6 Logik

6.1 Boolesche Formeln und Boolesche Funktionen

Einige Grundbegriffe der Aussagenlogik wurden bereits in der Einleitung 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 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. Letzlich führt uns das zu der Frage, ob, in welchem Sinne und wie die Tätigkeit von Mathematikern durch Computer ersetzt werden kann.

In diesem Abschnitt wird ein exakter und damit etwas formalerer Zugang zur Aussagenlogik besprochen. Wir beginnen mit der Definition von Booleschen Formeln, die durch die Einführung eines Rangs Induktionsbeweise ermöglicht.

Definition: *Primformeln* oder auch *atomare Formeln* sind die Booleschen Konstanten 0 und 1 sowie die Variablen $\{x_1, x_2, x_3, \ldots\}$. *Boolesche Formeln* (oder auch *Boolesche Terme*) und ihr *Rang* werden durch den folgenden induktiven Prozeß definiert:

1. Alle Primformeln sind Formeln vom Rang 0, d.h. $rg(0) = rg(1) = rg(x_i) = 0$.
2. Für jede Formel α ist $\neg\alpha$ eine Formel mit $rg(\neg\alpha) = rg(\alpha) + 1$.
3. Für alle Formeln α und β sind auch $(\alpha \wedge \beta)$ und $(\alpha \vee \beta)$ Formeln und man definiert $rg(\alpha \wedge \beta) = rg(\alpha \vee \beta) = max\{rg(\alpha), rg(\beta)\} + 1$.
4. Alle Booleschen Formeln können aus Primformeln durch wiederholte Anwendungen der Regeln (2) und (3) erzeugt werden.

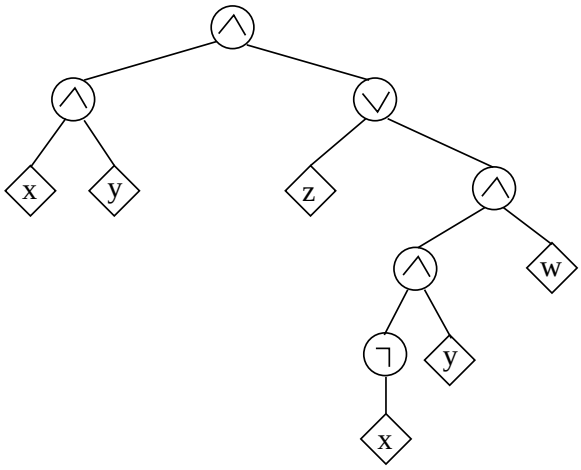
Anmerkungen:
1) Bei aussagenlogischen Formeln haben wir die Junktoren $\rightarrow, \leftrightarrow, \oplus$ kennengelernt. Spricht man über Boolesche Formeln, ist die Verwendung dieser Junktoren nicht üblich. Wir haben aber bereits gesehen, wie man sie logisch äquivalent durch $\vee, \wedge$ und $\neg$ ersetzen kann.

2) In der Literatur findet man häufig einen leicht modifizierten Ansatz, bei dem man auf die Booleschen Konstanten verzichtet und nur Variable als Primterme zulässt. Der Unterschied ist marginal, wir werden später noch genauer darauf eingehen.

3) Die Regeln zum Weglassen von Klammern wurden bereits besprochen. Zusätzlich wird vereinbart, dass bei gleichartigen Operationen jeder nichtgeklammerte Ausdruck als linksassoziativ geklammert angesehen wird.

4) Eine anschauliche Erklärung für den Aufbau von Termen und ihrem Rang bekommt man durch den sogenannten Syntaxbaum des Terms. Der Syntaxbaum einer Primformel besteht nur aus einem einzelnen Knoten, der mit der entsprechenden Variable markiert wird. Der Syntaxbaum eines Terms, der durch Regel 2 als $t = (\neg t')$ entsteht, hat einen mit $\neg$ markierten Wurzelknoten, unter dem der Syntaxbaum von t' steht. Für Terme die nach Regel 3 entstehen, wird der Wurzelknoten mit dem entsprechenden Junktor markiert und zu den Syntaxbäumen der zwei verwendeten Teilterme verbunden. Knoten, die mit Primformeln markiert sind, nennt man Blätter des Syntaxbaums. Der längste Weg vom Wurzelknoten zu einem Blatt beschreibt den Rang der Formel.

Beispiel: Der Term $((x \wedge y) \wedge (z \vee (((\neg x) \wedge y) \wedge w)))$ hat den folgenden Syntaxbaum:



Induktive Beweise können nun über den Rang geführt werden: Ist P eine Eigenschaft von Zeichenfolgen derart, daß

1) alle Primformeln p diese Eigenschaft haben und

2) für alle Formeln α, β, die die Eigenschaft P haben, auch die Formeln $\alpha \wedge \beta$, $\alpha \vee \beta$ und $\neg\alpha$ die Eigenschaft P haben,

dann haben alle Formeln die Eigenschaft P.

Auch die bereits besprochene Fortsetzung einer Wahrheitswertbelegung der Variablen auf zusammengesetzte Formeln, die sogennante Auswertung der Formel wird induktiv definiert.

Definition: Ist $Var_n = \{x_1, x_2 \ldots, x_n\}$ eine Menge von Primformeln, dann wird eine Abbildung $\omega : Var_n \longrightarrow \{0, 1\}$ eine *aussagenlogische Belegung* (auch kurz *Belegung, Realisierung*) genannt. Da man charakteristischen Funktionen von n–Mengen auch als n–Tupel von Nullen und Einsen darstellen kann, lässt sich die Menge Ω_n aller Belegungen der Varaiblenmenge $Var_n = \{x_1, x_2 \ldots, x_n\}$ auch durch $\{0, 1\}^n$ darstellen (die Bijektion ψ_n ordnet jeder Belegung ω das n-Tupel $(\omega(x_1), \omega(x_2), \ldots, \omega(x_n))$ zu).

Wir bezeichnen mit $\mathcal{F}_n$ die Menge aller aus Var_n erzeugbaren Formeln und definieren induktiv eine Funktion $\Phi_n : \Omega_n \times \mathcal{F}_n \longrightarrow \{0, 1\}$ durch

1) Für die Formeln vom Rang 0 und Belegungen $\omega \in \Omega_n$ ist $\Phi_n(\omega, 0) = 0$, $\Phi_n(\omega, 1) = 1$ und $\Phi_n(\omega, x_i) = \omega(x_i)$ für alle $x_i \in Var_n$.
2) Sei $\Phi_n(\omega, \alpha)$ und $\Phi_n(\omega, \beta)$ für alle Formeln $\alpha, \beta \in \mathcal{F}_n$ vom Rang $\leq k-1$ und Belegungen $\omega \in \Omega_n$ definiert, dann ist
$\Phi_n(\omega, \alpha \wedge \beta) = \Phi_n(\omega, \alpha) \wedge \Phi_n(\omega, \beta)$,
$\Phi_n(\omega, \alpha \vee \beta) = \Phi_n(\omega, \alpha) \vee \Phi_n(\omega, \beta)$ und
$\Phi_n(\omega, \neg\alpha) = \neg\Phi_n(\omega, \alpha)$,
womit $\Phi_n(\omega, \gamma)$ für alle Formeln γ von Grad $\leq k$ definiert ist.

Natürlich kann man diese Definition auch auf die Menge Ω der Belegungen aller Variablen $Var = \{x_1, x_2, \ldots\}$ und die Menge aller Formel $\mathcal{F}$ über Var ausweiten. Die dabei definierte Funktion $\Phi : \Omega \times \mathcal{F} \longrightarrow \{0,1\}$ stimmt auf den eingeschränkten Bereichen mit den Funktionen Φ_n überein.

Für jede feste Belegung $\omega_0 \in \Omega_n$ repräsentiert die Funktion $\Phi(\omega_0, \quad) : \mathcal{F}_{Var} \longrightarrow \{0,1\}$, die Forsetzung dieser Belegung auf alle Formeln. Dagegen stellt für jede feste Formel $\alpha_0 \in \mathcal{F}_n$ die Funktion $\Phi(\quad, \alpha_0) : \Omega_n \longrightarrow \{0,1\}$ den Wahrheitswertverlauf der Formel dar. Benutzt man vorher die Bijektion ψ^{-1} um n-Tupel in Belegungen umzuwandeln, so erhalten wir auf diese Weise die von α_0 repäsentierte Boolesche Funktion f_{α_0}, d.h. $f_{\alpha_0} = \Phi(\psi^{-1}(\quad), \alpha_0) : \{0,1\}^n \longrightarrow \{0,1\}$

Definition: Funktionen von $\{0,1\}^n$ nach $\{0,1\}$ werden *n–stellige Boolesche Funktionen* genannt. Die Menge dieser Funktionen wird mit $\boldsymbol{B_n}$ bezeichnet. Die Menge aller Booleschen Funktionen ergibt sich durch $\boldsymbol{B} = \bigcup_{i \in \mathbb{N}} \boldsymbol{B_i}$, wobei man unter $\boldsymbol{B_0}$ die Menge der Wahrheitswerte $\{0,1\}$ versteht.

Obwohl eine Aussagenverknüpfung und die entsprechende Boolesche Funktion nicht gleichzusetzen sind, werden für beide die gleichen Symbole verwendet. Der Unterschied wird aus dem Kontext klar. Sind p und q Aussagen (also Boolesche Formeln), dann bezeichnet $p \vee q$ die Disjunktion von p und q, also eine neue Formel. Sind a und b Wahrheitswerte, dann bezeichnet $a \vee b$ die Disjunktion von a und b, also einen Wahrheitswert.

Bereits mit wenigstelligen Booleschen Funktionen kann man relativ komplexe Zusammenhänge ausdrücken. Ein deutliches Indiz dafür ist die doppelt exponentiell wachsende Anzahl der n–stelligen Booleschen Funktionen. Die Bestimmung dieser Anzahl ist relativ einfach: Allgemein beschreibt die Potenz k^m die Anzahl der Funktionen, die einen bestimmten m–elementigen Definitionsbereich A in einen k–elementigen Wertebereich B abbilden, denn man hat für jedes der m Elemente aus A genau k Möglichkeiten der Zuordnung ihres Bildes, also insgesamt k^m Möglichkeiten.
Im konkreten Fall ist $A = \mathbb{B}^n$ eine 2^n–elementige Menge und $B = \mathbb{B}$ ist 2–elementig. Folglich gibt es $2^{(2^n)}$ n–stellige Boolesche Funktionen.

Definition: Zwei Formeln $\alpha, \beta \in \mathcal{F}_n$ heißen *logisch äquivalent* (auch *semantisch äquivalent, wertverlaufsgleich* oder kurz *äquivalent*), falls $\Phi_n(\omega, \alpha) = \Phi_n(\omega, \beta)$ für alle Belegungen $\omega \in \Omega_n$ gilt. Man schreibt dafür $\alpha \equiv \beta$.

Offensichtlich gilt $\alpha \equiv \beta$ genau dann, wenn α und β dieselbe Boolesche Funktion repräsentieren, d.h. wenn $f_\alpha = f_\beta$.

Wie wir bereits wissen, kann für sehr kleine Werte von n die Äquivalenz von Formeln aus $\mathcal{F}_n$ leicht durch *Wahrheitstafeln* überprüft werden. Diese Methode ist für große n nicht geeignet, denn die entsprechende Wahrheitstafel hat dann 2^n Zeilen. Eine wichtige Alternative bietet dann das folgende Ersetzungstheorem.

Definition: Die Menge der Subformeln einer Formel wird induktiv definiert. Es gilt
$Sf(p) = \{p\}$ für alle Primformeln p,
$Sf(\neg\alpha) = Sf(\alpha) \cup \{\neg\alpha\}$ für alle Formeln α
$Sf(\alpha \circ \beta) = Sf(\alpha) \cup Sf(\beta) \cup \{\alpha \circ \beta\}$ für alle Formeln α, β und alle Junktoren $\circ$.

Definition: Mit $\alpha[\,\beta/x]$ bezeichnet man den aus α enstehenden Term, wenn jedes Auftreten der Variable x in α durch den Term β ersetzt wird.

Beispiel: Für den Term $\alpha = x \wedge (y \vee x)$ ergibt sich durch Ersetzung von x durch den Term $\beta = x \wedge z$ der folgende Term:

$$\alpha[\,\beta/x] = \alpha[(x \wedge z)/x] = (x \wedge z) \wedge (y \vee (x \wedge z))$$

Eine besonders anschauliche Deutung dieser Definition erhält man durch Betrachtung der Syntaxbäume: Der Syntaxbaum des Terms $\alpha[\,\beta/x]$ entsteht aus dem Syntaxbaum von α, wenn jedes mit x markierte Blatt durch den Syntaxbaum von β ersetzt wird.

Substitutionstheorem: Seien α_1 und α_2 zwei semantisch äquivalente Terme und x eine Variable. Dann gelten für jeden Term β die folgenden Äquivalenzen:

$$\beta[\,\alpha_1/x] \equiv \beta[\,\alpha_2/x] \qquad \text{und} \qquad \alpha_1[\,\beta/x] \equiv \alpha_2[\,\beta/x].$$

Beispiele zur Anwendung dieses Theorems und eine Liste von Basisäquivalenzen findet man auf den ersten Seiten des Skripts.

6.2 Konjunktive und disjunktive Normalformen

Definition:

- Ein *Literal* ist eine Variable $x_i \in Var$ oder deren Negation.
- Ein Maxterm ist eine Disjunktion von Literalen (dazu zählt man auch ein einzelnes Literal) und ein Minterm ist ein Literal oder eine Konjunktion von Literalen (dazu zählt man auch ein einzelnes Literal).
- Eine Disjunktion von Mintermen wird *disjunktive Normalform* (kurz DNF) genannt.
- Eine Konjunktion von Maxtermen wird *konjunktive Normalform* (kurz KNF) genannt.

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.

Wir vereinbaren vorübergehend die folgende Notation: $x_i^1 = x_i$ und $x_i^0 = \neg x_i$ für alle $x_i \in Var$.

Satz: Jede Boolesche Funktion $f \in \boldsymbol{B_n}$ ist durch eine DNF α_f und durch eine KNF β_f repräsentierbar, wobei:

$$\begin{aligned} \alpha_f &= \bigvee_{(b_1,\ldots,b_n)\in f^{-1}(1)} x_1^{b_1} \wedge \ldots \wedge x_n^{b_n} \\ \beta_f &= \bigwedge_{(b_1,\ldots,b_n)\in f^{-1}(0)} x_1^{\neg b_1} \vee \ldots \vee x_n^{\neg b_n} \end{aligned}$$

Man nennt α_f die kanonische DNF und β_f die kanonische KNF von f. Im Spezialfall $f^{-1}(1) = \emptyset$ setzen wir $\alpha_f = 0$ und im Spezialfall $f^{-1}(0) = \emptyset$ setzen wir $\beta_f = 1$.

Beweis: Zuerst überzeugen wir uns davon, dass der Minterm $x_1^{b_1} \wedge \ldots \wedge x_n^{b_n}$ für die konkrete Belegung $\psi_n^{-1}(b_1, \ldots, b_n)$ 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 $\psi_n^{-1}(b_1, \ldots, b_n)$ erreicht. Die Disjunktion über alle $(b_1, \ldots, b_n) \in f^{-1}(1)$ führt dazu, dass alle Belegungen, die aus dieser Menge kommen, durch "ihren" Minterm akzeptiert werden, während Tupel aus $f^{-1}(0)$ von allen Mintermen aus α_f verworfen werden.
Der Beweis für β_f ist analog: Jede Disjunktion $x_1^{\neg b_1} \vee \ldots \vee x_n^{\neg b_n}$ wird für die konkrete Belegung $\psi_n^{-1}(b_1, \ldots, b_n)$ falsch (0) und wahr für alle anderen Belegungen. Durch die Disjunktion über alle $(b_1, \ldots, b_n) \in f^{-1}(0)$ erzeugt man eine Formel, welche die vorgegebene Funktion f repräsentiert.

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 KNF wird jeder Knoten einzeln (als 0-dimensionaler Unterwürfel) überdeckt. Man sieht, dass man diese KNF vereinfachen kann, wenn mehrere Knoten aus $f^{-1}(1)$ einen höherdimensionalen Unterwürfel bilden.

Verfolgt man diese Idee der Vereinfachung zurück zur Darstellung einer DNF als Disjunktion von Mintermen (bzw. einer KNF als Konjunktion von Maxtermen), dann kann man jeweils zwei Minterme (bzw. Maxterme) α und α' zu einem Minterm (bzw. Maxterm) β zusammenfassen, wenn α und α' identisch sind bis auf eine Variable x, die unnegiert in α und negiert in α' vorkommt. Der gemeinsame, identische Teil von α und α' bildet dann den Ersatzterm β.

Definition: Eine Menge von Junktoren, die man zur Formelbildung einsetzt, wird *logische Signatur* genannt.
Die Signatur $\{\neg, \vee, \wedge\}$ heißt *Boolesche Signatur*.
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.

Beweis: Die Vollständigkeit der Booleschen Signatur 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 Funktionen aus $\boldsymbol{B_0}$.

Mit den de Morganschen Regeln kann man die Disjuinktion (bzw. Konjunktion) durch Negation und Konjunktion (bzw. Negation und Disjunktion) eleminieren. So kann die Vollständigkeit der Signaturen $\{\neg, \wedge\}$ und $\{\neg, \vee\}$ auf die Vollständigkeit der Booleschen Signatur 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
($p|q \equiv \neg(p \wedge q)$, oft auch NAND genannt) bezeichnet, funktional vollständig.

6.3 Erfüllbarkeit, Tautologien und aussagenlogisches Folgern

Definition: Sei $\alpha \in \mathcal{F}_n$ eine Formel und $X \subseteq \mathcal{F}$ eine Menge von Formeln $\mathcal{F} = \bigcup_{i=1}^{\infty} \mathcal{F}_n$). α (bzw. X) wird erfüllbar genannt, wenn es eine Belegung $\omega \in \Omega_n$ (bzw. $\omega : Var \longrightarrow \{0,1\}$) gibt, so daß $\Phi_n(\omega, \alpha) = 1$ (bzw. $\Phi(\omega, \beta) = 1$ für alle $\beta \in X$) gilt. Man sagt dann, ω *erfüllt* α (bzw. ω *ist ein Modell für* X) und schreibt $\omega \models \alpha$ (bzw. $\omega \models X$).

Die Formel α wird *allgemeingültig, logisch gültig* oder eine *Tautologie* genannt, wenn $\omega \models \alpha$ für alle ω. Wir schreiben $\models \alpha$, falls α eine Tautologie ist, und $\not\models \alpha$, falls α keine Tautologie ist.
Eine Formel, die unerfüllbar ist, wird *Kontradiktion* genannt.

Satz: Eine Formel α ist eine Tautologie genau dann, wenn $\neg\alpha$ eine Kontradiktion ist.

Beispiele: Die Formeln $\alpha \vee \neg\alpha$ und $\alpha \leftrightarrow \alpha$ sind Tautologien (kurz: $\models \alpha \vee \neg\alpha$ und $\models \alpha \leftrightarrow \alpha$). Allgemein sind zwei Formeln α und β genau dann äquivalent, wenn $\alpha \leftrightarrow \beta$ eine Tautologie ist.
Die Formel $\alpha \wedge \neg\alpha$ ist eine Kontradiktion.

Für beliebige Formeln α, β, γ kann man die folgenden Tautologien bilden:

$\alpha \rightarrow \alpha$	(*Selbstimplikation*)
$\alpha \rightarrow (\beta \rightarrow \alpha)$	(*Prämissenbelastung*)
$(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow (\beta \rightarrow (\alpha \rightarrow \gamma))$	(*Prämissenvertauschung*)
$(\alpha \rightarrow \beta) \rightarrow ((\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma))$	(*gewöhnlicher Kettenschluß*)
$(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$	(*Fregescher Kettenschluß*)

Definition: Die Formel α wird eine *aussagenlogische Folgerung* aus der Formelmenge X genannt, falls für alle Modelle ω von X auch $\omega \models \alpha$ gilt. Man schreibt dafür

$X \models \alpha$. Ist $X = \{\alpha_1, \ldots, \alpha_n\}$, dann kann für $X \models \alpha$ auch $\alpha_1, \ldots, \alpha_n \models \alpha$ geschrieben werden.

Drei wichtige Folgerungsbeziehungen, die die Grundlage für zahlreiche Beweise in der Mathematik bilden, sollen hier besonders hervorgehoben werden,
der *Modus Ponens:*

$$\alpha, \alpha \to \beta \models \beta,$$

der *Beweis durch Fallunterscheidung:*

$$X, \alpha \models \beta \text{ und } X, \neg\alpha \models \beta \text{ impliziert } X \models \beta$$

und das *Deduktionstheorem:*

$$X, \alpha \models \beta \text{ impliziert } X \models \alpha \to \beta$$

Man kann beweisen, daß die Implikation im Deduktionstheorem sogar eine Äquivalenz ist.

Die Frage nach Algorithmen, die entscheiden, ob eine gegebene Formelmenge erfüllbar ist, gehört zu den grundlegenden Aufgabenstellungen in der künstlichen Intelligenz. Aber bereits das Erfüllbarkeitsproblem für einzelne Formeln ist *NP–vollständig.* Obwohl der Begriff der NP–Vollständigkeit hier nicht genauer erläutert werden kann, sollen einige Konsequenzen aus diesem Fakt genannt werden:

1) Ist eine gegebene Formel α erfüllbar, dann gibt es dafür einen kurzen Beweis (man rechnet für eine "geeignete" Bewertung ω nach, daß $\omega\alpha = 1$ ist).

2) Es ist kein in polynomieller Zeit laufender Algorithmus bekannt, der die Erfüllbarkeit von Formeln entscheidet (natürlich kann man $\omega\alpha$ für alle Bewertungen ω berechnen, aber es gibt exponentiell viele Berwertungen).

3) Würde man einen Polynomialzeit-Algorithmus finden, der die Erfüllbarkeit von Formeln entscheidet, so könnte man daraus Polynomialzeit-Algorithmen für alle anderen NP–vollständigen Probleme ableiten (z.B. Hamiltonkreis und TSP in Graphen und zahlreiche schwierige Optimierungsprobleme)

Auch die Probleme, zu entscheiden, ob eine gegebene Formel eine Tautologie bzw. eine Kontradiktion ist, weisen die gleichen Schwierigkeiten auf. Die Möglichkeit diese Probleme aufeinander zurückzuführen basiert auf der Tatsache, daß für jede Formel α die folgenden drei Aussagen äquivalent sind:
α ist erfüllbar
α ist keine Kontradiktion
$\neg\alpha$ ist keine Tautologie

Satz (Endlichkeitssatz, compactness theorem):

Eine Menge X von Formeln ist genau dann erfüllbar, wenn jede der endlichen Teilmengen von X erfüllbar ist.

Wir verzichten hier auf den Beweis des Satzes und merken an, dass seine typische Anwendung in Kontraposition erfolgt: Ist eine unendliche Formelmenge $\{\alpha_1, \alpha_2, \ldots\}$ nicht erfüllbar, dann gibt es ein n, so daß $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ nicht erfüllbar ist.

6.4 Resolutionskalkül

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 Erfüllbakeitsproblems man muß damit rechnen, daß das Verfahren für einige Eingaben exponentielle Laufzeit erfordert.

Sei eine endliche Formelmenge $X = \{\alpha_1, \ldots, \alpha_n\}$ gegeben. Wir setzen voraus, daß alle Formeln bereits in KNF vorliegen. Da die Formelmenge X genau dann erfüllbar (unerfüllbar) ist, wenn die Formel $\alpha = \alpha_1 \wedge \ldots \wedge \alpha_n$ erfüllbar (unerfüllbar) ist, reicht es aus, die Unerfüllbarkeit einer KNF–Formel

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

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

$$\mathcal{K}_\alpha = \{\{l_{1,1}, l_{1,2}, \ldots, l_{1,n_1}\}, \ldots\ldots, \{l_{k,1}, l_{k,2}, \ldots, l_{k,n_k}\}\}$$

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

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 α ein Formel in KNF, dargestellt als Klauselmenge $\mathcal{K}_\alpha$ und sei R ein Resolvent zweier Klauseln aus $\mathcal{K}_\alpha$. Dann sind α und die durch $\mathcal{K}_\alpha \cup \{R\}$ dargestellte Formel α' logisch äquivalent.

Beweis: Eine Richtung in dieser Äquivalenz ist einfach zu zeigen:
Wenn $\omega : Var \longrightarrow \mathbb{B}$ eine erfüllende Belegung für α' ist, dann nimmt jede Klausel aus α' unter ω den Wert 1 an. Damit ist ω aber auch erfüllende Belegung für α.

Für die Gegenrichtung ist eine etwas genauere Analyse notwendig. Wir nehmen an, dass $\omega : Var \longrightarrow \mathbb{B}$ eine erfüllende Belegung für α ist und wollen zeigen, dass dann auch alle Klauseln von α' 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. □

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

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

Beispiel: Sei $\mathcal{K} = Res^0(\mathcal{K}) = \{\{x_1, x_2, \neg x_3\}, \{\neg x_1, x_4\}, \{x_2, \neg x_4\}\}$.
Dann ist
$Res^1(\mathcal{K}) = \mathcal{K} \cup \{\{x_2, \neg x_3, x_4\}, \{\neg x_1, x_2\}\}$,
$Res^2(\mathcal{K}) = Res^1(\mathcal{K}) \cup \{\{x_2, \neg x_3\}\}$ und
$Res^*(\mathcal{K}) = Res^2(\mathcal{K}) =$
$= \{\{x_1, x_2, \neg x_3\}, \{\neg x_1, x_4\}, \{x_2, \neg x_4\}, \{x_2, \neg x_3, x_4\}, \{\neg x_1, x_2\}, \{x_2, \neg x_3\}\}$.
Man beachte, daß die durch die Klauselmenge $\mathcal{K}$ dargestellte Formel
$\alpha = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_4) \wedge (x_2 \vee \neg x_4)$
erfüllbar ist, denn jede Belegung ω mit $\omega(x_2) = \omega(x_4) = 1$ ist ein Modell für α.

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

Resolutionssatz: Eine Formel α in KNF, dargestellt durch die Klauselmenge $\mathcal{K}_\alpha$ ist genau dann unerfüllbar, wenn $\square \in Res^*(\mathcal{K}_\alpha)$.

Die Aussage, daß $\square \in Res^*(\mathcal{K}_\alpha)$ die Unerfüllbarkeit von α impliziert, wird als *Korrektheit des Resolutionskalküls* bezeichnet. Sie läßt sich leicht aus der Beobachtung

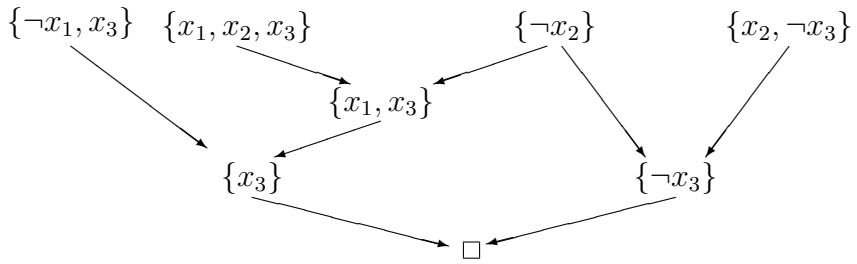
ableiten, daß $\square$ nur Resolvent von zwei Klauseln der Form $\{x_i\}$ und $\{\neg x_i\}$ sein kann. Da bereits $x_i \wedge \neg x_i$ nicht erfüllbar ist, folgt die Nichterfüllbarkeit von α aus dem Resolutionslemma (mehrfache Anwendung).

Die entgegengesetzte Implikation (aus der Unerfüllbarkeit von α folgt $\square \in Res^*(\mathcal{K}_\alpha)$) wird *Vollständigkeit des Resolutionskalküls* genannt. Sie kann durch Induktion über die Anzahl der in α 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 α entscheidet, die durch eine Klauselmenge $\mathcal{K}$ gegeben ist:

repeat
  $\mathcal{J} := \mathcal{K}$;
  $\mathcal{K} := Res(\mathcal{J})$;
until $(\square \in \mathcal{K})$ **or** $(\mathcal{J} = \mathcal{K})$;
if $\square \in \mathcal{K}$ **then** "α ist unerfüllbar" **else** "α ist erfüllbar";

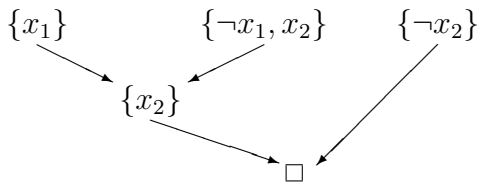
Generell sollte man beachten, daß zum Beweis der Unerfüllbarkeit einer Formel nicht unbedingt alle Resolventen gebildet werden müssen. Es reicht aus, nur die Resolventen zu bilden, die bei der *Deduktion* (Herleitung) von $\square$ eine Rolle spielen.
Beispiel: Sei $\mathcal{K} = \{\{x_1, x_2, x_3\}, \{\neg x_1, x_3\}, \{x_2, \neg x_3\}, \{\neg x_2\}\}$. Wir veranschaulichen die Deduktion der leeren Klausel durch einen sogenannten Resolutionsgraphen.



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

Eine weitere Anwendung für den Resolutionskalkül besteht im Beweis aussagenlogischer Folgerungen der Form $X \models \alpha$. Sie basiert auf der Beobachtung, daß α genau dann aussagenlogische Folgerung aus einer Formelmenge $X = \{\alpha_1, \ldots \alpha_n\}$ ist , wenn die Formel $\alpha_1 \wedge \ldots \wedge \alpha_n \wedge \neg\alpha$ nicht erfüllbar ist. Vor der eigentlichen Anwendung des Resolutionskalküls muss also wieder dafür sorgen, dass die Formeln $\alpha_1, \ldots \alpha_n$ und die Negation $\neg\alpha$ in KNF vorliegen.

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



Definition: Variable bezeichnet man auch als *positive Literale*, ihre Negationen nennt man *negative Literale*. Eine KNF–Formel α wird *Hornformel* genannt, falls jedes Disjunktionsglied höchstens ein positives Literal enthält.

Beispiel: Die folgende Formel ist eine Hornformel:
$\alpha = (x_1 \vee \neg x_3)(\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_4) \wedge x_2 \wedge \neg x_4$

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 $Res(\mathcal{K})$ und $Res^*(\mathcal{K})$ definieren 1–$Res(\mathcal{K})$ und 1–$Res^*(\mathcal{K})$ bezüglich der Einheitsresolution.

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

Auf einen Beweis wird an dieser Stelle verzichtet.

Da die Einheitsresolution die Länge der Klauseln ständig verkürzt, ist diese Methode (für Hornklauseln!) besonders effizient. Hornklauseln spielen eine sehr wichtige Rolle in der Logik–Programmierung.

6.5 Algebraische Strukturen und Prädikatenlogik

Der Aufbau der Prädikatenlogik hat starke Bezüge zur Beschreibung von mathematischen, insbesondere *algebraischen Strukturen*. Darunter versteht man eine Trägermenge (Individuenbereich) in der gewisse Relationen, Operationen und Konstanten ausgezeichnet sind, die bestimmte Eigenschaften aufweisen.

Beispiele:

a) Eine Menge G mit Operationen $* : G \times G \longrightarrow G, ^- : G \longrightarrow G$ und einer Konstanten $e \in G$ ist eine Gruppe, wenn die folgenden Bedingungen erfüllt sind:

1) für alle $x, y, z \in G$ gilt: $x * (y * z) = (x * y) * z$
2) für alle $x \in G$ gilt: $x * e = e * x = x$
3) für alle $x \in G$ gilt: $x * \bar{x} = \bar{x} * x = e$

Bekannte Beispiele sind $(\mathbb{Z}, +, -, 0)$ und $(\mathbb{R}^+, \cdot, ^{-1}, 1)$.

b) Eine Menge L mit einer Relation $\leq \subseteq \ L \times L$ und zwei Funktionen $sup, inf : L \times L \longrightarrow L$ ist ein Verband, falls $\leq$ eine Halbordnungsrelation ist und für je zwei Elemente $a, b \in L$ eine (eindeutig bestimmte) kleinste obere Schranke $sup(a, b)$ (das Supremum von a und b) und eine (eindeutig bestimmte) größte untere Schranke $inf(a, b)$ (das Infimum von a und b) existieren, d.h.

1) für alle $x \in L$ gilt: $x \leq x$
2) für alle $x, y \in L$ gilt: $x \leq y \ \wedge \ y \leq x \ \rightarrow x = y$
3) für alle $x, y, z \in L$ gilt: $x \leq y \ \wedge \ y \leq z \ \rightarrow \ x \leq z$
4) für alle $x, y, z \in L$ gilt: $x \leq sup(x, y) \ \wedge \ y \leq sup(x, y) \ \wedge$ $(x \leq z \ \wedge \ y \leq z \ \rightarrow \ sup(x, y) \leq z)$
5) für alle $x, y, z \in L$ gilt: $inf(x, y) \leq x \ \wedge \ inf(x, y) \leq y \ \wedge$ $(z \leq x \ \wedge \ z \leq y \ \rightarrow \ z \leq inf(x, y))$

Einfache Beispiele sind der Mengenverband $(\mathcal{P}(M), \subseteq, \cup, \cap)$ und der Teilbarkeitsverband $(\mathbb{Z}^+, |, kgV, ggT)$

Achtung: Man kann Verbände auch nur durch Eigenschaften der Operationen *sup* und *inf* definieren.

Wie diese Beispiele zeigen, werden quantifizierte Prädikate verwendet, um diese Strukturen zu charakterisieren. Der allgemeine Zugang zur Beschreibung von mathematischen Strukturen führt zu *Sprachen 1. Stufe*, auch *elementare Sprachen* genannt, deren Syntax im Folgenden beschrieben wird. Das Alphabet einer solchen Sprache besteht aus:

- einer Menge von Individuenvariablen $Var = \{x_1, x_2, \ldots\}$ wobei x, y, z beliebige Elemente aus dieser Menge bezeichnen sollen;
- den logischen Symbolen $\neg, \wedge$ sowie dem *Allquantor* $\forall$ (gelesen: "für alle") und dem Zeichen $\equiv$ für die Identitätsrelation in der zugrundeliegenden Struktur, weitere logische Symbole wie $\vee, \rightarrow$ und der *Existenzquantor* $\exists$ (gelesen: "es existiert ein") werden später durch Definition eingeführt;
- der *nichlogischen Signatur* L, die sich aus einer Menge von Konstantensymbolen, einer Menge von Funktionssymbolen (sie werden mit f, g bezeichnet und

beinhalten die Stelligkeit der symbolisierten Funktion) sowie einer Menge von Relationssymbolen (sie werden mit r bezeichnet und beinhalten die Stelligkeit der symbolisierten Relation)

Die Syntax der Sprache 1. Stufe über der Signatur L wird durch die (induktive) Definition von *Termen* und *Formeln* bestimmt.

Definition von Termen in L:

1) Alle Variablen und Konstanten sind Terme, sogenannte *Primterme*.

2) Ist $f \in L$ ein n–stelliges Funktionssymbol und sind $t_1, \ldots, t_n$ Terme, so ist auch $f(t_1, \ldots, t_n)$ ein Term.

Analog wie für Formeln der Aussagenlogik gilt auch hier ein Satz von der eindeutigen Termreduktion (Ist $f(t_1, \ldots, t_n) = g(s_1, \ldots, s_m)$ dann ist $f = g$, $n = m$ und $t_1 = s_1, \ldots, t_n = s_n$) und das Beweisprinzip durch Terminduktion (haben alle Primterme eine Eigenschaft und folgt, wenn Terme $t_1, \ldots, t_n$ diese Eigenschaft haben, daß für alle n–stelligen Funktionssymbole f auch $f(t_1, \ldots, t_n)$ diese Eigenschaft hat, dann haben alle Terme diese Eigenschaft). Auch die Mengen $St(t)$ (Subterme eines Terms t) und $var(t)$ (in t auftretende Variablen) werden induktiv definiert:

Ist c eine Konstante, dann ist $St(c) = \{c\}$ und $var(c) = \emptyset$.

Ist x eine Variable, dann ist $St(x) = \{x\}$ und $var(x) = \{x\}$.

Ist $t = f(t_1, \ldots, t_n)$, dann ist $St(t) = \{t\} \cup \bigcup_{i=1}^{n} St(t_i)$ und

$var(t) = \bigcup_{i=1}^{n} var(t_i)$.

Definition von Formeln in L:

1) Sind s, t Terme, so ist $s \equiv t$ eine Formel.

2) Sind $t_1, \ldots, t_n$ Terme und ist $r \in L$ ein n–stelliges Relationssymbol, so ist $r(t_1, \ldots, t_n)$ eine Formel.

3) Sind α, β Formeln und ist $x \in Var$, so sind auch $(\alpha \wedge \beta), \neg\alpha$ und $\forall x\ \alpha$ Formeln.

Bemerkungen:

a) Eine n–stellige Relation R in einer Menge M ist definiert als eine Teilmenge des n–fachen Kartesischen Produkts $M^n = M \times \ldots \times M$. Das Symbol r für eine solche Relation muß als charakteristische Funktion von R interpretiert werden: $r(t_1, \ldots, t_n) = 1$ (wahr) genau dann, wenn $(t_1, \ldots, t_n) \in R$.

b) die durch 1) und 2) gewonnenen Formeln werden als Primformeln bezeichnet.

c) Weitere logische Symbole können als Abkürzungen eingeführt werden:

$\alpha \vee \beta := \neg(\neg\alpha \wedge \neg\beta)$, $\alpha \rightarrow \beta := \neg(\alpha \wedge \neg\beta), \ldots$ und $\exists x\alpha := \neg\forall x\neg\alpha$

d) Wie die Formeldefinition deutlich macht, ist Quantifizierung nur für Individuenvariable vorgesehen. Erweitert man dieses Konzept um Variablen für Untermengen des Individuenbereichs und deren Quantifizierung, so erhält man *Sprachen 2. Stufe*, die jedoch andere semantische Eigenschaften haben.

Beispiele: Die drei Eigenschaften zur Charakterisierung einer Gruppe können wie folgt als Formeln geschrieben werden. Die Signatur L beinhaltet ein Konstantensymbol e, ein zweistelliges Funktionssymbol f und ein einstelliges Funktionssymbol g und keine Relationssymbole. Der Ausdruck $x * (y * z)$ wird durch den Term $f(x, f(y, z))$ repräsentiert.

1) $\forall x \forall y \forall z \quad f(x, f(y, z)) \equiv f(f(x, y), z)$
2) $\forall x \quad (f(e, x) \equiv x \ \wedge \ f(x, e) \equiv x)$
3) $\forall x \quad (f(x, g(x)) \equiv e \ \wedge \ f(g(x), x) \equiv e)$

Zur Definition der Semantik elementarer Sprachen benötig man die Begriffe L–Stuktur und L–Modell für eine gegebene nichtlogische Signatur L. Dazu muss eine konkrete Trägermenge festgelegt werden, auf der alle Funktions-, Relations- und Konstantensymbole durch konkrete Funktionen, Relationen und Konstanten realisiert werden. Gleichzeitig ist die Trägermenge der Individuenbereich für die Bewertung aller Variablen und für alle Quantoren.

Definition: Eine L–Struktur $\mathcal{A} = (A, L^{\mathcal{A}})$ besteht aus einer nichtleeren Trägermenge A und einer Menge $L^{\mathcal{A}}$, die für jedes Konstantensymbol $c \in L$ eine Konstante $c^{\mathcal{A}} \in A$, für jedes n–stellige Funktionssymbol $f \in L$ eine Funktion $f^{\mathcal{A}} : A^n \longrightarrow A$ und für jedes n–stellige Relationssymbol $r \in L$ die charakteristische Funktion $r^{\mathcal{A}}$ einer Relation $R^{\mathcal{A}} \subseteq A^n$ enthält.

Definition: Eine L–Struktur $\mathcal{A} = (A, L^{\mathcal{A}})$ mit einer zusätzlichen Abbildung $\omega : Var \longrightarrow A$ wird ein L–Modell genannt.

Man schreibt entsprechend $c^{\mathcal{M}} = c^{\mathcal{A}}$, $f^{\mathcal{M}} = f^{\mathcal{A}}$, $r^{\mathcal{M}} = r^{\mathcal{A}}$ und $x^{\mathcal{M}} = \omega(x)$.

Achtung: In einigen Büchern (z.B. Schöning) sind die Funktionen $\omega : Var \longrightarrow A$ bereits Bestandteil einer Struktur, d.h. eine Struktur in jenem Sinne ist ein L–Modell in Sinne unserer Definition.

Durch induktive Definition weist ein Modell $\mathcal{M}$ jedem Term t einen Wert $t^{\mathcal{M}}$ aus der Trägermenge A zu: Für Primterme sind $c^{\mathcal{M}} = c^{\mathcal{A}}$ und $x^{\mathcal{M}} = \omega(x)$ bereits definiert, für $t = f(t_1, \ldots, t_n)$ setzen wir $t^{\mathcal{M}} := f^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}})$.

Dadurch kann bei einem gegebenen Modell jeder Formel ein Wahrheitswert zugeordnet werden. Komplikationen bereiten Formeln der Form $\forall x \alpha$ denen der Wahrheitswert 1 genau dann zugeordnet wird, wenn die Formel α für jede Belegung von der Variablen x (mit einem Wert aus A) den Wahrheitswert 1 bekommt. Diese intuitiv einfache Idee erfordert aber wieder ein induktives Herangehen (für verschachtelte Quantoren), auf das wir hier aus Platzgründen verzichten. Wir sagen, dass ein Modell $\mathcal{M}$ die Formel α erfüllt, wenn α in $\mathcal{M}$ den Wert 1 bekommt und schreiben dann $\mathcal{M} \models \alpha$.

Ähnlich wie in der Aussagenlogik führen wir die folgenden Begriffe und Notationen ein:

- Zwei Formeln α und β heißen *logisch* oder *semantisch äquivalent* (Schreibweise $\alpha \equiv \beta$), falls für jedes Modell $\mathcal{M}$ gilt: $\mathcal{M} \models \alpha$ genau dann, wenn $\mathcal{M} \models \beta$.
- $\mathcal{M}$ ist ein *Modell für eine Formelmenge* X (Schreibweise: $\mathcal{M} \models X$), wenn $\mathcal{M}$ jede Formel aus X erfüllt. X heißt *erfüllbar*, wenn es ein Modell besitzt.
- Eine Formel α heißt *allgemeingültig* oder *Tautologie* (Schreibweise: $\models \alpha$), wenn $\mathcal{M} \models \alpha$ für jedes Modell $\mathcal{M}$.
- Eine Formel α *gilt in* $\mathcal{A}$ (Schreibweise: $\mathcal{A} \models \alpha$), falls $(\mathcal{A}, \omega) \models \alpha$ für alle $\omega : Var \longrightarrow A$ und entsprechend $\mathcal{A} \models X$ falls $\mathcal{A} \models \alpha$ für alle $\alpha \in X$.

- Man sagt, daß α aus X folgt (Schreibweise: $X \models \alpha$), falls jedes Modell für X auch α erfüllt.

Beispiele:
1) Wir betrachten die Formel $\alpha = \forall x \exists y \; \neg x \equiv y$. Wie man leicht sieht, ist α erfüllbar, denn jedes Modell mit einer zwei– oder mehrelementigen Trägermenge erfüllt α. Aber α ist keine Tautologie, weil Modelle mit einelementiger Trägermenge α nicht erfüllen. Ist $\mathcal{A} = (A, L^{\mathcal{A}})$ eine L–Struktur, so gilt $\mathcal{A} \models \alpha$, falls $|A| \geq 2$, und $\mathcal{A} \not\models \alpha$, falls $|A| = 1$.

2) Die Formel $\beta = \neg \forall x \forall y \; \neg x \equiv y$ ist eine Tautologie.

3) Sei $\mathcal{A}$ eine Struktur mit einer zweielementigen Trägermenge, dann gilt weder $\mathcal{A} \models x \equiv y$ noch $\mathcal{A} \models \neg x \equiv y$. Ist dagegen α eine beliebige Formel $\mathcal{M}$ ein beliebiges (passendes) Modell, dann gilt entweder $\mathcal{M} \models \alpha$ oder $\mathcal{M} \models \neg\alpha$.

Satz: Für beliebige Formeln α und β gelten die folgenden Äquivalenzen:

$$\begin{array}{rcl} \neg\forall x \; \alpha & \equiv & \exists x \neg\alpha \\ \forall x \; \alpha \wedge \forall x \; \beta & \equiv & \forall x \; (\alpha \wedge \beta) \\ \forall x \forall y \; \alpha & \equiv & \forall y \forall x \; \alpha \end{array} \qquad \begin{array}{rcl} \neg\exists x \; \alpha & \equiv & \forall x \neg\alpha \\ \exists x \; \alpha \vee \exists x \; \beta & \equiv & \exists x \; (\alpha \vee \beta) \\ \exists x \exists y \; \alpha & \equiv & \exists y \exists x \; \alpha \end{array}$$

Wie das Beispiel der Gruppentheorie deutlich macht, möchte man Stukturen betrachten, in denen die Gültigkeit einer bestimmten Formelmenge X vorausgesetzt wird. Ziel ist es dann, Aussagen zu erhalten, die aus X folgen. Dieser Ansatz führt zu der folgenden Begriffsbildung.
Definition: Eine *Theorie* ist eine Menge von Formeln $\mathcal{T}$, die bezüglich Folgerung abgeschlossen ist, d.h. $\mathcal{T} \models \alpha$ impliziert $\alpha \in \mathcal{T}$. Eine Formelmenge X wird *Axiomensystem* einer Theorie $\mathcal{T}$ genannt falls $\mathcal{T} = \{\alpha \,|\, X \models \alpha\}$.

Es sei angemerkt, daß auch die Menge aller (L–)Formeln eine Theorie ist. Man nennt dies eine entartete Theorie, da sie widersprüchliche Aussagen enthält und es kein Modell für sie gibt.

Sei $\mathcal{T}$ eine Theorie mit einem endlichen Axiomensystem $X = \{\alpha_1, \ldots \alpha_n\}$ und α eine Formel. α ist genau dann in $\mathcal{T}$, wenn die Formel
$\beta = \neg\alpha_1 \vee \ldots \vee \neg\alpha_n \vee \alpha$ eine Tautologie ist. Äquivalent formuliert, ist α genau dann in $\mathcal{T}$, wenn
$\neg\beta = \alpha_1 \wedge \ldots \wedge \alpha_n \wedge \neg\alpha$ nicht erfüllbar ist.

Es ergibt sich die Frage, ob es (ähnlich wie der Resolutionskalkül in der Aussagenlogik) ein Entscheidungsverfahren für die Erfüllbarkeit von Formeln elementarer Sprachen gibt. Die Antwort ist negativ: Das Erfüllbarkeitsproblem für Formeln elementarer Sprachen ist **nicht entscheidbar**.

Es konnte aber durch Anpassung und Verfeinerung der Resolutionsmethode ein sogenanntes Semientscheidungsverfahren entwickelt werden, das zumindest für unerfüllbare Formeln in endlicher Zeit mit der richtigen Antwort stoppt (aber für erfüllbare Formeln nie stoppt).