*
- *A ist* ein Widerspruch (widersprüchlich, unerfüllbar) *gdw. für alle Interpretationen I gilt: $I(A) = 0$.*
- *A ist* erfüllbar (konsistent) *gdw. es eine Interpretationen I gibt mit: $I \models A$.*
- *ein* Modell *für eine Formel A ist eine Interpretation I mit $I(A) = 1$.*

Man kann jede Aussage in den n Variablen $X_1, \ldots, X_n$ auch als eine Funktion mit den n Argumenten $X_1, \ldots, X_n$ auffassen. Dies entspricht dann *Booleschen Funktionen.*

Beispiel 2.5.

- $X \vee \neg X$ ist eine Tautologie.
- $(X \Rightarrow Y) \Rightarrow ((Y \Rightarrow Z) \Rightarrow (X \Rightarrow Z))$ ist eine Tautologie.
- $X \wedge \neg X$ ist ein Widerspruch.
- $X \vee Y$ ist erfüllbar.
- I mit $I(X) = 1, I(Y) = 0$ ist ein Modell für $X \wedge \neg Y$

Satz 2.6.

- *Es ist entscheidbar, ob eine Aussage eine Tautologie (Widerspruch, erfüllbar) ist.*
- *Die Frage "Ist A erfüllbar?" ist $\mathcal{NP}$-vollständig.*
- *Die Frage "Ist A Tautologie (Widerspruch)?" ist co-$\mathcal{NP}$-vollständig.*

Das einfachste und bekannteste Verfahren zur Entscheidbarkeit ist das der Wahrheitstafeln. Es werden einfache alle Interpretation ausprobiert.

Zur Erläuterung: $\mathcal{NP}$-vollständig bedeutet, daß jedes Problem der Problemklasse mit einem nicht-deterministischen Verfahren in polynomialer Zeit gelöst werden kann, und daß es keine bessere obere Schranke gibt.

Eine Problemklasse ist co-$\mathcal{NP}$-vollständig, wenn die Problemklasse die aus den Negationen gebildet wird, $\mathcal{NP}$-vollständig ist.

Sequentielle Algorithmen zur Lösung haben für beide Problemklassen nur Exponential-Zeit Algorithmen. Genaugenommen ist es ein offenes Problem der theoretischen Informatik, ob es nicht bessere sequentielle Algorithmen gibt.

Die Klasse der $\mathcal{NP}$-vollständigen Probleme ist vom praktischen Standpunkt aus leichter als co-$\mathcal{NP}$-vollständig: Für $\mathcal{NP}$-vollständige Probleme kann man mit Glück (d.h. Raten) oft schnell eine Lösung finden. Z.B. eine Interpretation einer Formel. Für co-$\mathcal{NP}$-vollständige Probleme muß man i.a. viele Möglichkeiten testen: Z.B. muß man i.a. alle Interpretation einer Formel ausprobieren.

2.1 Folgerungsbegriffe

Man muß zwei verschiedene Begriffe der Folgerungen für Logiken unterscheiden:

- semantische Folgerung
- syntaktische Folgerung (Herleitung, Ableitung) mittels einer prozeduralen Vorschrift. Dies ist meist ein nicht-deterministischer Algorithmus (Kalkül), der auf Formeln oder auf erweiterten Datenstrukturen operiert. Das Ziel ist i.a. die Erkennung von Tautologien (oder Folgerungsbeziehungen).

In der Aussagenlogik fallen viele Begriffe zusammen, die in anderen Logiken verschieden sind. Trotzdem wollen wir die Begriffe hier unterscheiden.

Definition 2.7. *Sei $\mathcal{F}$ eine Menge von (aussagenlogischen) Formeln und G eine weitere Formel.*

> *Wir sagen G* folgt semantisch *aus $\mathcal{F}$*
> *gdw.*
> *Für alle Interpretationen I gilt: wenn für alle Formeln $F \in \mathcal{F}$ die Auswertung $I(F) = 1$ ergibt, dann auch $I(G) = 1$.*

Die semantische Folgerung notieren wir auch als $\mathcal{F} \models G$

In der Aussagenlogik gibt es eine starke Verbindung von semantischer Folgerung mit Tautologien: Es gilt:

Satz 2.8. *$\{F_1, \ldots, F_n\} \models G$ gdw. $F_1 \wedge \ldots \wedge F_n \Rightarrow G$ ist Tautologie.*

Zwei Aussagen F, G nennen wir *äquivalent*, gdw. wenn $F \Leftrightarrow G$ eine Tautologie ist. Beachte, daß F und G genau dann äquivalent sind, wenn für alle Interpretationen I: $I \models F$ gdw. $I \models G$ gilt.

Es ist auch zu beachten, daß z.B. $X \wedge Y$ nicht zu $X' \wedge Y'$ äquivalent ist. D.h. bei diesen Beziehungen spielen die Variablennamen eine wichtige Rolle.

Definition 2.9. *Sei $\mathcal{F}$ eine Menge von (aussagenlogischen) Formeln und G eine weitere Formel. Gegeben sei ein (nicht-deterministischer) Algorithmus $\mathcal{A}$, der aus einer Menge von Formeln $\mathcal{H}$ eine neue Formel H berechnet, Man sagt auch, daß H syntaktisch aus $\mathcal{H}$ folgt (bezeichnet mit $\mathcal{H} \vdash_A H$ oder auch $\mathcal{H} \rightarrow_A H$).*

– *Der Algorithmus $\mathcal{A}$ ist* korrekt (sound), *gdw.* $\mathcal{H} \rightarrow_A H$ *impliziert* $\mathcal{H} \models H$
– *Der Algorithmus $\mathcal{A}$ ist* vollständig (complete), *gdw.* $\mathcal{H} \models H$ *impliziert, daß* $\mathcal{H} \vdash_A H$

Aus obigen Betrachtungen folgt, daß es in der Aussagenlogik für die Zwecke der Herleitung im Prinzip genügt, einen Algorithmus zu haben, der Aussagen auf Tautologieeigenschaft prüft. Gegeben $\{F_1, \ldots, F_n\}$, zähle alle Formeln F auf, prüfe, ob $F_1 \wedge \ldots \wedge F_n \Rightarrow F$ eine Tautologie ist, und gebe F aus, wenn die Antwort ja ist. Das funktioniert, ist aber nicht sehr effizient , wegen der Aufzählung aller Formeln. Außerdem kann man immer eine unendliche Menge von Aussagen folgern, da alle Tautologien immer dabei sind.

Es gibt verschiedene Methoden, aussagenlogische Tautologien (oder auch erfüllbare Aussagen) algorithmisch zu erkennen: Z.B. BDDs (binary decision diagrams) : eine Methode, Aussagen als Boolesche Funktionen anzusehen und möglichst kompakt zu repräsentieren; Genetische Algorithmen zur Erkennung erfüllbarer Formeln; Suchverfahren die mit statischer Suche und etwas Zufall und Bewertungsfunktionen operieren; Davis-Putnam Verfahren: Fallunterscheidung mit Simplifikationen. Tableaukalkül, der Formeln syntaktisch analysiert (analytic tableau).

In dieser Vorlesung sind die letzten beiden Verfahren von Interesse, da sie auf den prädikatenlogischen Fall verallgemeinerbar sind und wir diese Verallgemeinerungen auch betrachten wollen.

2.2 Tautologien und einige einfache Verfahren

Wir wollen im folgenden Variablen in Aussagen nicht nur durch die Wahrheitswerte T, F ersetzen, sondern auch durch Aussagen. Wir schreiben dann $A[B/x]$, wenn in der Aussage A die Aussagenvariable x durch die Aussage B ersetzt wird. In Erweiterung dieser Notation schreiben wir auch: $A[B_1/x_1, \ldots, B_n/x_n]$, wenn mehrere Aussagevariablen ersetzt werden sollen. Diese Einsetzung passiert gleichzeitig, so daß dies kompatibel zur Eigenschaft der Komposition als Funktionen ist: Wenn A n-stellige Funktion in den Aussagevariablen ist, dann ist $A[B_1/x_1, \ldots, B_n/x_n]$ die gleiche Funktion wie $A \circ (B_1, \ldots, B_n)$.

Satz 2.10. *Sind A_1, A_2 äquivalente Aussagen, und $B_1, \ldots, B_n$ weitere Aussagen, dann sind $A_1[B_1/x_1, \ldots, B_n/x_n]$ und $A_2[B_1/x_1, \ldots, B_n/x_n]$ ebenfalls äquivalent.*

Beweis. Man muß zeigen, daß alle Zeilen der Wahrheitstafeln für die neuen Ausdrücke gleich sind. Seien $y_1, \ldots, y_m$ die Variablen. Den Wert einer Zeile kann man berechnen, indem man zunächst die Wahrheitswerte für B_i berechnet und dann in der Wahrheitstabelle von A_i nachschaut. Offenbar erhält man für beide Ausdrücke jeweils denselben Wahrheitswert. □

Satz 2.11. *Sind A, B äquivalente Aussagen, und F eine weiter Aussage, dann sind $F[A]$ und $F[B]$ ebenfalls äquivalent.* [1]

Die Junktoren $\wedge$ und $\vee$ sind kommutativ, assoziativ, und idempotent, d.h. es gilt:

$$\begin{array}{ccc} \mathcal{F} \wedge \mathcal{G} & \Leftrightarrow & \mathcal{G} \wedge \mathcal{F} \\ \mathcal{F} \wedge \mathcal{F} & \Leftrightarrow & \mathcal{F} \\ \mathcal{F} \wedge (\mathcal{G} \wedge \mathcal{H}) & \Leftrightarrow & (\mathcal{F} \wedge \mathcal{G}) \wedge \mathcal{H} \\ \mathcal{F} \vee \mathcal{G} & \Leftrightarrow & \mathcal{G} \vee \mathcal{F} \\ \mathcal{F} \vee (\mathcal{G} \vee \mathcal{H}) & \Leftrightarrow & (\mathcal{F} \vee \mathcal{G}) \vee \mathcal{H} \\ \mathcal{F} \vee \mathcal{F} & \Leftrightarrow & \mathcal{F} \end{array}$$

Weiterhin gibt es für Aussagen noch Rechenregeln und Gesetze, die wir im folgenden auflisten wollen. Alle lassen sich mit Hilfe der Wahrheitstafeln beweisen, allerdings erweist sich das bei steigender Variablenanzahl als mühevoll, denn die Anzahl der Überprüfungen ist 2^n wenn n die Anzahl der Aussagenvariablen ist. Die Frage nach dem maximal nötigen Aufwand (in Abhängigkeit von der Größe der Aussagen) für die Bestimmung, ob eine Aussage erfüllbar ist, ist ein berühmtes offenes Problem der (theoretischen) Informatik, das $SAT \in \mathcal{P}$-Problem, dessen Lösung weitreichende Konsequenzen hätte, z.B. würde $\mathcal{P} = \mathcal{NP}$ daraus folgen. Im Moment geht man davon aus, daß dies nicht gilt.

Lemma 2.12 (Äquivalenzen:).

$$\begin{array}{lll} \neg(\neg A)) & \Leftrightarrow A & \\ A \Rightarrow B & \Leftrightarrow \neg A \vee B & \\ A \Leftrightarrow B & \Leftrightarrow A \Rightarrow B \wedge B \Rightarrow A & \\ \neg(A \wedge B) & \Leftrightarrow \neg A \vee \neg B & \text{(DeMorgansche Gesetze)} \\ \neg(A \vee B) & \Leftrightarrow \neg A \wedge \neg B & \\ A \wedge (B \vee C) & \Leftrightarrow (A \wedge B) \vee (A \wedge C) & \text{Distributivität} \\ A \vee (B \wedge C) & \Leftrightarrow (A \vee B) \wedge (A \vee C) & \text{Distributivität} \\ (A \Rightarrow B) & \Leftrightarrow (\neg B \Rightarrow \neg A) & \text{Kontraposition} \\ A \vee (A \wedge B) & \Leftrightarrow A & \text{Absorption} \\ A \wedge (A \vee B) & \Leftrightarrow A & \text{Absorption} \end{array}$$

Diese Regeln sind nach Satz 2.10 nicht nur verwendbar, wenn $A; B; C$ Aussagevariablen sind, sondern auch, wenn A, B, C für Aussagen stehen.

2.3 Normalformen

Für Aussagen der Aussagenlogik gibt es verschiedene Normalformen. U.a. die *disjunktive Normalform (DNF)* und die *konjunktive Normalform (CNF)*: Die letzte nennt man auch Klauselnormalform.

[1] Hierbei ist $F[B]$ die Aussage, die dadurch entsteht, daß in F die Subaussage A durch B ersetzt wird.

– *disjunktive Normalform (DNF).* Die Aussage ist eine Disjunktion von Konjunktionen von Literalen. D.h. von der Form

$$(L_{1,1} \wedge \ldots \wedge L_{1,n_1}) \vee \ldots \vee (L_{m,1} \wedge \ldots \wedge L_{m,n_m})$$

wobei $L_{i,j}$ Literale sind.

– *konjunktive Normalform (CNF).* Die Aussage ist eine Konjunktion von Disjunktionen von Literalen. D.h. von der Form

$$(L_{1,1} \vee \ldots \vee L_{1,n_1}) \wedge \ldots \wedge (L_{m,1} \vee \ldots \vee L_{m,n_m})$$

wobei $L_{i,j}$ Literale sind.

Die Klauselnormalform wird oft als Menge von Mengen notiert und auch behandelt. Dies ist gerechtfertigt, da sowohl $\wedge$ als auch $\vee$ assoziativ, kommutativ und idempotent sind, so daß Vertauschungen und ein Weglassen der Klammern erlaubt ist. Wichtig ist die Idempotenz, die z.B. erlaubt, eine Klausel $\{A, B, A, C\}$ als unmittelbar äquivalent zur Klausel $\{A, B, C\}$ zu betrachten. Eine Klausel mit einem Literal bezeichnet man auch als *1-Klausel*. Eine Klausel (in Mengenschreibweise) ohne Literale wird als *leere Klausel* bezeichnet. Diese ist äquivalent zu 0, dem Widerspruch.

Satz 2.13. *Eine Klausel C ist eine Tautologie genau dann wenn es eine Variable A gibt, so daß sowohl A als auch $\neg A$ in der Klausel vorkommen*

Beweis. Übungsaufgabe.

Es gilt:

Satz 2.14. *Zu jeder Aussage kann man eine äquivalente CNF finden und ebenso eine äquivalente DNF. Allerdings nicht in eindeutiger Weise.*

Satz 2.15.

– *Eine Aussage in CNF kann man in linearer Zeit auf Tautologie-eigenschaft testen.*
– *Eine Aussage in DNF kann man in linearer Zeit auf Unerfüllbarkeit testen.*

Beweis. Eine CNF $C_1 \wedge \ldots \wedge C_n$ ist eine Tautologie, gdw für alle Interpretationen I diese Formel stets wahr ist. D.h. alle C_i müssen ebenfalls Tautologien sein. Das geht nur, wenn jedes C_i ein Paar von Literale der Form $A, \neg A$ enthält.

Das gleiche gilt in dualer Weise für eine DNF, wenn man dualisiert, d.h. wenn man ersetzt: $\wedge \leftrightarrow \vee$, $0 \leftrightarrow 1$, CNF $\leftrightarrow$ DNF, Tautologie $\leftrightarrow$ Widerspruch. □

Der duale Test: CNF auf Unerfüllbarkeit bzw. DNF auf Allgemeingültigkeit ist die eigentliche harte Nuß.

Dualitätsprinzip Man stellt fest, daß in der Aussagenlogik (auch in der Prädikatenlogik) Definition, Lemmas, Methoden, Kalküle, Algorithmen stets eine duale Variante haben. Die Dualität ist wie im Beweis oben: durch Vertauschung zu sehen: $\wedge \leftrightarrow \vee$, $0 \leftrightarrow 1$, CNF $\leftrightarrow$ DNF, Tautologie $\leftrightarrow$ Widerspruch, Test auf Allgemeingültigkeit $\leftrightarrow$ Test auf Unerfüllbarkeit. D.h. auch, daß die Wahl zwischen Beweissystemen, die auf Allgemeingültigkeit testen und solchen , die auf Unerfüllbarkeit testen, eine Geschmacksfrage ist und keine prinzipielle Frage.

Satz 2.16. *Sei F eine Formel. Dann ist F allgemeingültig gdw. $\neg F$ ein Widerspruch ist*

Bemerkung 2.17. Aus Lemma 2.15 kann man Schlußfolgerungen ziehen über die zu erwartende Komplexität eines Algorithmus, der eine Formel (unter Äquivalenzerhaltung) in eine DNF (CNF) transformiert. Wenn dieser Algorithmus polynomial wäre, dann könnte man einen polynomialen Tautologietest durchführen:

> Zuerst überführe eine Formel in CNF und dann prüfe, ob diese CNF eine Tautologie ist.

Dies wäre ein polynomieller Algorithmus für ein co-$\mathcal{NP}$-vollständiges Problem.

Allerdings sehen wir später, daß die DNF (CNF-)Transformation selbst nicht der Engpass ist, wenn man nur verlangt, daß die Allgemeingültigkeit (Unerfüllbarkeit) erhalten bleibt.

Definition 2.18. Transformation in Klauselnormalform

Folgende Prozedur wandelt jede aussagenlogische Formel in konjunktive Normalform (CNF, Klauselnormalform) um:

1. *Elimination von $\Leftrightarrow$ und $\Rightarrow$:*

$$F \Leftrightarrow G \rightarrow F \Rightarrow G \wedge G \Rightarrow F$$

   und

$$F \Rightarrow G \rightarrow \neg F \vee G$$

2. *Negation ganz nach innen schieben:*

$$\begin{array}{lcl} \neg\neg F & \rightarrow & F \\ \neg(F \wedge G) & \rightarrow & \neg F \vee \neg G \\ \neg(F \vee G) & \rightarrow & \neg F \wedge \neg G \end{array}$$

3. *Distributivität (und Assoziativität, Kommutativität) iterativ anwenden, um $\wedge$ nach außen zu schieben ("Ausmultiplikation"). $F \vee (G \wedge H) \rightarrow (F \vee G) \wedge (F \vee H)$ (Das duale Distributivgesetz würde eine disjunktive Normalform ergeben.)*

Das Resultat dieser Prozedur ist eine Konjunktion von Disjunktionen (Klauseln) von Literalen:

$$\begin{array}{ll} & (L_{1,1} \vee \ldots \vee L_{1,n_1}) \\ \wedge & (L_{2,1} \vee \ldots \vee L_{2,n_2}) \\ \wedge & \\ \ldots & \\ \wedge & (L_{k,1} \vee \ldots \vee L_{k,n_k}) \end{array}$$

oder in (Multi-)Mengenschreibweise:

$$\begin{aligned}&\{\{L_{1,1},\ldots,L_{1,n_1}\},\\ &\{L_{2,1},\ldots,L_{2,n_2}\},\\ &\ldots\\ &\{L_{k,1},\ldots,L_{k,n_k}\}\}\end{aligned}$$

Die so hergestellte CNF ist eine Formel, die äquivalent zur eingegebenen Formel ist. Dieser CNF-algorithmus ist im schlechtesten Fall exponentiell, d.h. die Anzahl der Literale in der Klauselform wächst exponentiell mit der Größe der Ausgangsformel.

Es gibt zwei Schritte, die zum exponentiellem Anwachsen der Formel führen können,

- die Elimination von $\Leftrightarrow$: Betrachte die Formel $(A_1 \Leftrightarrow A_2) \Leftrightarrow (A_3 \Leftrightarrow A_4)$ und die Verallgemeinerung.
- Ausmultiplikation mittels Distributivgesetz:

  $$(A_1 \wedge \ldots \wedge A_n) \vee B_2 \vee \ldots \vee B_m \rightarrow ((A_1 \vee B_2) \wedge \ldots \wedge (A_n \vee B_2)) \vee B_3 \ldots \vee B_n$$

  Dies verdoppelt B_2 und führt zum Iterieren des Verdoppelns, wenn B_i selbst wieder zusammengesetzte Aussagen sind.

Beispiel 2.19.

$$\begin{aligned}&((A \wedge B) \Rightarrow C) \Rightarrow C\\ \rightarrow\ &\neg(\neg(A \wedge B) \vee C) \vee C\\ \rightarrow\ &(A \wedge B) \vee \neg C \vee C\\ \rightarrow\ &(A \vee \neg C \vee C) \wedge (B \vee \neg C \vee C)\end{aligned}$$

2.4 Lineare CNF

Wir geben einen Algorithmus an, der eine aussagenlogische Formel F in polynomialer Zeit in eine DNF F_{DNF} umwandelt, wobei die Formel F Tautologie ist gdw. F_{DNF} eine Tautologie ist. Es ist hierbei nicht gefordert, daß F und F_{DNF} äquivalent als Formeln sind! Beachte, daß bei diesem Verfahren die Anzahl der Variablen erhöht wird.

Der Trick ist: komplexe Subformeln iterativ durch neue Variablen abzukürzen. Sei $F[G]$ eine Formel mit der Subformel G. Dann erzeuge daraus $(G \Leftrightarrow A) \Rightarrow F[A]$, wobei A eine neue aussagenlogische Variable ist.

Satz 2.20. *$F[G]$ ist eine Tautologie gdw. $(G \Leftrightarrow A) \Rightarrow F[A]$ eine Tautologie ist. Hierbei muß A eine Variable sein, die nicht in $F[G]$ vorkommt.*

Beweis. "$\Rightarrow$" Sei $F[G]$ eine Tautologie und sei I eine beliebige Interpretation. Werte die Formel $(G \Leftrightarrow A) \Rightarrow F[A]$ unter I aus: Wenn $I(G \Leftrightarrow A) = 0$, dann ist die Formel unter I wahr. Wenn $I(G \Leftrightarrow A) = 1$, dann ist $I(F[A]) = I(F[G]) = 1$, da $F[G]$ eine Tautologie ist.

"$\Leftarrow$" Sei $(G \Leftrightarrow A) \Rightarrow F[A]$ eine Tautologie und I eine Interpretation der Variablen von $F[G]$. Wähle eine Interpretation I', die wie I ist, aber für A so gewählt ist, daß $I(A) = I(G)$. Da $(G \Leftrightarrow A) \Rightarrow F[A]$ eine Tautologie ist, erhält man $I'((G \Leftrightarrow A) \Rightarrow F[A]) = 1$. Da $I'(G \Leftrightarrow A) = 1$, kann nur $I'(F[A]) = 1$ gelten. Da Außerdem $I'(G) = I(A)$, erhält man $I'(F[G]) = I'(F[A]) = 1$. D.h Für alle Interpretationen I ist $F[G]$ gültig. $\square$

Zunächst benötigen wir den Begriff Tiefe einer Aussage und Subformel in Tiefe n. Beachte hierbei aber, daß es jetzt auf die genaue syntaktische Form ankommt: Es muß voll geklammert sein. Man kann es auch für eine flachgeklopfte Form definieren, allerdings braucht man dann eine andere Syntaxdefinition usw.

Definition 2.21. *Die Tiefe einer Subformel in Tiefe n definiert man entsprechend dem Aufbau der Syntax: Zunächst ist jede Formel F eine Subformel der Tiefe 0 von sich selbst. Sei H eine Subformel von F der Tiefe n. Dann definieren wir Subformeln von F wie folgt:*

- *Wenn $H \equiv \neg G$, dann ist G eine Subformel der Tiefe $n+1$*
- *Wenn $H \equiv (G_1\ op\ G_2)$, dann sind G_1, G_2 Subformeln der Tiefe $n+1$, wobei op einer der Junktoren $\vee, \wedge, \Rightarrow, \Leftrightarrow$ sein kann.*

Die Tiefe einer Formel sei die maximale Tiefe einer Subformel.

Wir wollen dieses Lemma jetzt auf folgende Art und Weise ausnutzen: Wegen der Dualität genügt es, den Algorithmus für DNF zu formulieren.

Definition 2.22. Schneller DNF-Algorithmus
Wenn eine Formel bereits in der Form $H_1 \vee \ldots \vee H_n$ ist, und H_j eine Tiefe ≥ 4 hat, dann ersetze H_j folgendermaßen: Ersetze alle Subformeln $G_1, \ldots, G_m$ von H_j die in Tiefe 3 auftauchen, durch neue Variablen A_i: D.h. Ersetze $H_j[G_1, \ldots, G_m]$ durch $\neg(G_1 \Leftrightarrow A_1) \vee \ldots \vee \neg(G_m \Leftrightarrow A_m) \vee H_j[A_1, \ldots, A_m]$ in der Formel $H_1 \vee \ldots \vee H_n$.

Iteriere diesen Schritt, bis er nicht mehr durchführbar ist.

Danach wandle die verbliebenen Teile der Disjunktion mit Tiefe ≤ 3 in DNF um.

Die Begründung der verbesserten Komplexität ist folgendermaßen: Aus einer Formel F der Tiefe n entsteht im ersten Schritt eine Formel der Tiefe höchstens 3 und weitere Disjunktionsglieder der Form $\neg(G \Leftrightarrow A)$ (mit kleinerer Tiefe als F). D.h. nur in G kann wieder ersetzt werden. D.h. die Anzahl der neu hinzugefügten Formel ist kleiner als die Anzahl aller Subformeln der ursprünglichen Formel F. Den Aufwand zur Umformung der kleinen Formeln kann man als konstant ansehen. Da die Größe der Formel F gerade die Anzahl der Subformeln ist, ist die Anzahl der durchzuführenden Schritte linear. Je nach Datenstruktur (zur Vermeidung von Suche) kann man den Algorithmus linear formulieren und implementieren.

Dual dazu ist die wieder die schnelle Herstellung einer CNF, wobei hierbei die Eigenschaft der Unerfüllbarkeit erhalten bleibt.

Beispiel 2.23. Wandle die Formel

$$(((((X \Leftrightarrow Y) \Leftrightarrow Y) \Leftrightarrow Y) \Leftrightarrow Y) \Leftrightarrow X)$$

um. Das ergibt:

$$(A \Leftrightarrow ((X \Leftrightarrow Y) \Leftrightarrow Y)) \Rightarrow ((((A \Leftrightarrow Y) \Leftrightarrow Y) \Leftrightarrow X)$$

Danach kann man diese Formel dann auf übliche Weise in DNF umwandeln.

Bemerkung 2.24. Ein Implementierung eines Algorithmus, der Formeln in Klauselmengen verwandelt, kann z.B. auf der www-Seite `http://spass.mpi-sb.mpg.de` gefunden werden. Das Programm heißt `FLOTTER`. Es benötigt einen gewissen Vorspann: Symbole, ...; es verlangt eine standardisierte Syntax und erzeugt dann als Ausgabe-datei eine Klauselmenge. Diese kann dann in einen weiteren Beweiser (`SPASS`) eingegeben werden. Allerdings hat FLOTTER einen aussagenlogischen check als Teilsystem eingebaut, so daß sich aussagenlogische Formeln sofort damit entscheiden lassen. Dieses System gibt im erfüllbaren Fall ein Modell aus.

2.5 Resolution für Aussagenlogik

Das Resolutionsverfahren dient zum Erkennen von Widersprüchen, wobei statt dem Test auf Allgemeingültigkeit einer Formel F die Formel $\neg F$ auf Unerfüllbarkeit getestet wird. Eine Begründung wurde bereits gegeben. Eine erweiterte liefert das folgende Lemma zum Beweis durch Widerspruch:

Satz 2.25. *Eine Formel $A_1 \wedge \ldots \wedge A_n \Rightarrow F$ ist allgemeingültig gdw. $A_1 \wedge \ldots \wedge A_n \wedge \neg F$ widersprüchlich ist.*

Beweis. Übungsaufgabe □

Die semantische Entsprechung ist:

Satz 2.26. $\{A_1, \ldots, A_n\} \models F$ *gdw. es keine Interpretation I gibt, so daß $I \models \{A_1, \ldots, A_n, \neg F\}$*

Die Resolution ist eine Regel mit der man aus zwei Klauseln einer Klauselmenge eine weitere herleiten und dann zur Klauselmenge hinzufügen kann.
Resolution:

$$\frac{\begin{array}{l} A \quad \vee B_1 \vee \ldots \vee B_n \\ \neg A \vee C_1 \vee \ldots \vee C_m \end{array}}{B_1 \vee \ldots \vee B_n \vee C_1 \vee \ldots \vee C_m}$$

Man nennt die ersten beiden Klauseln auch Elternklauseln und die neu hergeleitete Klausel *Resolvente.*

Auf der Ebene der Klauselmengen sieht das Verfahren so aus:

$$C \quad \rightarrow \quad C \cup \{R\}$$

wobei R eine Resolvente ist, die aus zwei Klauseln von C berechnet worden ist. Man nimmt der Einfachheit halber an, daß Klauseln Mengen sind; d.h. es kommen keine Literale doppelt vor. Außerdem nimmt man auch noch an, daß die Konjunktion der Klauseln eine Menge ist, d.h. nur neue Klauseln, die nicht bereits vorhanden sind, können hinzugefügt werden. Wird die leere Klausel hergeleitet, ist das Verfahren beendet. Denn ein Widerspruch wurde hergeleitet.

Eingesetzt wird die Resolution zur Erkennung unerfüllbarer Klauselmengen. Es werden solange Resolventen hergeleitet, bis entweder die leere Klausel hinzugefügt wurde oder keine neue Resolvente mehr herleitbar ist. Dies geschieht meist in der Form, daß man Axiome (als Konjunktion) eingibt und ebenso eine negierte Folgerung, so daß die Unerfüllbarkeit bedeutet, daß man einen Widerspruch hergeleitet hat.

Wenn man keine neuen Klauseln mehr herleiten kann, oder wenn besonders kurze (aussagekräftige) Klauseln hergeleitet werden, kann man diese als echte Folgerungen aus den eingegebenen Formeln ansehen, und evtl. ein Modell konstruieren. Allerdings ist das bei obiger Widerspruchsvorgehensweise nicht unbedingt ein Modell der Axiome, da die negiert eingegebene Folgerung dazu beigetragen haben kann.

Satz 2.27. *Wenn $C \to C'$ mit Resolution, dann ist C äquivalent zu C'.*

Beweis. Wir zeigen den nichttrivialen Teil:

Sei I eine Interpretation, die sowohl $A \vee B_1 \vee \ldots \vee B_n$ und $\neg A \vee C_1 \vee \ldots \vee C_m$ wahrmacht. Wenn $I(A) = 1$, dann gibt es ein C_j, so daß $I(C_j) = 1$. Damit ist auch die Resolvente unter I wahr. Analog für den Fall $I(A) = 0$.

□

Satz 2.28. *Die Resolution auf einer aussagenlogischen Klauselmenge terminiert, wenn man einen Resolutionsschritt nur ausführen darf, wenn sich die Klauselmenge vergrößert.*

Beweis. Es gibt nur endlich viele Klauseln, da Resolution keine neuen Variablen einführt. □

Übungsaufgabe 2.29. Gebe ein Beispiel an, so daß R sich aus C_1, C_2 mit Resolution herleiten läßt, aber $C_1 \wedge C_2 \Leftrightarrow R$ ist falsch

Da Resolution die Äquivalenz der Klauselmenge als Formel erhält, kann man diese auch verwenden, um ein Modell zu erzeugen, bzw. ein Modell einzuschränken. Leider ist diese Methode nicht immer erfolgreich: Zum Beispiel betrachte man die Klauselmenge $\{\{A, B\}, \{\neg A, \neg B\}\}$. Resolution ergibt:

$$\{\{A, B\}, \{A, \neg A\}, \{B, \neg B\}, \{\neg A, \neg B\}\}$$

D.h. es wurden zwei tautologische Klauseln hinzugefügt. Ein Modell läßt sich direkt nicht ablesen. Zum Generieren von Modellen ist die Davis-Putnam-Prozedur (siehe 2.6) oder ein Tableau-kalkül (siehe 2.7) geeigneter .

Was jetzt noch fehlt, ist ein Nachweis der naheliegenden Vermutung, daß die Resolution für alle unerfüllbaren Klauselmengen die leere Klausel auch findet.

Satz 2.30. *Für eine unerfüllbare Klauselmenge findet Resolution nach endlich vielen Schritten die leere Klausel.*

Beweis. Dazu genügt es anzunehmen, daß eine unerfüllbare Klauselmenge C existiert, die keine Herleitung der leeren Klausel mit Resolution erlaubt. Wir können annehmen, daß es eine kleinste Klauselmenge gibt bzgl. des Maßes $lxn(C)$ = Anzahl der Literale − Anzahl der Klauseln (Anzahl der überschüssigen Literale).

Im Basisfall (d.h. $lxn(C) = 0$) gibt es nur noch 1-Klauseln. Damit diese Klauselmenge unerfüllbar ist, muß es eine Variable A geben, so daß sowohl $\{A\}$ als auch $\{\neg A\}$ als Klausel vorkommt. Dann ist aber noch eine Resolution möglich, die die leere Klausel herleitet.

Sei also $lxn(C) > 0$. Dann betrachte eine Klausel $K \in C$, die mehr als ein Literal hat. Ersetzt man K durch K', wobei ein Literal gestrichen ist, d.h. $K = K' \cup \{L\}$, so erhält man ebenfalls eine unerfüllbare Klauselmenge C': Wäre C' erfüllbar mit der Interpretation I, dann auch $I \models C$. Da $lxc(C') < lxc(C)$, gibt es für C' eine Herleitung der leeren Klausel. Wenn diese Herleitung K' nicht (als Elternklausel) benötigt, dann kann man diese Herleitung bereits in C machen, also benötigt diese Herleitung die Klausel K. Übersetzt man diese Herleitung der leeren Klausel in eine Herleitung unter Benutzung von K, so erhält man eine Herleitung der 1-Klausel $\{L\}$.

Betrachtet man jetzt noch die Klauselmenge C'', die aus C entsteht, wenn man K durch $\{L\}$ ersetzt, so sieht man wie oben: C'' ist unerfüllbar und $lxc(C'') < lxc(C)$. Damit existiert eine Herleitung der leeren Klausel in C''. Zusammengesetzt erhält man eine Herleitung der leeren Klausel in C. □

Satz 2.31. *Resolution erkennt unerfüllbare Klauselmengen.*

Was wir hier nicht mehr durchführen wollen, sondern auf die Behandlung der allgemeinen Resolution verschieben, ist die Verwendung von Redundanzkriterien. Z.B. Löschung von Tautologien, unnötigen Klauseln usw.

Die Komplexität im schlimmsten Fall wurde von A. Haken [Hak85] (siehe auch [Ede92] nach unten abgeschätzt: Es gibt eine Folge von Formeln (die sogenannten Taubenschlag-formeln (pigeon hole formula, Schubfach-formeln), für die gilt, daß die kürzeste Herleitung der leeren Klausel mit Resolution eine exponentielle Länge (in der Größe der Formel) hat.

Betrachtet man das ganze Verfahren zur Prüfung der Allgemeingültigkeit einer aussagenlogischen Formel, so kann man eine CNF in linearer Zeit herstellen, aber ein Resolutionsbeweis muß mindestens exponentiell lange sein, d.h. auch exponentiell lange dauern.

Beachte, daß es formale Beweisverfahren gibt, die polynomial lange Beweise für die Schubfachformeln haben. Es ist theoretisch offen, ob es ein Beweisverfahren gibt, das für alle Aussagen polynomial lange Beweise hat: Dies ist äquivalent zum offenen Problem $\mathcal{NP}$ = co-$\mathcal{NP}$.

2.6 Davis-Putnam-Verfahren

Die Prozedur von Davis und Putnam zum Entscheiden der Erfüllbarkeit (und Unerfüllbarkeit) von aussagenlogische Klauselmengen beruht auf Fallunterscheidung und Ausnutzen und Propagieren der Information. Wenn man die Vollständigkeit des Resolutionskalküls für Prädikatenlogik inklusive einiger Redundanzregeln voraussetzt, (Subsumptionsregel, Isolationsregel), kann man die Korrektheit leicht begründen.

Definition 2.32. *Resolutionsverfahren für Aussagenlogik (Davis-Putnam Prozedur)*

Sei C eine aussagenlogische Klauselmenge. Dann wird die Davis-Putnam Entscheidungsprozedur DP folgendermaßen (rekursiv) definiert: Es ist ein Algorithmus, der eine Klauselmenge als Eingabe hat, und genau dann `true` *als Ausgabe hat, wenn die Klauselmenge unerfüllbar ist.*

Als Vorverarbeitungsschritt können wir annehmen, daß keine Tautologien in der Klauselmenge enthalten sind. D.h. es gibt keine Klausel, die gleichzeitig P und $\neg P$ für eine Variable P enthält.

1. *(a) Wenn die leere Klausel in C ist: RETURN* `true`.
   (b) Wenn C die leere Klauselmenge ist: RETURN `false`.
2. *wenn es in eine 1-Klausel $\{P\}$ ($\{\neg P\}$) gibt, wobei P eine Variable ist, dann verändere C wie folgt:*
   (a) Lösche alle Klauseln in denen P ($\neg P$) als Literal vorkommt.
   (b) Lösche alle Vorkommen des Literals $\neg P$ (P) in anderen Klauseln.
   Die resultierende Klauselmenge sei C'. RETURN $DP(C')$
3. *Wenn es isolierte Literale [2] gibt, wende die Löschregel für isolierte Literale an. D.h. lösche die Klauseln, in denen isolierte Literale vorkommen. Die resultierende Klauselmenge sei C'.*
   RETURN $DP(C')$
4. *Wenn keiner der obigen Fälle zutrifft, dann wähle eine noch in C vorkommende aussagenlogische Variable P aus.*
   RETURN $DP(C \cup \{P\}) \wedge DP(C \cup \neg P)$

Dies ist ein vollständiges, korrektes Entscheidungsverfahren für die (Un-)Erfüllbarkeit von aussagenlogischen Klauselmengen. Punkt 2a) entspricht der Subsumption, Punkt 2b) ist Resolution mit anschließender Subsumption. Punkt 3) ist der Spezialfall der isolierten Literale und 4)) ist eine Fallunterscheidung, ob P wahr oder falsch ist. Diese DP-Prozedur ist im allgemeinen sehr viel besser als eine vollständige Fallunterscheidung über alle möglichen Variablenbelegungen, d.h. besser als die Erstellung einer Wahrheitstafel. Die DP-Prozedur braucht im schlimmsten Fall exponentiell viel Zeit, was nicht weiter verwundern kann, denn ein Teil des Problems ist gerade SAT, das Erfüllbarkeitsproblem für aussagenlogische Klauselmengen, und das ist bekanntlich $\mathcal{NP}$-vollständig.

Der DP-Algorithmus ist erstaunlich schnell, wenn man bei Punkt 4) noch darauf achtet, daß man Literale auswählt, die in möglichst kurzen Klauseln vorkommen. Dies erhöht nämlich die Wahrscheinlichkeit, daß nach wenigen Schritten große Anteile der Klauselmenge gelöscht werden.

[2] P ist isoliert, wenn $\neg P$ nicht mehr vorkommt, entsprechend $\neg P$ ist isoliert, wenn P nicht mehr vorkommt

Der DP-Algorithmus kann leicht so erweitert werden, daß im Falle der Erfüllbarkeit der Klauselmenge auch ein Modell berechnet wird. Der Algorithmus arbeitet depth-first mit backtracking. Wenn die Antwort "erfüllbar"ist, kann man durch Rückverfolgung der folgenden Annahmen ein Modell bestimmen:

1. Isolierte Literale werden als wahr angenommen.
2. Literale in 1-Klauseln werden ebenfalls als wahr angenommen.

Beispiel 2.33. Betrachte folgende Klauselmenge, wobei jede Zeile einer Klausel entspricht.

$$\begin{array}{lll}
P, & Q & \\
\neg P, & Q & R \\
P, & \neg Q, & R \\
\neg, P & \neg Q, & R \\
P, & Q, & \neg R \\
\neg P, & Q, & \neg R \\
P, & \neg Q, & \neg R \\
\neg P, & \neg Q, & \neg R
\end{array}$$

Fall 1: Addiere die Klausel $\{P\}$. Das ergibt nach einigen Schritten:

$$\begin{array}{ll}
Q, & R \\
\neg Q, & R \\
Q, & \neg R \\
\neg Q, & \neg R
\end{array}$$

Fall 1.1: Addiere $\{Q\}$: ergibt die leere Klausel.
Fall 1.2: Addiere $\{\neg Q\}$: ergibt die leere Klausel.
Fall 2: Addiere die Klausel $\{\neg P\}$. Das ergibt nach einigen Schritten:

$$\begin{array}{ll}
Q & \\
\neg Q & R \\
Q & \neg R \\
\neg Q & \neg R
\end{array}$$

Weitere Schritte für Q ergeben

$$\begin{array}{l}
R \\
\neg R
\end{array}$$

Auch dies ergibt sofort die leere Klausel. Damit hat die DP-Prozedur die eingegebene Klauselmenge als unerfüllbar erkannt.

Übungsaufgabe 2.34. Wende die DP-Prozedur auf weitere Klauselmengen an. Z.B.

$$\begin{array}{lll} \neg P & Q & R \\ P & \neg Q & R \\ \neg P & \neg Q & R \\ P & Q & \neg R \\ \neg P & Q & \neg R \\ P & \neg Q & \neg R \\ \neg P & \neg Q & \neg R \end{array}$$

Beispiel 2.35. Wir nehmen uns ein Rätsel von Raymond Smullyan vor: Die Fragen nach dem Pfefferdieb. Es gibt drei Verdächtige: Den Hutmacher, den Schnapphasen und die Maus. Folgendes ist bekannt:

– Genau einer von ihnen ist der Dieb.
– Unschuldige sagen immer die Wahrheit
– Schnapphase: der Hutmacher ist unschuldig.
– Hutmacher: die Haselmaus ist unschuldig

Kodierung: H, S, M sind Variablen für Hutmacher, Schnapphase, Haselmaus und bedeuten jeweils "ist schuldig". Man kodiert das in Aussagenlogik und fragt nach einem Modell.

– $H \vee S \vee M$
– $H \Rightarrow \neg(S \vee M)$
– $S \Rightarrow \neg(H \vee M)$
– $M \Rightarrow \neg(H \vee S)$
– $\neg S \Rightarrow \neg H$
– $\neg H \Rightarrow \neg M$

Dies ergibt eine Klauselmenge (doppelte sind schon eliminiert):

1. H, S, M
2. $\neg H, \neg S$
3. $\neg H, \neg M$
4. $\neg S, \neg M$
5. $S, \neg H$
6. $H, \neg M$

Wir können verschiedene Verfahren zur Lösung verwenden.

1. Resolution ergibt: $2 + 5 : \neg H$, $3 + 6 : \neg M$. Diese beiden 1-Klauseln mit 1 resolviert ergibt: S. Nach Prüfung, ob $\{\neg H, \neg M, S\}$ ein Modell ergibt, können wir sagen, daß der Schnapphase schuldig ist.
2. Davis Putnam: Wir verfolgen nur einen Pfad im Suchraum:
   Fall 1: $S = 0$. Die 5-te Klausel ergibt dann $\neg H$. Danach die 6te Klausel $\neg M$. Zusammen mit (den Resten von) Klausel 1 ergibt dies ein Widerspruch.
   Fall 2: $S = 1$. Dann bleibt von der vierten Klausel nur $\neg M$ übrig, und von der zweiten Klausel nur $\neg H$. Diese ergibt somit das gleiche Modell.

Beispiel 2.36. Ein weiteres Rätsel von Raymond Smullyan:

Hier geht es um den Diebstahl von Salz. Die Verdächtigen sind: Lakai mit dem Froschgesicht, Lakai mit dem Fischgesicht, Herzbube.

Die Aussagen und die bekannten Tatsachen sind:

- Frosch: der Fisch wars
- Fisch: ich wars nicht
- Herzbube: ich wars
- Genau einer ist der Dieb
- höchstens einer hat gelogen

Man sieht, daß es nicht nur um die Lösung des Rätsels selbst geht, sondern auch um etwas Übung und Geschick, das Rätsel so zu formalisieren, daß es von einem Computer gelöst werden kann. Man muß sich auch davon überzeugen, daß die Formulierung dem gestellten Problem entspricht.
Wir wollen Aussagenlogik verwenden.
Wir verwenden Variablen mit folgenden Namen und Bedeutung:

FRW	Frosch sagt die Wahrheit
FIW	Fisch sagt die Wahrheit
HBW	Herzbube sagt die Wahrheit
FID	der Fisch ist der Dieb
FRD	der Frosch ist der Dieb
HBD	der Herzbube ist der Dieb

Die Formulierung ist:

höchstens einer sagt die Wahrheit:

$$\begin{array}{l}
\neg FRW \Rightarrow FIW \\
\neg FRW \Rightarrow HBW \\
\neg FIW \Rightarrow FRW \\
\neg FIW \Rightarrow HBW \\
\neg HBW \Rightarrow FRW \\
\neg HBW \Rightarrow FIW
\end{array}$$

genau einer ist der Dieb:

$$\begin{array}{l}
FID \vee FRD \vee HBD \\
FID \Rightarrow \neg FRD \\
FID \Rightarrow \neg HBD \\
FRD \Rightarrow \neg FID \\
FRD \Rightarrow \neg HBD \\
HBD \Rightarrow \neg FID \\
HBD \Rightarrow \neg FRD
\end{array}$$

Die Aussagen:

$$\begin{array}{l}
FRW \Rightarrow FID \\
FIW \Rightarrow \neg FID \\
HBW \Rightarrow HBD
\end{array}$$

Die Lösung kann mit dem Davis-Putnam Verfahren gefunden werden (eine Implementierung in Haskell ist auf der www-Seite der Vorlesung).

$$FRW = 1, FIW = 2, HBW = 3, FID = 4, FRD = 5, HBD = 6$$

Da kein Parser implementiert ist: Die Kodierung als eine Klauselmengen, wobei die Variablen durch positive Zahlen und die Negierung als negative Zahlen dargestellt sind, so daß n und $\neg n$ komplementäre Literale sind.

$$\begin{array}{l}
[[1,2],[1,3],[2,1],[2,3],[3,1],[3,2],\\
[-4,5,6],[-4,-5],[-4,-6],[-5,-4],[-5,-6],[-6,-4],[-6,-5],\\
[-1,4],[-2,-4],[-3,6]]
\end{array}$$

Die berechnete Lösung ist:

$$[-5, 6, -4, 3, 2, -1]$$

D.h FRW ist falsch, d.h. der Fisch hat gelogen und der Herzbube war der Dieb.

Beispiel 2.37. Anwendung auf ein Suchproblem: das n-Damen Problem.
Es sollen Königinnen auf einem quadratischen Schachbrett der Seitenlänge n so plaziert werden, daß diese sich nicht schlagen können. damit die Formulierung einfacher wird, erwarten wir, daß sich in jeder Zeile und Spalte eine Königin befindet.

Ein Programm zum Erzeugen der Klauselmenge erzeugt im Fall $n = 4$:

```
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16], [1, 5, 9, 13],
[2, 6, 10, 14], [3, 7, 11, 15], [4, 8, 12, 16],
[-1, -5], [-1, -9], [-1, -13],
[-1, -2], [-1, -6], [-1, -3], [-1, -11], [-1, -4], [-1, -16], [-5, -9], [-5, -13],
[-5, -2], [-5, -6], [-5, -10], [-5, -7], [-5, -15], [-5, -8], [-9, -13],
[-9, -6], [-9, -10], [-9, -14], [-9, -3], [-9, -11], [-9, -12], [-13, -10], [-13, -14],
[-13, -7], [-13, -15], [-13, -4], [-13, -16], [-2, -6], [-2, -10], [-2, -14],
[-2, -3], [-2, -7], [-2, -4], [-2, -12], [-6, -10], [-6, -14], [-6, -3],
[-6, -7], [-6, -11], [-6, -8], [-6, -16], [-10, -14], [-10, -7], [-10, -11], [-10, -15],
[-10, -4], [-10, -12], [-14, -11], [-14, -15], [-14, -8], [-14, -16], [-3, -7],
[-3, -11], [-3, -15], [-3, -4], [-3, -8], [-7, -11], [-7, -15], [-7, -4], [-7,-8],
[-7, -12], [-11, -15], [-11, -8], [-11, -12], [-11, -16], [-15, -12],
[-15, -16], [-4, -8], [-4, -12], [-4, -16], [-8, -12], [-8, -16], [-12, -16]]
```

Das Ergebnis der DP-Prozedur sind zwei Interpretationen:

```
[[-4, -8, -15, 5, -13, 14, -6, -2, 12, -9, -1, 3, -16, -10, -7, -11],
 [-4, 2, 8, -6, -1, 9, -12, -14, -13, -5, 15, -3, -16, -10, -7, -11]]
```

Die entsprechen den zwei möglichen Plazierungen im Fall $n = 4$.

2.7 Tableaukalkül für Aussagenlogik

Im folgenden betrachten wir einen Kalkül, der in verschiedenen Formen und Ausprägungen viele Einsatzbereiche hat: u.a Aussagenlogik, Prädikatenlogik, Modallogik, mehrwertige Logik, und Programmanalysen.

Ein Tableau ist i.a. eine baumförmig organisierte Datenstruktur, die mit (beliebigen) Formeln markiert ist, und die mit geeigneten Regeln aufgebaut wird. Die Formel an der Wurzel ist bewiesen (ein Tautologie), wenn das Tableau (der Baum) bestimmte Bedingungen erfüllt. Wir betrachten hier eine Variante des sogenannten analytischen Tableau-kalküls, der eine komplexe Formel Schritt für Schritt vereinfacht und am Ende Literale an den Blättern hat. Die zu überprüfende Bedingung betrifft jeweils die Formeln auf den Pfaden.

Grundbegriffe für den aussagenlogischen Tableaukalkül sind:

- α-Formeln (konjunktive Formeln) und
- β-Formeln (disjunktive Formeln)

Beachte, daß die Negationszeichen nicht nach innen gezogen sind.

Die direkten Unterformeln der α-Formeln sind:

α	α_1	α_2
$X \wedge Y$	X	Y
$\neg(X \vee Y)$	$\neg X$	$\neg Y$
$\neg(X \Rightarrow Y)$	X	$\neg Y$
$(X \Leftrightarrow Y)$	$X \Rightarrow Y$	$Y \Rightarrow X$

Die direkten Unterformeln der β-Formeln sind:

β	β_1	β_2
$X \vee Y$	X	Y
$\neg(X \wedge Y)$	$\neg X$	$\neg Y$
$X \Rightarrow Y)$	$\neg X$	Y
$\neg(X \Leftrightarrow Y)$	$\neg(X \Rightarrow Y)$	$\neg(Y \Rightarrow X)$

Es gilt: α ist äquivalent zu $(\alpha_1 \wedge \alpha_2)$, und β ist äquivalent zu $(\beta_1 \vee \beta_2)$.

Tableau-Kalkül für Aussagenlogik Ziel des Tableaukalküls ist: Beweise, daß eine Aussage A eine Tautologie ist. Damit kann man auch zeigen, daß eine Aussage B aus einer Menge von Aussagen $A_1, \ldots, A_n$ folgt: nämlich durch den Nachweis, daß $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ eine Tautologie ist.

Idee beim Tableau-Kalkül: Zeige, daß die Verneinung eine inkonsistente Aussage ist.

Definition 2.38. *Ein (aussagenlogisches) Tableau ist ein markierter Baum, wobei die Knoten mit aussagenlogischen Formeln markiert sind.*

- *Ein Pfad ist* geschlossen, *wenn* 0 *oder* $\neg 1$ *vorkommt, oder eine Formel* X *existiert, so daß auch* $\neg X$ *auf diesem Pfad ist.*

– *Ein Pfad ist* (atomar) geschlossen*, wenn* 0 *oder* $\neg 1$ *vorkommt, oder ein Atom* A *existiert, so daß auch* $\neg A$ *auf diesem Pfad ist. Ein Tableau ist* (atomar) geschlossen*, wenn alle Pfade* (atomar) geschlossen *sind.*

Man kann die Regeln ansehen als Tableau-aufbauregeln oder als Transformationsregeln auf Tableaus. Wir werden die Sichtweise der Tableau-aufbauregeln verfolgen, denn dann gibt es keinen Unterschied zwischen der Situation beim Aufbau und der Situation des fertigen (geschlossenen) Tableaus. Im Falle von Transformationsregeln könnte es sein, daß das fertige Tableau keine Überprüfung der Aufbauregeln zuläßt.

Definition 2.39. *Der Tableaukalkül* TK_A *hat als Eingabe eine Formel* F*. Initial wird ein Tableau mit einem Knoten und der Formel* $\neg F$ *erzeugt. Danach werden ausgehend vom initialen Tableau weitere Tableaus erzeugt mit folgenden* Expansionsregeln:

$$\frac{\neg\neg X}{X} \qquad \frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 \mid \beta_2} \qquad \frac{\neg 0}{1} \qquad \frac{\neg 1}{0}$$

Diese Regeln sind wie folgt zu verstehen: Sei θ *ein Pfad im Tableau* T*, und die obere Formel* F_o *eine Markierung eines Knotens auf diesem Pfad, dann erweitere das Tableau durch Verlängern des Pfades* θ *(d.h. Anhängen an das Blatt des Pfades) um einen mit der unteren Formel* F_u *markierten Knoten. Stehen unten zwei oder mehrere durch* $\mid$ *getrennte Formeln, dann sollen entsprechend viele Blätter als Töchter angehängt werden,mit der jeweiligen Formel markiert. Danach verzweigt der Pfad* θ *am alten Blatt zu mehreren Pfaden.*

Stehen zwei oder mehr Formeln untereinander, dann sollen in Folge an den Pfad θ *zwei oder mehrere Blätter (mit den jeweiligen Formeln markiert) angehängt werden.*

Wenn oben eine α*-Formel steht, erweitere erst um* α_1*, dann den neuen Knoten um* α_2*. Wenn oben eine* β*-Formel steht, hänge zwei markierte Knoten als Töchter an, eine mit* β_1 *eine mit* β_2 *markiert.*

Als Einschränkung wird verwendet, daß jede Formel auf jedem Pfad nur einmal analysiert (bzw. expandiert) wird.

Ein Formel F *ist bewiesen, wenn aus dem Tableau mit einem Knoten und der Formel* $\neg F$ *ein geschlossenes Tableau erzeugt worden ist.*

Im allgemeinen wird man die Formel direkt am Blatt markieren. Allerdings kommt es auch vor, daß eine Formel, die nicht das Blatt ist, expandiert wird.

Diese Regeln sind nicht-deterministisch, d.h. es gibt keine genaue Angabe, welche Formel zu expandieren ist. Diese Formulierung ist gewählt, um eine möglichst große Freiheit bei der Anwendung zu haben, mit der Garantie, daß die Anwendung korrekt bleibt. Allerdings sollte man in einer effizienten Implementierung zunächst die besten Schritte machen: D.h. möglichst wenig verzweigen. Dies wird durch Bevorzugung der α-Regeln erreicht. Außerdem sollte man Formeln nicht zweimal auf dem gleichen Pfad expandieren.

Beispiel 2.40. Wir zeigen ein (das) Tableau für $X \wedge \neg X$:

$$\begin{array}{c} X \wedge \neg X \\ | \\ X \\ | \\ \neg X \end{array}$$

Beispiel 2.41. Ein Tableau für $\neg(X \wedge Y \Rightarrow X)$:

$$\begin{array}{c} \neg(X \wedge Y \Rightarrow X) \\ | \\ X \wedge Y \\ | \\ \neg X \\ | \\ X \\ | \\ Y \end{array}$$

Beide Tableaus sind geschlossen.

Zur Optimierung der Analyse von Aussagen der Form $A \Leftrightarrow B$, gibt es eine bessere Alternative:

Erfinde neue Tableau-expansionsregeln

$$\frac{A \Leftrightarrow B}{\begin{array}{r|l} A & \neg A \\ B & \neg B \end{array}}$$

$$\frac{\neg(A \Leftrightarrow B)}{\begin{array}{r|l} A & \neg A \\ \neg B & B \end{array}}$$

Übungsaufgabe 2.42. Gebe die Tableauregeln für XOR an.

Beachte: $A\ XOR\ B$ ist äquivalent zu $\neg(A \Leftrightarrow B)$.

Hat man die Aufgabe zu zeigen, daß B aus $A_1, \ldots, A_n$ folgt, so kann man daraus sofort ein Tableau machen:

$$\begin{array}{c} A_1 \\ | \\ \ldots \\ | \\ A_n \\ | \\ \neg B \end{array}$$

Beispiel 2.43. Zeige, daß $P \Rightarrow (Q \Rightarrow P)$ eine Tautologie ist:

$$\begin{array}{c} \neg(P \Rightarrow (Q \Rightarrow P)) \\ | \\ P \\ | \\ \neg(Q \Rightarrow P) \\ | \\ Q \\ | \\ \neg P \end{array}$$

Das Tableau ist geschlossen, da P und $\neg P$ auf dem einen Pfad liegen.

Beispiel 2.44. $((P \Rightarrow Q) \wedge (Q \Rightarrow R)) \Rightarrow (P \Rightarrow R)$:

$$\begin{array}{ccc} & (P \Rightarrow Q) & \\ & | & \\ & (Q \Rightarrow R) & \\ & | & \\ & \neg(P \Rightarrow R) & \\ & | & \\ & P & \\ & | & \\ & \neg R & \\ \swarrow & & \searrow \\ \neg P & & Q \\ & & \swarrow \quad \searrow \\ \text{geschlossen} & & \neg Q \qquad R \\ & & \text{beide geschlossen} \end{array}$$

Der Nachweis der *algorithmischen Korrektheit des Tableaukalküls* für Aussagenlogik besteht aus zwei Teilen:

1. Korrektheit (Soundness): Der Kalkül erzeugt geschlossene Tableaus nur für Tautologien
2. Vollständigkeit (completeness): Für jede Tautologie kann der Tableaukalkül ein geschlossenes Tableau erzeugen.

Im folgenden verwenden wir "Korrektheit" im Sinne der soundness.

Definition 2.45. *Ein Pfad eines Tableaus ist* erfüllbar, *wenn die Konjunktion aller Formeln auf dem Pfad erfüllbar ist. Ein Tableau ist* erfüllbar, *wenn es einen Pfad gibt, der erfüllbar ist.*

Beachte: Wenn eine Menge die Formel 0 oder $\neg 1$ enthält, dann ist sie nicht erfüllbar.

Ein geschlossenes Tableau ist nicht erfüllbar.

Lemma 2.46. *Für die Tableau-Expansionsregeln gilt: Wenn T zu T' expandiert wird, dann ist T erfüllbar gdw. T' erfüllbar.*

Beweis. "$\Rightarrow$" Es genügt, sich einen erfüllbaren Pfad θ anzuschauen, und eine Interpretation I der aussagenlogischen Variablen zu wählen, die alle Formeln des Pfades wahr macht und alle Fälle der Expansionsregeln durchzugehen.

- Wenn $\neg\neg X$ in θ, dann ist $I(\neg\neg X) = 1$ wahr, also auch $I(X) = 1$
- Wenn für die α-Formel $X \wedge Y$ gilt, daß $I(X \wedge Y) = 1$, dann auch $I(X) = I(Y) = 1$.
- Wenn für die β-Formel $X \vee Y$ gilt daß $I(X \vee Y)$, dann gilt auch entweder $I(X) = 1$ oder $I(Y) = 1$. Somit ist einer der Pfade, die θ fortsetzen, erfüllbar, entweder der mit dem Blatt X, oder der mit dem Blatt Y.
- Analog für die anderen α und β-Formeln.

"$\Leftarrow$": Wenn ein Pfad in T' erfüllbar ist, dann ist auch der verkürzte Pfad in T, der den Pfad T' erzeugt hat, erfüllbar. □

Korollar 2.47. *Der Tableaukalkül TK_A ist sound.*

Beweis. Gegeben eine Formel F, die keine Tautologie ist. Dann ist $\neg F$ erfüllbar. Der Tableaukalkül startet mit $\neg F$, also ist das initiale Tableau erfüllbar, also auch alle daraus erzeugten, insbesondere kann kein geschlossenes Tableau erzeugt werden. □

Die Vollständigkeit kann im Fall der Aussagenlogik auf relativ einfache Weise gezeigt werden, allerdings ist diese Beweis-methode nicht auf allgemeinere Logiken übertragbar.

Zwischenziel: Zeige die Terminierung des Tableaukalkül für Aussagenlogik.

Bemerkung 2.48. Erinnerung:

Ein *fundierte (well-founded)* Ordnung ist eine partielle Ordnung $\geq$ auf einer Menge M, so daß es keine unendlich absteigenden Ketten $a_1 > a_2 > \ldots$ in M gibt.

Es gilt: Die *lexikographische Kombination* von fundierten Ordnungen ist wieder fundierte Ordnung. D.h. Seien $M_1, \geq_1$ und $M_2, \geq_2$ fundierte Ordnungen. Dann ist $M_1 \times M_2$ mit der Ordnung $(m_1, m_2) >_{12} (m_1', m_2')$ gdw. $m_1 >_1 m_1'$ oder ($m_1 =_1 m_1'$ und $m_2 >_2 m_2'$) fundiert.

Eine weitere nützliche Konstruktion von fundierten Ordnungen gibt es mittels Multimengen, sogenannte Multimengenordnungen: Sei $(M, >)$ eine Menge mit fundierter Ordnung, dann kann man auf Multimengen (Mengen bei denen mehrfaches Vorkommen von Elementen erlaubt ist) über M eine Ordnung erklären:

Seien A und B Multimengen über M, dann definiert man $A >> B$, wenn es weitere Multimengen X und Y gibt, so daß $B = (A \setminus X) \cup Y$ und es zu jedem Element von Y ein echt größeres Element in X gibt.

Es gilt: Die Multimengenordnung $>>$ ist eine partielle Ordnung. Sie ist fundiert, gdw. $>$ fundiert ist.

Z. B Nimmt man die natürlichen Zahlen mit der $>$-Ordnung, dann gilt $\{3,3,2,1\} >> \{3,2,2,2\}$, denn $\{3,2,2,2\} = \{3,3,2,1\} \setminus \{3,1\} \cup \{2,2,2\}$.

Mit folgender *Steuerung* terminiert der Tableaukalkül: Jede Formel wird auf jedem Pfad nur einmal expandiert.

Beachte, daß eine Formel die als Markierung eines Knoten auftritt, möglicherweise mehrfach expandiert werden muß, da mehrere Pfade hindurchgehen können. Dies ist unabhängig von der Art der Formel.

Jeden Pfad können wir einem Blatt zuordnen, dem Endknoten des Pfades.

Lemma 2.49. *Der Tableaukalkül für Aussagenlogik terminiert , wenn man jede Formel auf jedem Pfad höchstens einmal expandiert.*

Beweis. Wir konstruieren ein fundiertes Maß für die Größe eines Tableaus als Multimenge, so daß jede Expansionsregel dieses Maß verkleinert.

1. Die Größe einer Formel sei eine gewichtete Anzahl der Zeichen: Das Zeichen $\Leftrightarrow$ wird doppelt gezählt, Klammern werden nicht gezählt, alle anderen einfach.
2. Die Größe eines Pfades wird gemessen durch die Multimenge der Größe der Formeln an seinen Knoten, wobei der Expansions-status für diesen Pfad noch berücksichtigt werden muß: Ein Knoten wird nur dann in das Maß aufgenommen, wenn er auf diesem Pfad noch nicht expandiert worden ist.
3. Die Größe des Tableaus ist die Multimenge der Größe aller nicht geschlossenen Pfade.

Betrachte typische Fälle der Expansionsregeln:

- $\neg\neg X \to X$ macht einen Pfad kleiner, unabhängig von der Art der Formel X, da danach $\neg\neg X$ in diesem Pfad schon expandiert wurde. D.h. im Maß des Pfades wird die Größe von $\neg\neg X$ durch die Größe von X ersetzt.
- $X \wedge Y$ wird durch X, Y ersetzt. D.h. $gr(X) + gr(Y) + 1$ wird im Maß des Pfades durch $\{gr(X), gr(Y)\}$ ersetzt. Analog für die anderen α-Formeln.
- $X \vee Y$ wird durch $X \mid Y$ ersetzt. D.h. es werden zwei Pfade erzeugt. D.h ein Pfad θ wird durch zwei andere ersetzt. Im Maß wirkt sich das wie folgt aus: Da $gr(X \vee Y) > gr(X), gr(Y)$ für alle Formeln X, Y. Dadurch wird $gr(\theta) >> gr(\theta_1)$ und $gr(\theta) >> gr(\theta_2)$. Damit wird die Multimenge aller Pfade kleiner.
- $X \Leftrightarrow Y$ wird durch $X \Rightarrow Y$ und $Y \Rightarrow \ X$ ersetzt. Da $gr(X \Leftrightarrow Y) - 1 = gr(X \Rightarrow Y)$, kann man die gleiche Argumentation wie oben anwenden.
- Andere Fälle analog.

Das zugehörige Maß für das Tableau ist fundiert, also terminiert das Verfahren. □

Definition 2.50. *Eine* (aussagenlogische) Hintikka-Menge *ist eine Menge H von aussagenlogischen Formeln, für die folgendes gilt:*

1. *Es kann nicht gleichzeitig $A \in H$ und $\neg A \in H$ für ein Atom A gelten.*
2. $0 \notin H, \neg 1 \notin H$
3. $\neg\neg X \in H \Rightarrow X \in H$
4. $\alpha \in H \Rightarrow \alpha_1 \in H$ *und* $\alpha_2 \in H$
5. $\beta \in H$ *impliziert* $\beta_1 \in H$ *oder* $\beta_2 \in H$

Lemma 2.51. (Hintikka) *Jede aussagenlogische Hintikka-Menge ist erfüllbar.*

Beweis. Wir zeigen, daß es eine Interpretation gibt, so daß jede Formel in H den Wert 1 erhält. Wenn eine Variable $A \in H$, so definiere $I(A) := 1$, wenn $\neg A \in H$, dann definiere $I(A) := 0$, wenn weder A noch $\neg A$ in H, dann definiere $I(A)$ beliebig, z.B. $I(A) := 1$. Induktion über die Termstruktur zeigt jetzt, daß für alle Formeln $F \in H$, $I(F) = 1$ gilt. □

Lemma 2.52. *Sei T ein Tableau, auf das keine Expansionsregeln mehr angewendet werden können. Dann ist jeder Pfad entweder geschlossen oder entspricht einer aussagenlogischen Hintikka-Menge.*

Beweis. Wenn ein nicht geschlossener Pfad existiert, dann zeigen die Expansionsregeln, daß dieser Pfad dann eine Hintikka-Menge ist. Hierbei genügt entsprechend der oben angegebenen Steuerung daß jede Formel nur einmal expandiert wird. □

Satz 2.53. *Der Tableaukalkül für Aussagenlogik terminiert, ist korrekt und vollständig.*

Beweis. Wir haben schon die Korrektheit gezeigt. Wir betrachten den Fall, daß eine Tautologie eingegeben wird. In diesem Falle terminiert das Verfahren. Wenn ein Pfad existiert, der nicht geschlossen ist, so ist die Menge seiner Markierungen eine Hintikka-Menge, also erfüllbar, und somit auch die eingegebene Formel, was der Voraussetzung widerspricht. Also ist jeder Pfad geschlossen. □

Lemma 2.54. *Der Tableaukalkül konstruiert für erfüllbare Formeln ein Modell. Ein Modell kann man ablesen an jedem Pfad, der nicht geschlossen ist, aber auch nicht weiter expandiert werden kann.*

Begründung. Da erfüllbare Tableaus solange expandiert werden, bis alle Pfade entweder geschlossen sind oder eine Hintikka-Menge darstellen, erhält man zumindest einen Pfad mit einer Hintikka-Menge, wenn man das Tableauverfahren startet mit einer erfüllbaren Formel an der Wurzel . Dies ergibt ein Modell für einen Pfad. Beachtet man, daß im Beweis oben für erfüllbare Tableaus T , die zu T' expandiert werden, die Interpretation erhalten bleibt, kann man schließen, das die in diesem Pfad enthaltenen Atome ein Modell der Formel an der Wurzel definieren.