Syntax: Aussagenlogische Formeln

Das **Alphabet** der Aussagenlogik AL umfasst

(1) eine abzählbare Menge AV von **Aussagesymbolen** (**Aussagevariablen**), hier $A_1, A_2, \ldots$. Informell schreiben wir aber auch oft *A, B, C, ...* oder geben sogar selbsterklärende Namen wie *Es_regnet.*
Manchmal sprechen wir über „beliebige Aussagesymbole"; dann benutzen wir dafür stellvertretende Variablennamen *P, Q, ...*

(2) **Junktoren**

	Verwendung	Fachausdruck
$\neg$	*nicht* (einstellig!)	**Negation**
$\wedge$	*... und ...*	**Konjunktion**
$\vee$	*... oder ...* (incl. *oder beides*)	**Disjunktion**
$\rightarrow$	*wenn ... dann*	**Implikation**
$\leftrightarrow$	*... genau dann, wenn ...*	**Äquivalenz, Biimplikation**

(3) **Klammern** (,) bestimmen *Ausführungsreihenfolge.*

Die **Formeln** der AL sind induktiv definiert:

- Alle Aussagevariablen sind Formeln.
- Für alle Formeln φ, ψ sind $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, sowie $(\varphi \rightarrow \psi)$ Formeln (Klammern und Junktoren buchstäblich!).

Manche Klammern werden gerne **weggelassen**:

- zunächst die äußersten aus Bequemlichkeit, z.B. $A \wedge B$,
- weitere „wegen Assoziativität" (wegen → Semantik, später erklärt), z.B. $A \wedge B \wedge C$,
- weitere, wenn man ungeklammerte Junktoren nach bestimmten Prioritäten „abarbeitet" z.B. $A \wedge B \rightarrow C$ mit „$\wedge$ vor $\rightarrow$". Das tun wir **nicht**.

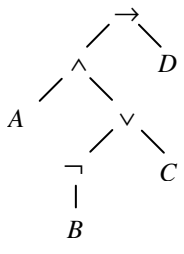
Die **Sprache** *ALForm* der AL-Formeln wird also durch folgende **Grammatik** erzeugt:

$N = \{S\}$ Nichtterminalsymbole
$T = \{ A_1, A_2, ..., \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,) \}$ Terminalsymbole
$S = S$ Startsymbol
$R = \{ S \rightarrow A_1 \mid A_2 \mid ...,$ Regeln
$S \rightarrow \neg S \mid (S \wedge S) \mid (S \vee S) \mid (S \rightarrow S) \mid (S \leftrightarrow S) \}$

Problem: Das Terminalalphabet und daher die Menge der Regeln sind unendlich.
Reparaturidee: Ersetze in der Sprache jedes A_i durch eine Kette *A...A* von i *A*'s. Wie lauten dann T und R?

Alternative Schreibweisen: Beispiel

Unsere (**Infix**):	$(A \wedge (\neg B \vee C)) \rightarrow D$
polnisch:	$\rightarrow \wedge A \vee \neg BCD$
umgekehrt polnisch:	$AB\neg C \vee \wedge D \rightarrow$
geordneter Baum	



Funktionen auf den Formeln

Die **Variablenmenge** einer Formel ist rekursiv definiert:
$Vars : Form \rightarrow \mathbf{P}\{A_1, A_2, ...\}$
$Vars(P) := \{P\}$,
$Vars(\neg\varphi) := Vars(\varphi)$
$Vars(\varphi \wedge \psi) := Vars(\varphi \vee \psi) :=$
$Vars(\varphi \rightarrow \psi) := Vars(\varphi \leftrightarrow \psi) := Vars(\varphi) \cup Vars(\psi)$
Ähnlich werden definiert:

- die Menge $Sub(\varphi)$ d. **Teilformeln** einer Formel φ,
- der **Grad** $Grad(\varphi)$ einer Formel φ, d.h. die Anzahl ihrer Junktorenanwendungen,
- die **Tiefe** $Tiefe(\varphi)$ einer Formel φ, d.h. ihre größte Schachtelungstiefe (mit $Tiefe(P) = 0$).

Semantik der Aussagenlogik

Die wichtigste Funktion auf AL-Formeln ist aber ihr **Wahrheitswert**, wahr oder falsch, in Abhängigkeit von den festgelegten Wahrheitswerten der beteiligten Aussagevariablen.
Die Menge möglicher **Wahrheitswerte** ist {W,F} (für **wahr**, **falsch**). Eine **Belegung** oder **Interpretation** ist eine Abbildung bel: $\text{Def}_{bel} \rightarrow \{W, F\}$ von einer Menge von Aussagevariablen $\text{Def}_{bel} \subseteq \{A_1, A_2, ...\}$ in {W,F}. bel ist **ausreichend** (oder **passend**) für eine Formel φ, wenn es alle Aussagevariablen der Formel belegt, d.h. $Vars(\varphi) \subseteq \text{Def}_{bel}$.

Jetzt ordnen wir rekursiv jeder Formel φ – in Abhängigkeit von einer gegebenen ausreichenden Belegung bel – ihren **Wahrheitswert** $wert_{bel}(\varphi)$ zu:
$wert_{bel}(A_i) := bel(A_i)$
$wert_{bel}(\neg\varphi) := sem_\neg(wert_{bel}(\varphi))$
und für unsere 4 zweistelligen logischen Junktoren j:
$wert_{bel}(\varphi\, j\, \psi) := sem_j(wert_{bel}(\varphi), wert_{bel}(\psi))$

Die **Junktorensematiken** $sem_j : \{W, F\}^2 \rightarrow \{W, F\}$
bzw. $sem_\neg : \{W, F\} \rightarrow \{W, F\}$ sind in der Tabelle (1) zusammengefasst. Dabei steht im Spaltenkopf oben die „mathematisch strenge" Inschrift, darunter die übliche, intuitiv verständlichere.
Man mache sich an etlichen Beispielen klar, wie die rekursive **Evaluation** einer Formel in deren Baum von den Blättern zur Wurzel aufsteigt.

φ	ψ	$sem_\neg(\varphi)$ $\neg\varphi$	$sem_\wedge(\varphi,\psi)$ $(\varphi \wedge \psi)$	$sem_\vee(\varphi,\psi)$ $(\varphi \vee \psi)$	$sem_\rightarrow(\varphi,\psi)$ $(\varphi \rightarrow \psi)$	$sem_\leftrightarrow(\varphi,\psi)$ $(\varphi \leftrightarrow \psi)$
W	W	F	W	W	W	W
W	F		F	W	F	F
F	W	W	F	W	W	F
F	F		F	F	W	W

Tab.1: Junktorensemantiken

Sprache und formale Logik

Wie hängen aussagenlogische Formeln und sprachliche Aussagen zusammen?
Eine (sprachliche) **Aussage** ist eine sprachliche Äußerung, der *man* in der *jeweiligen Situation* den Wert wahr oder falsch zuordnen *kann* (Problempunkte unterstrichen). Eine **atomare Aussage** ist eine Aussage ohne Negation und ohne Junktoren wie *nicht, und, oder, wenn dann, genau dann wenn*. Eine Aussage ist also aus atomaren Aussagen, Negationen und Junktoren **zusammengesetzt**. Sie erhält ihren Wahrheitswert in zwei Stufen:

(1) Eine **Situation** ordnet jeder atomaren Aussage den Wert wahr oder falsch zu, oft im Sinne von „hier und jetzt" (z.B. *Es regnet*.) bzw. unter Faktoren wie:

- vorherige sprachliche Situationsbeschreibung (Zeit, Ort, Subjekte, Objekte)
  („Gestern abend begegnete mir Frau Müller auf der Treppe.")
  – „Ihr Hund hat mich ganz schön angeknurrt."
- nichtsprachliche Zeigeaktionen
  (Kunde deutet auf Apfelkiste.)
  – „Die sehen nicht ganz frisch aus."
- zeitliche Nähe und Auffälligkeit (implizit)
  (Wumpa-wumpa!)
  – „Der hat aber seine Anlage im Auto voll aufgedreht."

(2) Der Wahrheitswert der Gesamtaussage ergibt sich rekursiv unter Berücksichtigung der sprachlichen Junktoren.
Sprachliche Aussagen **interpretieren** AL-Formeln, AL-Formeln **modellieren** sprachliche Aussagen.

Wahrheitstafelalgorithmus,

Werteverlaufsberechnung = Berechnung der Wahrheitswerte unter allen ausreichenden Belegungen

Gegeben sei eine Formel φ mit n Aussagevariablen.

Beispiel: $(A \to B) \vee \neg B$, n=2

(1) Lege eine Tabelle mit 2^n Zeilen und je einer Spalte für jede Teilformel von φ an und überschreibe die Spalten oben entsprechend so, dass alle deren Teilformeln links davon stehen.
(2) Trage in die Spalten für die Aussagevariablen zeilenweise alle 2^n möglichen W/F-Kombinationen ein.

[Übrigens: (1) und (2) sind nicht-deterministisch.]

Beispiel

A	B	$A \to B$	$\neg B$	$(A \to B) \vee \neg B$
W	W			
W	F			
F	W			
F	F			

(3) Trage in allen Spalten von links nach rechts die Wahrheitswerte der komplexeren Formeln gemäß den Wahrheitswerten ihrer (maximalen echten) Teilformeln und der Junktorensemantik ihres „obersten Junktors" ein.

Beispiel

A	B	$A \to B$	$\neg B$	$(A \to B) \vee \neg B$
W	W	W	F	**W**
W	F	F	W	**W**
F	W	W	F	**W**
F	F	W	W	**W**

Satz: **Korrektheit des Wahrheitstafelalgorithmus**
Der Wahrheitstafelalgorithmus **terminiert**. Anschließend enthält die φ-Spalte den **Wahrheitswerteverlauf** von φ, nämlich in jeder Zeile den Wahrheitswert von φ für die in der betreffenden Zeile stehende Belegung ihrer Aussagevariablen.

Alternative Vorgehensweise „in situ":

$(A$	$\to$	$B)$	$\vee$	$\neg$	B
W	W	W	**W**	F	W
W	F	F	**W**	W	F
F	W	W	**W**	F	W
F	W	F	**W**	W	F

… oder Abkürzungen für Teilformeln einführen und verwenden.

Andere Junktoren

Wir kennen jetzt 4 zweistellige Junktoren(semantiken), d.h. W-F-Belegungen einer 4-zeiligen Spalte. Es gibt aber 16, vgl. → Tabelle 2.

Beispiele: $\uparrow$ Sheffer-Verknüpfung, NAND
$\downarrow$ Peirce-Verknüpfung, NOR

Wo stehen z.B. …

- ψ nur dann wenn nicht φ
- entweder … oder (XOR)
- weder … noch
- wahr, egal ob φ oder ψ ?

Tab. 2: Alle zweistelligen Junktorensemantiken

φ	ψ	$\top$	$\vee$	$\leftarrow$	φ	$\to$	ψ	$\leftrightarrow$	$\wedge$	$\uparrow$	$\not\leftrightarrow$	$\neg\psi$	$\not\leftarrow$	$\neg\varphi$	$\not\to$	$\downarrow$	$\bot$
W	W	W	W	W	W	W	W	W	W	F	F	F	F	F	F	F	F
W	F	W	W	W	W	F	F	F	F	W	W	W	W	F	F	F	F
F	W	W	W	F	F	W	W	F	F	W	W	F	F	W	W	F	F
F	F	W	F	W	F	W	F	W	F	W	F	W	F	W	F	W	F

Es gibt $2^{(2^n)}$ Spaltenbelegungen für n-stellige Junktoren.
Wichtigster 3-stelliger Junktor (von 256) ist der **Entscheidungsoperator** $\varphi \to \psi_1 / \psi_2$, auch

"if φ then ψ_1 else ψ_2" geschrieben.

Er entspricht in der Wahrheitstafel $(\varphi \wedge \psi_1) \vee (\neg\varphi \wedge \psi_2)$ bzw. $(\varphi \to \psi_1) \wedge (\neg\varphi \to \psi_2)$.

Das **Programm-Statement**

A: IF <boolescher_term>
THEN <anweisung1>
ELSE <anweisung2>

bewirkt **logisch** ausgedrückt:

if boolescher_term(Zustand vor A)
then Zustand nach A = Zustand nach anweisung1
else Zustand nach A = Zustand nach anweisung2.

Wichtige semantische Begriffe

Eine für eine Formel φ ausreichende **Belegung** *bel* ist **Modell** von φ :$\Leftrightarrow$ $wert_{bel}(\varphi) = \mathrm{W}$. **Beispiel**sweise ist $bel(A) := \mathrm{F}$, $bel(B) := \mathrm{W}$ ein Modell von $A \to B$.
Weitere Rede- und Schreibweisen für „*bel* ist **Modell** von φ": *bel* **erfüllt** φ, φ **gilt unter** *bel* bzw. $bel \models \varphi$.
bel ist **Gegenbeispiel** zu φ :$\Leftrightarrow$ $wert_{bel}(\varphi) = F$. Man sagt oder schreibt auch: *bel* **widerlegt** φ bzw. $bel \not\models \varphi$.
Der **Wahrheitstafelalgorithmus** liefert **alle Modelle und Gegenbeispiele** einer Formel: Die **Zeilen** der Wahrheitstafel, in denen unter der Formel **W** (bzw. **F**) steht, enthalten unter den Aussagevariablen die **Modelle** (bzw. **Gegenbeispiele**).
Eine Formel φ nennt man **erfüllbar**, wenn sie ein Modell besitzt, und **unerfüllbar** (oder **widersprüchlich**) wenn nicht. $A \to B$ ist erfüllbar, $A \wedge \neg A$ ist unerfüllbar.
Eine Formel φ nennt man **kontingent**, wenn sie Modell(e) und Gegenbeispiel(e) besitzt. $A \wedge B$ ist kontingent.

Eine Formel nennt man **allgemeingültig** (oder eine **Tautologie**) und schreibt dann $\models \varphi$, wenn sie unter jeder für sie ausreichenden Belegung wahr ist, und **widerlegbar** (oder **falsifizierbar,** $\not\models \varphi$) wenn nicht.
$(A \to B) \vee (B \to A)$ ist allgemeingültig und erfüllbar;
$(A \to B) \wedge (B \to A)$ ist widerlegbar und erfüllbar und und daher kontingent.

φ ist:		In der φ-Spalte der W.-tafel kommt vor:
erfüllbar unerfüllbar kontingent allgemeingültig widerlegbar	$\Leftrightarrow$	W nur F W und F nur W F

Wichtige Tautologien (1) Implikationen

$(P \wedge \neg P) \to Q$	***Ex-Falso-Prinzip***
$((P \to Q) \to P) \to P$	*Peirce's Gesetz*
$(P \to Q) \leftrightarrow (\neg Q \to \neg P)$	***Kontraposition***
$(P \to Q) \to ((Q \to R) \to (P \to R))$	***Kettenschluss***
$((P \wedge Q) \to R) \leftrightarrow (P \to (Q \to R))$	*Prämissenverbindung*
$(P \to (Q \to R)) \leftrightarrow (Q \to (P \to R))$	***Präm.vertauschung***
$((P \to (Q \to R)) \leftrightarrow ((P \to Q) \to (P \to R))$	*Selbstdistributivität*

Beweisbeispiel für Kontraposition mit normaler (→ Tabelle 3 s.u.) und modifizierter Wahrheitstafel:

Tab. 3: Wahrheitstafel

P	Q	$P \to Q$	$\neg Q$	$\neg P$	$\neg Q \to \neg P$	$(P \to Q) \leftrightarrow (\neg Q \to \neg P)$
W	W	W	F	F	W	**W**
W	F	F	W	F	F	**W**
F	W	W	F	W	W	**W**
F	F	W	W	W	W	**W**

Tab 4: Modifizierte Wahrheitstafel

$(P$	$\to$	$Q)$	$\leftrightarrow$	$(\neg$	Q	$\to$	$\neg$	$P)$
W	W	W	**W**	F	W	W	F	W
W	F	F	**W**	W	F	F	F	W
F	W	W	**W**	F	W	W	W	F
F	W	F	**W**	W	F	W	W	F

Wichtige Tautologien (2): Äquivalenz/ Biimplikation
Regeln für Konjunktion und Disjunktion

$(P \wedge P) \leftrightarrow P$ $(P \vee P) \leftrightarrow P$	***Idempotenz***
$(P \wedge Q) \leftrightarrow (Q \wedge P)$ $(P \vee Q) \leftrightarrow (Q \vee P)$	***Kommutativität***
$((P \wedge Q) \wedge R) \leftrightarrow (P \wedge (Q \wedge R))$ $((P \vee Q) \vee R) \leftrightarrow (P \vee (Q \vee R))$	***Assoziativität***
$((P \wedge (Q \vee R)) \leftrightarrow ((P \wedge Q) \vee (P \wedge R))$ $((P \vee (Q \wedge R)) \leftrightarrow ((P \vee Q) \wedge (P \vee R))$	***Distributivität***

Negationsregeln

$\neg\neg P \leftrightarrow P$	***doppelte Negation***
$\neg(P \wedge Q) \leftrightarrow (\neg P \vee \neg Q)$ $\neg(P \vee Q) \leftrightarrow (\neg P \wedge \neg Q)$	***Anti-Distributivität***

Tautologie- und Kontradiktions-(Absorptions-)Regeln

$(P \wedge (Q \vee \neg Q)) \leftrightarrow P$ $(P \vee (Q \vee \neg Q)) \leftrightarrow (Q \vee \neg Q)$	***Tautologie-Absorption***
$(P \vee (Q \wedge \neg Q)) \leftrightarrow P$ $(P \wedge (Q \wedge \neg Q)) \leftrightarrow (Q \wedge \neg Q)$	***Kontradiktions-Absorption***

Implikations- und Äquivalenzauflösungsregeln

$(P \leftrightarrow Q) \leftrightarrow ((P \to Q) \wedge (Q \to P))$	***Äquivalenz-auflösung***
$(P \to Q) \leftrightarrow (\neg P \vee Q)$	***Implikations-auflösung***

Semantische Begriffe für Formelmengen
Eine Belegung *bel* ist **ausreichend** (oder **passend**) für eine Menge Φ von Formeln, wenn sie alle Aussagevariablen aller Formeln $\varphi \in \Phi$ belegt, d.h.

für alle $\varphi \in \Phi$: $\mathrm{Vars}(\varphi) \subseteq \mathrm{Def}_{bel}$.

Man nennt eine für eine Menge Φ von Formeln ausreichende Belegung, bei der jede Formel in Φ wahr ist, ein **Modell** von Φ. Sei beispielsweise Φ die Menge

$\{(A \to B_i) \mid i = 1,2,...\}$. Dann ist jede Belegung *bel* von $A, B_1, B_2, ...$ mit $bel(A) = F$ ein Modell von Φ.

Eine Menge von Formeln nennt man **erfüllbar**, wenn sie ein Modell besitzt, und **unerfüllbar** (oder **widersprüchlich**) wenn nicht. $\{(A \to B), \neg(C \vee B \vee \neg A)\}$ ist zum Beispiel unerfüllbar.

Diese Begriffe dienen u.a. der **„Logifizierung" kombinatorischer Aufgaben,** wie in folgenden Beispielen.

Rätsel 1: Der Kriminalinspektor sucht den oder die Täter unter vier Verdächtigen. Bei der Vernehmung werden folgende Aussagen gemacht:

- Ede: Jimmy hat's getan
- Jimmy: Carlo war's.
- Bomber: Ich war es jedenfalls nicht.
- Carlo: Jimmy log, als er sagte ich hätt's getan.

Der Inspektor weiß von der stets zuverlässigen Räuber-Jenny, dass genau eine der Aussagen stimmt und dass die vier stets nur einzeln agieren.

Wir arbeiten zur Formalisierung mit 4 Aussagevariablen: *B, C, E, J*: „Bomber/ Carlo/ Ede/ Jimmy war der Täter." Nächstes Ziel: Gewinnung der **Wissensbasis** = Menge wahrer Formeln, deren (ausrechenbare!) **Modelle** den bzw. die möglichen Täter benennen und damit die Aufgabe lösen.

Die vier Aussagen der Verdächtigen sind: J, C, $\neg B$ und $\neg C$. Die Wissensbasis des Inspektors ist aber $\{P,Q\}$ mit

P: Entweder stimmt genau die erste Aussage oder genau die zweite usw., also

$$(J \wedge \neg C \wedge B \wedge C) \vee (\neg J \wedge C \wedge B \wedge C) \vee (\neg J \wedge \neg C \wedge \neg B \wedge C) \vee (\neg J \wedge \neg C \wedge B \wedge \neg C).$$

Q: Es kommen keine zwei Täter in Frage, also

$$(B \wedge \neg C \wedge \neg E \wedge \neg J) \vee (\neg B \wedge C \wedge \neg E \wedge \neg J) \vee (\neg B \wedge \neg C \wedge E \wedge \neg J) \vee (\neg B \wedge \neg C \wedge \neg E \wedge J)$$

Mit einer 16-zeiligen Wahrheitstafel (oder weniger aufwändig) sind alle Modelle bestimmbar. Die Aufgabe ist gut gestellt: sie ergibt genau ein Modell (Lösung).

Rätsel 2: Sudoku → Beispiel s.u. Tabelle 5

Regel (1): Ein Sudoku-Feld ist eine 9×9-Matrix *M*, d.h. mit 9 Zeilen und 9 Spalten. Man unterscheidet darin 9 dis¬junkte Regionen, jede bestehend aus 3×3-Feldern.

Regel (2): Die möglichen Matrixeinträge sind {leer, 1, ... , 9}. Zu Beginn sind nur einige dieser Felder mit Zahlen gefüllt, also etliche Positionen leer.

Tab. 5: Sudoku-Beispiel

8				**2**	**9**	**5**		**7**
	5						**8**	
9		**4**			**5**	**2**		
1		**3**						
4				**3**				**8**
						7		**6**
		5	**6**			**8**		**3**
	7						**9**	
3		**1**	**4**	**7**				**2**

8	1	6	3	**2**	**9**	**5**	4	**7**
7	**5**	2	1	4	6	3	**8**	9
9	3	**4**	7	8	**5**	**2**	6	1
1	8	**3**	9	6	7	4	2	5
4	6	7	5	**3**	2	9	1	**8**
5	2	9	8	1	4	**7**	3	**6**
2	4	**5**	**6**	9	1	**8**	7	**3**
6	**7**	8	2	5	3	1	**9**	4
3	9	**1**	**4**	**7**	8	6	5	**2**

Regel (3): Die Aufgabe besteht darin, das Feld vollständig so auszufüllen, d.h. die M_{ik} =leer so zu verändern, dass in jeder Zeile, jeder Spalte und jeder Region jeweils jede Zahl zwischen 1 und 9 genau einmal vorkommt.

Wir verwenden für die Beschreibung der Anforderungen an die Lösung 729 atomaren Aussagevariablen A_{ijk}, $i, j, k = 1,...,9$, und eine Belegung *belLsg* mit

$$belLsg(A_{ijk}) = \mathrm{W} \quad \Leftrightarrow \quad M_{ij} = k \quad ,$$

d.h. A_{ijk} besagt: Auf Feld (i, j) befindet sich die Zahl k.

Nun können wir die Lösungs-Matrix-Eigenschaften und Sudoku-Spielregeln mit Hilfe der Aussagenlogik **formalisieren**:

(i) Auf jedem Feld befindet sich eine Zahl:

z.B. für *ij*=11: $\mathrm{ZahlDrin}_{11} :\Leftrightarrow A_{111} \vee A_{112} \vee \cdots \vee A_{119}$

also: $\mathrm{ZahlDrin}_{11} \wedge \mathrm{ZahlDrin}_{12} \wedge \ldots \mathrm{ZahlDrin}_{99}$

(ii) Auf keinem Feld befinden sich zwei Zahlen:

z.B. für *ij*=11: $\mathrm{NieZwei}_{11} :\Leftrightarrow$

$\neg(A_{111} \wedge A_{112}) \wedge \neg(A_{111} \wedge A_{113}) \wedge \cdots \wedge \neg(A_{118} \wedge A_{119})$

also: $\mathrm{NieZwei}_{11} \wedge \mathrm{NieZwei}_{12} \wedge \ldots \mathrm{NieZwei}_{99}$

Ähnlich:

(iii) In keiner Zeile, Spalte, Region kommt eine Zahl doppelt vor. …

(iv) Schließlich die Festlegung der Anfangszahlen, im Beispiel also

$$A_{118} \wedge A_{152} \wedge A_{169} \wedge \cdots \wedge A_{957} \wedge A_{992}$$

Dann „einfach" in einen logischen Problemlöser eintippen ☹ und nach einem **Modell**, d.h. nach den restlichen gültigen A_{ijk}, fragen. Die kommen dann schnell. ☺

Zum Glück geht es auch anders …

Ein gutes Sudoku hat **genau ein Modell**, d.h. genau eine Lösung.

Manche Denksportaufgaben haben null oder mehrere Lösungen oder sind sogar anderweitig problematisch.

Äquivalenzen

Zwei Formeln φ und ψ heißen **semantisch** (oder **logisch**) **äquivalent**, $\varphi \equiv \psi$ bzw. $\varphi \models\!\models \psi$, wenn sie unter allen (für beide ausreichenden) Belegungen den gleichen Wahrheitswert haben, wenn also φ und ψ in der Wahrheitstafel **identische Wahrheitswerteverläufe** haben.

Äquivalenzsatz
Zwei Formeln sind genau dann semantisch äquivalent, wenn $\varphi \leftrightarrow \psi$ allgemeingültig, also $\varphi \equiv \psi \Leftrightarrow \models \varphi \leftrightarrow \psi$ ist.

Beweisidee: Wodurch unterscheiden sich technisch die Wahrheitstafelproben beider Seiten?

Tautologien mit ↔ „oben" ergeben Äquivalenzen.

Syntaktisch-semantische Zusammenhänge

Der $\mathsf{T}\bot$-Dialekt. Die Sprache der **AL-Formeln mit den Konstanten** *top* und *bottom* wird durch folgende Grammatik erzeugt:
Nichtterminalsymbol-Menge $\{S\}$ und Startsymbol S wie gehabt.

- $\mathrm{T} = \{ \mathsf{T}, \bot, A_1, A_2, ..., \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,) \}$
  (Terminalsymbole erweitert)
- $\mathrm{R} = \{ S \rightarrow \mathsf{T} \mid \bot \mid A_1 \mid A_2 \mid ...,$
  $S \rightarrow \neg S \mid (S \wedge S) \mid (S \vee S) \mid (S \rightarrow S) \mid (S \leftrightarrow S) \}$
  (Regeln: Verwendung wie Variablen)

Der **Wert** ist auf den Konstanten **unabhängig von der Belegung** festgelegt:

$wert_{bel}(\mathsf{T}) := W$

$wert_{bel}(\bot) := F$

T und $\bot$ kommen in der natürlichen Sprache nicht vor. Punktuell spielen bestimmte bekräftigende Redewendungen diese Rolle:

- „... oder ich fress' einen Besen", für $\ldots \vee \bot$
- „Ach Quatsch", für $\ldots \wedge \bot$
- „weil es nun einmal so ist wie es ist", für $\ldots \wedge \mathsf{T}$

Zusätzlichen Tautologien im $\mathsf{T}\bot$-Dialekt

T	$\neg\bot$	***Numquam falsum***
$(P \vee \neg P) \leftrightarrow \mathsf{T}$	$(P \wedge \neg P) \leftrightarrow \bot$	***Tertium non datur***
$(P \wedge \mathsf{T}) \leftrightarrow P$	$(P \wedge \bot) \leftrightarrow \bot$	***Absorptionsgesetze***
$(P \vee \mathsf{T}) \leftrightarrow \mathsf{T}$	$(P \vee \bot) \leftrightarrow P$	
$(\mathsf{T} \rightarrow P) \leftrightarrow P$	$\bot \rightarrow P$	***Ex vero nonnisi verum*** ***Ex falso quodlibet***

Assoziativitätsaspekte
Wegen der Assoziativgesetze schreibt man Ketten von (d.h. „benachbarte") Konjunktionen bzw. Ketten von Disjunktionen meist ohne innere Klammern, denn alle Klammerungen ergeben äquivalente Terme
[Entgegen Gerüchten ist der Beweis nicht trivial!]
Doppeltes Beispiel:

$\big((A \wedge (B \wedge C)) \vee D\big) \vee E \Leftrightarrow (A \wedge B \wedge C) \vee D \vee E$.

Wo man umgekehrt eine *eindeutig* bestimmte Formel mit Klammern benötigt, genügt eine Standardinterpretation der „Sammelformel", z.B. „von links nach rechts":

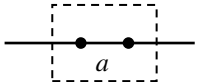
$\varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_n \Leftrightarrow ((...((\varphi_1 \wedge \varphi_2) \wedge \varphi_2) \wedge ... \wedge \varphi_{-1}) \wedge \varphi_n)$

Andere **Schreibweise**: $\varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_n \Leftrightarrow \bigwedge_{k=1}^{n} \varphi_k$,

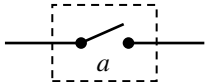
Analog für $\vee$ und $\bigvee$.

Anwendung: Boolesche Schaltwerke

Ein **simples Boolesches Schaltwerk** ist aus einer Anzahl von Elementen der Art **Arbeitskontakt** oder **Ruhekontakt** zusammengesetzt:



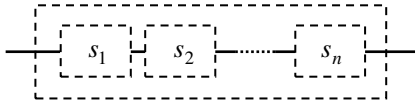
Ruhekontakt $\neg a$



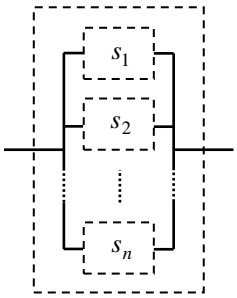
Arbeitskontakt a

(Schalter a steht auf AUS)

Die Zusammensetzung dieser Kontakte bzw. bereits zusammengesetzter Kontaktegruppen (aha: induktiv!) erfolgt **parallel** oder **in Reihe** (**seriell**):



Reihenschaltung



Parallelschaltung

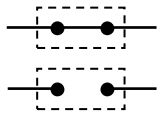
Heutige Schaltwerke enthalten natürlich auch komplexere Elemente (Gatter).

Konventionen

- Verschiedene Kontakte können den gleichen Namen tragen. Sie haben dann stets die gleiche Stellung.
- a- und $\neg a$-Kontakte haben stets entgegengesetzte Stellung.
- Alle Kontakte mit Anschrift a oder $\neg a$ werden vom gleichen Schalter a gesteuert.
- Die Kontakte sind in Stellung Schalter auf AUS gezeichnet.

1. Zuordnung: Boolesche Schaltwerke → AL-Formeln:

Arbeitskontakt m. Schalter a $\mapsto a$
Ruhekontakt mit Schalter a $\mapsto \neg a$
Reihenschaltg. v. $S_1, ..., S_n$ $\mapsto Z(S_1) \wedge ... \wedge Z(S_n)$
Parallelschaltg. v. $S_1, ..., S_n$ $\mapsto Z(S_1) \vee ... \vee Z(S_n)$
feste Verbindung $\mapsto \mathsf{T}$
dauernde Unterbrechung $\mapsto \bot$



2. Zuordnung:
Schalterstellungskombinationen → Belegungen

$Komb \mapsto$ Belegung bel_{Komb} mit

$bel_{Komb} : a \mapsto W :\Leftrightarrow$ Schalter a unter $Komb$ auf EIN.

Satz: S verbindet leitend bei Schalterstellung $Komb$
$\Leftrightarrow wert_{bel_{Komb}}(Z(S)) = \mathrm{W}$.

Heutige Schaltwerke enthalten natürlich auch komplexere Elemente (Gatter).

Formeln für AV'n einsetzen – Substitution

Substitutionssatz:
Werden in einer Tautologie (bzw. in einer unerfüllbaren Formel) für alle Aussagevariablen $A_i, i = 1,2,\ldots$ alle Vorkommen von A_i jeweils durch die gleiche Formel ψ_i ersetzt (Substitution, s.u.), so ist die entstehende Formel ebenfalls eine Tautologie (bzw. unerfüllbar).

Man schreibt meist nur die Ersetzungen mit $A_i \neq \psi_i$ und auch nur die für vorkommende A_i hin, also nur endlich viele.

Beispiel

$A \vee (A \rightarrow B)$ $\quad \begin{matrix} A \mapsto B \\ B \mapsto A \wedge B \end{matrix} \quad$ $B \vee (B \rightarrow (A \wedge B))$

Wozu dient der Substitutionssatz? Beweisen Sie doch $(A \wedge B \wedge C \wedge D) \rightarrow (A \wedge B \wedge C \wedge D)$, einmal mit Wahrheitstafel, einmal mit Substitutionssatz!

Rekursive Definition der (gleichzeitigen) Substitution von (paarweise verschiedenen) Aussagevariablen X_1, ..., X_n durch (nicht unbedingt verschiedene) Formeln $\psi_1, ..., \psi_n$, $\varphi_{[X_1, ..., X_n / \psi_1, ..., \psi_n]}$:

Für $X \in AV$ ist

$$X_{[X_1, ..., X_n / \psi_1, ..., \psi_n]} := \begin{cases} \psi_i & \text{für } X = X_i \\ X & \text{sonst} \end{cases},$$

$$(\neg\varphi)_{[X_1, ..., X_n / \psi_1, ..., \psi_n]} := \neg(\varphi_{[X_1, ..., X_n / \psi_1, ..., \psi_n]})$$

und für $\otimes \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ ist

$$(\varphi \otimes \rho)_{[X_1, ..., X_n / \psi_1, ..., \psi_n]} := \varphi_{[X_1, ..., X_n / \psi_1, ..., \psi_n]} \otimes \rho_{[X_1, ..., X_n / \psi_1, ..., \psi_n]}$$

Achtung: Reihenfolge-Effekte bei sequentieller Substitution: Oft sind $\varphi_{[A,B/\psi,\rho]}$, $\varphi_{[A/\psi][B/\rho]}$ und $\varphi_{[B/\rho][A/\psi]}$ *alle verschieden und nicht einmal äquivalent.*

Achtung: Die „Umkehrung" des Substitutionssatzes ist falsch, durch Substitution kann Tautologie neu entstehen.

Teilformeln ersetzen – Ersetzung

Ersetzungssatz (Satz über die äquivalente Ersetzung):
Werden in einer Formel φ ein oder mehrere Vorkommen einer Teilformel ψ durch eine zu ψ äquivalente Formel ρ ersetzt, so ist die entstehende Formel zu φ äquivalent.

Die Menge aller Möglichkeiten der Ersetzung, $Ers_{[\psi/\rho]}(\varphi)$, ist rekursiv definiert:

$$Ers_{[\psi/\rho]}(\varphi) := \{\varphi\} \cup \begin{cases} \text{if } \varphi = \psi & \text{then } \{\rho\} \\ \text{else if } \varphi = \neg\sigma & \text{then } \{\neg\tau \mid \tau \in Ers_{[\psi/\rho]}(\sigma)\} \\ \text{else if } \varphi = \varphi_1 \otimes \varphi_2 & \text{then } \{\sigma \otimes \tau \mid \sigma \in Ers_{[\psi/\rho]}(\varphi_1), \tau \in Ers_{[\psi/\rho]}(\varphi_2)\} \\ \text{else } \varnothing \end{cases}$$

Anwendungsbeispiel: Wegen $(B \rightarrow C) \equiv (\neg C \rightarrow \neg B)$ und $A \equiv \neg\neg A$ gilt $(B \rightarrow C) \vee A \equiv (\neg C \rightarrow \neg B) \vee \neg\neg A$.

Vergleich Ersetzung / Substitution
Bildlich im Syntaxbaum:
Substitution = alle gleichnamigen Blätter durch je denselben Ast ersetzen.
Ersetzung = einige gleiche Äste durch je denselben dazu äquivalenten Ast ersetzen.

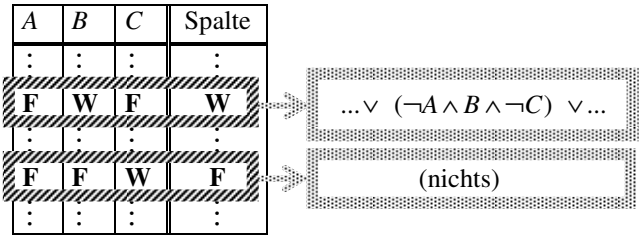
Transformation:	einfache Subst.	einfache Ersetzung
Was wird ersetzt?	Aussagevar.	Teilformel
Welche Vorkommen?	Alle	0, 1, mehrere, alle
Wodurch?	beliebige Formel	äquivalente Formel
Ergebnis äquivalent?	JA, falls Tautologie / Widerspr.	JA

Junktorenbasen

Mit den Junktoren einer Junktorenbasis können wir alle möglichen Spaltenbelegungen in Wahrheitstafeln erzeugen. Genauer gesagt ist eine **Junktorenbasis** *JuBa* eine Junktorenmenge der Art, dass wir zu jeder endlichen Menge *VarSet* von Aussagevariablen jeden gewünschte Wahrheitswerteverlauf (über alle Belegungen dieser Variablen) mit einer geeigneten **Formel „über** *JuBa* **und** *VarSet* " erzielen können.

Beispiel: $\{\neg, \wedge, \vee\}$ ist Junktorenbasis

Beweisidee: Der gewünschten 2^n-zeiligen Wahrheitstafel entspricht eine **Disjunktion von** 2^n **Konjunktionen** aus je n **Literalen** (= Variable/$\neg$ Variable), vgl. Schema:



Satz: **Von Junktorenbasen zu Junktorenbasen**
Seien M und N Junktorenmengen und M eine Junktorenbasis. Genau dann ist auch N eine Junktorenbasis, wenn zu jedem, sagen wir n-stelligen, Junktor $j \in M$ eine zu $j(A_1, ..., A_n)$ äquivalente Formel über N und $\{A_1, ..., A_n\}$ existiert.
Jede Junktoren-Obermenge einer Junktorenbasis ist daher auch eine Junktorenbasis.

Anwendungen:
$\{\neg, \wedge\}$ und $\{\neg, \vee\}$ sind Junktorenbasen
wegen $A \wedge B \equiv \neg(\neg A \vee \neg B)$ und $A \vee B \equiv \neg(\neg A \wedge \neg B)$,
Junktorenbasen-, Substitutions- und Ersetzungssatz

Es geht sogar noch sparsamer!

- Die **Sheffer**-Verknüpfung alleine, d.h. $\{\uparrow\}$, sowie
- die **Peirce**-Verknüpfung alleine, d.h. $\{\downarrow\}$, ist eine Junktorenbasis.

Beweistip: Suche für $\neg\varphi$ und für $\varphi \wedge \psi$ oder $\varphi \vee \psi$ eine Formel über neuer Basis.

- **If-then-else** alleine, d.h. $\{\rightarrow / \}$, ist eine Junktorenbasis im $\mathsf{T}\,\bot$-Dialekt

Beweistip: analog oder Shannon-Umformung, s.u.

Satz: **Shannon-Expansion**
Für jede AL-Formel φ und Aussagevariable P gilt:

$$\varphi \equiv P \rightarrow \mathrm{Subst}_{P \rightarrow \mathsf{T}}(\varphi) / \mathrm{Subst}_{P \rightarrow \bot}(\varphi)$$

Der Satz ist leicht einzusehen, wenn Sie sich die Wahrheitstafel in üblicher Schreibweise mit P in der ersten Spalte vorstellen.
Wie sieht die Shannon-Expansion (nach A) von φ aus, wenn A nicht in φ vorkommt?
Zwei Beispiele für Expansionsschritt

$$\neg A \equiv A \rightarrow \neg\mathsf{T} / \neg\bot$$
$$A \vee B \equiv A \rightarrow (\mathsf{T} \vee B) / (\bot \vee B)$$

Algorithmus ITE: Shannon-Umformung

Beginne mit der Ausgangsformel φ. Solange möglich:

- wähle eine noch nicht verarbeitete vorkommende Variable P,
- wende die Shannon-Expansion bzgl. P auf die zuletzt erhaltene Formel an
- und markiere P als verarbeitet.

Vereinfache schließlich die konstanten Teilterme zu $\top$ bzw. $\bot$.

Satz: **Korrektheit der Shannon-Umformung**
Die Shannon-Umformung **terminiert** und führt zu einer äquivalenten Formel in ITE-Form.

Zwei Beispiele für die Vereinfachung am Ende

$A \to \neg\top / \neg\bot \quad \equiv \quad A \to \bot / \top$

$A \to (\top \vee B)/(\bot \vee B) \quad \equiv \quad A \to \top / B$

Folgerungen

φ **folgt aus** (**ist „Folgerung aus“** = Folge von) einer Formelmenge M, geschrieben $M \models \varphi$,
$:\Leftrightarrow$ Jedes für φ ausreichende Modell von M ist auch ein Modell von φ, d.h. $bel \models M \Rightarrow bel \models \varphi$.
Bei einelementigem M schreibt man $\psi \models \varphi$ anstelle von $\{\psi\} \models \varphi$
Natürlich **folgt** jede Formel **in** M auch **aus** M. Folgt eine Formel aus einer Formelmenge M, dann auch aus jeder **Obermenge** von M.

Ist $M = \{\psi_1, \psi_2, ..., \psi_n\}$ eine endliche Menge von Formeln, gilt $M \models \varphi \Leftrightarrow (\psi_1 \wedge \psi_2 \wedge ... \wedge \psi_n) \models \varphi$.
$M \models \varphi$ erkennen wir bei einer **endlichen** Formelmenge M in der **Wahrheitstafel**:
In den Zeilen in denen **alle Formeln von** M den Wert W haben, hat **auch** φ den Wert W.

Beispiel: $\{A \to B, \neg B\} \models \neg A$

A	B	$A \to B$	$\neg B$	$\neg A$
W	W	W	F	F
W	F	F	W	F
F	W	W	F	W
F	F	**W**	**W**	**W**

Satz **Folgerung und Tautologie**
φ ist Tautologie
$\Leftrightarrow$ φ folgt aus der leeren Formelmenge.
$\Leftrightarrow$ φ folgt aus jeder Formelmenge.
$\Leftrightarrow$ φ folgt aus einer Menge von Tautologien.

Satz: **Folgerung und Unerfüllbarkeit**
$M \models \varphi \quad \Leftrightarrow \quad M \cup \{\neg\varphi\}$ unerfüllbar.
M ist genau dann unerfüllbar, wenn

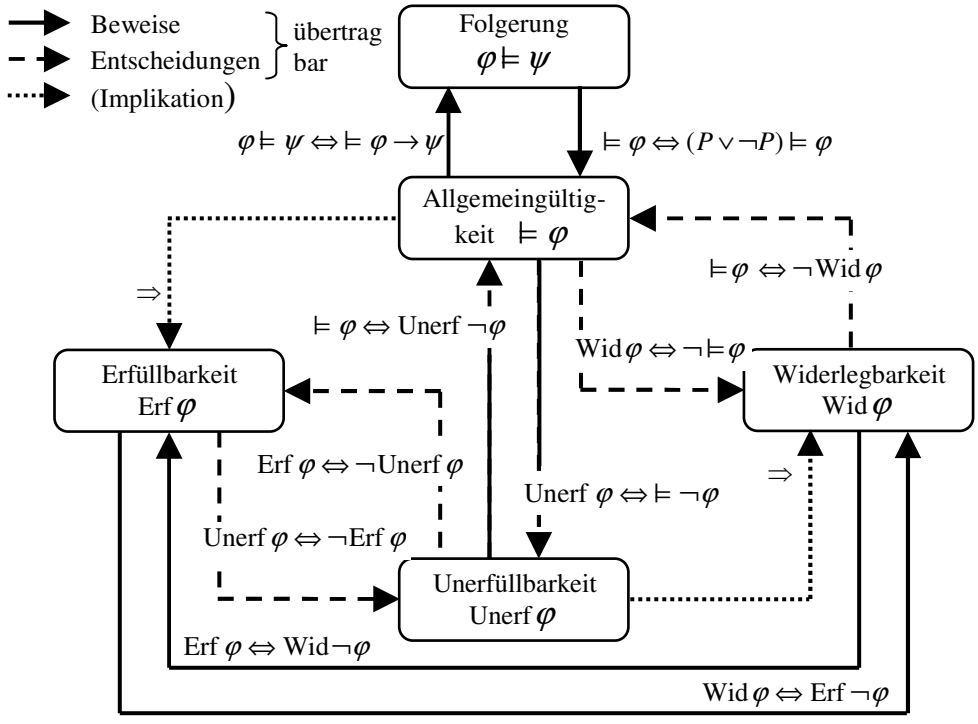
- jede Formel eine Folgerung aus M ist, bzw.
- mindestens eine unerfüllbare Formel eine Folgerung aus M ist.

Satz: Folgerung und Implikation, **Deduktionssätze**
$M \cup \{\psi\} \models \varphi \quad \Leftrightarrow \quad M \models (\psi \to \varphi)$
Für zwei Formeln φ, ψ gilt: $\varphi \models \psi \quad \Leftrightarrow \quad \models \varphi \to \psi$.
Für $m \geq 1$ und Formeln $\varphi_1, \varphi_2, \ldots, \varphi_{m-1}, \varphi_m$ und ψ gilt:
$\varphi_1, \varphi_2, \ldots, \varphi_{m-1}, \varphi_m \models \psi \Leftrightarrow \varphi_1, \varphi_2, \ldots, \varphi_{m-1} \models \varphi_m \to \psi$.

Satz: **Folgerung und Modelle**
Sei N **Menge von Folgerungen** aus M, d.h. für alle $\varphi \in N: M \models \varphi$, und *bel* sei für M und N ausreichende Belegung. Dann gilt:
bel Modell von $M \quad \Leftrightarrow \quad$ *bel* Modell von $M \cup N$.
Wir können dann also M um N erweitern, ohne an ihrer eventuellen Eigenschaft der

- Erfüllbarkeit,
- Unerfüllbarkeit oder

Abb.5: Gegenseitige Zurückführung semantischer Fragestellungen



- Allgemeingültigkeit aller ihrer Formeln

etwas zu ändern.

Abbildung 5 fasst zusammen, wie semantische Fragestellungen aufeinander zurückgeführt werden können.

Wissen

Sei M Formelmenge, $M \subseteq Form$. Wenn man weiß, dass alle Formeln in M gelten, d.h. wenn man über M als **Wissensbasis** verfügt, dann weiß man auch, dass alle Folgerungen aus M gelten. Letztere bilden das durch M gegebene (implizierte) **Wissen**.

$\text{Wissen}(M) := \{\varphi \in Form \mid M \models \varphi\}$

Alle anderen Formeln, sofern deren Negation nicht gewusst wird, könnten dann noch wahr oder falsch sein.

Satz: **Wissen**
Seien M und N Formelmengen.

- Wissen($\varnothing$) ist die Menge aller Tautologien von AL;
- $M \subseteq N \Rightarrow$ Wissen(M) $\subseteq$ Wissen(N);
- Wissen(Wissen(M)) = Wissen(M);
- M ist unerfüllbar $\Leftrightarrow$ Wissen(M) = Form.*

*) Aber wenn man glaubt, eine widersprüchliche Formelmenge zu wissen, dann irrt man sich.

Konjunktive Normalform und Resolution

Literal: eine Aussagevariable oder die Negation einer solchen: A, $\neg A$, ...
Klausel: eine Disjunktion (mit $\vee$ verknüpfte Kette) von Literalen $A \vee B$, $A \vee \neg B \vee \neg C$, ...
(**disjunktive Klausel**).
Horn-Klausel: Klausel, die höchstens ein positives Literal P enthält, z.B. $A \vee \neg B \vee \neg C$.
Dualklausel Konjunktion (mit $\wedge$ verknüpfte Kette) von Literalen, $A \wedge B$, $A \wedge \neg B \wedge \neg C$, ...
(**konjunktive Klausel**).

Klauseln erleichtern gewisse Prüfungen: Eine **Klausel** (bzw. eine ***Dualklausel***) ist genau dann **allgemeingültig** (bzw. ***unerfüllbar***), wenn sie für eine Aussagevariable P sowohl P als auch $\neg P$ enthält.

Beispiele: $A \vee \neg B \vee \neg A$, $A \wedge \neg B \wedge \neg A$ prüfen!

Eine Formel in **konjunktiver Normalform** (**KNF-Formel**) ist eine Konjunktion von Klauseln,

$$(Lit_{1,1} \vee Lit_{1,2} \vee ... \vee Lit_{1,k_1}) \wedge (Lit_{2,1} \vee Lit_{2,2} \vee ... \vee Lit_{2,k_2}) \wedge ... \wedge (Lit_{n,1} \vee Lit_{n,2} \vee ... \vee Lit_{n,k_n})$$

Andere **Schreibweise**: $\bigwedge_{i \in I} \bigvee_{k \in K_i} Lit_{ik}$ (mit endl. Indexmengen I und $K_i, i \in I$).

Beispiele:
- $(\neg A \vee B) \wedge (A \vee \neg B \vee C) \wedge A \wedge \neg C$,
- alle Klauseln,
- alle Dualklauseln.

Grammatik der KNF-Formeln (ohne äußere Klammern):

- KNF → (Kl) | (Kl) $\wedge$ KNF
- Kl → Lit | Lit $\vee$ Kl
- Lit → AV | $\neg$ AV
- AV → A_1 | A_2 | A_3 | ...

Man betrachtet eine KNF-Formel oft als **Menge von Klauseln** und jede dieser Klauseln als **Menge von Literalen**:

KNFset$((\neg A \vee B) \wedge (A \vee \neg B \vee C) \wedge A \wedge \neg C)$
$= \{\{\neg A, B\}, \{A, \neg B, C\}, \{A\}, \{\neg C\}\}_{KNF}$.

Rechtfertigung: Wie die Mengen ist auch der Wahrheitswert der Original-KNF-Formel **invariant** unter Umordnung und Wiederholung innerhalb der Mengen/Ketten.

Um **allen** Mengen von Mengen von Literalen unter einer passenden Belegung einen Wahrheitswert zuteilen zu können, vereinbart man: Die **leere** KNF-Formel $\{\}_{KNF}$ ist immer **wahr;** die **leere** Klausel $\{\}_{Kl}$ ist immer **falsch**.

Satz: **Allgemeingültigkeit einer KNF-Formel**
Eine KNF-Formel φ ist **allgemeingültig**

- $\Leftrightarrow$ alle Klauseln von φ sind allgemeingültig
- $\Leftrightarrow$ **jede Klausel** von φ enthält für mindestens eine Variable P die Literale P und $\neg P$.

Beweisidee:
(1) $\Rightarrow$ (2): Wäre $KNF = Kl \wedge Rest$, und Kl unter *bel* falsch, dann auch $Kl \wedge Rest$.
(2) $\Rightarrow$ (3): Wären alle Variablen in Klausel $\{Lit_1, Lit_2, ..., Lit_k\}_{Kl}$ unterschiedlich, dann belege alle „anders als in der Klausel“, und sie würde falsch.
(3) $\Rightarrow$ (1): $P \vee \neg P$ ist stets wahr, also auch $P \vee \neg P \vee Rest$, also jede Klausel sowie $\bigwedge(Klauseln)$.

Wegen der leichten Ablesbarkeit der Allgemeingültigkeit, wäre es sehr schön, alle Formeln nach KNF „äquivalent umformen“ zu können.

Normalformensatz (für KNF)
Zu jeder AL-Formel gibt es mindestens eine äquivalente KNF-Formel.

Algorithmus KNF1 mit Wahrheitstafel

Synthese „der KNF einer Formel φ“ **über den Wahrheitswerteverlauf**

1. Berechne den Wahrheitswerteverlauf von φ mit Hilfe der **Wahrheitstafel**.
2. Ist die Formel allgemeingültig (alle Formelwerte = W), so wähle eine Aussagevariable P und bilde $P \vee \neg P$ – fertig!
   In der Mengenform reicht auch die leere KNF-Menge $\{\}_{KNF}$ – (2) wird Teil von (3). Andernfalls gibt es Zeilen mit dem W.-Wert F.
3. Für jede Zeile mit dem Formelwert F bilde eine Klausel, die für jede vorkommende Aussagevariable P folgendes enthält:
   P, wenn der P-Wert in der Zeile F ist, und $\neg P$, wenn er W ist.
4. Die Menge bzw. Konjunktion dieser Klauseln ist eine KNF-Darstellung von φ.

A	B		$\neg(B \to A) \wedge \neg(A \wedge B)$
W	W		W
W	F		W
F	**W**		**F**
F	**F**		**F**

$(A \vee \neg B) \quad \wedge \quad (A \vee B)$

Beispiele für #3

bzw. $\{\{A, \neg B\}, \{A, B\}\}_{KNF}$

Achtung: Es gibt im Beispiel eine kürzere Lösung: A !

Algorithmus KNF2 mit äquivalenten syntaktischen Umformungen

a) Führe ggf. die wegen Assoziativität weggelassenen Klammern wieder ein.

b) Wende, so lange es geht, immer wieder irgendeine der folgenden Ersetzungen auf Teilformeln an:

1. Elimination von Äquivalenz und Implikation
   $\varphi \leftrightarrow \psi \quad \mapsto \quad (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
   $\varphi \rightarrow \psi \quad \mapsto \quad \neg\varphi \vee \psi$
2. Negation beseitigen oder näher an die Aussagevariablen rücken
   $\neg\neg\varphi \quad \mapsto \quad \varphi$
   $\neg(\varphi \wedge \psi) \quad \mapsto \quad \neg\varphi \vee \neg\psi$
   $\neg(\varphi \vee \psi) \quad \mapsto \quad \neg\varphi \wedge \neg\psi$
3. Konjunktion von den Literalen wegrücken
   $\varphi \vee (\psi \wedge \rho) \quad \mapsto \quad (\varphi \vee \psi) \wedge (\varphi \vee \rho)$
   $(\psi \wedge \rho) \vee \varphi \quad \mapsto \quad (\psi \vee \varphi) \wedge (\rho \vee \varphi)$

Beschleunigungsmöglichkeit: #2 und #3 auch auf 3- und mehrgliedrige Ketten anwenden.

In #3 nur $\vee$ über $\wedge$ verteilen, nicht umgekehrt!

Satz: **Korrektheit von KNF1 und KNF2 (→ Normalformensatz)**

KNF1 und KNF2 terminieren stets. Die durch Umformung erhaltene Formel ist in KNF (ggf. durch geeignetes Weglassen von Klammern) und zur Ausgangsformel äquivalent.

Zur Entscheidung über die *Erfüllbarkeit* oder *Unerfüllbarkeit* von KNF-Formeln existiert ein besonderer Algorithmus.

Algorithmus: AL-Resolution aus KNF

Sei φ KNFset-Formel. Wir setzen $\varphi_0 := \varphi$.

Für k = 0,1,... (bis zum unten definierten Abbruch) gehen wir wie folgt vor:

Resolutionsschritt: Sind zwei der Klauseln, κ_1, κ_2, von φ_k von der Form $\kappa_1 = \{P, Rest_1\}$ und $\kappa_2 = \{\neg P, Rest_2\}$ (auch mit evtl. leerem $Rest_{1/2}$), und ist $Rest_1 \cup Rest_2 \notin \varphi_k$, so setzen wir $\varphi_{k+1} := \varphi_k \cup \{Rest_1 \cup Rest_2\}$ (**Resolvente**);

- Variante: Ist $Rest_1 \cup Rest_2 = \varnothing$ → STOP.

(In der Original-Formelschreibweise erzeugen wir mit $\kappa_1 = A \vee Rest_1$ und $\kappa_2 = \neg A \vee Rest_2$ die erweiterte Formel $\varphi_{k+1} := \varphi \wedge (Rest_1 \vee Rest_2)$.)

Ist mit φ_k stattdessen kein Resolutionsschritt mehr möglich: STOP.

Das zuletzt erhaltene φ_i ist eine (bzw. *die*, s.u.) **Resolventenmenge Res(φ)**.

Schema:

2 Klauseln: $Rest_1 \quad A$; $\neg A \quad Rest_2$

mit passenden Literalen $Rest_1$: A ; $\neg A$; $Rest_2$

ergeben Resolvente: $Rest_1 \quad Rest_2$

Beispiel: $(\neg A \vee B) \wedge (\neg B \vee C) \wedge A \wedge \neg C$,

mögliche Resolutionsschritte

je: KNF-Set-Formel

1. Klausel 2. Klausel Reste 1/2

$\varphi_0 = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}\}$
$\{A\} \quad \{\neg A, B\} \quad \varnothing / \{B\}$

$\varphi_1 = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}, \{B\}\}$
$\{\neg B, C\} \quad \{\neg C\} \quad \{\neg B\} / \varnothing$

$\varphi_2 = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}, \{B\}, \{\neg B\}\}$
$\{B\} \quad \{\neg B\} \quad \varnothing / \varnothing$ Variante:

Variante:

$\varphi_3 = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}, \{B\}, \{\neg B\}, \varnothing\}$ **(stop)**
$\{\neg A, B\} \quad \{\neg B\} \quad \{\neg A\} / \varnothing$

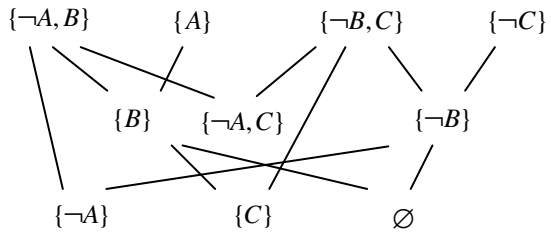
$\varphi_4 = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}, \{B\}, \{\neg B\}, \varnothing, \{\neg A\}\}$
$\{\neg B, C\} \quad \{B\} \quad \{C\} / \varnothing$

$\varphi_5 = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}, \{B\}, \{\neg B\}, \varnothing, \{\neg A\}, \{C\}\}$
$\{\neg A, B\} \quad \{\neg B, C\} \quad \{\neg A\} / \{C\}$

$\varphi_6 = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}, \{B\}, \{\neg B\}, \varnothing, \{\neg A\}, \{C\}, \{\neg A, C\}\}$

Achtung: die vorige Version ist bequem für die mathematische Beschreibung und den Korrektheitsbeweis des Algorithmus. Manuell führen wir ihn eher graphisch aus s.u., und programmieren würden wir vielleicht eine Folgen-Version, s.u.

Resolution, **graphisch**:



Eine andere willkürliche Reihenfolge führt evtl. zu einem anders geformten Resolutionsgraph, jedoch stets zur gleichen Resolventenmenge.

Resolutionssatz der AL

1. Es sind stets nur endlich viele Resolutionsschritte möglich. Der Algorithmus terminiert (STOP).
2. Alle konstruierbaren Resolventen von φ sind (in Mengensicht) identisch, also logisch äquivalent. Man spricht daher von **der** Resolventenmenge.
3. φ ist semantisch äquivalent zu Res(φ).
4. φ ist genau dann **unerfüllbar**, wenn Res(φ) die **leere Klausel enthält** bzw. – in Formelschreibweise – zwei Klauseln der Form P und $\neg P$ enthält. (das frühe *stop*)

Beweisideen

1. φ enthalte m Literale. Jeder RS erzeugt neue Teilmenge, $\leq 2^m$ Möglichkeiten.
2. Jeder einmal mögliche Resolutionsschritt bleibt weiter möglich, bis seine Resolvente erzeugt ist, also höchstens Reihenfolgeunterschiede.
3. Res. erzeugt Folgerungen aus φ:
   Resolutionsregel $((P \vee Q) \wedge (\neg P \vee R)) \rightarrow (Q \vee R)$.
4. $\Leftarrow$: einfach. $\Rightarrow$: siehe Literatur/Web

Wegen der Rückführungsmöglichkeiten eignet sich Resolution auch zur Entscheidung über Folgerung und Allgemeingültigkeit

Disjunktive Normalform und Tableaux

Eine Formel in **disjunktiver Normalform** (**DNF-Formel**) ist eine Disjunktion von Dualklauseln.

$(Lit_{1,1} \wedge ... \wedge Lit_{1,k_1}) \vee ... \vee (Lit_{n,1} \wedge ... \wedge Lit_{n,k_n})$, bzw.

$\bigvee_{i \in I} \bigwedge_{k \in K_i} Lit_{ik}$

Beispiele: $(\neg A \wedge B) \vee (\neg B \wedge C) \vee A \vee \neg C$,
alle Klauseln,
alle Dualklauseln.

Grammatik: analog der Grammatik der KNF-Formeln

Mengenschreibweise:

DNFset$((\neg A \wedge B) \vee (\neg B \wedge C) \vee A \vee \neg C)=$
$\{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}\}_{DNF}$.

Vereinbarung: Die **leere** DNF-Formel $\{\}_{DNF}$ ist immer **falsch;** die **leere** Dualklausel $\{\}_{Dk}$ ist immer **wahr** (auch allgemein die leere Disjunktion bzw. Konjunktion → ⊥-⊤-Dialekt)

Im Gegensatz zu KNF-Formeln ist hier die Un-/Erfüllbarkeit unmittelbar abzulesen:

Satz: **(Un-) Erfüllbarkeit einer DNF-Formel**
Eine DNF-Formel φ ist unerfüllbar

- ⇔ alle Dualklauseln von φ sind unerfüllbar (z.B. bei leerem φ)
- ⇔ jede Dualklausel von φ enthält für mindestens eine Variable P die Literale P und $\neg P$.

Normalformensatz (für DNF)
Zu jeder AL-Formel gibt es mindestens eine äquivalente DNF-Formel.

Algorithmus DNF1 – Synthese aus Wahrheitstafel

1. Berechne den Wahrheitswerteverlauf von φ mit Hilfe der **Wahrheitstafel**.
2. Ist die Formel unerfüllbar (alle Formelwerte = F), so wähle eine Aussagevariable P und bilde $P \wedge \neg P$ – fertig!
   In der Mengenform reicht auch die leere DNF-Menge $\{\}_{DNF}$ – (2) wird Teil von (3).
   Andernfalls gibt es Zeilen mit dem W.-Wert W.
3. Für jede Zeile mit dem Formelwert **W** bilde eine Klausel, die für jede vorkommende Aussagevariable P folgendes enthält:
   P, wenn der P-Wert in der Zeile W ist, und $\neg P$, wenn er F ist.
4. Die Menge bzw. Disjunktion dieser Dualklauseln ist die gesuchte DNF-Darstellung.

A	B		$\neg(B \rightarrow A) \wedge \neg(A \wedge B)$
W	**W**		**W**
W	**F**		**W**
F	W		F
F	F		F

$(A \wedge B)$ $\vee$ $(A \wedge \neg B)$ Beispiel

bzw. $\{\{A, B\}, \{A, \neg B\}\}_{DNF}$

Achtung: Es gibt eine kürzere Lösung: A !

Algorithmus DNF2 mit äquivalenten syntaktischen Umformungen

Wende, so lange es geht, immer wieder irgendeine der folgenden Ersetzungen auf Teilformeln an:

1. Elimination von Äquivalenz und Implikation (wie bei KNF)
2. Negation beseitigen oder näher an die Aussagevariablen rücken (wie bei KNF)
3. Disjunktion von den Literalen wegrücken
   $\varphi \wedge (\psi \vee \rho) \quad \mapsto \quad (\varphi \wedge \psi) \vee (\varphi \wedge \rho)$
   $(\psi \vee \rho) \wedge \varphi \quad \mapsto \quad (\psi \wedge \varphi) \vee (\rho \wedge \varphi)$

Beschleunigungsmöglichkeit: #2 und #3 auch auf 3- und mehrgliedrige Ketten anwenden.

⚠ In #3 nur ∧ über ∨ verteilen, nicht umgekehrt!

Satz: **Korrektheit von DNF1 und DNF2 (und damit Normalformensatz!)**

DNF1 und DNF2 terminieren stets. Die durch Umformung erhaltene Formel ist in DNF und zur Ausgangsformel äquivalent. (Klammern aus ∧- bzw. ∨-Ketten wegl.)

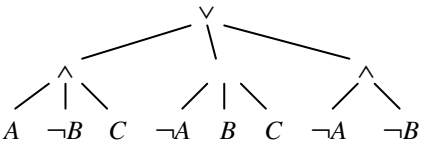
Formeln, Bäume, Automaten

Wir suchen nach Graphen (z.B. Bäumen oder Automaten), mit denen Formeln (ihre Syntax), bzw. ihr Werteverlauf (ihre Semantik) kodiert werden können.

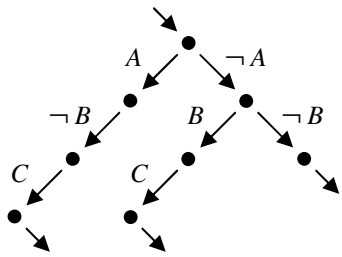
→ Tableaux → OBDDs

Für die DNF-Beispielformel
$\varphi = (A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (\neg A \wedge \neg B)$
eignen sich

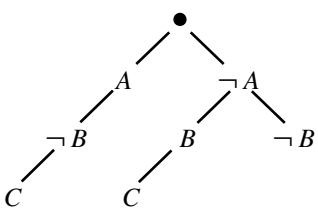
- Syntaxbaum(φ)



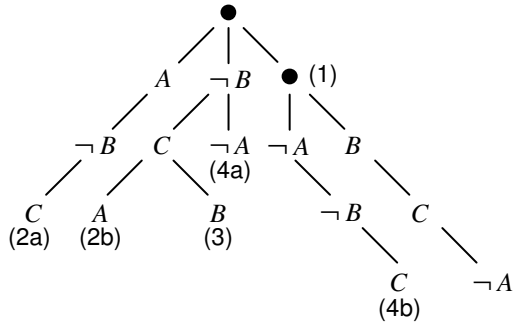
- sparsamer: ein (ABC-reihenfolgeabh.) Automat (ND bzw. unvollst.) der Dualklauseln von φ.



- nach Rollentausch Kante/Knoten ähnlich, als knotenetikettierter Baum, **Tableaubaum** genannt.



- als Tableaubaum mit mehr Freiheiten:
  - gemischten Reihenfolgen,
  - Leerknoten (neben Wurzel) (1)
  - Doppelproduktionen (2)
  - Widerspruchzweigen (3)
  - Oberbelegungen (4)



Tableauverfahren (*tableau*, pl. *tableaux* = Französisch für *Tabelle* – obwohl es unten um Bäume geht)
Grundgedanke: *Welche Voraussetzungen müssen gelten, damit eine Formel gilt*?

Dazu schreiben wir 9 Tautologien leicht verändert als **Tableauregeln**:

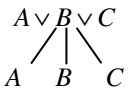
1. $\dfrac{\neg(\varphi \to \psi)}{\begin{array}{l}\varphi \\ \neg\psi\end{array}}$ 2. $\dfrac{\varphi \wedge \psi}{\begin{array}{l}\varphi \\ \psi\end{array}}$ 3. $\dfrac{\neg(\varphi \vee \psi)}{\begin{array}{l}\neg\varphi \\ \neg\psi\end{array}}$

4. $\dfrac{\varphi \leftrightarrow \psi}{\begin{array}{l}\varphi \to \psi \\ \psi \to \varphi\end{array}}$ 5. $\dfrac{\neg\neg\varphi}{\varphi}$ 6. $\dfrac{\varphi \to \psi}{\neg\varphi \mid \psi}$ 7. $\dfrac{\neg(\varphi \wedge \psi)}{\neg\varphi \mid \neg\psi}$

8. $\dfrac{\varphi \vee \psi}{\varphi \mid \psi}$ 9. $\dfrac{\neg(\varphi \leftrightarrow \psi)}{\neg(\varphi \to \psi) \mid \neg(\psi \to \varphi)}$

Beispielsweise die #1 und die #6:
$\neg(\varphi \to \psi) \leftrightarrow (\varphi \wedge \neg\psi)$: übereinander ≅ UND,
$(\varphi \to \psi) \leftrightarrow (\neg\varphi \vee \psi)$: nebeneinander ≅ ODER.

Beschleunigungsmöglichkeit bei #2, 3, 7 und 8:
längere Ketten, z.B. #8 erweitert:

$A \vee B \vee C \xrightarrow{\frac{\varphi_1 \vee \ldots \vee \varphi_n}{\varphi_1 \mid \ldots \mid \varphi_n}}$ 

Wiederholte Anwendung der Tableauregeln
- erzeugt einen Baum,
- „ersetzt" Biimplikation durch Implikationen,
- „ersetzt" Implikation durch UND oder ODER,
- bringt Negationen „nach innen", vor die Aussagevariablen.

Dies ähnelt der äquivalenten Umformung in DNF. Wie dort kann man
- alle Modelle ablesen und
- über die Erfüllbarkeit entscheiden.

Die (gedachte) Anwendung der Distributivität $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ entspricht dem „Auseinanderzupfen" des Tableaubaumes, bis er nur noch in der Wurzel verzweigt. → Oder-Kette von Und-Ketten → DNF

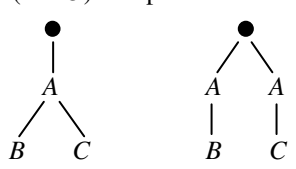


Tableau-Algorithmus zur DNF-Umformung

I. *Aus der Ausgangsformel φ einen **Tableaubaum** **Tab(φ)** konstruieren, dessen **Knoten** mit je einer Formel **beschriftet** und am Ende als **erledigt** markiert sind. Jeder Knoten ist dabei zunächst **unerledigt**.*

(1) ***Baum*** := Wurzelknoten, mit φ beschriftet (*und unerledigt, Startbaum*)

(2) So lange es geht, wähle einen unerledigten Knoten *k* in ***Baum***.
  a) Ist *k* **kein Literal**, dann **expandiere** *k,* d.h.
    - Hat *k*'s Formel die Struktur einer Kopf-Formel in der **oberen** Tableauregel-Reihe (1–5), dann **hänge an alle Blätter** unterhalb *k* die „entsprechenden" Formeln **untereinander** (*als Kind + Enkel etc., im Falle* $\neg\neg\psi$ *nur* ψ *als Kind*) an.
    - Hat *k*'s Formel die Struktur einer Kopf-Formel in der ***unteren*** Tableauregelreihe (6–9), dann **hänge an alle Blätter** unterhalb *k* die „entsprechenden" Formeln ***nebeneinander*** (als Kinder) an.
  b) Markiere Knoten *k* als **erledigt**. (*Literale sind also gleich erledigt.*)

(3) **Tab(φ)** := ***Baum***

II. *Aus dem **Tableaubaum** eine **DNF-Darstellung** ablesen.*

(4) Bilde für jeden **Zweig** von Tab(φ) eine **Dualklausel**, nämlich die **Konjunktion aller „Literal-Knoten"** des Zweiges.

(5) Bilde die **Disjunktion** dieser Dualklauseln für alle Zweige bzw. Blätter.

Satz: Das **Tableauverfahren** ist nicht-deterministisch, **terminiert** und liefert mit Schritt 5 eine **DNF-Darstellung** der Ausgangsformel.
Die Dualklauseln aus Schritt 3 liefern – mit allen möglichen Fortsetzungen auf Vars(φ) – die Modelle von φ.

Zur weiter gehenden Anwendung von Tableaux nennen wir während der Ausführung des Algorithmus
- einen Knoten **abgeschlossen,** wenn auf seinem Pfad ab der Wurzel zwei erkennbar widersprüchliche Formeln, („persönlicher Nichtdeterminismus" oder auf jeden Fall erkennbar: *P* und ¬ *P*) an Knoten stehen;
- ein Blatt **offen**, wenn es nicht abgeschlossen ist und alle Knoten auf seinem Zweig erledigt sind.

Tableau-Algorithmus zur Überprüfung der Erfüllbarkeit von Formeln
wie bisher, aber *nach dem ersten offenen Blatt aufhören*: die Formel ist dann *erfüllbar*; sie ist genau dann *unerfüllbar*, wenn nach Terminierung des Algorithmus *alle Blätter abgeschlossen* sind.

Tableau-Algorithmus zur Überprüfung der Allgemeingültigkeit von Formeln: die *Negation* bilden und deren *Unerfüllbarkeit* per Tableau überprüfen.

Der folgende Tableaubaum beweist die Erfüllbarkeit der Wurzelformel. Die Expansionsschritte sind mithilfe der Zeilennummern nachvollziehbar.

Nr.		von Nr.
(1)	$\neg((\neg A \to (B \vee \neg C)) \to (C \to A))$ ✓	
(2)	$\neg A \to (B \vee \neg C)$ ✓	(1)
(3)	$\neg(C \to A)$ ✓	(1)
(4)	C ✓	(3)
(5)	$\neg A$ ✓	(3)
(6,7)	$\neg\neg A$ ✓ $B \vee \neg C$ ✓	(2,2)
(8)	A ✓	(6)
(9,10)	B ✓ $\neg C$ ✓	(2,2)

Der Nichtdeterminismus kann zur Effizienzsteigerung (Beschleunigung, Raumersparnis, Übersichtlichkeit) genutzt werden, z.B.:

- *Übergehe* bei der Wahl in (2) abgeschlossene Knoten, denn unerfüllbare Alternativen können in der DNF entfallen.
- *Wähle* Knoten mit ($\wedge$)-*Schritten* (Tab.-Reg. 1–5, ohne weitere Aufspaltung des Baumes) *vor* den Knoten mit ($\vee$)-*Schritten* (Tab.-Reg. 6–9) aus ...

Beweisprogramme arbeiten teils mit Resolutions-, teils mit Tableau-Methoden.

Komplexität von SAT

Womit gelangen wir am schnellsten zum Ergebnis? :

- Äquiv. DNF-Umformung + Inspektion
- Äquiv. KNF-Umformung + Resolution
- Wahrheitstafel + Inspektion
- Tableau
- (später noch: Deduktion)

Das hängt von den Formeln ab!

- DNF-nah oder KNF-nah?
- Formel lang oder kurz?
- Viele oder wenige Aussagevariablen?
- Sowie von den nichtdeterministischen Auswahlen und Reihenfolgen.

Alle 5 Methoden entscheiden zw. Erfüllbarkeit und Widersprüchlichkeit, also auch über (endliche) Folgerung und zw. Tautologie und Widerlegbarkeit (vgl. Reduktionsdiagramm).

Alle bekannten Algorithmen zur Entscheidung der Erfüllbarkeit (**SAT**) erfordern (bis zu) exponentiell wachsenden Zeit- bzw. Rechenaufwand in Abhängigkeit von der Komplexität der Formel. Das Problem SAT ist **NP-vollständig**, d.h. NP-komplex und „mindestens so komplex wie jedes NP-komplexe". NP-komplex bedeutet: es erfordert polynomialen Aufwand, die richtige Lösung zu *verifizieren*. Sie lässt sich aber evtl. nicht mit polynomialem Aufwand *bestimmen*.

Polynomialer Aufwand bedeutet: es existiert ein Algorithmus mit der Eigenschaft: Problemgröße = g $\Rightarrow$ Die Anzahl der Lösungsschritte ist geringer als ein bestimmtes Polynom n-ten Grades von g.

Wer beweist, dass SAT nicht mit polynomialem Aufwand zu lösen ist, beweist damit **P ≠ NP** und wird berühmt. Wer einen polynomial komplexen Algorithmus für SAT findet, beweist **P = NP** und wird auch berühmt – aber nicht unbedingt beliebt (kryptologische Katastrophe!). In beiden Fällen wird man Dollarmillionär:
http://www.claymath.org/millennium/P_vs_NP/

Logik-Programmierung: PROLOG-Stil

Logik-Programmierung verwendet Programmiersprachen, in denen wir

- uns Logik-nah ausdrücken,
- unser Wissen in Formeln eingeben und
- Fragen nach Formeln stellen

können.

AL-Prolog, Beispieldialog

Anfangszustand: *Wissen*:= ∅

Eingaben: `es_regnet.`
`es_regnet, kein_schirm -> werde_nass.`
für: *Wissen* := *Wissen* ∪ { *es_regnet*, *es_regnet* ∧ *kein_schirm* → *werde_nass* }
`werde_nass?`
für: (*Wissen* ⊨ *werde_nass*)?

Ausgabe: `nein.`
nicht für: *Wissen* ⊨ ¬ *werde_nass*
sondern für: *Wissen* ⊭ *werde_nass*

Eingabe: `kein_schirm.`
für: *Wissen* := *Wissen* ∪ { *kein_schirm*}

Eingabe: `werde_nass?`
Ausgabe: `yes.`
für: *Wissen* ⊨ *werde_nass*

Diese Logik-Programmierung …

- akzeptiert drei Arten von Eingaben:
  - Fakt (Aussagevariable**.**)
  - Regel (Konjunktion von Aussagevar'en → Aussagevariable**.**)
  - Frage/Ziel (Aussagevariable**?**)
- sammelt die eingegebenen Fakten und Regeln in einer Wissensbasis: Alle bisher eingegebenen Fakten und Regeln sollen wahr sein.
- beantwortet die Frage nach einer Auss.-variable *P*
  - mit **ja**, wenn aus der Wissensbasis *P* folgt;
  - mit **nein**, wenn aus der Wissensbasis (noch) nicht *P* folgt;

Die beschränkten Ausdrucksmittel erlauben eine besonders effiziente Beantwortung der Fragen (Folgerung aus Wissensbasis oder nicht?)

Einfache **Erweiterungsmöglichkeiten**:

- Die Prämisse (linke Seite) einer Regel darf Disjunktion von Konjunktionen von Aussagevariablen sein, denn z.B. ist $((A_1 \wedge A_2) \vee ((B_1 \wedge B_2 \wedge B_3)) \to C.$ ersetzbar durch $(A_1 \wedge A_2) \to C.$ $(B_1 \wedge B_2 \wedge B_3) \to C.$
  Vgl.: *Wenn es regnet oder jemand mit der Gießkanne gießt, ist die Erde nass.*
  Wenn es regnet, ist die Erde nass, und wenn jemand mit der Gießkanne gießt, dann auch.
- Die Konklusion (rechte Seite) einer Regel, sowie auch ein Ziel (Frage), darf Konjunktion von Aussagevariablen sein, denn z.B. ist $((A_1 \wedge A_2) \to C \wedge D.$ ersetzbar durch $(A_1 \wedge A_2) \to C.$ $(A_1 \wedge A_2) \to D.$
  Vgl.: *Wenn es regnet, ist die Erde nass und Regenwürmer kommen heraus.*
  Wenn es regnet, ist die Erde nass, und außerdem kommen dann Regenwürmer heraus.
  Und die Frage $E \wedge F?$ ist ersetzbar durch die Regel $(E \wedge F) \to G$ (G sonst unbenutzt) und die Frage $G?$

Markierungsalgorithmus

Gegeben sei eine Wissensbasis aus Fakten und Regeln, gefolgt von einer Frage (Ziel) $A_1 \wedge \ldots \wedge A_n$. Modelliere das Ziel als weitere Regel $A_1 \wedge \ldots \wedge A_n \rightarrow$ **ja** .

1. Schreibe eine Liste der vorkommenden Aussagevariablen plus **ja**. Jede von ihnen ist zunächst unmarkiert.
2. Markiere für jeden Fakt A_n. dessen Aussagevariable A_n.
3. Wenn **ja** markiert ist, so endet die Berechnung mit dem Ergebnis **JA**.
4. Wenn
   - eine Regel $A_1 \wedge \ldots \wedge A_n \rightarrow A_k$ /**ja** in der Wissensbasis steht, und
   - alle $A_1, ..., A_n$ sind markiert, und
   - A_k /**ja** ist noch nicht markiert,

   dann markiere auch A_k /**ja** und gehe nach 3.
5. (*Wenn also die Berechnung nicht in 3. stoppte und 4. nicht (mehr) anwendbar ist*:)
   Die Berechnung endet mit dem **Ergebnis NEIN**.

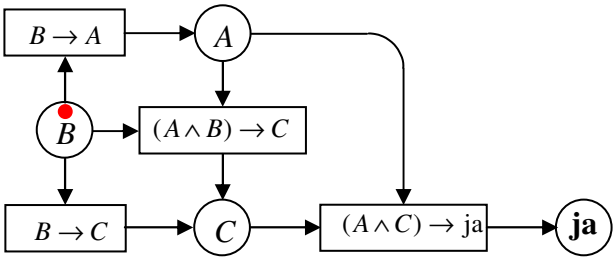
Satz: **Markierungsalgorithmus**
Der Markierungsalgorithmus **terminiert** und beantwortet die Frage, ob das Ziel eine **logische Folge** der bisherigen Wissensbasis – d.h. der eingegebenen Fakten und Regeln – ist.

Markierungsalgorithmus grafisches **Beispiel**:
$B \rightarrow A$. $(A \wedge B) \rightarrow C$. $B \rightarrow C$. B. $A \wedge C$?

Stellen ◯ modellieren Aussagevariablen und **ja**. Man markiert sie mit •, sobald man „sie weiß“ – anfangs B.

Jede Transition □ (1 pro Regel) markiert ihre Ausgangsstellen, sobald alle ihre Eingangsstellen markiert sind.

ja markiert? → Antwort **ja**.



Traditionelle Sichtweise: PROLOG prüfte, ob die Eingabenmenge

es_regnet
kein_schirm
$\neg$ *es_regnet* $\vee$ $\neg$ *kein_schirm* $\vee$ *werde_nass*
$\neg$ *werde_nass*

unerfüllbar ist. Die Antwort lautete richtig: Ja.

Jede Eingabe besteht aus Klauseln mit höchstens einem positiven Literal**,** sog. **Horn-Klauseln (Hoern-Logik)**:
Regeln: ein positives + mindestens ein negatives Literal
Beispiel: $\neg$ *es_regnet* $\vee$ $\neg$ *kein_schirm* $\vee$ *werde_nass*
als Implikation: *es_regnet* $\wedge$ *kein_schirm* $\rightarrow$ *werde_nass.*
Fakten: ein positives Literal, keine negativen
Beispiel: *es_regnet.*
als Implikation: $\top \rightarrow$ *es_regnet*
Fragen, Ziele: nur negative Literale
Beispiel: *werde_nass?* → $\neg$*werde_nass*
als Implikation: *werde_nass* $\rightarrow \bot$.
Vorteile der Horn-Logik: Der Markierungsalgorithmus erfordert nur polynomialen Aufwand. **Nachteil** der Horn-Logik: Nicht zu allen Formeln existiert eine äquivalente Horn-Klausel.
Gegenbeispiel: $A \vee B$!

Binary Decision Diagrams

Unser ITE-Algorithmus (→ Junktorenbasen!) liefert eine spezielle ITE-Formel, nämlich

- mit ausschließlich Konstanten in den THEN/ELSE-Blatt-Positionen,
- mit ausschließlich Variablen in den IF-Positionen,
- und alle Variablen werden in jedem Zweig des Formelbaumes in derselben (willkürlich gewählten) Reihenfolge abgefragt (**geordnet**).

Beispiel:

$(A \vee B) \rightarrow C$ (1) Sh-Exp. nach C

$\equiv C \rightarrow ((A \vee B) \rightarrow \top) \ / \ ((A \vee B) \rightarrow \bot))$ (2) Sh-Exp. nach B

$\equiv B \rightarrow \big(C \rightarrow ((A \vee \top) \rightarrow \top) / ((A \vee \top) \rightarrow \bot)\big) / \big(C \rightarrow ((A \vee \bot) \rightarrow \top) / ((A \vee \bot) \rightarrow \bot)\big)$ (3) Sh-Exp. nach A

$\equiv A \rightarrow (B \rightarrow (C \rightarrow \boxed{((\top \vee \top) \rightarrow \top)} \ / \ ...$ usw.

(4) konstante Terme auswerten

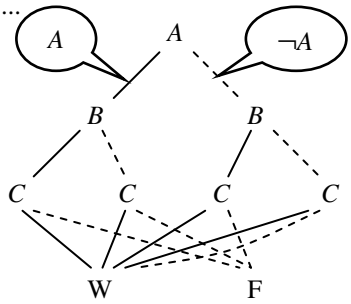
$\equiv A \rightarrow (B \rightarrow (C \rightarrow \boxed{\top}) / \ ...$ usw.

ITE-Form-Satz
Zu jeder Formel gibt es mindestens eine äquivalente (sogar **geordnete)** ITE-Formel.

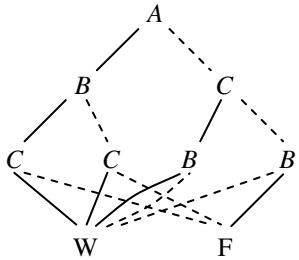
Darstellung als Baum*: ...

Ordnung:
hier A vor B vor C – müssen aber nicht in allen Zweigen alle vorkommen
(s.u.: ROBDD)

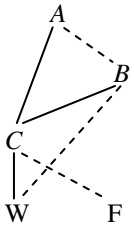
*) W-Blätter und F-Blätter jeweils übereinandergelegt.



ein **OBDD**,
Ordered Binary Decision Diagram



ein äquivalenter **BDD** (ungeordnet aber ohne Variablenwiederholung, -auslassung und „unmögliche Kanten“)



ein äquivalenter **reduzierter „R“OBDD**

Bei allen dreien ...

W-Fälle: ABC, A-BC, -ABC, -A-BC, -A-B-C
F-Fälle: AB-C, A-B-C, -AB-C

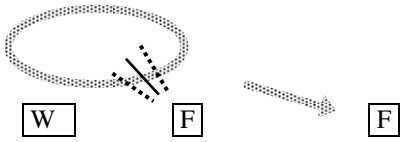
Ein (AL-) **BDD – Binary Decision Diagram** – ist ein Kanten- und Knoten-beschrifteter verwurzelter gerichteter zyklenfreier Graph *G* mit:

- Jeder terminale Knoten ist mit W oder mit F beschriftet;
- jeder nicht-terminale Knoten
  - ist mit einer Aussagevariablen beschriftet und
  - hat zwei Ausgangskanten, eine mit W eine mit F beschriftet,
- er ist (in der Regel) widerspruchsfrei:
  z.B nicht P–F–Q–W–W und Q–W–P–F–F.

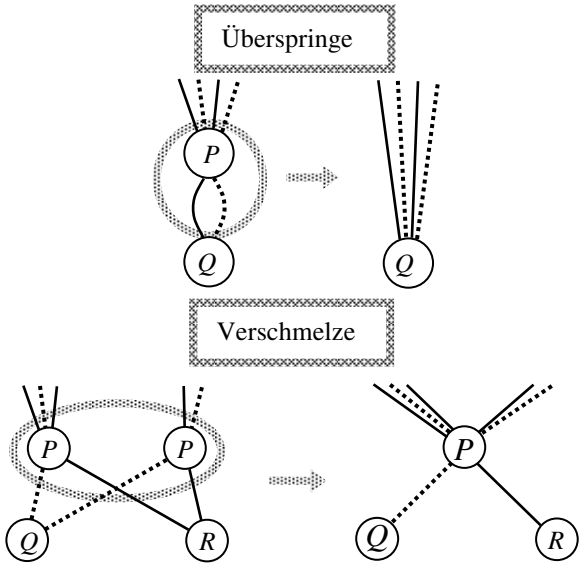
Zeichenkonventionen
Alle Kanten laufen abwärts, ohne Pfeilspitzen.
W-Kanten sind durchgehend und F-Kanten gestrichelt gezeichnet – als Ersatz für die Beschriftung.
Nichtterminalknoten werden oft durch Kreise, Terminalknoten durch Quadrate eingerahmt.
Ein BDD heißt **geordnet** – **Ordered Binary Decision Diagram, OBDD** – wenn ...
auf den Aussagevariablen eine lineare Ordnung π der Art existiert, dass für jede Kante von einem nichtterminalen Knoten mit Beschriftung P zu einem nichtterminalen Knoten mit Beschriftung Q gilt: $P <_\pi Q$.
Ein OBDD heißt **reduziert** – **Reduced** Ordered Binary Decision Diagram, **ROBDD**, wenn er die kleinstmögliche Knotenzahl (für diese Ordnung π !) hat.

Algorithmus RedOBDD: Reduktion eines OBDD

- Isolierter W/F-Knoten? Alles auf den anderen!



- Solange möglich:
  - Überspringe eine „folgenlose Fallunterscheidung“
  - Verschmelze zwei Knoten mit „gleicher Fallunterscheidung“:

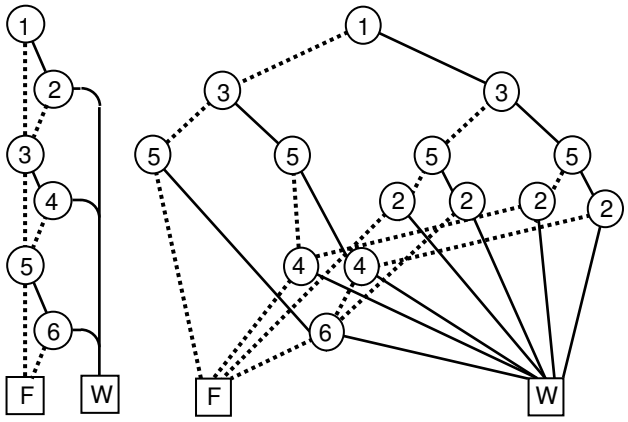


Satz: **Reduktion eines OBDD**
RedOBDD terminiert stets.
Die durch Umformung dann erhaltene Formel ist

- ein ROBDD,
- eindeutig (kanonisch) für Ordnung π und
- zum Ausgangs-OBDD **äquivalent**.

Satz: **ROBDD-Eigenschaften**
Äquivalent sind Formeln genau dann, wenn sie den gleichen ROBDD haben
Erfüllbar ist eine Formel genau dann, wenn im ROBDD der W-Knoten vorkommt.

ROBDDs als Datenstruktur erlauben oft rechnerisch *effiziente Darstellungen* von Formeln *und* effiziente *Berechnungen* von Erfüllbarkeit und Äquivalenz.
Man kann Junktoren auch direkt auf OBDDs anwenden, und man kann den Umgang mit OBDDs optimieren, z.B. durch Verwendung mehrerer Zeiger.
Nutzung: VLSI-CAD, Model Checking
SAT bleibt allgemein exponentiell aufwändig, wird aber in vielen prakt. Fällen mittels OBDDs machbar.
Die erreichte Effizienz ist stark Reihenfolge-abhängig, der Aufwand kann von linear bis exponentiell reichen, z.B. hat $\varphi = (A_1 \wedge A_2) \vee (A_3 \wedge A_4) \vee (A_5 \wedge A_6)$ je nach Reihenfolge als ROBDD …



Axiomatisch-deduktive Kalküle

In einem (korrekten) **axiomatisch-deduktiven AL-Kalkül** (*Ju*, *Ax*, *Reg*) verwendet man

- eine Junktorenbasis ***Ju*** –
  die Menge Form(*Ju*) aussagenlogischer Formeln über *Ju* ist „repräsentativ“ insofern, als mit ihr alle Wahrheitswerteverläufe erzeugbar sind bzw. zu jeder AL-Formel eine äquivalente *Ju*-Formel existiert,
- eine ausgewählte Menge ***Ax*** von AL-Tautologien über *Ju*, genannt **Axiome**, und
- eine ausgewählte Menge ***Reg*** von AL-Folgerungen aus endlichen Formelmengen $\{\varphi_1, ..., \varphi_n\} \models \psi$, allesamt *Ju*-Formeln, genannt **Inferenz-** oder **Ableitungsregeln**,

Anwendung eines Axioms ψ bedeutet:

Wahl einer Substitution $z = A_1, ..., A_k / \pi_1, ..., \pi_k$ und Aufnahme von $\psi_{[z]}$ in die Menge der abgeleiteten Formeln. Man schreibt auch $\vdash_\psi \psi_{[z]}$.

Anwendung einer Regel R: $\{\varphi_1, ..., \varphi_n\} \models \psi$ bedeutet: Wahl einer Substitution $z = A_1, ..., A_k / \pi_1, ..., \pi_k$, derart, dass $\varphi_{1[z]}, ..., \varphi_{n[z]}$ bereits abgeleitet sind, und Aufnahme von $\psi_{[z]}$ in die Menge der abgeleiteten Formeln. Man schreibt auch $\varphi_{1[z]}, ..., \varphi_{n[z]} \vdash_R \psi_{[z]}$.

Ist beispielsweise
$R: A \wedge B \models A \vee B$ und $z = A, B / C, D \to E$ und die Formel $C \wedge (D \to E)$ bereits abgeleitet, so erzeugt ein Ableitungsschritt mit R die Formel $C \vee (D \to E)$.

Axiome sind praktisch spezielle Regeln $\varnothing \models \psi$ mit leerer Menge von Prämissen. Wir reden daher jetzt oft einfach nur noch von **Regeln**.

Die wichtigste Inferenzregel ist der **Modus Ponens**
$\boxed{MP: A,\ A \to B \ \models\ B}$.

Den Kalkül nennt man **vollständig**, wenn es möglich ist, durch wiederholte Anwendung der Axiome und Regeln **alle Tautologien** über *Ju* und, bei Vorgabe einer Wissensbasis $M \subseteq$ Form(*Ju*), **alle Folgerungen** aus M (über *Ju*) abzuleiten.

Für die Menge der aus M **ableitbaren Formeln** gibt es zwei gleichwertige Beschreibungen:

Induktiv

φ ist **mittels** einer Menge ***Reg*** von Inferenzregeln **aus *M* ableitbar**, wenn:

- $\varphi \in M$, oder
- es existiert eine Menge M' aus M ableitbarer Formeln und eine Regel $R \in Reg$ so, dass φ mittels R (und Substitution) in einem Schritt aus M' ableitbar ist, $M' \vdash_R \varphi$.

Iterativ

Menge der **mittels *Reg* aus *M* ableitbar**en Formeln:

$$Abl^0_{Reg}(M) := M$$

$$Abl^{n+1}_{Reg}(M) := Abl^n_{Reg}(M) \cup \bigcup_{R \in Reg} \{\varphi \in Form \mid Abl^n_{Reg}(M) \vdash_R \varphi\}$$

$$Abl_{Reg}(M) := \bigcup_{n=1}^{\infty} Abl^n_{Reg}(M).$$

Eine **Ableitung/Beweis** einer Formel φ (aus M mit *Reg*) ist eine endliche Folge $\varphi_1, ..., \varphi_n$ mit

- $\varphi = \varphi_n$ und
- jedes φ_k in M oder aus davorstehenden Formeln unmittelbar ableitbar:
  $\varphi_k \in M \cup Abl^1_{Reg}(\{\varphi_1, ..., \varphi_{k-1}\})$.

Im Falle korrekter Kalküle (hier: aus echten Tautologien und gültigen Folgerungen) **beweist** die Ableitung, dass $M \models \varphi$.

Zwei korrekte und vollständige Kalkülbeispiele sind

Hilbert-Ackermann

Junktorenbasis
$\neg, \vee$
Axiome
$\neg(A \vee A) \vee A$
$\neg A \vee (A \vee B)$
$\neg(A \vee B) \vee (B \vee A)$
$\neg(\neg A \vee B) \vee (\neg(C \vee A) \vee (C \vee B))$
Inferenzregeln
Modifizierter Modus Ponens: $MMP = \dfrac{A, \neg A \vee B}{B}$

Mendelson

Junktorenbasis
$\neg, \to$
Axiome
$A \to (B \to A)$
$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
$(\neg B \to \neg A) \to ((\neg B \to A) \to B)$
Inferenzregeln
Modus Ponens

Mit Geschick reicht *ein* (recht langes) Axiom! Es gab schon „sportliche" Wettläufe um das jeweils kürzeste Einzelaxiom in einer Junktorenbasis.

Andere Axiomensysteme spendieren mehr Axiome:
Hilbert-Bernays, Kleene, de Swart, ...
Beweisbeispiele (= Ableitungen aus den Axiomen!)

$\boxed{A \to A}$ – im Mendelson-Kalkül

1. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$	Axiom
2. $(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$	1, $[B, C / A \to A, A]$
3. $A \to (B \to A)$	Axiom
4. $A \to ((A \to A) \to A)$	3, $[B / A \to A]$
5. $((A \to (A \to A)) \to (A \to A))$	2, 4, MP
6. $A \to (A \to A)$	3, $[B / A]$
7. $A \to A$	5, 6, MP

$\boxed{A \to A}$ (d.h. $\neg A \vee A$) – im Hilbert-Ackermann-Kalkül

1. $\neg(\neg A \vee B) \vee (\neg(C \vee A) \vee (C \vee B))$	Axiom
2. $\neg(\neg A \vee B) \vee (\neg(\neg C \vee A) \vee (\neg C \vee B))$	1, $C / \neg C$
3. $\neg A \vee (A \vee B)$	Axiom
4. $\neg A \vee (A \vee A)$	3, B / A
5. $\neg(\neg(A \vee A) \vee A) \vee (\neg(\neg A \vee (A \vee A)) \vee (\neg A \vee A))$	2, $A, B, C / A \vee A, A, A$
6. $\neg(\neg A \vee (A \vee A)) \vee (\neg A \vee A)$	4,5,MMP
7. $\neg A \vee A$	4,6,MMP

Beweise sind offenbar **leichter nachzuprüfen als zu finden**!

Kritische Fragen zur Behandlung der Logik

- Wozu Kalküle und Beweise? Wir haben doch die Wahrheitstafeln und sonstigen Algorithmen!

(1) Komplexität
Man beweist lieber $(A \wedge B \wedge C \wedge D \wedge E \wedge F \wedge G) \to A$ in wenigen Schritten im Kalkül als mittels Wahrheitstafel mit 128 Zeilen und 1792 Feldern.

(2) Berechenbarkeit
In der Prädikatenlogik handeln die „Formeln" von Eigenschaften von potentiell unendlich vielen verschiedenen Objekten. Die kann man nicht mehr alle in einer Tabelle erfassen und durchrechnen. Da gibt es (beweisbar!) keinen sicher terminierenden Algorithmus zur Überprüfung der Erfüllbarkeit etc. – wir sind auf im Kopf gefundene Beweise angewiesen.

- Wozu dann überhaupt diese ganzen Wahrheitstafeln und sonstigen Algorithmen?

In vielen praktischen Fragen gelingt es, das Ganze in Aussagenlogik auszudrücken. Und dann kann man den Computer die Arbeit erledigen lassen.

Natürliches Schließen und unser Werkzeugkasten

In den 1930er Jahren erweiterte Gerhard Gentzen die axiomatischen Kalküle gewissermaßen um das Deduktionstheorem und erhielt so seinen (formalen) **Kalkül des natürlichen Schließens**, der weitgehend das formlose Beweisen in Mathematikbüchern nachbildet.

Seine Beweise sind nicht reine Folgen von Formeln, wie im Vorigen, sondern haben eine **Blockstruktur** (wie Programmiersprachen): Man kann einen Block eröffnen, indem man eine neue Formel φ **annimmt** und sie im Block mitbenutzt, um eine Formel ψ abzuleiten. Dann schließt man den Block, wobei man festhält, dass (gemäß **Deduktionstheorem,** und zwar unter der Voraussetzung der bisher abgeleiteten Formeln, und der per Blockstruktur noch geltenden Annahmen) $\varphi \to \psi$ abgeleitet ist, ein sog. **bedingter Beweis**. Anschließend gilt die Annahme φ nicht mehr – sie ist auf den Block beschränkt!

Hier ist das sog. „Natürliche Schließen" um einige korrekte Schlussregeln zu einem logischen **Werkzeugkasten** erweitert, in Anlehnung an Gary Hardegree, http://courses.umass.edu/phil110-gmh/text/c05_3-99.pdf

Beispiel einer Werkzeugkasten-Ableitung:

	Zeige $(A \to B) \to ((B \to C) \to (A \to C))$	
1.	| $A \to B$	Ann
	| Zeige $(B \to C) \to (A \to C)$	
2.	| | $B \to C$	Ann
	| | Zeige $(A \to C)$	
3.	| | | A	Ann
4.	| | | B	MP,1,3
5.	| | | C	MP,2,4
6.	| | $A \to C$	BB
7.	| $(B \to C) \to (A \to C)$	BB
8.	$(A \to B) \to ((B \to C) \to (A \to C))$	BB

Eine **Werkzeugkasten-Ableitung** einer Formel φ aus einer endlichen Formelmenge M ist eine endliche **Folge mit Blockstruktur**.

- Die **Glieder dieser Folge** sind
  ohne Ableitungsbegründung
  - voran die gegebenen **Prämissen**$\in M$ (φ `Geg`), aus denen gefolgert werden soll (Bei Beweisen von Tautologien $M = \varnothing$.)
  - **Zielformeln** (`Zeige` φ), zwingend eine unmittelbar nach den Prämissen.
  - ad hoc **angenommene** Formeln (φ `Ann`)

  sowie jeweils **mit** Ableitungsbegründung
  - **abgeleitete** Formeln (φ Regel & Prämissennummer(n) oder – bei Erfüllungszeilen – der Beweistyp),

  alle (außer Zielformeln) jeweils durchnummeriert.
- Auf die Prämissen folgt der **Haupt-Block**.
- Ein **Block** beginnt mit einer **Zielzeile** (`Zeige` φ), dann kommt (eingerückt) der Blockkörper, am Ende (nicht mehr eingerückt) die abgeleitete Erfüllungszeile φ. Die letzte Formel im Blockkörper muss jeweils *abgeleitet* sein.
- Es gibt 4 **Blockschemata**, die Beweismethoden entsprechen, vgl. Tabelle.
- Im Blockkörper stehen *angenommene* Formeln und Erfüllungszeilen nur dort, wo im Blockschema gezeigt (ggf. entsprechend in Unterblöcken), abgeleitete Formeln und Unterblöcke (Zwischenbeweise) aber nach Belieben.

Wie wird beim Werkzeugkasten abgeleitet?

- per Blockschema
- per Ableitungsregeln, s. Tabelle. Die *Prämissen* müssen *davor* und im gleichen oder einem übergeordneten Blockkörper stehen (*vgl. Deklarationen in Programmiersprachen*).

Die 4 Beweis- (und Block-) Schemata im Werkz.-K.

Direkter Beweis:	Bedingter Beweis:
Zeige φ	Zeige $\varphi \to \psi$
| ⋮	| φ (Ann)
(| φ)	| ⋮
φ (DB)	| ψ
	$\varphi \to \psi$ (BB)

Indirekter Beweis 1:	Indirekter Beweis 2:
Zeige φ	Zeige $\neg\varphi$
| $\neg\varphi$ (Ann)	| φ (Ann)
| ⋮	| ⋮
| $\bot$	| $\bot$
φ (IB1)	$\neg\varphi$ (IB2)

Die Werkzeugkasten-Ableitungsregeln

Oder-Einführung	OE $\frac{\varphi}{\varphi \vee \psi}$, $\frac{\psi}{\varphi \vee \psi}$
Oder-Benutzung 1	OB_1 $\frac{\varphi \vee \psi, \neg\varphi}{\psi}$, $\frac{\varphi \vee \psi, \neg\psi}{\varphi}$
Oder-Benutzung 2	OB_2 $\frac{\varphi \vee \psi, \varphi \to \rho, \psi \to \rho}{\rho}$
Und-Einführung	UE $\frac{\varphi, \psi}{\varphi \wedge \psi}$
Und-Benutzung	UB $\frac{\varphi \wedge \psi}{\varphi}$, $\frac{\varphi \wedge \psi}{\psi}$
Doppelte-Negations-Einführung	DNE $\frac{\varphi}{\neg\neg\varphi}$
Doppelte-Negation-Benutzung	DNB $\frac{\neg\neg\varphi}{\varphi}$
Gdw-Einführung	GE $\frac{\varphi \to \psi, \psi \to \varphi}{\varphi \leftrightarrow \psi}$
Gdw-Benutzung links/ rechts	GB $\frac{\varphi \leftrightarrow \psi}{\varphi \to \psi}$, $\frac{\varphi \leftrightarrow \psi}{\psi \to \varphi}$
Widerspr. Einführung	WE $\frac{\varphi, \neg\varphi}{\bot}$
Widerspruch-Benutzung	WB $\frac{\bot}{\varphi}$

Folgerungsbenutzung 1, Modus Ponens	MP $\dfrac{\varphi, \varphi \to \psi}{\psi}$
Folgerungsben. 2, Modus Tollens	MT $\dfrac{\varphi \to \psi, \neg\psi}{\neg\varphi}$
Nicht-Und-Benutzung	NUB $\dfrac{\neg(\varphi \wedge \psi)}{\varphi \to \neg\psi}, \dfrac{\neg(\varphi \wedge \psi)}{\psi \to \neg\varphi}$
Nicht-Oder-Benutzung	NOB $\dfrac{\neg(\varphi \vee \psi)}{\neg\varphi}, \dfrac{\neg(\varphi \vee \psi)}{\neg\psi}$
Nicht-Folgerungs-Benutzung	NFB $\dfrac{\neg(\varphi \to \psi)}{\varphi \wedge \neg\psi}$
Nicht-Gdw-Benutzung	NGB $\dfrac{\neg(\varphi \leftrightarrow \psi)}{\neg\varphi \leftrightarrow \psi}, \dfrac{\neg(\varphi \leftrightarrow \psi)}{\varphi \leftrightarrow \neg\psi}$
Wiederholung	WDH $\dfrac{\varphi}{\varphi}$

Weitere Regeln sind nach Bedarf aus Äquivalenz- und Implikations-Tautologien herleitbar. Sinnvoll ist in der Praxis auch das Einfügen *anderswo bewiesene* Tautologien oder Folgerungen aus *M* (mit einer Quellenangabe an Stelle der **Ableitungsbegründung**, vgl. Mathematiktexte)

Beim direkten Beweis wird gerne die vorletzte Zeile des Schemas weggelassen (daher oben eingeklammert), aber dann ihre Begründung (Regel + verwendete Regelprämissen) an die Erfüllungszeile angehängt. Direkte Beweis-Blöcke dienen meist der freiwilligen Zerlegung des Beweises in Teilaufgaben.

Einige Beweisstrategien:

Zeige $\varphi \wedge \psi$		Zeige $\varphi \vee \psi$		Zeige $\varphi \leftrightarrow \psi$	
| Zeige φ		| $\neg(\varphi \vee \psi)$	AE	| Zeige $\varphi \to \psi$	
| | ⋮		| $\neg\varphi$	NOB	| | ⋮	
| φ	...	| $\neg\psi$	NOB	| $\varphi \to \psi$	
| Zeige ψ		| ⋮		| Zeige $\psi \to \varphi$	
| | ⋮		| $\bot$	...	| | ⋮	
| ψ	...	$\varphi \vee \psi$	IB1	| $\psi \to \varphi$	
$\varphi \wedge \psi$	UE			$\varphi \leftrightarrow \psi$	GE

Man beachte dass die Blockstruktur- und Sichtbarkeitsprinzipien denen bei Programmiersprachen entsprechen.

Satz: Der **Werkeugkasten für Aussagenlogik** ist **korrekt** und **vollständig**.

Der Werkeugkasten formalisiert die manuell üblichen Beweise und macht sie dadurch **maschinell überprüfbar**. Aber er führt (wie manuelle Beweisversuche) **nicht zwingend** zum Beweis einer korrekten Tautologie oder Folgerung.
Er verwendet implizit das **Deduktionstheorem**, und dies erlaubt **kürzere Beweise** als z.B. nur mit Axiomen und Modus Ponens.

Man vergleiche den folgenden Beweis für $A \to A$ mit dem im Mendelson-Kalkül:

$A \to A$ – mit Werkzeugkasten

	Zeige $A \to A$	
1.	| A	Ann
2.	| A	Wdh, 1
3.	$A \to A$	BB

Mathematisch interessante Sätze

Interpolationssätze der AL

Craigs Interpolationssatz für AL

Wenn für zwei Formeln φ , ψ gilt: $\models \varphi \to \psi$, dann kommt entweder eine Aussagevariable *P* sowohl in φ als auch in ψ vor, und dann existiert auch eine Formel π, deren sämtliche Aussagevariablen sowohl in φ als auch in ψ vorkommen, derart dass $\models \varphi \to \pi$ und $\models \pi \to \psi$.
oder sie haben keine Aussagevariable gemeinsam, und $\models \neg\varphi$ oder $\models \psi$.

Beweis (mit **Konstruktion** für ein π): Wikipedia (englisch) – Craig interpolation

Lyndons Interpolationssatz für AL

Wenn sich zwei Theorien *S*, *T* widersprechen, d.h. wenn für zwei Theorien *S*, *T* Theo($S \cup T$) unerfüllbar ist, dann existiert eine Formel φ , deren sämtliche Aussagevariablen sowohl in *S* als auch in *T* vorkommen und derart dass $S \models \varphi$ und $T \models \neg\varphi$ ... (+ weitere Eigenschaften von φ)

Kompaktheitssätze der AL

Eine Formelmenge ***M*** ist **erfüllbar** genau dann, wenn **jede endliche Teilmenge von *M* erfüllbar** ist.

Aus einer Formelmenge ***M*** **folgt** eine Formel ψ, $M \models \psi$, genau dann, wenn ψ bereits aus **einer endlichen Teilmenge** $N \subseteq M$ folgt: $N \models \psi$.

Ein möglicher Beweisweg der oberen Aussage für abzählbare *M* verwendet Königs Lemma. Für beliebige M aber verwendet man das **Lemma von Zorn**:

> In jeder nicht-leeren partiellen Ordnung, in der jede Kette nach oben beschränkt ist, existiert ein maximales Element.

Das geht aber nur in einer Mengenlehre mit **Auswahlaxiom**.