

Vorlesung Informatik A

**Boolesche Aussagenlogik, Umgang mit Quantoren
und der Resolutionskalkül**

**Wintersemester 2014/15
Institut für Informatik, Freie Universität Berlin**

Dozent: F. Hoffmann

(Stand 20.10.2014)

1 Boolesche Aussagenlogik

1.1 Grundbegriffe; Vom Booleschen Term zur Booleschen Funktion

Die klassische Boolesche Aussagenlogik (George Boole, engl. Mathematiker 1815 - 1864) beruht auf zwei Grundprinzipien, dem *Zweiwertigkeitsprinzip*, welches fordert, dass jede Aussage einen eindeutig bestimmten Wahrheitswert hat, der nur *wahr* oder *falsch* sein kann, und dem *Extensionalitätsprinzip*, nach dem der Wahrheitswert einer zusammengesetzten Aussage nur von den Wahrheitswerten ihrer Bestandteile abhängt. Wir werden im folgenden eine 1 für den Wahrheitswert *wahr* und eine 0 für *falsch* verwenden.

Zunächst sind folgende Fragen zu klären:

- Was sind Aussagen?
- Nach welchen Regeln kann man Aussagen zusammensetzen? Das ist die Syntax der Aussagenlogik.
- Nach welchen Regeln werden zusammengesetzte Aussagen interpretiert, das ist die Frage nach der Semantik.

Auf die alten Griechen, genauer Aristoteles (384 - 322 v.Chr.), geht die folgende Definition zurück:

Definition: Eine Aussage ist ein Satz (ein formalsprachliches Gebilde), von dem es sinnvoll ist zu sagen, er sei entweder wahr oder falsch.

Beispiele: Hier einige Beispiele aus der Mathematik.

1. Der Satz "*7 ist eine Primzahl.*" und der Satz "*7 ist eine ungerade Zahl.*" sind wahre Aussagen. Dagegen ist der Satz "*7 ist eine gerade Zahl.*" eine falsche Aussage. Genauer gesehen ist der letzte Satz die Negation des zweiten Satzes, denn *nicht ungerade* zu sein, ist (zumindest für ganze Zahlen) das gleiche, wie *gerade* zu sein.
2. "*Jede natürliche Zahl > 1 ist das Produkt von Primzahlen.*" ist eine wahre Aussage. Das ist die bekannte Primzahlzerlegung von natürlichen Zahlen, und wenn die Zahl selbst Primzahl ist, so besteht das Produkt nur aus einem Faktor, nämlich der Zahl selbst.
3. Der Satz " *$\sqrt{2}$ ist eine rationale Zahl.*" ist – wie man aus der Schulmathematik weiß – eine falsche Aussage, aber es bedarf schon einiger Überlegungen, um das zu zeigen.
4. Der Satz "*Jede gerade natürliche Zahl größer als 2 ist die Summe zweier Primzahlen*" ist eine Aussage, denn entweder es gibt eine gerade Zahl, die sich nicht als Summe zweier Primzahlen darstellen lässt (dann ist die Aussage falsch), oder es gibt keine solche Zahl (dann ist die Aussage wahr). Man nimmt an, dass die Aussage wahr ist (Goldbach-Vermutung), konnte das aber bisher noch nicht beweisen.

5. Der Satz *“Dieser Satz ist falsch.”* ist als Russells Paradoxon bekannt. Durch die spezielle Art des Bezugs auf sich selbst ist er weder wahr noch falsch (beides führt zum Widerspruch) und ist deshalb **keine** Aussage. Genauso hat ein Satz wie *“Sei ruhig!”* keinen zugeordneten Wahrheitswert.

Die Zuordnung des Wahrheitswertes zu Einzelaussagen ist Sache der Einzelwissenschaft (Mathematik, Biologie, Physik etc.). Die Logik interessiert sich für den Wahrheitswert zusammengesetzter Aussagen.

Wie kann man Aussagen zu neuen Aussagen zusammensetzen?:

Das Zusammensetzen von Aussagen erfolgt durch die Verwendung von logischen Verknüpfungswörtern wie:

und, oder, nicht, wenn ... dann, genau dann wenn, entweder ... oder

Dafür gibt es Operationssymbole und -namen, hier die zunächst wichtigen:

- die **Negation** einer Aussage A , Notation: $\neg A$, gesprochen: “nicht A ”
- die **Konjunktion** von A und B , Notation: $A \wedge B$, gesprochen: “ A und B ”,
- die **Disjunktion** von A und B , Notation $A \vee B$, gesprochen: “ A oder B ”,
- die **Implikation**, Notation: $A \Rightarrow B$, gesprochen: “ A impliziert B ”,
- die **Äquivalenz**, Notation: $A \Leftrightarrow B$, gesprochen: “ A genau dann, wenn B ”,
- die **Antivalenz**, Notation: $A \oplus B$, gesprochen: “entweder A oder B ”

Es folgt die induktive Definition der Syntax der Aussagenlogik, also die Antwort auf die Frage, welche Konstrukte wir zulassen wollen. Sei dazu A eine Menge von Einzelaussagen, deren Wahrheitswert schon feststeht. Dazu soll immer wenigstens die wahre Aussage “*true*” und die falsche Aussage “*false*” gehören. Weiterhin sei V eine abzählbare Menge von Variablen, also einfach eine Menge von Platzhaltern für Aussagen.

Definition:(Syntax der Aussagenlogik)

Die Menge der *Booleschen Formeln (Booleschen Terme)* der Aussagenlogik über der Variablenmenge V und der Aussagenmenge A ist induktiv definiert:

1. Jedes $a \in A$ und jedes $v \in V$ sind Boolescher Term.
2. Wenn t Boolescher Term ist, so ist auch $(\neg t)$ Boolescher Term.
3. Wenn t_1 und t_2 Boolesche Terme sind, so sind es auch die Konjunktion $(t_1 \wedge t_2)$ und die Disjunktion $(t_1 \vee t_2)$.
4. (Minimalitätsprinzip) Nur Konstrukte, die sich durch endlich oft Anwenden der Regeln (1), (2) bzw. (3) erzeugen lassen, sind Boolesche Terme.

Bemerkungen:

1. Man beachte, dass diesen Konstrukten bisher keinerlei Interpretation bezüglich des Wahrheitswertes zugewiesen wurde.

- Die runden Klammern sind wichtig! Nur durch sie kann ggf. nachvollzogen werden, wie ein Term entstanden ist.
- Wir haben nicht alle oben eingeführten Operatoren benutzt! Der Grund wird später klar, tatsächlich könnte man es ohne weiteres tun und wir werden die anderen auch verwenden. Die Menge $\{\wedge, \vee, \neg\}$ von Operatoren heißt *Standardsignatur* für Boolesche Formeln.

Boolesche Algebra, Rechnen mit Wahrheitswerten:

Sei $\mathbb{B} = \{0, 1\}$ die Menge der beiden Booleschen Wahrheitswerte. Mit $\mathbb{B} \times \mathbb{B}$ bezeichnen wir die Menge der geordneten Paare von Wahrheitswerten (das sogenannte kartesische Produkt von \mathbb{B} und \mathbb{B}), das heißt $\mathbb{B} \times \mathbb{B} = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$.

Wir definieren die Negation $\neg p$ für einen Wahrheitswert $p \in \mathbb{B}$ durch: $\neg 0 = 1$ und $\neg 1 = 0$. Die Negation kann also aufgefasst werden als Funktion $\neg : \mathbb{B} \rightarrow \mathbb{B}$.

Des weiteren betrachten wir die folgenden zweistelligen Booleschen Funktionen von $\mathbb{B} \times \mathbb{B}$ nach \mathbb{B} gegeben durch ihren Werteverlauf, die Konjunktion, Disjunktion, Implikation, Äquivalenz und Antivalenz von zwei Wahrheitswerten festlegen.

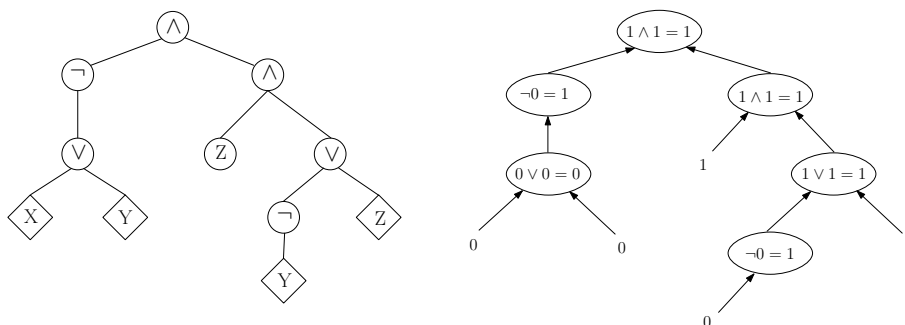
p	q	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$	$p \oplus q$
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	0	0	1
1	1	1	1	1	1	0

Man beachte, dass wir dieselben Operatorzeichen benutzen: einmal bei der Verknüpfung von Booleschen Termen (Syntax) und zum anderen beim Rechnen mit Wahrheitswerten. Dies ist eigentlich unzulässig, aber aus dem jeweiligem Kontext wird klar, auf welcher Ebene wir uns befinden, und bei der Definition der Semantik eines Booleschen Terms werden wir die inhaltliche Rechtfertigung finden.

An folgendem Beispiel wollen wir die einfache Konstruktion der sogenannten *Baumdarstellung eines Booleschen Terms* zeigen. Wir betrachten einen vollständig geklammerten Booleschen Term, zum Beispiel

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

Die Klammerung widerspiegelt eine Hierarchie. Als letzte Operation wurde eine Konjunktion ausgeführt. Diese greift links zu auf die Negation einer Disjunktion, rechts auf eine Konjunktion, usw. Auf der untersten Ebene stehen die Variablen.



Es sollte klar sein, dass ein vollständig geklammerter Term die Baumdarstellung eindeutig bestimmt und umgekehrt. Das rechte Bild zeigt die bottom-up-Auswertung für eine konkrete Belegung, s.u.

Ein vollständig geklammerter Term t hat einen Rang $rg(t)$. Dies ist eine natürliche Zahl, die die maximale Schachtelungstiefe von Klammern angibt, bzw. in der zugehörigen Baumdarstellung entspricht dies der maximalen Anzahl von Schritten, die man vom obersten Niveau nach unten laufen kann, bis man bei einer Variablen angekommen ist. In obigem Beispiel ist der Rang 4. Formal folgt die Definition des Ranges dem induktiven Aufbau der Terme über einer Menge A von Einzelaussagen und V von Variablen.

Definition:(Rang eines Booleschen Terms)

1. Für jedes $a \in A$ und jedes $v \in V$ sei $rg(a) = rg(v) = 0$.
2. Falls $t = (\neg t_1)$, so sei $rg(t) = rg(t_1) + 1$.
3. Falls $t = (t_1 \vee t_2)$ oder falls $t = (t_1 \wedge t_2)$, so sei $rg(t) = \max\{rg(t_1), rg(t_2)\} + 1$

Wir werden später die Rangfunktion benutzen, um Aussagen über Boolesche Terme mittels vollständiger Induktion über den Formelrang zu beweisen.

Die Interpretation eines Booleschen Terms:

Zunächst hängt die Interpretation eines Terms t von der konkreten Belegung der Variablen V mit Aussagen, genauer mit dem Wahrheitswert dieser Aussagen zusammen.

Sei $\beta : V \rightarrow \mathbb{B}$ eine solche konkrete Zuordnung. Ziel ist es einen Wahrheitswert $I_\beta(t)$ zu finden, dies ist die Interpretation von t bezüglich β .

Wieder folgt die Bestimmung dieses Wertes dem induktiven Aufbau der Formeln.

Definition: Die Interpretation von t über V bezüglich der konkreten Belegung β ist gegeben durch:

1. Falls $t = a$ mit $a \in A$ so ist $I_\beta(a)$ nach Annahme bekannt, das ist der Wahrheitswert der Einzelaussage a . Insbesondere ist $I_\beta(true) = 1, I_\beta(false) = 0$
Falls $t = v$ für $v \in V$ so setzen wir $I_\beta(v) = \beta(v)$.
2. Falls $t = (\neg t_1)$, so sei $I_\beta(t) = \neg I_\beta(t_1)$.
3. Falls $t = (t_1 \vee t_2)$, so sei $I_\beta(t) = I_\beta(t_1) \vee I_\beta(t_2)$.
Analog, falls $t = (t_1 \wedge t_2)$, so sei $I_\beta(t) = I_\beta(t_1) \wedge I_\beta(t_2)$.

Anmerkungen:

1. Schaut man sich die Baumdarstellung des Termes an, so entspricht die Interpretation des Termes bezüglich β einer bottom-up-Auswertung des Baumes. An die Stelle der Variablen treten die konkreten Wahrheitswerte und die Operatoren sind dann jene aus der Booleschen Algebra.
2. Nochmal zu Verdeutlichung: In der Gleichung $I_\beta(t_1 \vee t_2) = I_\beta(t_1) \vee I_\beta(t_2)$ steht auf der linken Seite das \vee aus der Syntaxdefinition der Terme, das \vee auf der rechten Seite operiert auf der Ebene der Wahrheitswerte!

Wenn $V = \{x_1, \dots, x_n\}$, so kann man eine konkrete Belegung β identifizieren mit einem sogenannten n -Tupel (b_1, \dots, b_n) von Wahrheitswerten. Das ist einfach die geordnete Aufzählung der Werte von β , genauer $b_i = \beta(x_i)$. Die Menge aller 2^n verschiedenen solcher n -Tupel bezeichnet man mit $\underbrace{\mathbb{B} \times \mathbb{B} \times \dots \times \mathbb{B}}_{n \text{ mal}}$.

Definition: Eine n -stellige Boolesche Funktion f ist eine Funktion

$$f : \underbrace{\mathbb{B} \times \mathbb{B} \times \dots \times \mathbb{B}}_{n \text{ mal}} \rightarrow \mathbb{B}$$

Damit definiert jeder Boolesche Term t über einer n -elementigen Variablenmenge eine n -stellige Boolesche Funktion

$$f_t : \mathbb{B} \times \mathbb{B} \times \dots \times \mathbb{B} \rightarrow \mathbb{B}$$

$$f_t(b_1, \dots, b_n) = I_\beta(t)$$

Merke: Jeder Boolesche Term definiert eine Boolesche Funktion, das ist seine Interpretation, seine abstrakte Bedeutung, seine **Semantik**.

Definition:

1. Zwei Boolesche Terme t_1, t_2 heißen *semantisch äquivalent*, geschrieben $t_1 \equiv t_2$, falls ihre Interpretationen gleich sind, das heißt, der Werteverlauf (die Wertetabellen) der Funktionen f_{t_1} und f_{t_2} sind gleich.
2. Ein Boolescher Term t heißt *erfüllbar*, falls es eine konkrete Belegung (b_1, \dots, b_n) der Variablen gibt, so dass $f_t(b_1, \dots, b_n) = 1$. Diese Belegung heißt dann *erfüllend*.
3. Ein Boolescher Term heißt *Tautologie* oder auch *logisch gültig*, falls jede Belegung erfüllend ist.
4. Ein Boolescher Term heißt *Kontradiktion*, falls keine Belegung erfüllend ist.

Wie überprüft man algorithmisch, ob Terme diese Eigenschaften haben? Eine brute-force-Lösung besteht in der Konstruktion der Wertetabellen. Man beachte, dass bei $n = 64$ die Tabelle 2^{64} Zeilen hat! Dauert das Auswerten einer Zeile $10^{-6}s$ auf einem Rechner, so dauert das Auswerten der gesamten Tabelle ca. $5 \cdot 10^5$ Jahre!!!

Für kleine n kann man das natürlich noch machen und so ist etwa der folgende Term $t' = (((\neg x) \wedge (\neg y)) \wedge z)$ zu obigen Term t in der Baumdarstellung semantisch äquivalent.

Konventionen I:

Zur vereinfachten Darstellung von Booleschen Termen werden folgende Vereinbarungen getroffen.

1. Das äußere Klammerpaar kann weggelassen werden.
2. Die Bindungskraft der logischen Operatoren $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ nimmt in dieser Reihung von links nach rechts ab!

Bsp.: Der Term $t = (((\neg x) \wedge (\neg y)) \wedge z)$ lässt sich somit schreiben als $t = (\neg x \wedge \neg y) \wedge z$

3. Auch die Verwendung verschiedener Klammerarten erleichtert die Lesbarkeit.

4. Im Zweifelsfall immer Klammern setzen!!!

Einige semantische Äquivalenzen sind besonders wichtig und heißen deshalb **Boolesche Gesetze**.

Satz: Für beliebige Formeln α, β, γ gelten die folgenden semantischen Äquivalenzen:

$$\begin{array}{ll}
 (\alpha \wedge \beta) \wedge \gamma & \equiv \alpha \wedge (\beta \wedge \gamma) \\
 (\alpha \vee \beta) \vee \gamma & \equiv \alpha \vee (\beta \vee \gamma) & \text{(Assoziativität)} \\
 \alpha \wedge \beta & \equiv \beta \wedge \alpha \\
 \alpha \vee \beta & \equiv \beta \vee \alpha & \text{(Kommutativität)} \\
 \alpha \wedge (\beta \vee \gamma) & \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \\
 \alpha \vee (\beta \wedge \gamma) & \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma) & \text{(Distributivität)} \\
 \alpha \wedge \alpha & \equiv \alpha \\
 \alpha \vee \alpha & \equiv \alpha & \text{(Idempotenz)} \\
 \\
 \alpha \wedge (\alpha \vee \beta) & \equiv \alpha \\
 \alpha \vee (\alpha \wedge \beta) & \equiv \alpha & \text{(Absorption)} \\
 \neg(\alpha \wedge \beta) & \equiv \neg\alpha \vee \neg\beta \\
 \neg(\alpha \vee \beta) & \equiv \neg\alpha \wedge \neg\beta & \text{(deMorgansche Regel)} \\
 \neg\neg\alpha & \equiv \alpha & \text{(Doppelnegation)} \\
 \alpha \Rightarrow \beta & \equiv \neg\alpha \vee \beta \\
 \alpha \Rightarrow \beta & \equiv \neg\beta \Rightarrow \neg\alpha & \text{(Kontraposition)} \\
 \alpha \Leftrightarrow \beta & \equiv (\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta) \\
 \alpha \Rightarrow \beta \wedge \gamma & \equiv (\alpha \Rightarrow \beta) \wedge (\alpha \Rightarrow \gamma) \\
 \alpha \Rightarrow \beta \vee \gamma & \equiv (\alpha \Rightarrow \beta) \vee (\alpha \Rightarrow \gamma) \\
 \alpha \wedge \beta \Rightarrow \gamma & \equiv (\alpha \Rightarrow \gamma) \vee (\beta \Rightarrow \gamma) \\
 \alpha \vee \beta \Rightarrow \gamma & \equiv (\alpha \Rightarrow \gamma) \wedge (\beta \Rightarrow \gamma)
 \end{array}$$

Die Beweise können einfach mit Hilfe von Wertetabellen geführt werden. (s. Übung)

Beispiel: Man beachte, dass im obigen Satz α, β, γ ganze Formeln sein können, nicht nur einzelne Variable. Der Beweis der folgenden Äquivalenz mit Wahrheitstafeln würde 16 Zeilen erfordern. Verwendet man dagegen die Absorption und die doppelte Negation zur Ersetzung von Subformeln, so erhält man einen einfachen und kurzen Beweis.

$$\begin{aligned}
 p_1 \vee ((p_2 \vee p_3) \wedge \neg(\neg p_1 \wedge (\neg p_1 \vee p_4))) & \equiv p_1 \vee ((p_2 \vee p_3) \wedge \neg\neg p_1) \\
 \equiv p_1 \vee ((p_2 \vee p_3) \wedge p_1) & \equiv p_1
 \end{aligned}$$

Wir wollen das zugrundeliegende Substitutionsprinzip noch einmal explizit formulieren. Bezeichne $t[x/t']$ den Booleschen Term, der entsteht, wenn ich im Term t jedes Vorkommen der Variable x durch den Term t' ersetze.

Satz:(Substitution und semantische Äquivalenz)

Sei t ein Boolescher Term, x eine darin vorkommende Variable und t_1, t_2 zwei semantisch äquivalente Terme. Es gilt:

$$t[x/t_1] \equiv t[x/t_2] \quad \text{und} \quad t_1[x/t] \equiv t_2[x/t]$$

Hinweis: Man mache sich dies an den zugehörigen Baumdarstellungen klar!

Wer das Formale nachlesen will: Broy, Informatik, eine grundlegende Einführung, Band 1, Springer, 1998

Konvention II: Wegen der Assoziativität von Konjunktion bzw. Disjunktion schreiben wir $\alpha \wedge \beta \wedge \gamma$ bzw. $\alpha \vee \beta \vee \gamma$ auch völlig ohne Klammern, obwohl natürlich dies weiterhin für uns zweistellige Operatoren sind!

1.2 Von der Booleschen Funktion zum Booleschen Term

Wir haben gesehen, wie man einem Booleschen Term t über einer Variablenmenge $V = \{x_1, \dots, x_n\}$ seine Interpretation zuordnet, dies ist eine n -stellige Boolesche Funktion $f_t : \mathbb{B} \times \dots \times \mathbb{B} \rightarrow \mathbb{B}$.

Beobachtung 1: Es gibt unendlich viele syntaktisch verschiedene Boolesche Terme, die ein und dieselbe Boolesche Funktion repräsentieren.

Beweis: Wie wir wissen ist zum Beispiel $t \equiv t \vee t \equiv t \vee t \vee t \equiv \dots$. Alle diese syntaktisch verschiedenen Booleschen Terme haben dieselbe semantische Interpretation f_t . \square

Beobachtung 2: Es gibt genau 2^{2^n} viele verschiedene n -stellige Boolesche Funktionen.

Beweis: Eine Funktion ist eineindeutig bestimmt durch ihre Wertetabelle. Statt Funktionen zu zählen, zählen wir Wertetabellen. Bei einer n -stelligen Booleschen Funktion hat diese 2^n Zeilen. Das sind die n -Tupel aus $\mathbb{B} \times \dots \times \mathbb{B}$. Jeder Zeile wird durch die Funktion einer von zwei möglichen Werten aus \mathbb{B} zugeordnet. Insgesamt gibt es

$$\underbrace{2 \cdot 2 \cdot 2 \cdot \dots \cdot 2}_{2^n \text{ Faktoren}} = 2^{2^n}$$

Möglichkeiten, die Wertetabelle zu vervollständigen.

Beobachtung 1 sagt, wenn wir einen Term gefunden haben, der eine konkrete Funktion repräsentiert, so gibt es unendlich viele, syntaktisch verschiedene Terme, die dies auch tun. Bleibt die Frage, gibt es für jede Boolesche Funktion wenigstens einen Term, der sie repräsentiert?

Aufgabe: Gegeben sei eine n -stellige Boolesche Funktion g . Finde einen Booleschen Term $t(g)$, so dass $f_{t(g)} = g$, der also genau die vorgegebene Semantik hat.

Wir betrachten zunächst folgende Teilaufgabe:

Sei g eine n -stellige Boolesche Funktion, die genau für das konkrete Argument

$(b_1, b_2, \dots, b_n) \in \mathbb{B} \times \dots \times \mathbb{B}$ den Wert 1 annimmt.

Finde einen Term über der Variablenmenge $V = \{x_1, x_2, \dots, x_n\}$, der diese Funktion repräsentiert!

Beispiel:

Sei $n = 3$ und $(b_1, b_2, b_3) = (1, 1, 0)$. Wir betrachten den Term $t = x_1 \wedge x_2 \wedge \neg x_3$ und überlegen uns, dass er das Gewünschte leistet. Zunächst ist $(1, 1, 0)$ erfüllende Belegung.

Weiterhin ist keine andere Belegung erfüllend, denn sie müsste sich von $(1, 1, 0)$ unterscheiden. Das heißt, an mindestens einer der ersten beiden Stellen steht eine 0 oder die dritte Stelle ist 1. Mithin wird die Konjunktion zu 0 ausgewertet.

Wir verallgemeinern das Beispiel zur Lösung unserer Teilaufgabe: Dazu vereinbaren wir folgende Notation: $x_i^{b_i} = x_i$ für $b_i = 1$ und $x_i^{b_i} = \neg x_i$ für $b_i = 0$.

Nun betrachten wir den Term $t = x_1^{b_1} \wedge \dots \wedge x_n^{b_n}$. Er wird bei der Belegung mit (b_1, \dots, b_n) zu 1 ausgewertet, während jede andere Belegung die Konjunktion zu 0 macht.

Der Term $\neg t = \neg(x_1^{b_1} \wedge \dots \wedge x_n^{b_n}) \equiv x_1^{-b_1} \vee \dots \vee x_n^{-b_n}$ wird also dann bei Belegung mit (b_1, \dots, b_n) zu 0 ausgewertet, bei allen anderen Belegungen zu 1.

Was hilft uns dies bei der Lösung unserer eigentlichen Aufgabe?

Sei $g^{-1}(1) = \{(b_1, \dots, b_n) | g(b_1, \dots, b_n) = 1\}$ das Urbild der 1 und $g^{-1}(0)$ das Urbild der 0 bei der Funktion g . Wir haben zumindestens die folgenden zwei Möglichkeiten, zu einem Term t zu kommen:

- Variante I: Sei t ein Boolescher Term, der genau dann zu 1 ausgewertet wird, wenn man diese Belegung **oder** diese Belegung **oder**...**oder** diese Belegung aus $g^{-1}(1)$ wählt.
- Variante II: Sei t ein Boolescher Term, der genau dann zu 1 ausgewertet wird, wenn man **nicht** diese Belegung **und nicht** diese Belegung **und**...**und nicht** diese Belegung aus $g^{-1}(0)$ gewählt hat

Bevor wir das formalisieren, noch ein paar Begriffe:

Definition: Ein *Literal* ist eine Variable oder deren Negation.

Eine Disjunktion $\alpha_1 \vee \dots \vee \alpha_n$ ($n \geq 1$) wird *disjunktive Normalform* (kurz DNF) genannt, wenn jedes α_i eine Konjunktion von Literalen oder ein einzelnes Literal ist.

Eine Konjunktion $\alpha_1 \wedge \dots \wedge \alpha_n$ ($n \geq 1$) wird *konjunktive Normalform* (kurz KNF) genannt, wenn jedes α_i eine Disjunktion von Literalen oder ein einzelnes Literal ist.

Beispiele: $(x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3) \vee x_2$ ist eine DNF, aber keine KNF.

Die Formeln $x_1 \vee x_2$ und $\neg x_1 \wedge x_4 \wedge \neg x_6$ sind sowohl DNF's als auch KNF's.

Satz: Jede n -stellige Boolesche Funktion f ist durch die sogenannte kanonische DNF $\text{dnf}(f)$ und durch die kanonische KNF $\text{knf}(f)$ über der Variablenmenge $V = \{x_1, x_2, \dots, x_n\}$ repräsentierbar, wobei:

$$\begin{aligned} \text{dnf}(f) &= \bigvee_{(b_1, \dots, b_n) \in f^{-1}(1)} (x_1^{b_1} \wedge \dots \wedge x_n^{b_n}) \\ \text{knf}(f) &= \bigwedge_{(b_1, \dots, b_n) \in f^{-1}(0)} (x_1^{-b_1} \vee \dots \vee x_n^{-b_n}) \end{aligned}$$

Im Spezialfall $f^{-1}(1) = \emptyset$ setzen wir $\text{dnf}(f) = \text{false}$ und im Spezialfall $f^{-1}(0) = \emptyset$ setzen wir $\text{knf}(f) = \text{true}$.

Beweis: Zunächst stellen wir fest, dass das genau die Umsetzung der obigen Varianten I und II ist, man störe sich nicht daran, dass die Funktion jetzt f heißt...

Nochmal formal für die DNF: Wir überzeugen uns davon, dass $x_1^{b_1} \wedge \dots \wedge x_n^{b_n}$ für die konkrete Belegung (b_1, \dots, b_n) wahr (1) und für alle anderen Belegungen falsch (0) ist. In der Tat wird eine Konjunktion von Literalen genau dann 1, wenn jedes Literal 1 ist, und diese Situation wird nur bei der Belegung (b_1, \dots, b_n) erreicht. Die Disjunktion über alle $(b_1, \dots, b_n) \in f^{-1}(1)$ führt dazu, dass alle Belegungen, die aus dieser Menge kommen, durch "ihre" Konjunktion akzeptiert werden, während Tupel aus $f^{-1}(0)$ von allen Konjunktionen aus $\text{dnf}(f)$ verworfen werden.

Der Beweis für $\text{knf}(f)$ ist analog: Jede Disjunktion $x_1^{-b_1} \vee \dots \vee x_n^{-b_n}$ wird für die konkrete Belegung (b_1, \dots, b_n) falsch (0) und wahr für alle anderen Belegungen. Durch die Konjunktion über alle $(b_1, \dots, b_n) \in f^{-1}(0)$ erzeugt man eine Formel, welche die vorgegebene Funktion f repräsentiert. \square

Korollar: Jeder Boolesche Term t ist semantisch äquivalent zu einem Term in DNF und zu einem Term in KNF.

Beweis: Wir bilden zu t die zugehörige Interpretation f_t und für diese Funktion dann die kanonische DNF bzw. kanonische KNF. \square

Alternativ kann man einen Beweis mittels vollständiger Induktion über den Formelrang von t führen (später).

Ausblick: Zum Verständnis von DNF und KNF kann man auch geometrische bzw. graphentheoretische Hilfsmittel einsetzen: Die Menge der n -Tupel $\{0, 1\}^n$ bildet auch die Knotenmenge des n -dimensionalen Würfelgraphen Q_n . Jede Konjunktion von k Literalen repräsentiert einen $(n - k)$ -dimensionalen Unterwürfel. Eine DNF muss durch die Unterwürfel der in ihr auftretenden Konjunktionen die Knotenmenge $f^{-1}(1)$ überdecken. In der kanonischen DNF wird jeder Knoten einzeln (als 0-dimensionaler Unterwürfel) überdeckt. Man sieht, dass man diese DNF vereinfachen kann, wenn mehrere Knoten aus $f^{-1}(1)$ einen höherdimensionalen Unterwürfel bilden.

Wir kommen darauf im 2. Logikteil am Ende des Semesters zurück.

Definition: Eine Menge von logischen Junktoren, die man zur Formelbildung einsetzt, wird *logische Signatur* genannt. Die Signatur $\{\neg, \vee, \wedge\}$ heißt *Boolesche Standardsignatur*. Eine logische Signatur ist *funktional vollständig*, wenn jede Boolesche Funktion durch eine mit dieser Signatur gebildeten Formel repräsentierbar ist.

Satz: Die Boolesche Signatur sowie die Signaturen $\{\neg, \wedge\}$ und $\{\neg, \vee\}$ sind funktional vollständig.

Die Vollständigkeit der Booleschen Standardsignatur folgt aus der Existenz von DNF's bzw. KNF's. Man beachte, daß die Formeln $p \wedge \neg p$ und $p \vee \neg p$ Funktionen repräsentieren, die konstant 0 bzw. konstant 1 sind. So erhält man auch die 0-stelligen Funktionen. Mit den deMorganschen Regeln kann man die Disjunktion (bzw. Konjunktion) durch Negation und Konjunktion (bzw. Negation und Disjunktion) eliminieren. So kann die Vollständigkeit der Signaturen $\{\neg, \wedge\}$ und $\{\neg, \vee\}$ auf die Vollständigkeit der Booleschen Standardsignatur zurückgeführt werden.

Es ist leicht zu sehen, daß die Signaturen $\{\vee\}$, $\{\wedge\}$ und $\{\vee, \wedge\}$ nicht funktional vollständig sind. Etwas schwerer ist der Beweis der Unvollständigkeit der Signatur $\{\neg, \Leftrightarrow\}$.

Dagegen ist die Signatur $\{|\}$, wobei $|$ den “nicht und”-Junktor (bekannt auch als NAND bzw. Sheffer–Strich) bezeichnet, funktional vollständig. Ebenso ist das NOR (Negation der Disjunktion) allein schon vollständig.

2 Der Gebrauch von Quantoren

Definition: Eine *Aussageform* $P(x_1, \dots, x_n)$, man spricht auch von einem Prädikat, über den Universen U_1, U_2, \dots, U_n ist ein Satz mit freien Variablen x_1, x_2, \dots, x_n , der zur Aussage wird (also einen Wahrheitswert hat), wenn jedes x_i durch einen konkreten Wert aus U_i ersetzt wird.

Zum Beispiel sind “ $P(x) : x > 1$ ” oder “ $Q(x) : x + 0 = x$ ” bzw. “ $R(x, y) : x + y = x$ ” Aussageformen für den Bereich der ganzen Zahlen. Dies sind so noch keine Aussagen, dazu bedarf es der Belegung der Variablen mit konkreten Werten!

Also, $P(1)$ ist falsch, $Q(0)$ ist wahr, $R(1, y)$ ist noch keine Aussage und $R(1, 0)$ ist wahr.

Es gibt weitere Möglichkeiten Aussagen mit Aussageformen zu verbinden.

Sei $P(x), x \in U$ eine Aussageform.

- Wir betrachten die Aussage (!): “Für jedes konkrete $x \in U$ gilt $P(x)$ ”. Dies ist eine Aussage, denn entweder stimmt es oder es stimmt nicht. In der mathematischen Notation wird dafür der sogenannte **Allquantor** \forall benutzt:

$$\forall x \in U : P(x)$$

- Weiterhin betrachten wir die Aussage “Es gibt (wenigstens) ein $x \in U$, für das $P(x)$ wahr ist”. Zur Notation benutzt man den sogenannten **Existenzquantor** \exists :

$$\exists x \in U : P(x)$$

Wenn das zugrundeliegende Universum aus dem Kontext klar ist, schreibt man auch verkürzt $\forall x P(x)$ bzw. $\exists x P(x)$.

Beispiele: Die Aussagen “ $\forall x \in \mathbb{N} : x + 0 = x$ ” und “ $\exists x \in \mathbb{N} : x^2 = x$ ” sind wahr, die Aussagen “ $\exists x \in \mathbb{N} : x + 1 = x$ ” und “ $\forall x \in \mathbb{N} : x^2 = x$ ” sind falsch.

Da wir es mit Aussagen zu tun haben, können wir diese auch mittels logischer Junktoren zu neuen Aussagen zusammensetzen und wir können von semantischer Äquivalenz sprechen.

Satz: Für beliebige Prädikate $P(x)$ und $Q(x)$ gelten die folgenden semantischen Äquivalenzen:

$$\begin{array}{ll} \neg \forall x P(x) & \equiv \exists x \neg P(x) & \neg \exists x P(x) & \equiv \forall x \neg P(x) \\ \forall x P(x) \wedge \forall x Q(x) & \equiv \forall x (P(x) \wedge Q(x)) & \exists x P(x) \vee \exists x Q(x) & \equiv \exists x (P(x) \vee Q(x)) \\ \forall x \forall y P(x, y) & \equiv \forall y \forall x P(x, y) & \exists x \exists y P(x, y) & \equiv \exists y \exists x P(x, y) \end{array}$$

Achtung: Die folgenden Formelpaare sind im allgemeinen nicht semantisch äquivalent:

$$\begin{array}{ll} (\forall x P(x) \vee \forall x Q(x)) & \not\equiv \forall x (P(x) \vee Q(x)) \\ (\exists x P(x) \wedge \exists x Q(x)) & \not\equiv \exists x (P(x) \wedge Q(x)) \end{array}$$

Konkrete Beispiele für die letzte Bemerkung erhält man für den Bereich der natürlichen Zahlen, wenn $P(x)$ (bzw. $Q(x)$) aussagt, daß x eine gerade (bzw. ungerade) Zahl ist.

Nochmal zur Schachtelung von mehreren Quantoren:

Für ein Prädikat $P(x, y)$ mit zwei freien Variablen können wir neue Prädikate bilden, zum Beispiel das Prädikat $Q(x) : \forall y P(x, y)$. Wenn man jetzt für x konkrete Werte einsetzt, bekommt man Aussagen. $\forall x \forall y P(x, y)$ beschreibt dann die Aussage, dass $Q(x)$ für jeden konkreten Wert von x wahr ist.

Quantifizierung von zwei Variablen

Aussage	Wann wahr?	Wann falsch?
$\forall x \forall y P(x, y)$ $\forall y \forall x P(x, y)$	$P(x, y)$ wahr für jedes Paar (x, y)	Es gibt wenigstens ein Paar (x, y) , für das $P(x, y)$ falsch ist
$\forall x \exists y P(x, y)$	Für jedes x gibt es ein (sein) y , so dass $P(x, y)$ wahr ist	Es gibt ein x , so dass für jedes y $P(x, y)$ falsch ist
$\exists x \forall y P(x, y)$	Es gibt ein x , das für jedes y $P(x, y)$ wahr macht	Für jedes x gibt es ein y , so dass $P(x, y)$ falsch ist.
$\exists x \exists y P(x, y)$ $\exists y \exists x P(x, y)$	Es gibt ein Paar (x, y) , so dass $P(x, y)$ wahr ist.	$P(x, y)$ ist falsch für jedes Paar (x, y)

Die oben angegebene Regel zur Negation von quantifizierten Aussagen gelten auch für geschachtelte Quantoren. Insbesondere gilt, vgl. letzte Spalte der Tabelle:

$$\begin{aligned}
 \neg \forall x \forall y P(x, y) &\equiv \exists x (\neg \forall y P(x, y)) \equiv \exists x \exists y \neg P(x, y) \\
 \neg \forall x \exists y P(x, y) &\equiv \exists x \forall y \neg P(x, y) \\
 \neg \exists x \forall y P(x, y) &\equiv \forall x \exists y \neg P(x, y) \\
 \neg \exists y \exists x P(x, y) &\equiv \forall x \forall y \neg P(x, y)
 \end{aligned}$$

Wir sprechen von *Negationsnormalform*, wenn die Negationsjunktoren nicht außerhalb von Quantoren stehen, sondern sich nur auf atomare Formeln beziehen und ansonsten die Standardsignatur verwendet wird.

Beispiele:

- Wir negieren die Aussage, dass es für alle $x < y$ immer ein z gibt mit $x < z < y$.

$$\begin{aligned}
 &\neg (\forall x (\forall y (x < y \Rightarrow (\exists z (x < z \wedge z < y)))))) \\
 \equiv &\exists x \exists y \neg ((x < y \Rightarrow (\exists z (x < z \wedge z < y)))) \\
 \equiv &\exists x \exists y (\neg (\neg (x < y) \vee (\exists z ((x < z \wedge z < y))))) \\
 \equiv &\exists x \exists y ((x < y) \wedge (\forall z \neg ((x < z) \wedge (z < y)))) \\
 \equiv &\exists x \exists y ((x < y) \wedge (\forall z ((x \geq z) \vee (z \geq y))))
 \end{aligned}$$

- Gelegentlich wird der Quantor $\exists!$ verwendet um auszudrücken, dass es genau ein Individuum aus dem Grundbereich gibt, das das Prädikat zur wahren Aussage macht. Dies kann man äquivalent schreiben als

$$\exists! x P(x) \equiv \exists x (P(x) \wedge \forall y (x \neq y \Rightarrow \neg P(y)))$$

Und damit

$$\neg \exists! x P(x) \equiv \forall x (\neg P(x) \vee \exists y (x \neq y) \wedge P(y))$$

Zum Schluss dieses Abschnitts noch ein Hinweis auf algorithmische Aspekte:

Allgemein ist die Frage, ob eine durch Quantoren gebildete Aussage wahr oder falsch ist, algorithmisch nicht entscheidbar. In vielen konkreten Fällen kann man die Frage aber durch genauere Überlegungen beantworten. Wie kann man in solchen Fällen sich selbst und andere von der Richtigkeit seiner Überlegungen überzeugen? Der typische Beweis dafür, dass eine quantifizierte Aussage wahr ist, erfolgt in drei Stufen. Zuerst wird die Aussage durch Anwendung von äquivalenten Umformungen aus der Aussagenlogik und aus dem obigen Satz über Äquivalenzen in eine Standardform gebracht, bei der alle auftretenden Quantoren am Anfang stehen (man nennt dies eine *Pränexform*). Danach erfolgt die Belegung der Variablen in Form eines Spiels zwischen zwei Parteien: Einem *Beweiser* und seinem *Gegenspieler*, der nachzuweisen versucht, dass die Aussage falsch ist. Dabei darf der Gegenspieler bei jedem Allquantor die entsprechende Variable x durch ein beliebiges Objekt a aus dem Individuenbereich belegen. Sollte die Aussage doch falsch sein (also nicht für alle Objekte gelten), würde der Gegenspieler gerade ein solche Objekt wählen. Ist die Aussage wahr, dann ist es (für den Beweiser) egal, welches Objekt a der Gegenspieler gewählt hat. Der Beweiser ist bei allen Existenzquantoren am Zuge und muss ein passendes Objekt (in Abhängigkeit von den vorher vom Gegenspieler gewählten Objekten) finden, für welches die nachfolgende Aussage wahr ist. Nachdem alle Variablen belegt sind, haben wir eine (variablenfreie) Aussage. Im letzten Schritt muss diese Aussage verifiziert (als wahr bewiesen) werden.

Wir wollen die drei Stufen eines solchen Beweises an einem einfachen Beispiel demonstrieren und die folgende Aussage für den Bereich der rationalen Zahlen beweisen.

Aussage: Für beliebige $x, y \in \mathbb{Q}$ mit $x < y$ gibt es eine von x und y verschiedene Zahl $z \in \mathbb{Q}$, die zwischen x und y liegt.

Die Beweisidee liegt klar auf der Hand: man setzt für z den Mittelwert aus x und y und kann dann die Behauptung nachrechnen. Wir stellen die Aussage als Formel mit Quantoren dar. Der Bereich \mathbb{Q} ist durch Verabredung festgelegt und wird nicht explizit genannt:

$$\forall x \forall y [(x < y) \Rightarrow \exists z (x < z \wedge z < y)]$$

In diesem Fall könnte man die erste Stufe überspringen und gleich mit dem Spiel der Festlegung der Werte beginnen, bei dem der Gegner x und y vorgibt und wir im Fall $x < y$ (als Beweiser) $z = \frac{x+y}{2}$ setzen. Um das dreistufige Verfahren formal korrekt umzusetzen, würden wir aber zuerst die Implikation hinter den beiden Allquantoren äquivalent umformen:

$$\begin{aligned} (x < y) \Rightarrow \exists z (x < z \wedge z < y) &\equiv \neg(x < y) \vee \exists z (x < z \wedge z < y) \\ &\equiv \exists z (x \geq y) \vee \exists z (x < z \wedge z < y) \\ &\equiv \exists z (x \geq y \vee (x < z \wedge z < y)) \end{aligned}$$

Im ersten Schritt wurde die Regel $p \Rightarrow q \equiv \neg p \vee q$ angewendet. Im zweiten Schritt wurde die Negation von $x < y$ in $x \geq y$ umgewandelt und der Existenzquantor $\exists z$ davorgesetzt - das kann man machen, weil $x \geq y$ in keiner Weise von z abhängt. Im letzten Schritt wurde die vierte Regel aus obigem Satz verwendet. Jetzt liegt die Aussage in Pränexform vor:

$$\forall x \forall y \exists z [x \geq y \vee (x < z \wedge z < y)].$$

Das Spiel beginnt mit der Vorgabe von zwei Werten $x = a$ und $y = b$ durch den Gegenspieler. Der Beweiser setzt $z = (a + b)/2$.

Nun erfolgt die Verifikation der Aussage $a \geq b \vee (a < (a + b)/2 \wedge (a + b)/2 < b)$ durch Betrachtung von zwei Fällen: Entweder es gilt $a \geq b$, dann wird die Aussage durch den linken Term der Disjunktion wahr oder es gilt $a < b$. In diesem Fall müssen beide Ungleichungen auf der rechten Seite der Disjunktion erfüllt sein, aber beides folgt über den Zwischenschritt $a/2 < b/2$ jeweils kombiniert mit den Ungleichungen $a/2 \leq a/2$ bzw. $b/2 \leq b/2$.

Bleibt die Frage, warum man diese Argumentation nicht für den Bereich \mathbb{N} der natürlichen Zahlen wiederholen kann. Die Antwort ist wieder nicht schwer: Der Ausdruck $\frac{a+b}{2}$ führt in vielen Fällen aus dem Bereich \mathbb{N} heraus. Das ist aber noch kein Beweis dafür, dass die Aussage über diesem Bereich falsch ist. Um das zu zeigen, beweist man, dass die negierte Aussage wahr ist. Wir bilden die Negation mit Hilfe der Regeln (1) und (2) aus obigem Satz und der deMorganschen Regel:

$$\exists x \exists y \forall z [(x < y) \wedge (x \geq z \vee z \geq y)].$$

Da die Aussage bereits in Pränexform vorliegt, kann das Spiel der Variablenbelegung beginnen. Wir setzen als Beweiser $x = 0$ und $y = 1$. Der Gegenspieler belegt z mit einem Wert c . Wir müssen jetzt die Aussage $0 < 1 \wedge (0 \geq c \vee c \geq 1)$ verifizieren. Offensichtlich ist $0 < 1$ wahr und wir müssen nur noch zeigen, dass mindestens eine der Ungleichungen aus $(0 \geq c \vee c \geq 1)$ wahr ist. Wenn aber die erste Ungleichung nicht erfüllt ist, muss c eine natürliche Zahl größer als 0, also mindestens 1 sein und damit ist die zweite Ungleichung erfüllt.

3 Resolutionskalkül

Die Grundlagen der Aussagenlogik wurden bereits Kapitel 1 besprochen. Bisher dienten Aussageverknüpfungen nur als Mittel, um bestimmte Sachverhalte in eine formale Sprache zu übertragen und um die Struktur von mathematischen Beweisen besser zu verstehen. Die Hauptmotivation zur Beschäftigung mit diesem Gebiet greift aber wesentlich weiter: Zum einen geht es um das formale (und möglichst automatische) Ableiten von neuen Aussagen aus einer gegebenen Menge von Aussagen (Voraussetzungen), zum anderen um Verfahren zur Prüfung, ob eine Aussage allgemeingültig (Tautologie) oder erfüllbar ist. Wir werden sehen, dass beide Problemstellungen eng zusammenhängen. Das führt uns auf die Suche nach Methoden zum automatisierten Beweisen vom mathematischen Theoremen und letztlich zur Frage, ob, in welchem Sinne und wie die Tätigkeit von Mathematikern durch Computer ersetzt werden kann.

Ein *Kalkül* ist eine Kollektion von syntaktischen Umformungsregeln, die unter gegebenen Voraussetzungen aus bereits vorhandenen Formeln neue Formeln erzeugen. Der *Resolutionskalkül* besteht aus einer einzigen Umformungsregel – der sogenannten *Resolution*. Das gesamte Verfahren dient dazu, die Unerfüllbarkeit einer Formelmenge zu testen und gegebenenfalls nachzuweisen. Das Verfahren ist einfach, aber auf Grund der NP-Vollständigkeit des Erfüllbarkeitsproblems muss man damit rechnen, dass auch dieses Verfahren für einige Eingaben exponentielle Laufzeit erfordert.

Sei eine endliche Formelmenge $X = \{F_1, \dots, F_n\}$ gegeben. Wir setzen voraus, daß alle Formeln bereits in KNF vorliegen. Da die Formelmenge X genau dann erfüllbar (unerfüllbar) ist, wenn die Formel $F = F_1 \wedge \dots \wedge F_n$ erfüllbar (unerfüllbar) ist, reicht es aus, die Unerfüllbarkeit einer KNF-Formel

$$F = (l_{1,1} \vee l_{1,2} \vee \dots \vee l_{1,n_1}) \wedge \dots \wedge (l_{k,1} \vee l_{k,2} \vee \dots \vee l_{k,n_k})$$

zu testen, wobei alle $l_{i,j} \in \{x_1, x_2, \dots\} \cup \{\neg x_1, \neg x_2, \dots\}$ Literale sind. Zur Vereinfachung wird F als eine Menge \mathcal{K}_F von Klauseln geschrieben, welche die einzelnen Disjunktionsglieder repräsentieren:

$$\mathcal{K}_F = \{\{l_{1,1}, l_{1,2}, \dots, l_{1,n_1}\}, \dots, \{l_{k,1}, l_{k,2}, \dots, l_{k,n_k}\}\}$$

Für jedes Literal l definieren wir $\bar{l} = \begin{cases} \neg p_i & \text{falls } l = p_i \\ p_i & \text{falls } l = \neg p_i \end{cases}$

Definition: Seien K_1, K_2 Klauseln und l ein Literal mit $l \in K_1$ und $\bar{l} \in K_2$. Dann wird die Klausel $R = (K_1 \setminus \{l\}) \cup (K_2 \setminus \{\bar{l}\})$ ein *Resolvent* von K_1 und K_2 genannt.

Zur Darstellung nutzt man die folgende Diagrammschreibweise:



Die leere Klausel wird explizit als Resolvent zugelassen. Sie wird durch das Symbol \square bezeichnet. Die leere Klausel gilt als nicht erfüllbar, also als eine Kontradiktion.

Resolutions-Lemma: Sei F eine Formel in KNF, dargestellt als Klauselmeng \mathcal{K}_F und sei R ein Resolvent zweier Klauseln aus \mathcal{K}_F . Dann sind F und die durch $\mathcal{K}_F \cup \{R\}$ dargestellte Formel F' logisch äquivalent.

Beweis: Eine Richtung in dieser Äquivalenz ist einfach zu zeigen:

Wenn $\omega : \text{Var} \rightarrow \mathbb{B}$ eine erfüllende Belegung für F' ist, dann nimmt jede Klausel aus F' unter ω den Wert 1 an. Damit ist ω aber auch erfüllende Belegung für F .

Für die Gegenrichtung ist eine etwas genauere Analyse notwendig. Wir nehmen an, dass $\omega : \text{Var} \rightarrow \mathbb{B}$ eine erfüllende Belegung für F ist und wollen zeigen, dass dann auch alle Klauseln von F' unter ω den Wert 1 annehmen. Bis auf den Resolventen R folgt das aus der Voraussetzung. Um es auch für R zu zeigen, betrachten wir seine Entstehung

$$R = (K \setminus \{l\}) \cup (K' \setminus \{\bar{l}\}), \text{ wobei } l \in K \text{ und } \bar{l} \in K'$$

und machen eine Fallunterscheidung danach, welchen Wert das Literal l unter der Belegung ω hat:

- $\omega(l) = 0$: Die Klausel K kann nicht durch das Literal l den Wert 1 bekommen, also muss ein anderes Literal l' in K auftreten, das unter ω den Wert 1 hat. Dann ist $l' \in R$ und folglich nimmt R unter ω den Wert 1 an.
- $\omega(l) = 1$, d.h. $\omega(\bar{l}) = 0$: Die Klausel K' kann nicht durch das Literal \bar{l} den Wert 1 bekommen, also muss ein anderes Literal l' in K' auftreten, das unter ω den Wert 1 hat. Dann ist $l' \in R$ und folglich nimmt R unter ω den Wert 1 an. \square

Definition: Für eine beliebige Klauselmeng \mathcal{K} definiert man:

$$\begin{aligned} \text{Res}(\mathcal{K}) &= \mathcal{K} \cup \{R \mid R \text{ ist Resolvent zweier Klauseln aus } \mathcal{K}\} \\ \text{Res}^0(\mathcal{K}) &= \mathcal{K} \\ \text{Res}^{n+1}(\mathcal{K}) &= \text{Res}(\text{Res}^n(\mathcal{K})) \\ \text{Res}^*(\mathcal{K}) &= \bigcup_{n=1}^{\infty} \text{Res}^n(\mathcal{K}) \end{aligned}$$

Beispiel: Sei $\mathcal{K} = \text{Res}^0(\mathcal{K}) = \{\{x_1, x_2, \neg x_3\}, \{\neg x_1, x_4\}, \{x_2, \neg x_4\}\}$.

Dann ist

$$\begin{aligned} \text{Res}^1(\mathcal{K}) &= \mathcal{K} \cup \{\{x_2, \neg x_3, x_4\}, \{\neg x_1, x_2\}\}, \\ \text{Res}^2(\mathcal{K}) &= \text{Res}^1(\mathcal{K}) \cup \{\{x_2, \neg x_3\}\} \text{ und} \\ \text{Res}^*(\mathcal{K}) &= \text{Res}^2(\mathcal{K}) = \\ &= \{\{x_1, x_2, \neg x_3\}, \{\neg x_1, x_4\}, \{x_2, \neg x_4\}, \{x_2, \neg x_3, x_4\}, \{\neg x_1, x_2\}, \{x_2, \neg x_3\}\}. \end{aligned}$$

Man beachte, dass die durch die Klauselmeng \mathcal{K} dargestellte Formel

$F = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_4) \wedge (x_2 \vee \neg x_4)$ erfüllbar ist, denn jede Belegung ω mit $\omega(x_2) = \omega(x_4) = 1$ ist ein Modell für F .

Da eine endliche Klauselmeng \mathcal{K} nur endlich viele Literale enthält, ist auch die Menge der ableitbaren Resolventen endlich (eine Untermeng der Potenzmeng aller vorkommenden Literale). Folglich kann auch $\text{Res}^*(\mathcal{K})$ in endlich vielen Schritten erzeugt werden, denn gilt $\text{Res}^{n+1}(\mathcal{K}) = \text{Res}^n(\mathcal{K})$ für ein $n \in \mathbb{N}$, dann ist $\text{Res}^*(\mathcal{K}) = \text{Res}^n(\mathcal{K})$. So liefert der folgende Satz die Grundlage für ein endliches Verfahren, das die Nichterfüllbarkeit von Formeln (und Formelmengen) entscheidet.

Resolutionssatz: Eine Formel F in KNF, dargestellt durch die Klauselmeng \mathcal{K}_F ist genau dann unerfüllbar, wenn $\square \in \text{Res}^*(\mathcal{K}_F)$.

Die Aussage, dass $\square \in \text{Res}^*(\mathcal{K}_F)$ die Unerfüllbarkeit von F impliziert, wird als *Korrektheit des Resolutionskalküls* bezeichnet. Sie lässt sich leicht aus der Beobachtung ableiten, dass \square nur Resolvent von zwei Klauseln der Form $\{x_i\}$ und $\{\neg x_i\}$ sein kann. Da bereits $x_i \wedge \neg x_i$ nicht erfüllbar ist, folgt die Nichterfüllbarkeit von F aus dem Resolutionslemma (mehrfache Anwendung).

Die entgegengesetzte Implikation (aus der Unerfüllbarkeit von F folgt $\square \in \text{Res}^*(\mathcal{K}_F)$) wird *Vollständigkeit des Resolutionskalküls* genannt. Sie kann durch Induktion über die Anzahl der in F auftretenden Primformeln bewiesen werden. Wir verzichten an dieser Stelle auf den Beweis und verweisen auf das Buch von Schöning.

Der folgende Pseudocode beschreibt einen Algorithmus, der die Unerfüllbarkeit einer Formel F entscheidet, die durch eine Klauselmenge \mathcal{K} gegeben ist:

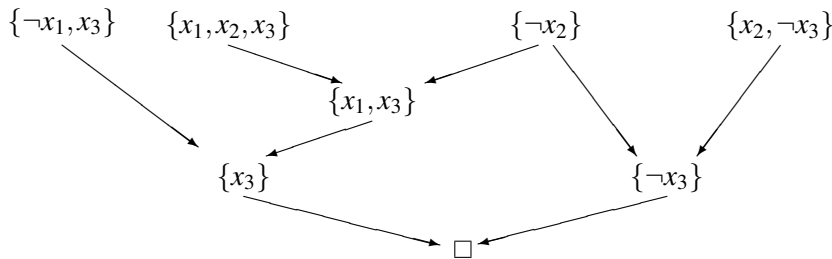
```

repeat
   $\mathcal{J} := \mathcal{K}$ ;
   $\mathcal{K} := \text{Res}(\mathcal{J})$ ;
until ( $\square \in \mathcal{K}$ ) or ( $\mathcal{J} = \mathcal{K}$ );
if  $\square \in \mathcal{K}$  then “ $F$  ist unerfüllbar” else “ $F$  ist erfüllbar”;

```

Generell sollte man beachten, dass zum Beweis der Unerfüllbarkeit einer Formel nicht unbedingt alle Resolventen gebildet werden müssen. Es reicht aus, nur die Resolventen zu bilden, die bei der *Deduktion* (Herleitung) von \square eine Rolle spielen.

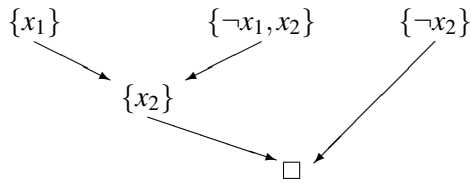
Beispiel: Sei $\mathcal{K} = \{\{x_1, x_2, x_3\}, \{\neg x_1, x_3\}, \{x_2, \neg x_3\}, \{\neg x_2\}\}$. Wir veranschaulichen die Deduktion der leeren Klausel durch einen sogenannten Resolutionsgraphen.



Auf Grund der bekannten Tatsache, dass F genau dann eine Tautologie ist, wenn $\neg F$ unerfüllbar ist, kann die Resolutionsmethode auch zum Tautologietest eingesetzt werden. Dazu muss aber die Negation $\neg F$ in KNF vorliegen. Gerade wenn F bereits als KNF-Formel gegeben ist, wird es oft sehr aufwändig sein, $\neg F$ in eine äquivalente KNF-Formel zu verwandeln, aber wenn F als DNF-Formel gegeben ist, kann man $\neg F$ durch doppelte Anwendung der deMorganschen Regel leicht in eine KNF verwandeln.

Eine weitere Anwendung für den Resolutionskalkül besteht im Beweis aussagenlogischer Folgerungen der Form $X \models F$. Sie basiert auf der Beobachtung, dass F genau dann aussagenlogische Folgerung aus einer Formelmeng $X = \{F_1, \dots, F_n\}$ ist, wenn die Formel $F_1 \wedge \dots \wedge F_n \wedge \neg F$ nicht erfüllbar ist. Vor der eigentlichen Anwendung des Resolutionskalküls muss man also wieder dafür sorgen, dass die Formeln F_1, \dots, F_n und die Negation $\neg F$ in KNF vorliegen.

Beispiel: Zum Beweis der Abtrennungsregel $\{x_1, x_1 \Rightarrow x_2\} \models x_2$ besteht der erste Schritt darin, $x_1 \Rightarrow x_2$ durch die äquivalente KNF $\neg x_1 \vee x_2$ zu ersetzen, und dann muss man die Deduktion von \square aus $\{\{x_1\}, \{\neg x_1, x_2\}, \{\neg x_2\}\}$ finden:



3.1 Hornformel und Einheitsresolventen

Definition: Variablen werden *positive Literale* genannt, ihre Negationen nennt man *negative Literale*. Ein KNF-Term F wird *Hornformel* genannt, falls jeder Maxterm (Klausel) in F höchstens ein positives Literal enthält.

Beispiel: Die folgende Formel ist eine Hornformel:

$$F = (x_1 \vee \neg x_3)(\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_4) \wedge x_2 \wedge \neg x_4$$

Eine besondere Eigenschaft der Klauseln einer Hornformel besteht darin, dass man sie als spezielle Implikationen ohne negierte Variable schreiben kann. Wir unterscheiden dazu drei Fälle:

- Der Maxterm enthält mindestens ein negatives Literal und genau ein positives Literal y :

$$G = \neg x_1 \vee \dots \vee \neg x_k \vee y \equiv \neg(x_1 \wedge \dots \wedge x_k) \vee y \equiv (x_1 \wedge \dots \wedge x_k) \rightarrow y$$
- Der Maxterm enthält nur negative Literale:

$$G = \neg x_1 \vee \dots \vee \neg x_k \equiv \neg(x_1 \wedge \dots \wedge x_k) \vee 0 \equiv (x_1 \wedge \dots \wedge x_k) \rightarrow 0$$
- Der Maxterm besteht nur aus einem positiven Literal:

$$G = y \equiv 0 \vee y \equiv \neg 1 \vee y \equiv 1 \rightarrow y$$

Ein Nachteil der Hornformeln liegt in ihrer eingeschränkten Ausdruckskraft, d.h. es gibt Boolesche Funktionen, die man nicht durch Hornformeln darstellen kann (ein einfaches Beispiel ist die Disjunktion). Dafür kann man das Erfüllbarkeitsproblem für Hornformeln relativ leicht lösen. Grundlage für dieses Verfahren sind die folgenden drei Beobachtungen:

1. Wenn in jeder Klausel einer Hornformel F ein positives Literal auftritt, dann ist F erfüllbar (die erfüllende Belegung setzt alle Variable auf 1).
2. Wenn in jeder Klausel einer Hornformel F ein negatives Literal auftritt, dann ist F erfüllbar (die erfüllende Belegung setzt alle Variable auf 0).
3. Wenn eine Hornformel F eine Klausel enthält, die nur aus einem positiven Literal x_i besteht, dann muss x_i in jeder erfüllenden Belegung von F den Wert 1 bekommen.

Die algorithmische Idee besteht nun darin, schrittweise alle Variablen zu markieren, die in einer erfüllenden Belegung den Wert 1 annehmen müssen. Man markiert eine Variable x_i also nur dann, wenn $\{x_i\}$ eine Klausel der aktuellen Formel ist. Wird x_i irgendwann markiert, hat das zwei Konsequenzen (Beobachtung 3):

- 1) Alle Klauseln, die x_i enthalten, sind automatisch erfüllt und werden deshalb gestrichen.
- 2) Alle negativen Literale $\neg x_i$ können nicht mehr zur Erfüllung ihrer Klauseln beitragen, d.h. man streicht alle Vorkommen von $\neg x_i$ (aber nicht die Klauseln selbst!). Entsteht bei diesem Prozess eine leere Klausel, dann ist die Hornformel nicht erfüllbar (Abbruch mit Antwort unerfüllbar), denn wir hatten davor eine Klausel $\{x_i\}$ - als Anlass für die Markierung - und eine Klausel $\{\neg x_i\}$, aus der nach der Streichung die leere Klausel entstanden ist.

Es gibt zwei weitere Abbruchbedingungen für den Algorithmus, nämlich wenn alle Klauseln gestrichen wurden (erfüllbar) und wenn keine weitere Klausel der Form $\{x_j\}$ existiert (erfüllbar nach Beobachtung 2).

Man kann die Idee für den Markierungsalgorithmus auch auf einen speziellen Resolutionskalkül für Hornformeln übertragen. Dabei wird auf das Streichen der erfüllten Klauseln verzichtet, d.h. es gibt wie bisher nur die Abbruchbedingungen, dass man die leere Klausel deduzieren kann oder dass man keine neuen Resolventen bilden kann. Der Unterschied zum vorher beschriebenen Resolutionskalkül besteht aber darin, dass man für unerfüllbare Hornformeln die leere Klausel herleiten kann, indem man ausschließlich Resolventen aus einer positiven Klausel $\{x_i\}$ mit einer anderen Klausel bildet.

Definition: Sei \mathcal{K} die Klauselmenge einer Hornformel. Ein Resolvent R aus den Klauseln $K_i, K_j \in \mathcal{K}$ wird *Einheitsresolvent* genannt, falls $|K_i| = 1$ oder $|K_j| = 1$. Analog zu $\text{Res}(\mathcal{K})$ und $\text{Res}^*(\mathcal{K})$ definiert man $1\text{-Res}(\mathcal{K})$ und $1\text{-Res}^*(\mathcal{K})$ mit Einheitsresolventen anstelle von Resolventen. Der Resolutionssatz tritt dann in folgender Gestalt auf.

Satz: Eine durch die Klauselmenge \mathcal{K} dargestellte Hornformel ist genau dann unerfüllbar, wenn $\square \in 1\text{-Res}^*(\mathcal{K})$.

Da die Einheitsresolution die Länge der Klauseln ständig verkürzt, ist diese Methode (für Hornklauseln!) besonders effizient.

Hornformeln und der Markierungsalgorithmus sind eine wichtige Grundlage von logischen Programmiersprachen, insbesondere von der Sprache PROLOG.