Logik

Prof. Dr. Madlener

TU Kaiserslautern

SS 2005

Logik

Studiengang „Informatik“, „Technoinformatik“und „ WiWi/Inf“
SS'05
Prof. Dr. Madlener TU - Kaiserslautern

Vorlesung:
Mi 11.45-13.15 52/207 **Achtung:** Mi 4.5 46/215

- Informationen
  http://www-madlener.informatik.uni-kl.de/teaching/ss2005/logik/logik.html
- Grundlage der Vorlesung: Skript
  Einführung in die Logik und Korrektheit von Programmen.

- Bewertungsverfahren:
  Übungen:
  Aufsichtsarbeit: max. 30 Punkte,
  Abschlussklausur: max. 70 Punkte.
- Übungen: Hörsaalübung Montag 10:00–11:30 1/106
  Sprechzeiten siehe Homepage

Grundlagen der Aussagenlogik

- Syntax
- Semantik
- Deduktiver Aufbau der Aussagenlogik
- Natürliche Kalküle

Algorithmischer Aufbau der Aussagenlogik

- Semantische Tableaux
- Normalformen
- Davis-Putman-Algorithmen
- Resolutions-Verfahren

Grundlagen der Prädikatenlogik

- Beziehungen zwischen Eigenschaften von Elementen
- Semantik der P-Logik 2-Stufe – Interpretationen, Belegungen, Bewert
- Transformationen von Termen und Formeln

Entscheidbarkeit in der Prädikatenlogik

- Unentscheidbarkeit der Allgemeingültigkeit
- Hauptsätze der Prädikatenlogik erster Stufe
- Theorien erster Stufe

Einleitung

Methoden zur Lösung von Problemen mit Hilfe von Rechnern Formalisierung ($\equiv$ Festlegung)

- **Logik::** „Lehre vom folgenrichtigen Schließen“ bzw. „Lehre von formalen Beziehungen zwischen Denkinhalten“

Zentrale Fragen: Wahrheit und Beweisbarkeit von Aussagen $\rightsquigarrow$ **Mathematische Logik**.

- **Logik in der Informatik**:
  - **Aussagenlogik**: Boolsche Algebra. Logische Schaltkreise (Kontrollsystemen), Schaltungen, Optimierung.
  - **Prädikatenlogik**: Spezifikation und Verifikation von Softwaresystemen.
  - **Modal- und Temporallogik**: Spezifikation und Verifikation reaktiver Systeme.

Logik in der Informatik

1. Semantik von Programmiersprachen (Hoarscher Kalkül).
2. Spezifikation von funktionalen Eigenschaften.
3. Verifikationsprozess bei der SW-Entwicklung.
   Beweise von Programmeigenschaften.
4. Spezielle Programmiersprachen (z.B. PROLOG)

- **Automatisierung des logischen Schließens**
  1. Automatisches Beweisen (Verfahren,...)
  2. Grundlagen von Informationsystemen (Verarbeitung von Wissen, Reasoning,...)

Voraussetzungen

1. **Mathematische Grundlagen**. Mengen, Relationen, Funktionen. Übliche Formalisierungen: „Mathematische Beweise“, Mathematische Sprache, d.h. Gebrauch und Bedeutung der üblichen Operatoren der naiven Logik. Also Bedeutung von **nicht, und, oder, impliziert, äquivalent, es gibt, für alle**.
2. **Grundlagen zur Beschreibung formaler Sprachen**. Grammatiken oder allgemeiner **Kalküle** (Objektmenge und Regeln zur Erzeugung neuer Objekte aus bereits konstruierter Objekte), Erzeugung von Mengen, Relationen und Funktionen, Hüllenoperatoren (Abschluss von Mengen bzgl. Relationen).
3. **Vorstellung von Berechenbarkeit**, d.h. entscheidbare und rek.aufzählbare Mengen, Existenz nicht entscheidbarer Mengen und nicht berechenbarer Funktionen.

Berechnungsmodelle/Programmiersprachen

Algorithmische Unlösbarkeit?

prinzipielle Lösbarkeit

↓

effiziente Lösbarkeit

↓

algorithmischer Entwurf

↓

P: Programm in einer HPS

Problem
Spezifikation

(Formalisiert)

Syntaktische und semantische Verifikation von P.

- **Syntaxanalyse**
  Sprachen Chomski-Hierarchie
  Kontext freie Sprachen
  Grammatiken/Erzeugungsprozess

- **Programmverifikation**
  Tut P auch was erwartet wird.
  Gilt $P \rightsquigarrow$ Problem Spezifikation

Typische Ausdrücke

- $(x+1)(y-2)/5$ Terme als Bezeichner von Objekten.
- $3+2=5$ Gleichungen als spezielle Formeln.
- „29 ist (k)eine Primzahl “ Aussagen.
- „$3+2=5$ und 29 ist keine Primzahl “ Aussage.
- „wenn 29 keine Primzahl ist, dann ist $0=1$ “ Aussage.
- „jede gerade Zahl, die größer als 2 ist, ist die Summe zweier Primzahlen “ Aussage.
- $2 \leq x$ und $(\forall y \in \mathbb{N})$
  $((2 \leq y \text{ und } y+1 \leq x) \rightarrow \text{nicht}(\exists z \in \mathbb{N}) y * z = x)$
  Aussage.

Typische Ausdrücke (Fort.)

- $(\forall X \subseteq \mathbb{N})(0 \in X \wedge (\forall x \in \mathbb{N})(x \in X \rightarrow x + 1 \in X) \rightarrow X = \mathbb{N})$ **Induktionsprinzip**.
- $(\forall X \subseteq \mathbb{N})(X \neq \emptyset \rightarrow X$ hat ein kleinstes Element)
  Jede nichtleere Menge natürlicher Zahlen enthält ein minimales Element.

Zweiwertige Logik Jede Aussage ist entweder **wahr** oder **falsch**.

- Es gibt auch andere Möglichkeiten (Mehrwertige Logik).
- Prädikatenlogik erster Stufe (PL1): Nur Eigenschaften von Elementen und Quantifizierung von Elementvariablen erlaubt.

Kapitel I

Grundlagen der Aussagenlogik

Aussagenlogik

- Aussagen $\rightsquigarrow$ Bedeutung **wahr**(1), **falsch** (0)
- Aufbau von Aussagen $\rightsquigarrow$ **Syntax**.
- Bedeutung von Aussagen $\rightsquigarrow$ **Semantik**.

Syntax

Definition 1.1 (Syntax)

Die Sprache der Aussagenlogik
Sei $\Sigma = V \cup O \cup K$ Alphabet mit $V = \{p_1, p_2, ...\}$ abzählbare Menge von **Aussagevariablen**, $O = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, ...\}$ Menge von **Verknüpfungen** mit Stelligkeiten (Junktoren, Operatoren) und $K = \{(,)\}$ **Klammern** Hilfssymbole.
Die Menge der **Aussageformen** (Formeln der Aussagenlogik) $F \subset \Sigma^*$ wird induktiv definiert durch:

1. $V \subset F$ Menge der **atomaren Aussagen**
2. $A, B \in F \quad$ so $(\neg A), (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B) \in F$
3. F ist die kleinste Menge die V enthält und 2. erfüllt (Hüllenoperator)

Bemerkung 1.2

- *Eigenschaften von Elementen in F werden durch* **strukturelle Induktion**, *d.h. durch Induktion über den Aufbau der Formeln, nachgewiesen.*
- *Beispiele für Eigenschaften sind:*
  1. *Für $A \in F$ gilt: A ist atomar (ein p_i) oder beginnt mit „(“ und endet mit „)“.*
  2. *Sei $f(A,i) = \sharp$ „(“$- \sharp$ „)“ in den ersten i Buchstaben von A, dann gilt $f(A,i) > 0$ für $1 \leq i < |A|$ und $f(A,i) = 0$ für $i = |A|$.*
- *F kann als Erzeugnis einer Relation $R \subset U^* \times U$ mit $U = \Sigma^*$ oder eines Kalküls dargestellt werden. Dabei wird F frei von dieser Relation erzeugt, da für alle $u, v \in U^*$ und $A \in F$ gilt: uRA und vRA so $u = v$.*
- *$F = L(G)$ für eine eindeutige kontextfreie Grammatik G.*

Satz 1.3 (Eindeutigkeitssatz)

Jede A-Form $A \in F$ ist entweder atomar oder sie lässt sich eindeutig darstellen als
$A \equiv (\neg A_1)$ *oder* $A \equiv (A_1 * A_2)$ *mit* $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ *wobei* $A_1, A_2 \in F$.
Beweis: Induktion über Aufbau von F.

Vereinbarungen

Schreibweisen, Abkürzungen, Prioritäten.

- Äußere Klammern weglassen.
- A-Formen sind z.B.: p_1, p_{101},
  $p_1 \vee p_{12}$ als Abkürzung für $(p_1 \vee p_{12})$,
  $(((p_1 \to p_2) \wedge (\neg p_2)) \to (\neg p_1))$, $p_1 \vee (\neg p_1)$
- Zur besseren Lesbarkeit: Prioritäten: $\neg, \wedge, \vee, \to, \leftrightarrow$ d.h.
  $A \wedge B \to C$ steht für $((A \wedge B) \to C)$
  $A \vee B \wedge C$ steht für $(A \vee (B \wedge C))$
  $\neg A \vee B \wedge C$ steht für $((\neg A) \vee (B \wedge C))$
  $A \vee B \vee C$ steht für $((A \vee B) \vee C)$ (Linksklammerung).
- Andere Möglichkeiten. „Präfix“- oder „Suffix“- Notation
  Für $(A * B)$ schreibe $*AB$ und für $(\neg A)$ schreibe $\neg A$

Semantik

Definition 1.4 (Bewertung, Belegung)

Eine **Bewertung** der A-Formen ist eine Funktion $\varphi : F \to \{0,1\} = \mathbb{B}$ mit

1. $\varphi(\neg A) = 1 - \varphi(A)$
2. $\varphi(A \vee B) = max(\varphi(A), \varphi(B))$
3. $\varphi(A \wedge B) = min(\varphi(A), \varphi(B))$
4. $\varphi(A \to B) = \begin{cases} 0 & \text{falls } \varphi(A) = 1 \text{ und } \varphi(B) = 0 \\ 1 & \text{sonst} \end{cases}$
5. $\varphi(A \leftrightarrow B) = \begin{cases} 0 & \text{falls } \varphi(A) \neq \varphi(B) \\ 1 & \text{falls } \varphi(A) = \varphi(B) \end{cases}$

Belegungen und Bewertungen (Fort.)

- Sprechweise: A ist „falsch“ unter φ, falls $\varphi(A) = 0$
  A ist „wahr“ unter φ oder φ „erfüllt“ A, falls $\varphi(A) = 1$.
- Darstellung von Bewertungen durch **Wahrheitstafeln**:

A	$\neg A$
1	0
0	1

A	B	$A \vee B$	$A \wedge B$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	0	0	1	1
0	1	1	0	1	0
1	0	1	0	0	0
1	1	1	1	1	1

Belegungen und Bewertungen (Fort.)

- Eine **Belegung** der A-Variablen V ist eine Funktion $\psi : V \to \mathbb{B}$.
- Offenbar induziert jede Bewertung eine Eindeutige Belegung durch $\psi(p_i) := \varphi(p_i)$.

Lemma 1.5

Jede Belegung $\psi : V \to \mathbb{B}$ lasst sich auf genau eine Weise zu einer Bewertung $\varphi : F \to \mathbb{B}$ fortsetzen. Insbesondere wird jede Bewertung durch die Werte auf V eindeutig festgelegt.

Bewertungen

Folgerung 1.6

Die Bewertung einer Aussageform $A \in F$ hängt nur von den Werten der in ihr vorkommenden Aussagevariablen aus V ab. Das heißt, will man $\varphi(A)$ berechnen, genügt es, die Werte $\varphi(p)$ zu kennen für alle Aussagevariablen p, die in A vorkommen.

- Beispiel: Sei $\varphi(p) = 1, \varphi(q) = 1, \varphi(r) = 0$. Dann kann $\varphi(A)$ iterativ berechnet werden:

$$A \equiv \underbrace{\underbrace{\underbrace{(\underbrace{p}_{1} \to \underbrace{(\underbrace{q}_{1} \to \underbrace{r}_{0})}_{0}}_{0} \to \underbrace{\underbrace{(\underbrace{p}_{1} \wedge \underbrace{q}_{1})}_{1} \to \underbrace{r}_{0}}_{0}}_{1}$$

Also gilt $\varphi(A) = 1$.

- **Frage** Welche Werte kann $\varphi(A)$ annehmen, wenn φ alle Belegungen durchläuft. Ist etwa $\varphi(A) = 1$ für alle Belegungen φ? Um das nachzuprüfen, „genügt“ es, die endlich vielen unterschiedlichen Belegungen der Variablen, die in A vorkommen, zu überprüfen. Kommen n Variablen in A vor, so gibt es 2^n verschiedene Belegungen.

  A definiert eine Boolesche Funktion $f_A : \mathbb{B}^n \to \mathbb{B}$.

- Beispiel: Für die drei Variablen p, q und r aus A im obigen Beispiel gibt es 8 Belegungen, die betrachtet werden müssen.

Bewertungen

p	q	r	$q \to r$	$p \wedge q$	$p \to (q \to r)$	$(p \wedge q) \to r$	A
0	0	0	1	0	1	1	1
0	0	1	1	0	1	1	1
0	1	0	0	0	1	1	1
0	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1
1	0	1	1	0	1	1	1
1	1	0	0	1	0	0	1
1	1	1	1	1	1	1	1

A ist „wahr“ unabhängig von den Werten von p, q, r, d.h. für jede Bewertung φ. Weitere solche Formeln sind etwa:
$(A \to (B \to A))$, $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ oder $((\neg A \to \neg B) \to (B \to A))$.

Wichtige Begriffe

Definition 1.7

Sei $A \in F$, $\Sigma \subseteq F$.

1.(a) A heißt **Tautologie** (**allgemeingültig**), falls $\varphi(A) = 1$ für jede Bewertung φ gilt. (Schreibweise "$\models A$")

(b) A ist **erfüllbar**, falls es eine Bewertung φ gibt, mit $\varphi(A) = 1$.

(c) A ist **widerspruchsvoll**, falls $\varphi(A) = 0$ für jede Bewertung φ.

(d) Schreibe $\mathrm{Taut} = \{A \mid A \in F \text{ ist Tautologie}\}$, die Menge der Tautologien oder „Theoreme" der Aussagenlogik, bzw. $\mathrm{Sat} := \{A \mid A \in F \text{ und } A \text{ ist erfüllbar}\}$ die Menge der erfüllbaren Formeln.

Definition (Fort.)

2.(a) Σ ist **erfüllbar**, falls es eine Bewertung φ gibt mit $\varphi(A) = 1$ für alle $A \in \Sigma$. ("φ **erfüllt** Σ")

(b) **Semantischer Folgerungsbegriff:** A ist **logische Folgerung** von Σ, falls $\varphi(A) = 1$ für jede Bewertung φ, die Σ erfüllt.

Man schreibt "$\Sigma \models A$".
Ist $\Sigma = \{A_1, \ldots, A_n\}$, ist die Kurzschreibweise "$A_1, \ldots, A_n \models A$" üblich.

(c) Die Menge Folg(Σ) der Folgerungen aus Σ ist definiert durch:

$$\text{Folg}(\Sigma) := \{A \mid A \in F \text{ und } \Sigma \models A\}.$$

Einfache Folgerungen

Bemerkung 1.8

Beispiele

1. $(p \vee (\neg p))$, $((p \to q) \vee (q \to r))$, $p \to (q \to p)$, $(p \to p)$, $(p \to \neg\neg p)$ *und A aus Folgerung 1.6 sind Tautologien.*

   $(p \wedge (\neg p))$ *ist widerspruchsvoll.*
   $A \in$ Taut *gdw* $\neg A$ *widerspruchsvoll*

   $(p \wedge q)$ *ist erfüllbar jedoch keine Tautologie und nicht widerspruchsvoll.*

   Die Mengen Taut, Sat *sind entscheidbar.*

   Beachte Taut $\subset$ Sat.

Bemerkung (Fort.)

2.(a) *Sei* $\Sigma = \{p\}$ *und* $A = p \vee q$. *Dann gilt* $\Sigma \models A$, *denn falls* $\varphi(p) = 1$, *dann auch* $\varphi(p \vee q) = 1$. *Jede Bewertung, die* Σ *erfüllt, erfüllt also auch* A.

(b) *Ist* $\Sigma = \emptyset$, *dann gilt* $\Sigma \models A$ *genau dann, wenn* A *Tautologie ist, d.h.* $\mathrm{Folg}(\emptyset) = \mathrm{Taut}$.

(c) *Ist* Σ *nicht erfüllbar, dann gilt* $\Sigma \models A$ *für alle* $A \in F$, *d.h.* $\mathrm{Folg}(\Sigma) = F$. *Insbesondere* $\Sigma \models A, \neg A$ *für ein* A.

(d) *Sei* $\Sigma \subseteq \Sigma'$. *Ist* Σ' *erfüllbar, dann ist auch* Σ *erfüllbar.*

(e) *Es gilt* $\Sigma \subseteq \mathrm{Folg}(\Sigma)$ *und* $\mathrm{Folg}(\mathrm{Folg}(\Sigma)) = \mathrm{Folg}(\Sigma)$.

(f) *Falls* $\Sigma \subseteq \Sigma'$, *dann gilt* $\mathrm{Folg}(\Sigma) \subseteq \mathrm{Folg}(\Sigma')$.

3. $\Sigma \models A$ *gilt genau dann, wenn* $\Sigma \cup \{\neg A\}$ *nicht erfüllbar.*
Ist Σ *endlich, dann ist es entscheidbar, ob* Σ *erfüllbar ist, und die Menge* $\mathrm{Folg}(\Sigma)$ *ist entscheidbar.*

Deduktionstheorem und Modus-Ponens Regel

Lemma 1.9

a) **Deduktionstheorem**:

$$\Sigma, A \models B \qquad \text{gdw} \qquad \Sigma \models (A \to B).$$

(Σ, A ist Kurzschreibweise für $\Sigma \cup \{A\}$)

b) **Modus-Ponens-Regel**:
Es gilt $\{A, A \to B\} \models B$.

Insbesondere ist B eine Tautologie, falls A und $(A \to B)$ Tautologien sind.

Übliche Notationen für Regeln der Form "$A_1, \ldots, A_n \models B$" sind:

$$\frac{\begin{array}{c} A_1 \\ \vdots \\ A_n \end{array}}{B} \quad \text{und} \quad \frac{A_1, \ldots, A_n}{B}$$

Für die Modus Ponens Regel also:

$$\frac{A, (A \to B)}{B} \ (\mathbf{MP})$$

Kompaktheitssatz der Aussagenlogik

Satz 1.10 (Kompaktheitssatz)

$\Sigma \subseteq F$ ist erfüllbar genau dann, wenn jede endliche Teilmenge von Σ erfüllbar ist.

$\Sigma \subseteq F$ ist unerfüllbar genau dann, wenn es eine unerfüllbare endliche Teilmenge von Σ gibt.

Insbesondere gilt $\Sigma \models A$ genau dann, wenn es eine endliche Teilmenge $\Sigma_0 \subseteq \Sigma$ gibt mit $\Sigma_0 \models A$.

Anwendungen Kompaktheitssatz

Beispiel 1.11

Sei $\Sigma \subseteq F$. Gibt es zu jeder Bewertung φ ein $A \in \Sigma$ mit $\varphi(A) = 1$, so gibt es $A_1, ..., A_n \in \Sigma$ $(n > 0)$ mit $\models A_1 \vee ... \vee A_n$.

- Betrachte die Menge $\Sigma' = \{\neg A \mid A \in \Sigma\}$, nach Voraussetzung ist sie unerfüllbar. Also gibt es eine endliche nichtleere Teilmenge $\{\neg A_1, ..., \neg A_n\}$ von Σ' die unerfüllbar ist. Also gilt für jede Bewertung φ gibt es ein i mit $\varphi(\neg A_i) = 0$ oder $\varphi(A_i) = 1$ und somit $\varphi(A_1 \vee ... \vee A_n) = 1$.

- Der zweite Teil des Satzes ist die Grundlage für Beweisverfahren für $\Sigma \models A$. Dies ist der Fall wenn $\Sigma \cup \{\neg A\}$ unerfüllbar ist.
  Widerspruchbeweise versuchen systematisch eine endliche Menge $\Sigma_0 \subset \Sigma$ zu finden, so dass $\Sigma_0 \cup \{\neg A\}$ unerfüllbar ist.

Logische Äquivalenz

Definition 1.12 (Logische Äquivalenz)

Seien $A, B \in F$ heißen **logisch äquivalent** mit der Schreibweise $A \models\!\dashv B$, falls für jede Bewertung φ gilt: $\varphi(A) = \varphi(B)$.

Insbesondere ist dann $A \models B$ und $B \models A$.

- Einige Beispiele für logisch äquivalente Formeln:
  1. $A \models\!\dashv \neg(\neg A)$, $\quad A \models\!\dashv A \vee A$, $\quad A \models\!\dashv A \wedge A$
  2. $A \wedge B \models\!\dashv B \wedge A$ und $A \vee B \models\!\dashv B \vee A$,
  3. $A \wedge (B \wedge C) \models\!\dashv (A \wedge B) \wedge C$ und $A \vee (B \vee C) \models\!\dashv (A \vee B) \vee C$,
  4. $A \wedge (B \vee C) \models\!\dashv (A \wedge B) \vee (A \wedge C)$ und
     $A \vee (B \wedge C) \models\!\dashv (A \vee B) \wedge (A \vee C)$ (**Distributiv**)

Logische Äquivalenz (Fort.)

5. $\neg(A \wedge B) \models\!\!\models (\neg A) \vee (\neg B)$ und $\neg(A \vee B) \models\!\!\models (\neg A) \wedge (\neg B)$ (**De Morgan**)
6. $A \to B \models\!\!\models (\neg A) \vee B$,
   $A \wedge B \models\!\!\models \neg(A \to (\neg B))$ und
   $A \vee B \models\!\!\models (\neg A) \to B$.
7. $A \leftrightarrow B \models\!\!\models (A \to B) \wedge (B \to A)$

- Man beachte, dass $\models\!\!\models$ reflexiv, transitiv und symmetrisch, d.h. eine Äquivalenzrelation ist.
- Ersetzt man in einer Formel eine Teilformel durch eine logisch äquivalente Formel, so erhält man eine logisch äquivalente Formel.

Logische Äquivalenz (Fort.)

- Folgende Aussagen sind äquivalent:
  - $\models (A \leftrightarrow B)$
  - $A \models B$ und $B \models A$
  - $A \models\!\models B$
  - $\mathrm{Folg}(A) = \mathrm{Folg}(B)$

Folgerung 1.13

Zu jedem $A \in F$ gibt es $B, C, D \in F$ mit

1. $A \models\!\models B$, *B enthält nur $\rightarrow$ und $\neg$ als log. Verknüpfungen*
2. $A \models\!\models C$, *C enthält nur $\wedge$ und $\neg$ als log. Verknüpfungen*
3. $A \models\!\models D$, *D enthält nur $\vee$ und $\neg$ als log. Verknüpfungen*

Logische Äquivalenz (Fort.)

Definition 1.14 (Vollständige Operatorenmengen)

Eine Menge $OP \subseteq \{\neg, \vee, \wedge, \rightarrow, \leftrightarrow, ..\}$ heißt **vollständig**, falls es zu jedem $A \in F$ eine logisch äquivalente A-Form $B \in F(OP)$ gibt.

- Vollständige Operatorenmengen für die Aussagenlogik sind z.B.:
  $\{\neg, \rightarrow\}, \{\neg, \vee\}, \{\neg, \wedge\}, \{\neg, \vee, \wedge\}, \{\mathsf{false}, \rightarrow\}$
- Dabei ist false eine Konstante, mit $\varphi(\mathsf{false}) = 0$ für jede Bewertung φ. Offenbar gilt $\neg A \models\!\!\!\not\models (A \rightarrow \mathsf{false})$.
- **Normalformen**:: DNF (Disjunktive Normalform), KNF (Konjunktive Normalform), KDNF, KKNF (Kanonische Formen).

Boolsche Funktionen

Jede Aussageform $A(p_1, ..., p_n)$ stellt in natürlicher Form eine Boolsche Funktion $f_A : \mathbb{B}^n \to \mathbb{B}$ dar. Nämlich durch die Festlegung $f_A(b_1, ..., b_n) = \varphi_{\vec{b}}(A)$ mit der Bewertung $\varphi_{\vec{b}}(p_i) = b_i$

- Man kann leicht durch Induktion nach n zeigen, dass jede Boolsche Funktion $f : \mathbb{B}^n \to \mathbb{B}$ $(n > 0)$ sich in obiger Form durch eine Aussageform in $p_1, ..., p_n$ und einer vollständigen Operatorenmenge darstellen lässt.
- Die Boolsche Algebra über $\mathbb{B}$ hat als übliche Operatormenge **true, false, not, or, and** mit der standard Interpretation.
- Für andere Operatormengen die etwa **nand, nor** enthalten, siehe Digitale Logik
  (Gatter: Ein- Ausgabesignale, Verzögerung. **nand, nor** Gattern werden bevorzugt, da nur zwei Transistoren nötig).

Beispiel Patientenüberwachungssystem

Beispiel Patientenüberwachungssystem erhält gewisse Daten über den Zustand eines Patienten. Z.B. Temperatur, Blutdruck, Pulsrate. Die Schwellenwerte für die Daten seien etwa wie folgt festgelegt:

Zustände	
Ein/ Ausgaben	Bedeutung
A	Temperatur außerhalb 36-39°C.
B	Blutdruck außerhalb 80-160 mm.
C	Pulsrate außerhalb 60-120 Schläge pro min.
O	Alarmaktivierung ist notwendig.

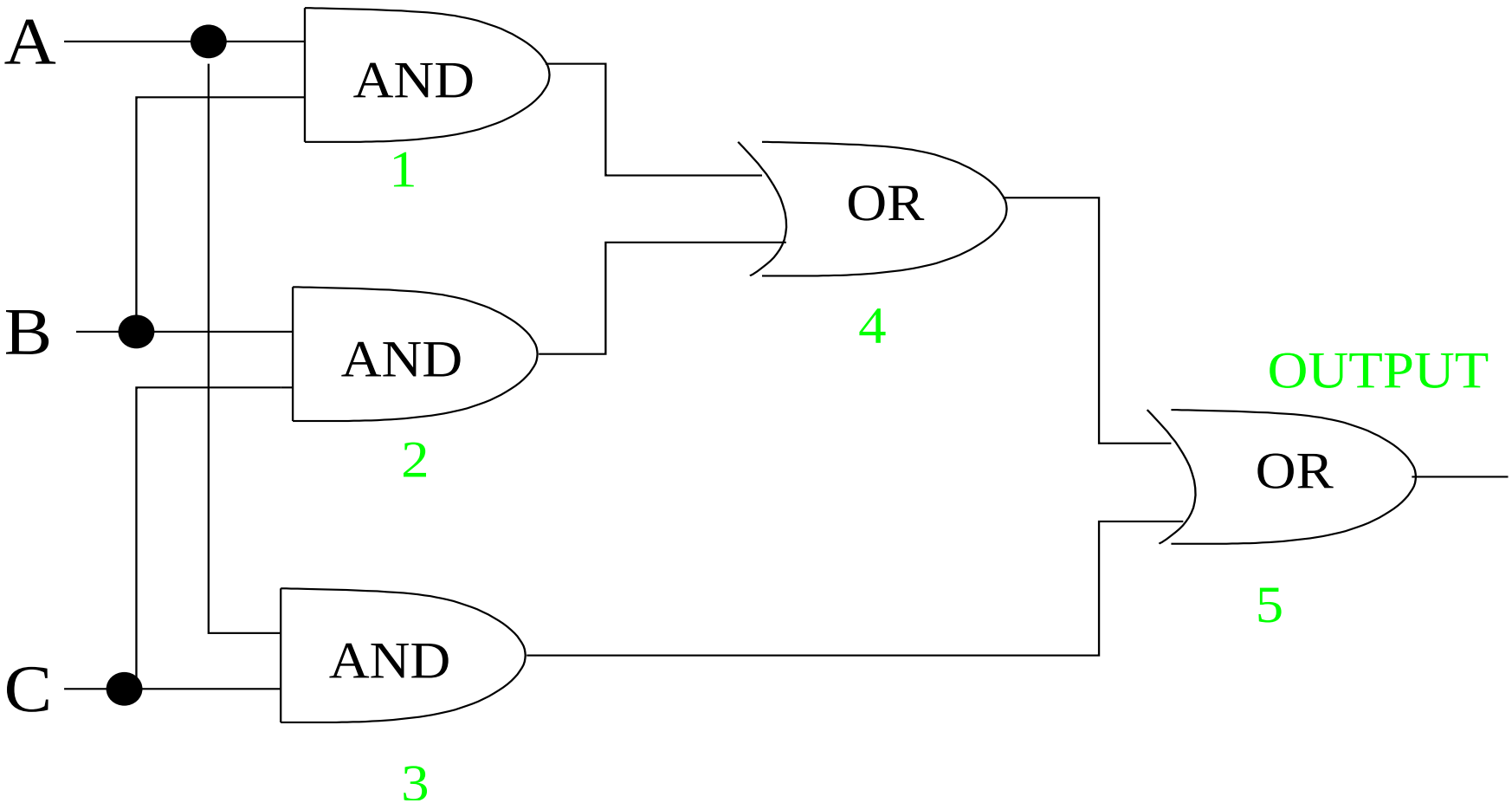
Die Anforderungen, d.h. bei welchen Kombinationen der Werte der Zustände eine Alarmaktivierung notwendig ist, werden durch den Medizin-Experten festgelegt. Sie seien in folgender Tabelle fixiert.

I/O - **Tabelle**

A	B	C	O
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Logischer Entwurf: Betrachte die Zeilen in denen O den Wert 1 hat und stelle eine KDNF auf (Disjunktion von Konjunktionen von Literalen, wobei ein Literal eine atomare Form oder die Negation einer solchen ist).
$(\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C)$

Als eine Realisierung könnte man das folgende Schaltnetz nehmen:



Deduktiver Aufbau der Aussagenlogik

Dieser Abschnitt beschäftigt sich mit einem axiomatischen Aufbau der Aussagenlogik mittels eines **"Deduktiven Systems"** oder eines **„Kalküls"**.
Eine syntaktisch korrekte Formel in einem Deduktiven System wird "Theorem" genannt, wenn sie durch rein mechanische Anwendungen der Regeln des Systems aus den Axiomen des Systems "abgeleitet" werden kann.
Man kann mehrere deduktive Systeme angeben, in denen aussagenlogische Formeln genau dann Theoreme sind, wenn sie auch Tautologien sind.

Deduktive Systeme-Kalküle

Definition 1.15 (Deduktives System)

Ein **Deduktives System** $\mathcal{F} = \mathcal{F}(\mathrm{Ax}, R)$ besteht aus

- einem Alphabet Δ (hier $\Delta = V \cup K \cup \{\rightarrow, \neg\}$),
- $F \subseteq \Delta^\star$, einer Menge von (wohldefinierten) Formeln (hier die Aussageformen),
- $\mathrm{Ax} \subseteq F$, einer Menge von Axiomen und
- R, einer Menge von Regeln der Form $\frac{A_1, \ldots, A_n}{A}$ $(n \in \mathbb{N}^+)$. $(A_1, ..., A_n, A \in F)$

Die Mengen F, Ax und R sind im allgemeinen rekursiv entscheidbar.

Deduktive Systeme-Kalküle

- Die Menge $T = T(\mathcal{F})$ der **Theoreme** ist definiert durch:
  1. $Ax \in T$ (d.h. alle Axiome sind Theoreme)
  2. Sind $A_1, \ldots, A_n \in T$ und ist die Regel $\frac{A_1, \ldots, A_n}{A}$ in R, dann ist $A \in T$.
  3. T ist die kleinste Menge von Formeln, die (1) und (2) erfüllt.
- Man schreibt für $A \in T(\mathcal{F})$ auch $\vdash_{\mathcal{F}} A$ oder einfach $\vdash A$ und sagt "A ist in $\mathcal{F}$ herleitbar".
- **Deduktiver Folgerungsbegriff:** Sei $\Sigma \subseteq F$, $A \in F$, dann bedeutet $\Sigma \vdash_{\mathcal{F}(Ax,R)} A$ nichts anderes als $\vdash_{\mathcal{F}(Ax \cup \Sigma,R)} A$. Sprechweise: "$A$ ist in $\mathcal{F}$ aus Σ herleitbar".
- Sind $\mathcal{F}_1$ und $\mathcal{F}_2$ deduktive Systeme über der Formelmenge F und gilt $T(\mathcal{F}_1) = T(\mathcal{F}_2)$ so nennt man sie **äquivalent.**

Bemerkung

Bemerkung 1.16

1. *Eigenschaften der Elemente von T werden durch strukturelle Induktion bewiesen.*
   T wird von einer Relation $R' \subseteq F^\star \times F$ erzeugt.
   Eine Formel A ist ein Theorem oder ist in $\mathcal{F}$ herleitbar, falls es eine endliche Folge von Formeln $B_0, \ldots, B_n$ gibt mit $A \equiv B_n$ und für $0 \leq i \leq n$ gilt:
   $B_i \in Ax$ oder es gibt l und $i_1, \ldots, i_l < i$ und eine Regel $\frac{B_{i_1} \ldots B_{i_l}}{B_i} \in R$.
   - *Die Folge $B_0, \ldots, B_n$ heißt auch* **Beweis (Herleitung)** *für A in $\mathcal{F}$.*
   - *Das bedeutet $\vdash A$ gilt genau dann, wenn es einen Beweis $B_0, \ldots, B_n$ mit $A \equiv B_n$ gibt.*

Bemerkung (Fort.)

2. *Die Menge T der Theoreme ist rekursiv aufzählbar (denn Ax und R sind rekursiv). Die Menge der Beweise*

$$\text{Bew} := \{B_1 \star B_2 \star \ldots \star B_n \mid B_1, \ldots, B_n \text{ ist Beweis}\}$$

ist rekursiv. (Siehe Argumentation von $L(G)$ ist rekursiv aufzählbar für Grammatiken G).

   - *Ist Σ rekursiv entscheidbar, so gelten obige Aussagen entsprechend. Insbesondere ist $Folg_{\mathcal{F}}(\Sigma) = \{A \mid \Sigma \vdash_{\mathcal{F}(Ax,R)} A\}$ rekursiv aufzählbar.*
   - **Beachte**: *Beweise sind im allgemeinen nicht eindeutig. Es wird im allgemeinen nicht verlangt, dass T von R frei erzeugt wird.*

Bemerkung (Fort.)

3. *Gibt es ein deduktives System $\mathcal{F}_0$, so dass $\vdash_{\mathcal{F}_0} A$ genau dann gilt, wenn $\models A$ gilt?*

- *Hierzu werden Ax und R häufig endlich beschrieben durch Schemata.*

Beispielsweise beschreibt das Axiom $(A \to (B \to A))$ die Menge
$\{A_0 |$ *es gibt* $A, B \in F$ *mit* $A_0 \equiv (A \to (B \to A))\}$
und die Regel
$\frac{A, A \to B}{B}$ *beschreibt die Menge von Regeln*

$$\left\{ \frac{A_0, A_1}{B_0} \mid \text{Es gibt } A, B \in F \text{ mit } A_0 \equiv A, B_0 \equiv B \text{ und } A_1 \equiv A \to B \right\}.$$

Das deduktive System $\mathcal{F}_0$

Definition 1.17 (Das deduktive System $\mathcal{F}_0$)

Das deduktive System $\mathcal{F}_0$ für die Aussagenlogik besteht aus der Formelmenge F_0 der Formeln in $V, \neg, \rightarrow, ($ und $)$. Die Axiomenmenge Ax wird durch folgende Axiomenschemata beschrieben:

Ax1: $A \rightarrow (B \rightarrow A)$

Ax2: $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

Ax3: $((\neg A) \rightarrow (\neg B)) \rightarrow (B \rightarrow A)$

Dabei beschreiben Ax1, Ax2 und Ax3 disjunkte Formelmengen.

Das deduktive System $\mathcal{F}_0$

Die Regelmenge R wird beschrieben durch das Regelschema

$$\text{MP:} \quad \frac{A, (A \to B)}{B} \quad \text{(modus ponens).}$$

- Beachte: Ax und R sind rekursiv entscheidbar.
- Es genügt zunächst nur Axiome für Formeln in $\to$ und $\neg$ zu betrachten, da alle anderen Formeln zu einer Formel in $\to$ und $\neg$ logisch äquivalent sind.
- Die Menge der Theoreme von $\mathcal{F}_0$ wird nicht frei erzeugt. Die Modus-Ponens-Regel ist hochgradig nicht eindeutig. $\frac{A, A \to B}{B}$ und $\frac{A', A' \to B}{B}$ sind beides Regeln mit gleicher Folgerung. Das erschwert sehr das Finden von Beweisen.

Beispiel

Beispiel 1.18

Für jedes $A \in F_0$ gilt $\vdash (A \to A)$, d.h. $(A \to A) \in T(\mathcal{F}_0)$

Beweis:

$$
\begin{array}{lll}
B_0 \equiv (A \to ((A \to A) \to A)) \to & & \\
\qquad ((A \to (A \to A)) \to (A \to A)) & & \text{Ax2} \\
B_1 \equiv A \to ((A \to A) \to A) & & \text{Ax1} \\
B_2 \equiv (A \to (A \to A)) \to (A \to A) & & \text{MP}(B_0, B_1) \\
B_3 \equiv A \to (A \to A) & & \text{Ax1} \\
B_4 \equiv A \to A & & \text{MP}(B_2, B_3)
\end{array}
$$

■

- Wie findet man Beweise im System $\mathcal{F}_0$?
  Einziger Hinweis: Die Zielformel B, sofern sie kein Axiom ist, muss in der Form $(A_1 \rightarrow (...(A_n \rightarrow B)...))$ vorkommen. Wähle geeignete A´s.
- **Beachte:** Alle Axiome sind Tautologien der Aussagenlogik. Da diese abgeschlossen gegenüber Modus Ponens sind, sind alle Theoreme von $\mathcal{F}_0$ Tautologien. D.h. $T(\mathcal{F}_0) \subseteq Taut(F_0)$.
- Will man in ganz F Beweise führen, so muss man weitere Axiome einführen.
  Z.B.
  $Ax1\wedge: \quad (A \wedge B) \rightarrow (\neg(A \rightarrow (\neg B)))$
  $Ax2\wedge: \quad (\neg(A \rightarrow (\neg B))) \rightarrow (A \wedge B)$

Deduktiver Folgerungsbegriff

Definition 1.19 (Axiomatischer Folgerungsbegriff)

Sei $\Sigma \subseteq F_0, A \in F_0$.

1. A ist aus Σ in $\mathcal{F}_0$ **herleitbar**, wenn A sich aus $\mathrm{Ax} \cup \Sigma$ mit den Regeln aus R herleiten lässt, d.h. A ist Theorem im deduktiven System $\mathcal{F}$ mit Axiomenmenge $\mathrm{Ax} \cup \Sigma$ und gleicher Regelmenge wie $\mathcal{F}_0$. Schreibweise $\Sigma \vdash_{\mathcal{F}_0} A$, einfacher $\Sigma \vdash A$.

   $B_0, \ldots, B_n$ ist ein **Beweis** für $\Sigma \vdash A$, falls $A \equiv B_n$ und für alle $0 \leq i \leq n$ gilt: $B_i \in \mathrm{Ax} \cup \Sigma$ oder es gibt $j, k < i$ mit $B_k \equiv (B_j \to B_i)$.
2. Σ heißt **konsistent**, falls für keine Formel $A \in F_0$ gilt $\Sigma \vdash A$ und $\Sigma \vdash \neg A$.
   Gibt es eine solche Formel, so heißt Σ **inkonsistent**.

Folgerung 1.20 (Beweishilfsmittel)

1. *Gilt* $\Sigma \vdash A$, *so folgt unmittelbar aus der Definition 1.19, dass es eine endliche Teilmenge* $\Sigma_0 \subseteq \Sigma$ *gibt mit* $\Sigma_0 \vdash A$.
   Dies entspricht dem Kompaktheitssatz für "$\models$*".*
2. *Ist* Σ *inkonsistent, dann gibt es eine endliche Teilmenge* $\Sigma_0 \subseteq \Sigma$, *die inkonsistent ist*
   (denn ist $\Sigma \subseteq \Gamma$ *und* $\Sigma \vdash A$, *dann gilt auch* $\Gamma \vdash A$*).*
3. *Ist* $\Sigma \subseteq \Gamma$ *so* $Folg_{\mathcal{F}_0}(\Sigma) \subseteq Folg_{\mathcal{F}_0}(\Gamma)$.
4. *Aus* $\Sigma \vdash A$ *und* $\Gamma \vdash B$ *für alle* $B \in \Sigma$ *folgt* $\Gamma \vdash A$.
   Ist also $\Sigma \subseteq Folg_{\mathcal{F}_0}(\Gamma)$ *so* $Folg_{\mathcal{F}_0}(\Sigma) \subseteq Folg_{\mathcal{F}_0}(\Gamma)$.
   Beweise lassen sich also zusammensetzen.
5. *Gilt* $\Sigma \vdash A$, *so ist* $\{\Sigma, \neg A\}$ *inkonsistent.*
   Gilt auch die Umkehrung?
6. *Es gilt stets* $T(\mathcal{F}_0) \subseteq Folg_{\mathcal{F}_0}(\Sigma)$ *für jede Menge* Σ.

Satz 1.21 (Deduktionstheorem)

Sei $\Sigma \subseteq F_0$ *und seien* $A, B \in F_0$.
Dann gilt: $\Sigma, A \vdash B \quad \text{gdw} \quad \Sigma \vdash (A \to B)$

Beweis:

„ $\Leftarrow$ “ Klar wegen MP-Regel.
„ $\Rightarrow$ “ Sei $B_0, ..., B_m$ ein Beweis für $\Sigma, A \vdash B$, d.h. $B \equiv B_m$.
Beh.: Für $i = 0, ..., m$ gilt $\Sigma \vdash (A \to B_i)$

Induktion nach i und Fallunterscheidung, je nachdem ob B_i gleich A ist, in $\mathrm{Ax} \cup \Sigma$ liegt oder mit MP-Regel aus B_j, B_k mit $j, k < i$ entsteht. ■

Anwendungen des Deduktionstheorems

Beispiel 1.22 (Beweistransformationen. Wiederverwendung von Beweisen.)

- Vereinbarungen zur Darstellung von Beweisen:
  $B_1, \ldots, B_n$ heißt **abgekürzter Beweis** für $\Sigma \vdash B_n$, falls für jedes j mit $1 \leq j \leq n$ gilt: $\Sigma \vdash B_j$ oder es gibt $j_1, \ldots, j_r < j$ mit $B_{j_1}, \ldots, B_{j_r} \vdash B_j$.
- Gibt es einen abgekürzten Beweis für $\Sigma \vdash A$, dann gibt es auch einen Beweis für $\Sigma \vdash A$.

1. $\vdash (A \to A)$ folgt aus dem Deduktionstheorem, da $A \vdash A$ gilt.
2. Um $A \to B, B \to C \vdash A \to C$ zu zeigen, zeige $A, A \to B, B \to C \vdash C$.

Anwendungen des Deduktionstheorems (Fort.)

3. $\vdash (\neg\neg A \rightarrow A)$ dazu genügt es zu zeigen
   $\neg\neg A \vdash A$

Beweis:

$B_1 \equiv \neg\neg A$	
$B_2 \equiv \neg\neg A \rightarrow (\neg\neg\neg\neg A \rightarrow \neg\neg A)$	Ax1
$B_3 \equiv \neg\neg\neg\neg A \rightarrow \neg\neg A$	MP
$B_4 \equiv (\neg\neg\neg\neg A \rightarrow \neg\neg A) \rightarrow (\neg A \rightarrow \neg\neg\neg A)$	Ax3 ■
$B_5 \equiv \neg A \rightarrow \neg\neg\neg A$	MP
$B_6 \equiv (\neg A \rightarrow \neg\neg\neg A) \rightarrow (\neg\neg A \rightarrow A)$	Ax3
$B_7 \equiv \neg\neg A \rightarrow A$	MP
$B_8 \equiv A$	MP

Anwendungen des Deduktionstheorems (Fort.)

4. $\vdash (A \to B) \to ((B \to C) \to (A \to C))$
   (zeige: $A \to B, B \to C \vdash A \to C$)
5. $\vdash (B \to ((B \to A) \to A))$
6. $\vdash (\neg B \to (B \to A))$ (zu zeigen: $\neg B, B \vdash A$)

Beweis:

$B_1 \equiv \neg B$	Vor
$B_2 \equiv \neg B \to (\neg A \to \neg B)$	Ax1
$B_3 \equiv \neg A \to \neg B$	MP
$B_4 \equiv (\neg A \to \neg B) \to (B \to A)$	Ax3
$B_5 \equiv B \to A$	MP
$B_6 \equiv B$	Vor
$B_7 \equiv A$	MP ■

7. $\vdash B \rightarrow \neg\neg B$
8. $\vdash ((A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A))$ und
   $\vdash ((\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B))$
9. $\vdash (B \rightarrow (\neg C \rightarrow \neg(B \rightarrow C)))$
10. $\vdash ((B \rightarrow A) \rightarrow ((\neg B \rightarrow A) \rightarrow A))$
11. $\vdash (A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$

Frage: Lassen sich alle Tautologien als Theoreme im System $\mathcal{F}_0$ herleiten ?

Vollständigkeit und Entscheidbarkeit von $\mathcal{F}_0$

Satz 1.23 (Korrektheit und Vollständigkeit von $\mathcal{F}_0$)

Sei $A \in F_0$ eine Formel der Aussagenlogik.

a) Korrektheit: *Aus $\vdash_{\mathcal{F}_0} A$ folgt $\models A$, d.h. nur Tautologien können als Theoreme in $\mathcal{F}_0$ hergeleitet werden.*

b) Vollständigkeit: *Aus $\models A$ folgt $\vdash_{\mathcal{F}_0} A$, d.h. alle Tautologien lassen sich in $\mathcal{F}_0$ herleiten.*

Vollständigkeit und Entscheidbarkeit von $\mathcal{F}_0$ (Fort.)

Als Hilfsmittel dient:

Lemma 1.24

Sei $A \equiv A(p_1, \ldots, p_n) \in F_0, n > 0$, *wobei* $p_1, \ldots, p_n$ *die in* A *vorkommenden Aussagevariablen sind. Sei* φ *eine Bewertung. Ist*

$$P_i := \begin{cases} p_i & , \textit{falls } \varphi(p_i) = 1 \\ \neg p_i & , \textit{falls } \varphi(p_i) = 0 \end{cases} \qquad A' := \begin{cases} A & , \textit{falls } \varphi(A) = 1 \\ \neg A & , \textit{falls } \varphi(A) = 0 \end{cases}$$

$(1 \leq i \leq n)$, *dann gilt* $P_1, \ldots, P_n \vdash A'$.

Angenommen das Lemma gilt und sei $\models A$, d.h. $\varphi(A) = 1$ für alle Bewertungen φ. Sei φ eine Bewertung mit $\varphi(p_n) = 1$. Es gilt $P_1, \ldots, P_n \vdash A$ und wegen $P_n \equiv p_n$ gilt $P_1, \ldots, P_{n-1}, p_n \vdash A$. Betrachtet man eine Bewertung φ' mit $\varphi'(p_n) = 0$ und sonst gleich φ , erhält man $P_1, \ldots, P_{n-1}, \neg p_n \vdash A$.

Vollständigkeit und Entscheidbarkeit von $\mathcal{F}_0$ (Fort.)

- Durch Anwenden des Deduktionstheorems entstehen daraus
  $P_1, \ldots, P_{n-1} \vdash p_n \to A$ und
  $P_1, \ldots, P_{n-1} \vdash \neg p_n \to A$.
  Gleichzeitig gilt nach dem 10. Beispiel von 1.22 auch
  $P_1, \ldots, P_{n-1} \vdash ((p_n \to A) \to ((\neg p_n \to A) \to A))$.
- Durch zweimaliges Anwenden des Modus-Ponens entsteht
  $P_1, \ldots, P_{n-1} \vdash A$.
- Dies gilt für jede Wahl der $P_i, i = 1, ..., n-1$ und somit lässt sich das Argument iterieren. D.h. in einer Herleitung von A muss kein p_i verwendet werden, also $\vdash A$.
- Das Lemma wird durch Induktion über den Aufbau von A nachgewiesen. D.h. für $A \equiv p_1, \neg C, B \to C$ unter Verwendung von Deduktionen aus Beispiel 1.22.

Folgerung

Folgerung 1.25

Sei $\Sigma \subseteq F_0, A \in F_0$.

1. $\Sigma \vdash_{\mathcal{F}_0} A$ *gilt genau dann, wenn* $\Sigma \models A$ *gilt.*
2. Σ *ist genau dann konsistent, wenn* Σ *erfüllbar ist.*
3. Σ *ist genau dann inkonsistent, wenn* Σ *unerfüllbar ist.*
4. *Ist* Σ *endlich und* $A \in F_0$*, dann ist* $\Sigma \vdash_{\mathcal{F}_0} A$ *entscheidbar.*

Beweis der Folgerung

Beweis:

1.

$$\Sigma \vdash_{\mathcal{F}_0} A$$

$$\overset{1.20}{\Longleftrightarrow} \quad \text{Es gibt } A_1, \ldots, A_n \in \Sigma \text{ mit } A_1, \ldots, A_n \vdash_{\mathcal{F}_0} A$$

$$\overset{D.T.}{\Longleftrightarrow} \quad \text{Es gibt } A_1, \ldots, A_n \in \Sigma \text{ mit } \vdash_{\mathcal{F}_0} (A_1 \to (A_2 \to \ldots (A_n \to A) \ldots))$$

$$\overset{1.23}{\Longleftrightarrow} \quad \text{Es gibt } A_1, \ldots, A_n \in \Sigma \text{ mit } \models (A_1 \to (A_2 \to \ldots (A_n \to A) \ldots))$$

$$\overset{D.T.}{\Longleftrightarrow} \quad \text{Es gibt } A_1, \ldots, A_n \in \Sigma \text{ mit } A_1, \ldots, A_n \models A$$

$$\overset{K.S.}{\Longleftrightarrow} \quad \Sigma \models A$$

Beweis der Folgerung

2. Σ ist konsistent. $\iff$
   Es gibt kein A mit $\Sigma \vdash A$ und $\Sigma \vdash \neg A$. $\iff$
   Es gibt kein A mit $\Sigma \models A$ und $\Sigma \models \neg A$. $\iff$
   Σ ist erfüllbar (Bemerkung 1.8 c)).

Natürliche Kalküle

Es gibt andere deduktive Systeme, für die Satz 1.23 gilt. Das deduktive System $\mathcal{F}_0$ wurde von S.C. Kleene eingeführt. Das folgende deduktive System geht auf G. Gentzen zurück.

Definition 1.26 (Gentzen-Sequenzenkalkül)

Eine Sequenz ist eine Zeichenreihe der Form $\Gamma \vdash \Delta$ mit zwei endlichen Mengen von Formeln Γ und Δ.
Seien $\Gamma, \Delta \subseteq F$ endliche Mengen von Formeln und $A, B \in F$.
Der Kalkül für Objekte der Form $\Gamma \vdash_G \Delta$ wird definiert durch folgende Axiome und Regeln:

Gentzen-Sequenzenkalkül: Axiome und Regeln

Ax1: $\Gamma, A \vdash_G A, \Delta$

Ax2: $\Gamma, A, \neg A \vdash_G \Delta$

Ax3: $\Gamma \vdash_G A, \neg A, \Delta$

$R_{\wedge,\vee}$: $\dfrac{\Gamma, A, B \vdash_G \Delta}{\Gamma, A \wedge B \vdash_G \Delta}$ $\qquad$ $\dfrac{\Gamma \vdash_G A, B, \Delta}{\Gamma \vdash_G (A \vee B), \Delta}$

$R_{\rightarrow}$: $\dfrac{\Gamma, A \vdash_G \Delta, B}{\Gamma \vdash_G (A \rightarrow B), \Delta}$ $\qquad$ $\dfrac{\Gamma \vdash_G A, \Delta \,; \quad \Gamma, B \vdash_G \Delta}{\Gamma, (A \rightarrow B) \vdash_G \Delta}$

$R_{\neg}$: $\dfrac{\Gamma, A \vdash_G \Delta}{\Gamma \vdash_G \neg A, \Delta}$ $\qquad$ $\dfrac{\Gamma \vdash_G A, \Delta}{\Gamma, \neg A \vdash_G \Delta}$

$R_{\wedge'}$: $\dfrac{\Gamma \vdash_G A, \Delta \,; \quad \Gamma \vdash_G B, \Delta}{\Gamma \vdash_G (A \wedge B), \Delta}$

$R_{\vee'}$: $\dfrac{\Gamma, A \vdash_G \Delta \,; \quad \Gamma, B \vdash_G \Delta}{\Gamma, (A \vee B) \vdash_G \Delta}$

Gentzen-Sequenzenkalkül

$\Gamma \vdash_G \Delta$ ist ableitbar bedeutet: Es gibt ein $r \in \mathbb{N}$ und eine Folge von Sequenzen $\Gamma_1 \vdash_G \Delta_1, \ldots, \Gamma_r \vdash_G \Delta_r$ mit

1. $\Gamma_r \equiv \Gamma$ und $\Delta_r \equiv \Delta$
2. Jedes $\Gamma_j \vdash_G \Delta_j$ mit $1 \leq j \leq r$ ist Axiom oder geht aus vorangehenden Folgegliedern aufgrund einer Regel hervor.

Bemerkung 1.27 (**Semantische Interpretation**)

Die Aussage $\Gamma \vdash_G \Delta$ kann wie folgt anschaulich interpretiert werden: Für jede Bewertung φ gibt es eine Formel $A \in \Gamma$ mit $\varphi(A) = 0$ oder es gibt eine Formel $B \in \Delta$ mit $\varphi(B) = 1$. Sind $\Gamma = \{A_1, \ldots, A_n\}$ und $\Delta = \{B_1, \ldots, B_m\}$, also endlich, entspricht dies also der Formel $(A_1 \wedge \cdots \wedge A_n) \rightarrow (B_1 \vee \cdots \vee B_m)$.

- Der Semantische Folgerungsbegriff $\Sigma \models A$ für eine Menge von Formeln $\{\Sigma, A\}$ kann wie folgt auf Mengenpaare Γ, Δ erweitert werden:

$$\Gamma \models \Delta \quad \text{gdw} \quad \Gamma \models A$$

  wobei A die Disjunktion der Formeln in Δ ist.
- Interpretiert man in einer Sequenz $\Gamma \vdash_G \Delta$ die Menge Γ als Voraussetzungen, und die Menge Δ als Konklusion, so lässt sich die Korrektheit des Kalküls leicht nachweisen.
- Es gilt also:
  Aus $\Gamma \vdash_G \Delta$ folgt $\Gamma \models \Delta$. (**Übung**)
- Es gilt auch die Umkehrung dieser Aussage, d.h. der Sequenzenkalkül von Gentzen ist korrekt und vollständig. (Bew. siehe z.B. Kleine Büning/Lettmann: Aussagenlogik, Deduktion und Algorithmen)

Beispiel

Beispiel 1.28

Es gilt $p \vee q, (\neg p) \vee r \vdash_G q \vee r$

Beweis:

$B_1 \equiv q, r \vdash q, r$	Ax1
$B_2 \equiv q, \neg p \vdash q, r$	Ax1
$B_3 \equiv q, (\neg p) \vee r \vdash q, r$	$R_{\vee'}(1,2)$
$B_4 \equiv p, r \vdash q, r$	Ax1
$B_5 \equiv \neg p, p \vdash q, r$	Ax2
$B_6 \equiv p, (\neg p) \vee r \vdash q, r$	$R_{\vee'}(4,5)$
$B_7 \equiv p \vee q, (\neg p) \vee r \vdash q, r$	$R_{\vee'}(3,6)$
$B_8 \equiv p \vee q, (\neg p) \vee r \vdash q \vee r$	$R_{\vee}(7)$

■

Bemerkung 1.29 (**Weitere Kalküle für die Aussagenlogik**)

Man findet in der Literatur eine Vielzahl von „natürlichen“ Kalkülen (deduktiven Systemen), die ebenfalls korrekt und vollständig sind. Für diese werden auch Beweisstrategien für so genannte „Goals“ und „Subgoals“ vorgestellt.

Als Beispiel Hilberts Kalkül, das z.B. für jeden Operator eine Regel für die Einführung und eine für die Entfernung des Operators enthält.

Hilberts Kalkül

- Konjunktion $\wedge\_I: \dfrac{p, q}{p \wedge q}$ $\wedge\_E: \dfrac{p \wedge q}{p}$
- Disjunktion $\vee\_I: \dfrac{p}{p \vee q}$ $\vee\_E: \dfrac{p \vee q, \neg p}{q}$
- Implikation $\rightarrow\_E: \dfrac{p, p \rightarrow q}{q}$ $\rightarrow\_E: \dfrac{\neg q, p \rightarrow q}{\neg p}$
  Modus Ponens Modus Tollens
- Negation $\neg\_E: \dfrac{p, \neg p}{q}$ $\neg\_E: \dfrac{\neg\neg p}{p}$
  Widerspruchsregel Doppelnegation
- Äquivalenz $\leftrightarrow\_E: \dfrac{p \leftrightarrow q}{p \rightarrow q}$ $\leftrightarrow\_E: \dfrac{p \leftrightarrow q}{q \rightarrow p}$

- Transitivität $\quad \leftrightarrow_{\perp}I : \dfrac{p \leftrightarrow q, q \leftrightarrow r}{p \leftrightarrow r}$
- Deduktions - Theorem $\quad \rightarrow_{\perp}I : \dfrac{p, ..., r, \underline{s} \vdash t}{p, ..., r \vdash s \rightarrow t}$
- Reductio ad absurdum $\quad \neg_{\perp}I : \dfrac{p, ..., r, \underline{s} \vdash t, p, ..., r, \underline{s} \vdash \neg t}{p, ..., r \vdash \neg s}$
- Hypothetischer Syllogismus $\quad \dfrac{p \rightarrow q, q \rightarrow r}{p \rightarrow r}$
- Konstruktives Dilemma $\quad \dfrac{p \rightarrow q, r \rightarrow s, p \vee r}{q \vee s}$

Hinzukommen die üblichen Assoziativitäts-, Kommutativitäts-, Distributivitäts-, Negations-, Implikations- und de Morgan Regeln.

Beispiele in Hilberts Kalkül

Beispiel 1.30

Zeige $(p \wedge q) \vee r \vdash \neg p \rightarrow r$

Beweis:

- **Transformationsbeweis**

1.	$(p \wedge q) \vee r$	Prämisse
2.	$r \vee (p \wedge q)$	Kommutativität
3.	$(r \vee p) \wedge (r \vee q)$	Distributivität
4.	$(r \vee p)$	$\wedge\_E$
5.	$(p \vee r)$	Kommutativität
6.	$(\neg\neg p \vee r)$	Negations-Gesetz
7.	$\neg p \rightarrow r$	Implikations-Gesetz

■

Beispiele in Hilberts Kalkül

Zeige $(p \wedge q) \vee r \vdash \neg p \rightarrow r$

Beweis:

- **Bedingter Beweis**

1.	$(p \wedge q) \vee r$	Prämisse
2.	$\neg(\neg p \vee \neg q) \vee r$	Doppelnegation, de Morgan
3.	$(\neg p \vee \neg q) \rightarrow r$	Implikationsgesetz
4.	$\neg p$	Annahme
5.	$\neg p \vee \neg q$	$\vee\_I$
6.	r	Modus Ponens $\rightarrow E$ aus 3. und 5.
7.	$\neg p \rightarrow r$	Aus 4, 5, 6 mit Ded. Theo. $\rightarrow I$

■

Beispiele in Hilberts Kalkül

Zeige $(p \wedge q) \vee r \vdash \neg p \rightarrow r$

Beweis:

- **Indirekter Beweis**

1.	$(p \wedge q) \vee r$	Prämisse
2.	$(p \vee r) \wedge (q \vee r)$	Distributivgesetz
3.	$(p \vee r)$	$\wedge\_E$
4.	$\neg(\neg p \rightarrow r)$	Annahme
5.	$\neg(p \vee r)$	Implikations-und Negationsgesetz
6.	$\neg\neg(\neg p \rightarrow r)$	Red. Abs. $\rightarrow\!\!\!\!\!\!+ I$ aus 3, 4, 5.
7.	$\neg p \rightarrow r$	Doppelnegation

■

Algorithmischer Aufbau der Aussagenlogik

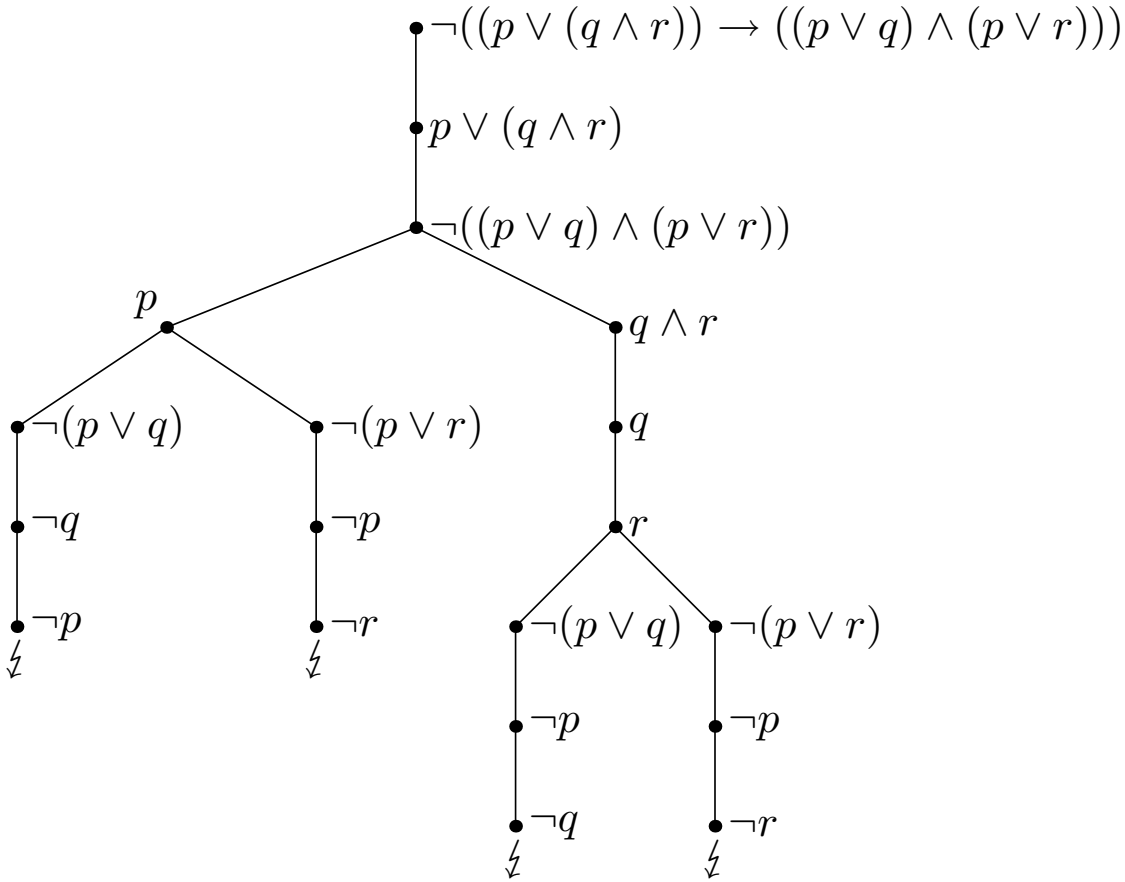
In diesem Abschnitt betrachten wir Verfahren die bei gegebener endlichen Menge Σ und A-Form A entscheiden ob $\Sigma \models A$ gilt. Die bisher betrachteten Verfahren prüfen alle Belegungen der in den Formeln vorkommenden Variablen oder zählen effektiv die Theoreme eines geeigneten deduktiven Systems auf. Dies ist sicherlich recht aufwendig. Obwohl die Komplexität dieses Problems groß ist (Entscheidbarkeit von *SAT* ist bekanntlich NP-vollständig), ist die Suche nach Verfahren, die „oft“ schneller als die „brute force Methode“ sind, berechtigt.
Wir betrachten drei solcher Verfahren die alle Erfüllbarkeitsverfahren sind, d.h. sie basieren auf:
$\Sigma \models A \quad \text{gdw} \quad \{\Sigma, \neg A\}$ unerfüllbar:
Semantische Tableaux Davis-Putman Resolution

Beispiel

Beispiel 2.1 (Semantische Tableaux)

Um die Allgemeingültigkeit einer Formel A zu zeigen, konstruiere einen binären Baum für $\neg A$, dessen Knoten jeweils eine Klasse möglicher Belegungen repräsentieren die diesen Knoten erfüllen. Die Wurzel des Baumes repräsentiert alle möglichen Belegungen und die Vereinigung der Klassen der Söhne eines inneren Knotens des Baumes ist die Klasse der Belegungen, die der Knoten repräsentiert. Gelingt es, einen solchen Baum derart zu konstruieren, dass sämtliche Blätter des Baumes zu einem Widerspruch führen, ist gezeigt, dass es keine Belegung gibt, die $\neg A$ erfüllt. Somit gilt, dass A Tautologie ist.

$\models (p \vee (q \wedge r) \rightarrow (p \vee q) \wedge (p \vee r))$ gilt genau dann, wenn $\neg((p \vee (q \wedge r)) \rightarrow ((p \vee q) \wedge (p \vee r)))$ unerfüllbar ist.



Da alle Äste zu Widersprüchen führen, gibt es keine Belegung, die die Formel erfüllt!

Feststellungen

Zwei Arten von Formeln, solche, die zu Verzweigungen führen (β-Formeln), und solche, die nicht zu Verzweigungen führen (α-Formeln).

- α-Formeln mit Komponenten α_1 und α_2, die zu Knoten mit den Markierungen α_1 und α_2 führen:

α	$\neg\neg A$	$A_1 \wedge A_2$	$\neg(A_1 \vee A_2)$	$\neg(A_1 \rightarrow A_2)$
α_1	A	A_1	$\neg A_1$	A_1
α_2	(A)	A_2	$\neg A_2$	$\neg A_2$

- β-Formeln mit Komponenten β_1 und β_2, die zu Verzweigungen führen mit Knotenmarkierungen β_1 und β_2:

β		$\neg(A_1 \wedge A_2)$		$A_1 \vee A_2$		$A_1 \rightarrow A_2$	
β_1	β_2	$\neg A_1$	$\neg A_2$	A_1	A_2	$\neg A_1$	A_2

Feststellungen (Fort.)

- **Beachte:** Jede Aussageform ist entweder atomar (d.h. eine Variable) oder die Negation einer atomaren Formel (d.h. ein **Literal**) oder eine α- oder eine β-Formel, und genau von einem dieser drei Typen.

- Es gilt:
  Eine α-Formel ist genau dann erfüllbar, wenn beide Komponenten α_1 und α_2 erfüllbar sind.

  Eine β-Formel ist genau dann erfüllbar, wenn eine der Komponenten β_1 oder β_2 erfüllbar ist.

Feststellungen (Fort.)

- Insbesondere gilt für $\Gamma \subseteq F$ und α-Formel α mit Komponenten α_1 und α_2 und β-Formel β mit Komponenten β_1 und β_2:
  $\Gamma \cup \{\alpha\}$ erfüllbar gdw $\Gamma \cup \{\alpha_1, \alpha_2\}$ erfüllbar und
  $\Gamma \cup \{\beta\}$ erfüllbar gdw $\Gamma \cup \{\beta_1\}$ oder $\Gamma \cup \{\beta_2\}$ erfüllbar.
- Ein **Literal** ist eine Aussagevariable p_i oder eine negierte Aussagevariable $\neg p_i$. Für eine A-Form A sind A und $\neg A$ **komplementär** oder **konjugiert**.
- Enthält Γ komplementäre Formeln (Literale) A und $\neg A$, so ist Γ nicht erfüllbar.
  Im Beispiel enthält jeder Ast komplementäre Literale, also ist die Astformelmenge für kein Ast erfüllbar.

Formalisierung der Tableaux

Definition 2.2 (Tableaux)

Tableaux sind binäre Bäume mit Knoten, die mit Formeln aus F markiert sind. Sei $\Sigma \subseteq F$.

1. Die Menge der Tableaux τ_Σ für Σ wird induktiv definiert durch:
   (a) $\tau_{\{A\}}$ ist der Baum mit einem Knoten, der mit $A \in \Sigma$ markiert ist. In diesem Fall schreibt man auch τ_A statt $\tau_{\{A\}}$.
   Graphisch:

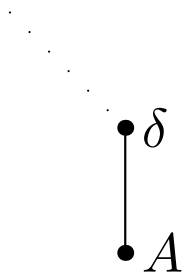
   $$\bullet A$$

   (b) Ist τ Tableau für Σ und δ Marke eines Blattes von τ, so lässt sich τ wie folgt zu einem Tableau τ' für Σ fortsetzen:
   τ' entsteht aus τ indem man als Nachfolger von δ:

Formalisierung (Fort.)

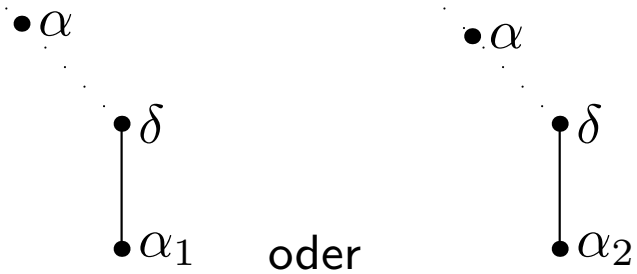
1. (b) (Σ) einen Knoten hinzufügt, der mit einer Formel $A \in \Sigma$ markiert ist. (A soll nicht bereits als Marke im Ast von δ vorkommen.)

Graphisch:

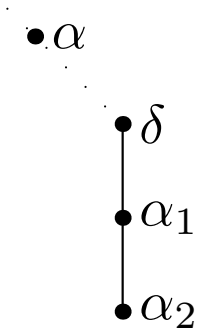


Formalisierung (Fort.)

1. (b) (α) einen Knoten hinzufügt, der mit α_1 oder α_2 markiert ist, falls eine α-Formel α auf dem Ast zu δ vorkommt und α_1 und α_2 die Komponenten von α sind.
   Graphisch:



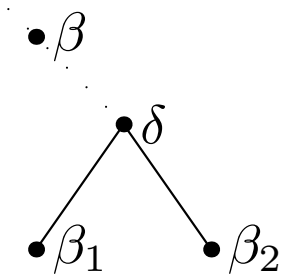
   In der Praxis werden jedoch an δ nacheinander die Knoten für beide Komponenten hinzugefügt:



Formalisierung (Fort.)

1. (b) (β) zwei Knoten hinzufügt, die mit den Komponenten β_1 bzw. β_2 einer β-Formel β markiert sind, falls β auf dem Ast zu δ vorkommt.

   Graphisch:



   Entsteht τ' aus τ durch Anwendung einer der Regeln $(\Sigma), (\alpha)$ oder (β), so heißt τ' **direkte Fortsetzung** von τ.

1. (c) $\tau \in \tau_\Sigma$ genau dann, wenn $\tau = \tau_A$ für ein $A \in \Sigma$ oder es gibt eine Folge $\tau_0, \ldots, \tau_n (= \tau)$, $n \in \mathbb{N}$, so dass τ_{j+1} eine direkte Fortsetzung von τ_j ist für $j = 0, \ldots, n-1$ und $\tau_0 = \tau_A$ für ein $A \in \Sigma$.

Formalisierung (Fort.)

2. Ein **Ast** eines Tableaus τ heißt **abgeschlossen**, falls er zwei konjugierte Formeln enthält (d.h. für ein $A \in F$ sowohl A als auch $(\neg A)$ enthält), sonst heißt der Ast **offen**.
   - Ein Tableau τ heißt **abgeschlossen**, wenn jeder Ast von τ abgeschlossen ist.
   - τ heißt **erfüllbar**, wenn τ einen **erfüllbaren Ast** (d.h. die Marken entlang des Ast bilden eine erfüllbare Formelmenge) enthält.
3. Sei $\Gamma \subseteq F, A \in F$. Dann ist A **Tableau-Folgerung** aus Γ Schreibe: $\Gamma \vdash_\tau A$ genau dann, wenn für $\Sigma = \Gamma \cup \{\neg A\}$ jedes Tableau aus τ_Σ sich zu einem abgeschlossenen Tableau aus τ_Σ fortsetzen lässt.

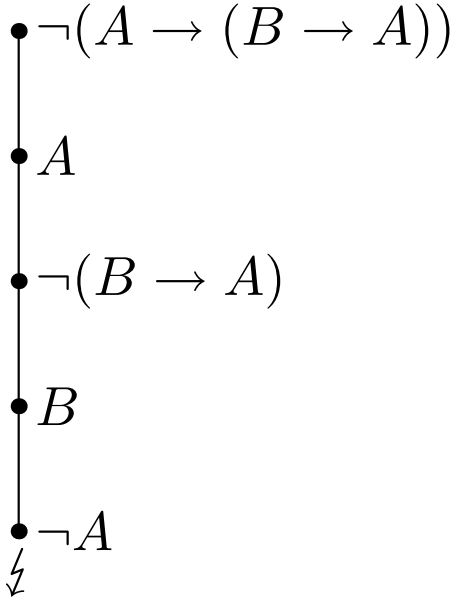
Bemerkung 2.3

Ziel ist es zu zeigen: $\Gamma \vdash_\tau A \iff \Gamma \models A$.

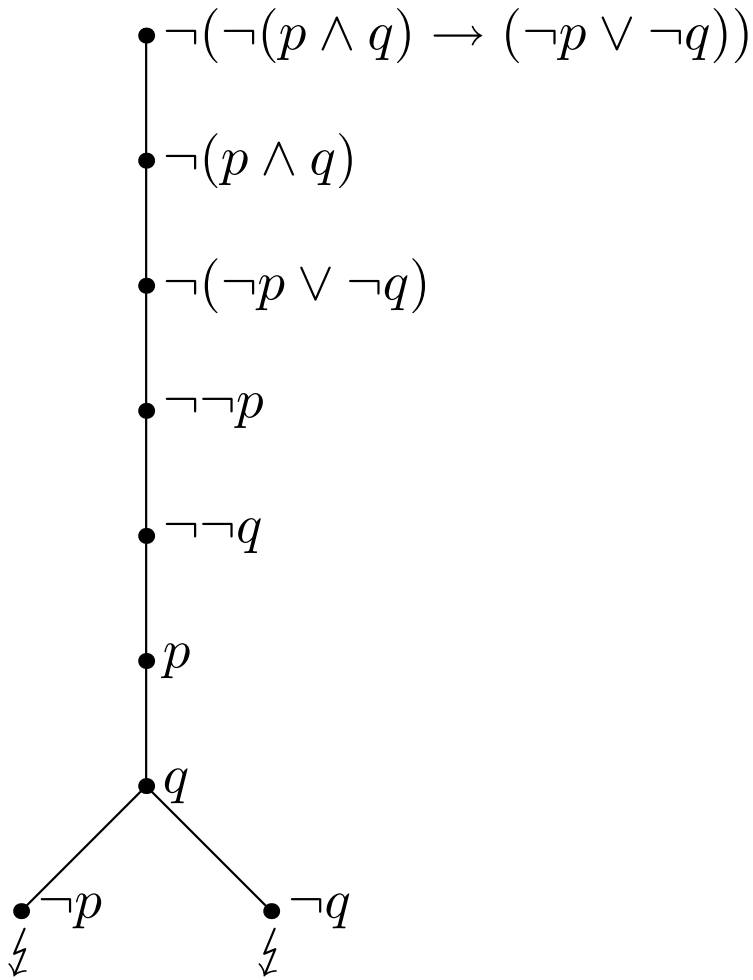
1. *Abgeschlossene Äste und Tableaux sind nicht erfüllbar.*
2. *Ist* Γ *erfüllbar, so ist jedes Tableau aus* τ_Γ *erfüllbar (und insbesondere nicht abgeschlossen).*
3. *Gilt* $\Gamma \vdash_\tau A$, *so ist* $\Sigma = \Gamma \cup \{\neg A\}$ *nicht erfüllbar. Insbesondere sind Tableau-Folgerungen korrekt (aus* $\Gamma \vdash_\tau A$ *folgt* $\Gamma \models A$).
4. *Gibt es ein abgeschlossenes Tableau in* τ_Γ, *so lässt sich jedes Tableau aus* τ_Γ *zu einem abgeschlossenen Tableau fortsetzen.*
5. *Tableaux sind endliche Bäume. Ist* $\tau \in \tau_\Sigma$, *so kommen als Marken nur (negierte oder unnegierte) Teilformeln von Formeln aus* Σ *vor.*
   Unendliche Tableaux können als Grenzfälle (falls Σ *unendlich) betrachtet werden.*

Beispiel 2.4

$\vdash_\tau A \to (B \to A)$:

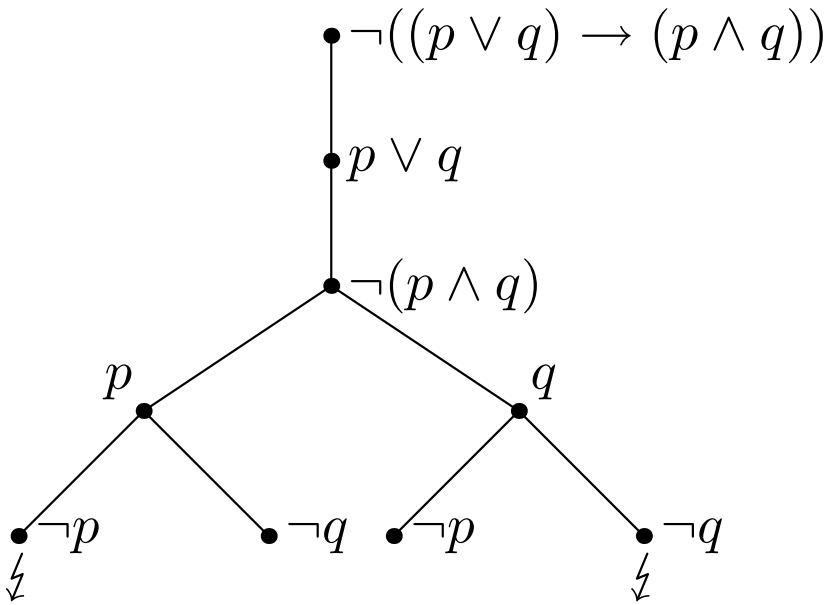


$\vdash_\tau \neg(p \wedge q) \to (\neg p \vee \neg q)$:



Beispiel 2.5

$\vdash_\tau (p \vee q) \rightarrow (p \wedge q)$ gilt nicht:

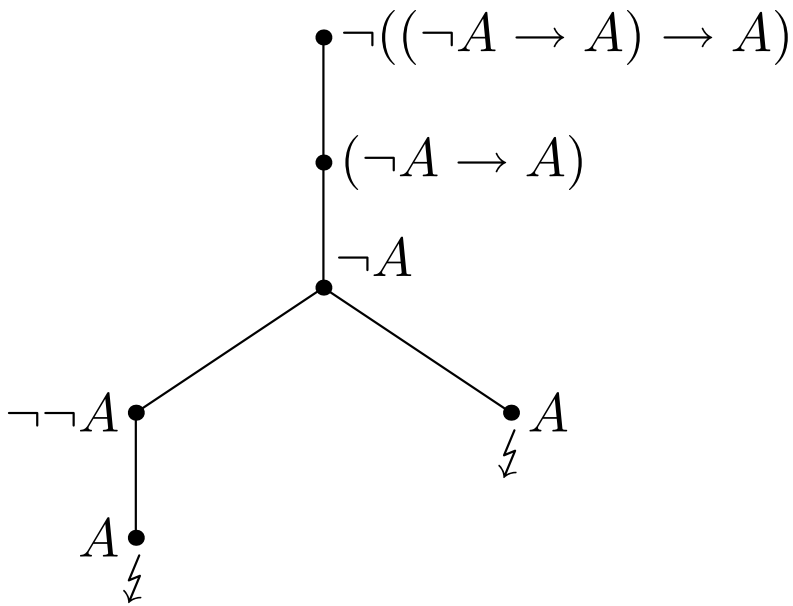


Es gibt Belegungen, die $\neg((p \vee q) \rightarrow (p \wedge q))$ erfüllen, nämlich φ mit $\varphi(p) = 1$ und $\varphi(q) = 0$ und φ' mit $\varphi'(p) = 0$ und $\varphi'(q) = 1$. Also gilt nicht $\vdash_\tau (p \vee q) \rightarrow (p \wedge q)$.

Beispiele (Fort.)

Beispiel 2.6

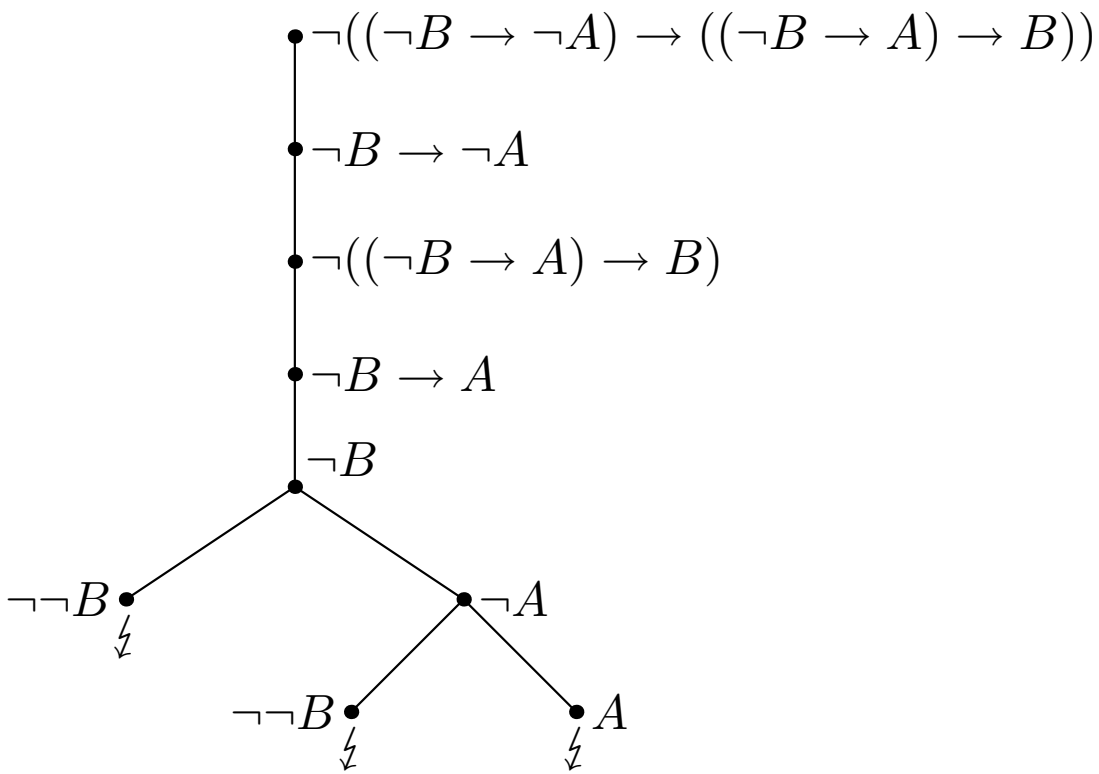
$\vdash_\tau (\neg A \to A) \to A$



Beispiele (Fort.)

Beispiel 2.7

$\vdash_\tau (\neg B \to \neg A) \to ((\neg B \to A) \to B)$



Vollständige Tableaux

Definition 2.8

Sei τ ein Tableau, Θ ein Ast von τ.

- Θ heißt **vollständig**, falls für die Menge der Formeln in Θ gilt: Mit jeder α-Formel $\alpha \in \Theta$ ist stets $\{\alpha_1, \alpha_2\} \subseteq \Theta$ und mit jeder β-Formel $\beta \in \Theta$ ist stets $\beta_1 \in \Theta$ oder $\beta_2 \in \Theta$.
- τ heißt **vollständig**, falls jeder Ast in τ abgeschlossen oder vollständig ist.
- Sei $\Sigma \subseteq F, \Sigma$ endlich. $\tau \in \tau_\Sigma$ heißt **vollständig für Σ**, falls τ vollständig ist und jeder offene Ast Σ enthält.
- Sei $\Sigma \subseteq F, \Sigma$ unendlich, so verallgemeinerte Tableaux erlaubt (d.h. jeder offene Ast ist unendlich und enthält Σ).

Bemerkung 2.9

1. *Ziel: Ist Σ endlich, so lässt sich jedes Tableau aus τ_Σ zu einem vollständigen Tableau für Σ mit Hilfe von Σ-, α- und β-Regeln erweitern.*
   Beachte, dass α- und β- Regeln nur (negierte) Teilformeln einführen und dass eine Formel nur endlich viele Teilformeln enthalten kann.
   (Gilt entsprechend für Σ unendlich mit verallg. Tableaux).
2. *Sei Γ die Menge der Formeln eines vollständigen offenen Astes von Γ. Dann gilt:*
   - 2.1 *Es gibt kein $p \in V$ mit $\{p, \neg p\} \subseteq \Gamma$.*
   - 2.2 *Ist $\alpha \in \Gamma$, so auch $\alpha_1, \alpha_2 \in \Gamma$.*
   - 2.3 *Ist $\beta \in \Gamma$, so ist $\beta_1 \in \Gamma$ oder $\beta_2 \in \Gamma$.*

Vollständige Tableaux (Fort.)

Lemma 2.10

Jede Menge Σ von Formeln, die 1, 2 und 3 aus der Bemerkung 2.9.2 genügt, ist erfüllbar. Insbesondere sind vollständige offene Äste von Tableaux erfüllbar.
Gibt es offene vollständige Tableaux für Γ, so ist Γ erfüllbar.

Beweis:

Definiere:

$$\varphi(p) = \begin{cases} 0 & \neg p \in \Sigma \\ 1 & \text{sonst} \end{cases}$$

Offensichtlich ist φ wohldefiniert.
Beh.: Falls $A \in \Sigma$, dann $\varphi(A) = 1$. (Induktion) ■

Satz 2.11

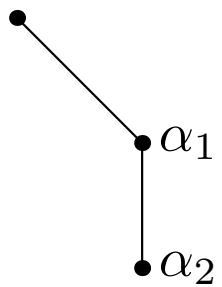
Sei $\Gamma \subseteq F$. Dann gilt:

1. Γ *ist nicht erfüllbar* gdw τ_Γ *enthält ein abgeschlossenes Tableau.*
2. *Äquivalent sind*
   - $\Gamma \models A$ *(oder* $\Gamma \vdash A$*)*
   - $\tau_{\{\Gamma, \neg A\}}$ *enthält ein abgeschlossenes Tableau.*
3. *Äquivalent sind*
   - $\models A$ *(oder* $\vdash A$*)*
   - $\tau_{\neg A}$ *enthält ein abgeschlossenes Tableau.*

Beachte: Der Kompaktheitssatz (1.10) folgt aus 1., denn ist Γ nicht erfüllbar, enthält τ_Γ ein abgeschlossenes Tableau und abgeschlossene Tableaux sind stets endliche Bäume, d.h. eine endliche Teilmenge von Γ ist nicht erfüllbar.

Systematische Tableaukonstruktion

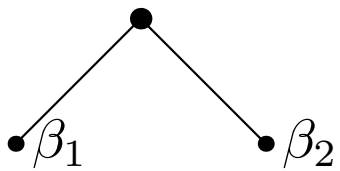
- Sei $\Gamma \subseteq F$, dann ist Γ abzählbar. Sei also $\Gamma = \{A_1, A_2, \ldots\}$. Konstruktion einer Folge von Tableaux τ_n $(n \in \mathbb{N})$:
  1. $\tau_1 \equiv A_1$. Ist A_1 Literal, dann wird der Knoten markiert.
  2. Sind alle Äste von τ_n abgeschlossen, dann Stopp!
     τ_{n+1} entsteht aus τ_n wie folgt:
  3. Ist Y die erste unmarkierte α-Formel in τ_n, durch die ein offener Ast geht, so markiere Y und erweitere jeden offenen Ast, der durch Y geht, um die Teilformeln α_1 und α_2 von Y.



α_1 und α_2 werden markiert, falls sie Literale sind. Dadurch werden möglicherweise Äste abgeschlossen.

oder:

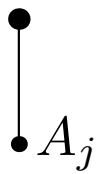
4. Ist Y die erste unmarkierte β-Formel in τ_n, durch die ein offener Ast geht, so markiere Y und erweitere jeden offenen Ast, der durch Y geht, um



Markiere β_1 und/oder β_2, falls diese Literale sind. Dadurch werden möglicherweise Äste abgeschlossen.

oder:

5. Gibt es eine Formel $A_j \in \Gamma$, die noch nicht in jedem offenen Ast vorkommt, so erweitere alle diese Äste um:



Falls möglich, Knoten markieren und Äste abschließen.

- **Verfahren**: Beginne mit τ_1. Wiederhole 3. solange wie möglich. Dann 4.. Sind weder 3. noch 4. möglich so 5. Geht nichts mehr, so stopp.
- Aus τ_n $(n \geq 1)$ erhält man kein weiteres Tableau, falls τ_n abgeschlossen ist oder alle Formeln von τ_n markiert sind und Γ endlich und ausgeschöpft ist.
- Setze $\tau_\infty := \bigcup_{n \in \mathbb{N}} \tau_n$. Dann ist τ_∞ ein binärer Baum.

Behauptung: τ_∞ ist vollständig!

Beweis:

1. $\tau_\infty = \tau_k$ für ein $k \in \mathbb{N}$.
   - Ist τ_k abgeschlossen, gilt die Behauptung.
   - Ist τ_k nicht abgeschlossen, so ist τ_k vollständig: Alle Formeln sind markiert und Γ muss endlich sein. Alle Formeln von Γ sind in den offenen Ästen von τ_k. Somit ist Γ nach Lemma 2.10 erfüllbar.
2.
   - Es gibt kein $k \in \mathbb{N}$ mit $\tau_\infty = \tau_k$. Dann ist τ_∞ ein unendlicher Baum.
   - Es gibt eine Folge von Knoten $\{Y_n\}, n \in \mathbb{N}$, die unendlich viele Nachfolger haben: Setze $Y_1 = A_1$, die Wurzel mit unendlich vielen Nachfolgerknoten. Ist Y_n bereits gefunden, dann hat Y_n entweder einen oder zwei direkte Nachfolger, von denen einer unendlich viele Nachfolger hat. Wähle als Y_{n+1} diesen Knoten. Dann ist der Ast $\{Y_n | n \in \mathbb{N}\}$ in τ_∞, offen, vollständig und enthält Γ, d.h. Γ ist erfüllbar.

Bemerkung und Folgerung

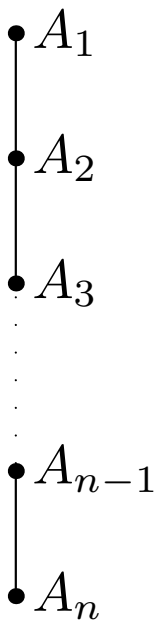
Bemerkung 2.12

1. *Ist Γ eine rekursiv aufzählbare Menge, so ist das Hinzufügen einer Formel $A_n \in \Gamma$ zu einem Tableau effektiv, d.h. falls Γ rekursiv aufzählbar aber nicht erfüllbar ist, so stoppt die systematische Tableau-Konstruktion. Insbesondere stoppt die systematische Tableau-Konstruktion immer, wenn Γ endlich ist. Sie liefert dann entweder:*
   - *Γ ist nicht erfüllbar, d.h. es gibt eine $n \in \mathbb{N}$, so dass τ_n abgeschlossen ist, oder:*
   - *Γ ist erfüllbar und die (offenen) Äste von τ_n liefern alle Belegungen, die Γ erfüllen.*

   Die systematische Tableau-Konstruktion liefert also für endliche Mengen in den offennen vollständigen Äste alle Belegungen der wesentlichen Variablen, die Γ erfüllen.

Folgerungen (Fort.)

2. *Zur Vereinfachung der systematischen Tableau-Konstruktion für eine Menge* $\Gamma = \{A_1, \ldots, A_n\}$ *beginne mit*



als Anfangstableau.

Folgerungen (Fort.)

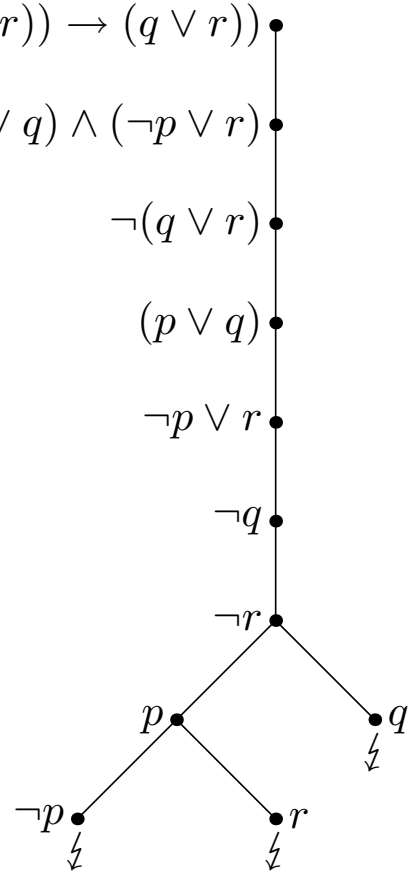
3.

$$\begin{aligned} \Gamma \models A &\iff \Gamma \cup \{\neg A\} \textit{ unerfüllbar} \\ &\iff \tau_{\{\Gamma, \neg A\}} \textit{ enthält abgeschlossenes Tableau} \\ &\iff \Gamma \vdash_\tau A \end{aligned}$$

Für Γ endlich beginne also mit Anfangstableau für $\{\neg A, A_1, \ldots, A_n\}$

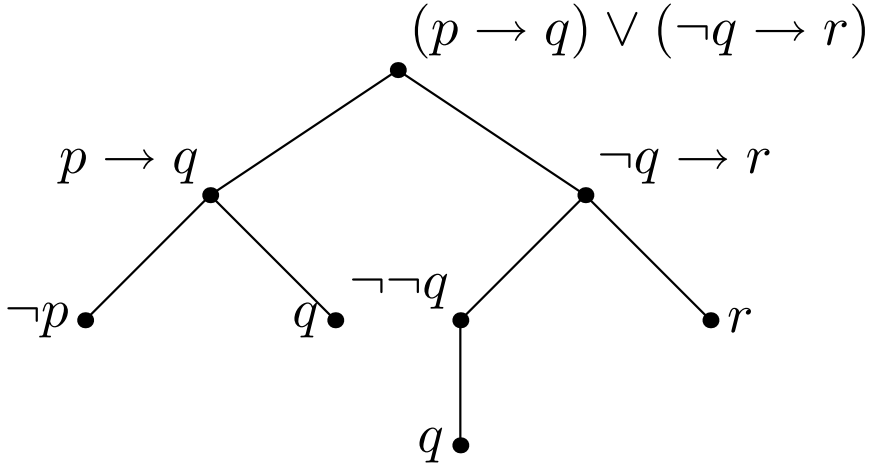
Folgerungen (Fort.)

4. •(a) $\models ((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$ *oder*
$(p \vee q) \wedge (\neg p \vee r) \models (q \vee r)$



Folgerungen (Fort.)

- (b) *Bestimme alle Belegungen, die* $A \equiv (p \to q) \vee (\neg q \to r)$ *erfüllen!*



Demnach ist $\{\varphi \mid \varphi$ *ist Bewertung mit* $\varphi(p) = 0$ *oder* $\varphi(q) = 1$ *oder* $\varphi(r) = 1\}$ *die Menge aller Belegungen, die A erfüllen. An den Blättern des Baumes lässt sich auch eine äquivalente Disjunktive Normalform (DNF) zur Formel A ablesen, nämlich* $\neg p \vee q \vee r$.

Normalformen

- Normalformen haben oft den Vorteil, dass man aus Formeln in dieser Form gewisse Informationen leicht ablesbar sind. So lassen sich z.B. aus einer KDNF (kanonische disjunktive Normalform) alle erfüllende Belegungen aus den Elementarkonjunktionen direkt ablesen. Aus minimalen DNF lassen sich leicht die Schaltnetze (mit UND, ODER, NEG Gattern) herleiten. Die systematische Tableux Konstruktion erlaubt es diese Normalformen aus einem vollständigen Tableau abzulesen.

Normalformen

Motivation: Oft will man eine beliebige A-Form in eine Form transformieren die „einfachere“ Gestalt hat und spezielle Algorithmen zur Lösung einer bestimmten Fragestellung für Formeln in dieser Gestalt verfügbar sind. Die Transformation sollte nicht zu teuer sein, sonst würde sich der Aufwand dafür nicht lohnen.

- Transformiert werden kann in einer
  - logisch äquivalenten Formel, d.h. $A \models\!\!\!\models T(A)$ oder
  - erfüllungs äquivalenten Formel, d.h.
    A erfüllbar gdw. $T(A)$ erfüllbar
- Wir behandeln drei dieser Normalformen:
  - Negationsnormalform (**NNF**) Form in $\neg, \vee, \wedge$
  - Konjunktive Normalform (**KNF**) Form in $\neg, \vee, \wedge$
  - Disjunktive Normalform (**DNF**) Form in $\neg, \vee, \wedge$

Normalformen

Definition 2.13 (NNF)

Eine Formel A ist in NNF gdw. jedes Negationszeichen direkt vor einem Atom (A-Variable) steht und keine zwei Negationszeichen direkt hintereinander stehen. Also:

1. Für $p \in V$ sind p und $\neg p$ in NNF
2. Sind A, B in NNF, so auch $(A \vee B)$ und $(A \wedge B)$

Beachte $(A \rightarrow B)$ wird durch $(\neg A \vee B)$ und $\neg(A \rightarrow B)$ durch $(A \wedge \neg B)$ ersetzt.

Lemma 2.14

Zu jeder Formel $A \in F$ gibt es eine logisch äquivalente Formel $B \in F(\neg, \vee, \wedge)$ in NNF mit $|B| \in O(|A|)$.

Beweis:

Übung. Verwende Doppelnegationsregel, de Morgan. ■

Klauseln

Definition 2.15 (Klausel)

Eine Formel $A \equiv (L_1 \vee ... \vee L_n)$ mit Literalen L_i für $i = 1, ..., n$ wird **Klausel** genannt.

- Sind alle Literale einer Klausel negativ, so ist es eine **negative** Klausel. Sind alle Literale positiv, so ist es eine **positive** Klausel. Klauseln die maximal ein positives Literal enthalten, heißen **Horn Klauseln**.
- A wird k-Klausel genannt, falls A maximal k Literale enthält. 1-Klauseln werden auch Unit-Klauseln genannt.
- Eine Formel A ist in **KNF** gdw. A eine Konjunktion von Klauseln ist. D.h.
  $A \equiv (A_1 \wedge ... \wedge A_m)$ mit Klauseln A_i für $i = 1, ..., m$.
- Sind die A_i k-Klauseln, so ist A in k-**KNF**.

Normalformen (Fort.)

Beispiel 2.16

$A \equiv (p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee q) \wedge (\neg p \vee \neg q)$ ist in 2-KNF (Beachte ist unerfüllbar). Betrachtet man Klauseln als Mengen von Literalen, so lassen sich Formeln in KNF als Mengen von Literalmengen darstellen, etwa
$A \equiv \{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}, \{\neg p, \neg q\}\}$.

Lemma 2.17

Zu jeder Formel $A \in F$ gibt es eine logisch äquivalente Formel B in KNF mit $|B| \in O(2^{|A|})$.

- Beachte: Es gibt eine Folge von Formeln A_n mit $|A_n| = 2n$, für die jede logisch äquivalente Formel B_n in KNF mindestens die Länge 2^n besitzt.

Definition 2.18 (DNF)

Eine A-Form A ist in **DNF** gdw. A eine Disjunktion von Konjunktionen von Literalen ist, d.h. $A \equiv (A_1 \vee ... \vee A_i)$ mit $A_i \equiv (L_{i1} \wedge ... \wedge L_{im_i})$.

Definition 2.19 (Duale Formel)

Die **duale Formel** von A, $d(A)$ (auch A^*) ist definiert durch:

- $d(p) \equiv p \quad$ für $p \in V$
- $d(\neg A) \equiv \neg d(A)$
- $d(B \vee C) \equiv (d(B) \wedge d(C))$
- $d(B \wedge C) \equiv (d(B) \vee d(C))$

Lemma 2.20

Für jede A-Form A gilt:

1. *Sei A in KNF, dann ist NNF($\neg A$) in DNF.*
2. *Ist A in KNF, so ist $d(A)$ in DNF.*
3. *A ist Tautologie gdw. $d(A)$ widerspruchsvoll.*
4. *A ist erfüllbar gdw. $d(A)$ ist keine Tautologie.*

Setzt man $\varphi'(p) = 1 - \varphi(p)$, so gilt $\varphi'(d(A)) = 1 - \varphi(A)$

Davis-Putman-Algorithmen

- Erfüllbarkeits-Algorithmen
- Formeln in NNF ($\neg, \wedge, \vee$)
- Bottom-Up Verfahren - Festlegung einer erfüllenden Bewertung durch Auswahl der Werte der Atome

Definition 2.21

Sei A A-Form, $p \in \mathbb{V}$ definiere $A[p/1]$ (bzw. $A[p/0]$) als Ergebnis des folgenden Ersetzungsprozesses:

1. Ersetze in A jedes Vorkommen von p durch 1.
2.
   - Tritt nun eine Teilform $\neg 1$ auf, ersetze sie durch 0,
   - $\neg 0$ ersetze durch 1.
   - Teilformeln $B \wedge 1$, sowie $B \vee 0$ werden durch B ersetzt,
   - Teilformeln $B \vee 1$ durch 1 und
   - Teilformeln $B \wedge 0$ durch 0 ersetzt.
3. Schritt 2 wird so lange durchgeführt, bis keine weitere Ersetzung möglich ist.

Analog für $A[p/0]$.

Allgemeiner verwende $A[l/1]$ bzw. $A[l/0]$ für Literale l.

- **Beachte**: Für A in KNF und Literal l gilt:
  $A[l/1]$ entsteht aus A durch Streichen aller Klauseln, die das Literal l enthalten und durch Streichen aller Vorkommen des Literals $\neg l$ in allen anderen Klauseln.
- $A[p/1]$ (bzw. $A[p/0]$) sind wohldefiniert. (Warum ?)
- Als Ergebnis des Ersetzungsprozesses $A[p/i]$ $i = 1, 0$ erhält man:
  - eine A-Form (in NNF bzw. KNF wenn A diese Form hatte)
  - 1 die „leere Formel“
  - 0 die „leere Klausel“ ($\sqcup, \square$)

  Die leere Formel wird als wahr interpretiert. Die leere Klausel als falsch (nicht erfüllbar), d. h. $A[p/i]$ als A-Form behandelbar

Für jedes Atom $p \in \mathbb{V}$ und $A \in F$ gilt:

1. $A[p/i]$ $i \in 1, 0$ ist entweder die leere Formel oder die leere Klausel oder eine A-Form in NNF in der p nicht vorkommt.
2. $A \wedge p \models\!\!\!\models A[p/1] \wedge p$ $\qquad\qquad$ $A \wedge \neg p \models\!\!\!\models A[p/0] \wedge \neg p$
3. A ist erfüllbar gdw $A[p/1] = 1$ oder $A[p/0] = 1$ oder eine der Formeln $A[p/1], A[p/0]$ erfüllbar ist.

$\hookrightarrow$ Durch Testen der Teilbewertungen $A[p/1]$ und $A[p/0]$ kann rekursiv die Erfüllbarkeit von A entschieden werden.

Regelbasierter Aufbau von DP-Algorithmen

Definition 2.22 (**Regeln für Formeln in NNF**)

1. **Pure-Literal Regel** Kommt ein Atom $p \in \mathbb{V}$ in einer A-Form A nur positiv oder nur negativ vor, so können wir p mit 1 bzw. 0 belegen und die Formel dementsprechend kürzen.

   $\hookrightarrow$ (Es gilt $A[p/0] \models A[p/1]$ bzw. $A[p/1] \models A[p/0]$), genauer A erfüllungsäquivalent $A[p/1]$ bzw. $A[p/0]$.

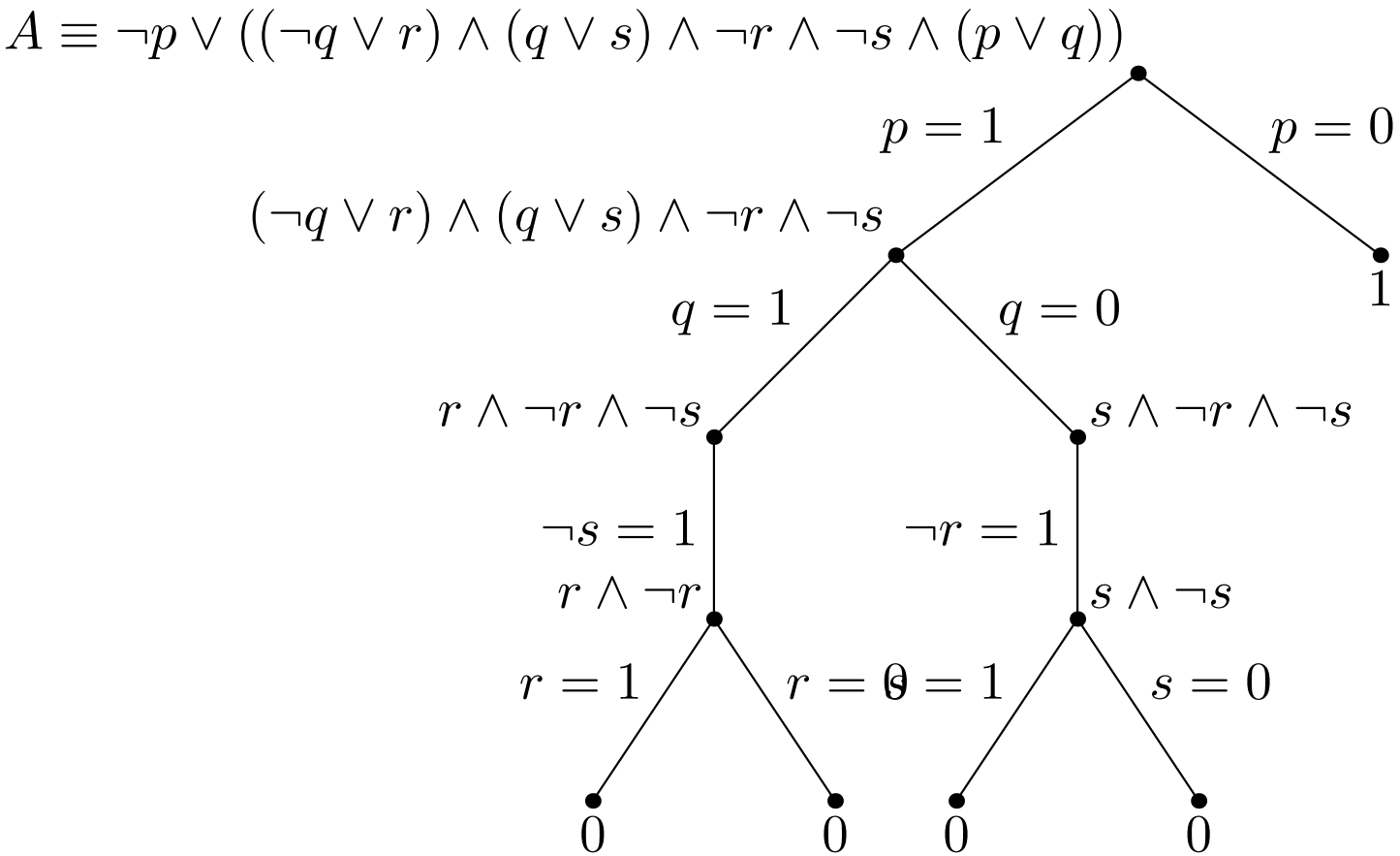
2. **Splitting-Regel** Kommt jedes Atom sowohl positiv als auch negativ vor, so wähle ein solches Atom p in A aus und bilde aus A die zwei A-Formen $A[p/1]$ und $A[p/0]$.

   $\hookrightarrow$ Die Ausgangsformel A ist genau dann erfüllbar, wenn bei einer der Kürzungen der Wert 1 oder eine erfüllbare Formel als Ergebnis auftritt.

Regelbasierter Aufbau von DP-Algorithmen

- Regeln reduzieren das Erfüllbarkeitsproblem für eine Formel mit n Atomen auf EP für Formeln mit maximal $(n-1)$ Atomen.
- Algorithmen, die mit Hilfe dieser beiden Regeln mit verschiedenen Heuristiken (zur Auswahl des splitting Atoms) und weiteren Verfeinerungen arbeiten, werden als **Davis-Putman-Algorithmen** bezeichnet.

Beispiel 2.23 (Darstellung der Abarbeitung als Baum)



Weitere Verfeinerungen

Definition 2.24 (Regeln für Formeln in KNF)

3. **Unit-Regel**

Sei A in KNF und A enthält eine Unit Klausel $A_i \equiv l$. Bilde $A[l/1]$ (A ist erfüllbar gdw $A[l/1]$ erfüllbar), da das Literal einer Unit-Klausel durch eine erfüllende Bewertung auf wahr gesetzt werden muss.

$(\neg q \vee r) \wedge (q \vee s) \wedge \neg r \wedge \neg s$

$\neg r = 1$

$\neg q \wedge (q \vee s) \wedge \neg s$

$\neg q = 1$

$s \wedge \neg s$

$s = 1$

0

- Seien A_1 und A_2 Klauseln. A_1 **subsumiert** A_2 ($A_1 \subseteq A_2$) gdw jedes Literal aus A_1 auch in A_2 auftritt.
- Aus der Erfüllbarkeit einer Klausel folgt sofort die Erfüllbarkeit aller Klauseln, die sie subsummiert.

4. **Subsumption-Rule**
   Sei A in KNF. Streiche aus A alle Klauseln, die von anderen subsumiert werden:: $SR(A)$.
   - Da in KNF alle Klauseln konjunktiv verknüpft sind, braucht man nur diejenigen zu berücksichtigen, die von keiner anderen subsumiert werden.

procedure Davis/Putman
//Eingabe: A in KNF//
//Ausgabe: Boolscher Wert für Erfüllbarkeit (1,0)//
begin
if $A \in \{0,1\}$ **then** **return**(A);
p:=**pure**(A,s);
//*liefert Atom und Belegung, falls nur positiv oder nur negativ vorkommt sonst* **null**//
if $p \neq$ **null** **then** **return**(DPA(A[p/s]));
p:=**unit**(A,s); //*Unit Klausel mit Belegung sonst* **null**//
if $p \neq$ **null** **then** **return**(DPA(A[p/s]));
A:=Subsumption_Reduce(A); //*entfernt subs. Klauseln*//
p:=**split**(A); //*liefert Atom in A*//
if DPA($A[p/1]) = 1$ **then** **return**(1);
return(DPA(A[p/0]));
end

Auswahlkriterien für die Splitting Regel

- Wähle das erste in der Formel vorkommende Atom,
- wähle das Atom, welches am häufigsten vorkommt,
- $\cdots$ das in den kürzesten Klauseln am häufigsten vorkommt,
- wähle Atom mit $\sum_{p \text{ in } A_i} |A_i|$ minimal,
- berechne die Anzahl der positiven und negativen Vorkommen in den kürzesten Klauseln und wähle das Atom mit der größten Differenz.

Resolutions-Verfahren

- Das **Resolutionskalkül** als Deduktionssystem operiert auf Klauselmengen, d. h. Formeln in KNF mit nur einer Schlussregel:
  Aus Klauseln $(A \vee l)$ und $(B \vee \neg l)$ kann eine neue Klausel $(A \vee B)$ erzeugt werden.
- **Ziel:** Leere Klausel zu erzeugen.

- Klauseln als Mengen $(p \vee \neg q \vee p) \leftrightarrow \{p, \neg q\}$
  $l \equiv p$ so $\neg l \equiv \neg p$ $\quad$ $l \equiv \neg p$ so $\neg l \equiv p$

Resolutions-Verfahren

Definition 2.25 (**Resolutionsregel** (Resolventenregel))

Seien A, B Klauseln, l ein Literal. l kommt in A und $\neg l$ kommt in B vor. Dann können A und B über l (bzw. $\neg l$) resolviert werden.

- Die **Resolvente** der Klauseln A und B ist die Klausel $(A \backslash \{l\}) \cup (B \backslash \{\neg l\})$.
  A und B sind die Elternklauseln der Resolvente

Schema $\dfrac{A \quad , \quad B}{Res_l(A, B) \equiv (A \backslash \{l\}) \cup (B \backslash \{\neg l\})}$ (Resolutionsregel)

Eigenschaften der Resolvente

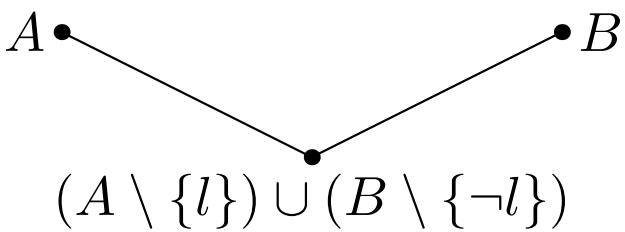
Beachte:

- Enthält die Resolvente ein Literal l', so muss dieses bereits in A oder B enthalten sein.
- Schreibe auch $A \vee l,\ B \vee \neg l \underset{Res}{\vdash} A \vee B$.
- $$A \wedge B \text{ erfüllbar} \quad \text{gdw} \quad A \wedge B \wedge Res_l(A,B) \text{ erfüllbar.}$$
  $$\text{gdw} \quad Res_l(A,B) \text{ erfüllbar.}$$
- $A \wedge B \models Res(A,B)$.
- Resolvente kann leere Klausel $\square$ sein.

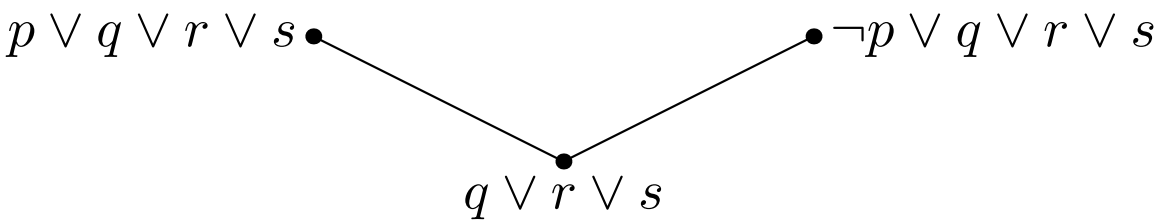
Darstellung - Beispiele

Beispiel 2.26

Darstellung für Klauseln A, B, die über l resolvieren

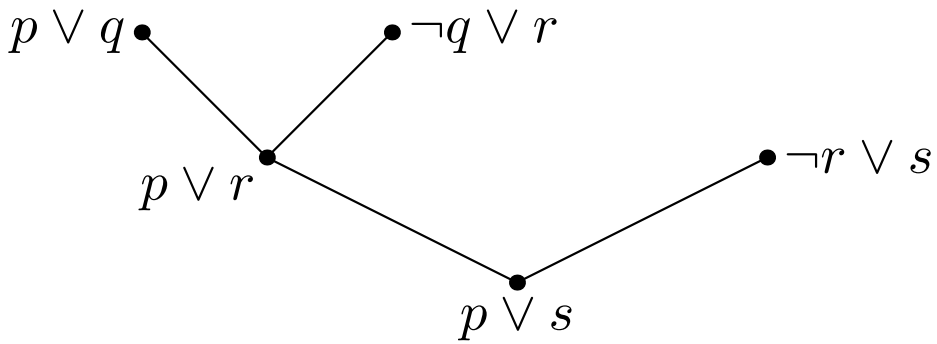


a) Formel $A = \{(p \vee q \vee r \vee s), (\neg p \vee q \vee r \vee s)\}$

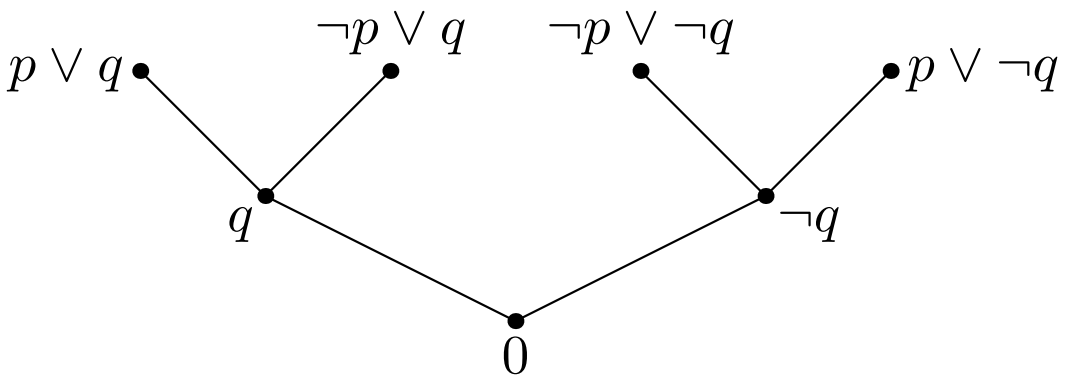


Beispiele

b) $A \equiv \{(p \vee q), (\neg q \vee r), (\neg r \vee s)\}$

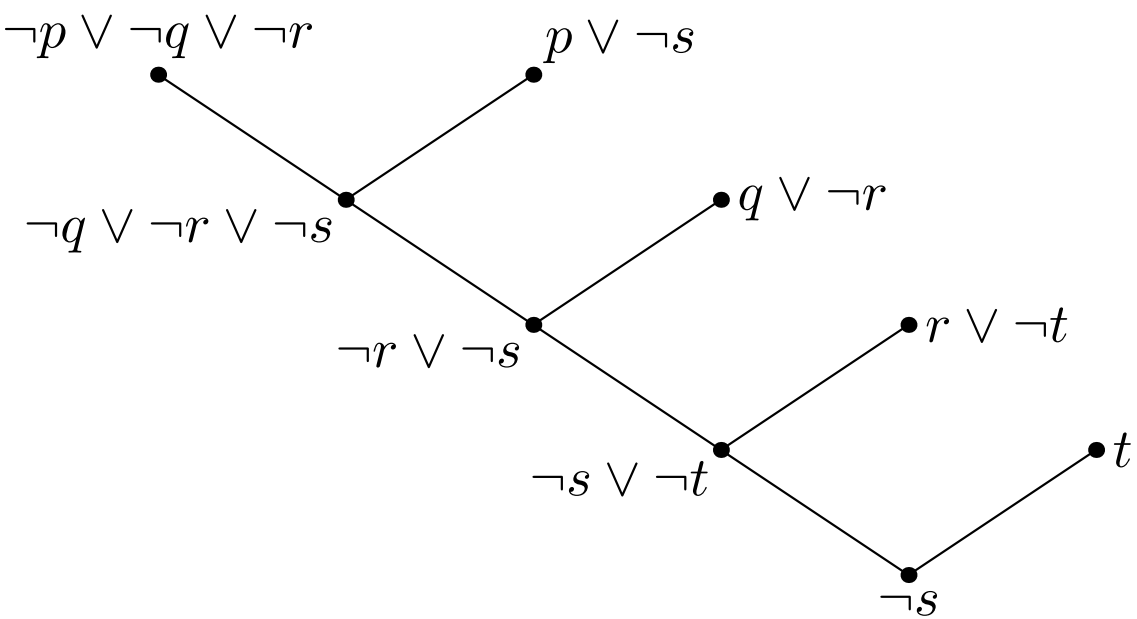


c) $A \equiv \{(p \vee q), (\neg p \vee q), (p \vee \neg q), (\neg p \vee \neg q)\}$



Beispiele

d) $A \equiv \{(\neg p \vee \neg q \vee \neg r), (p \vee \neg s), (q \vee \neg r), (r \vee \neg t), t\}$
(Horn-Klauseln)



Ableitungen im Resolutionskalkül

Definition 2.27 (Herleitungen (Ableitungen))

Sei $A \equiv \{C_1, \ldots, C_n\}$ eine Formel in KNF und C ein Klausel. Eine Folge $D_1, \ldots, D_k$ von Klauseln ist eine **Herleitung** der Klausel C aus A. Wenn $C \equiv D_k$ und für alle j mit $1 \leq j \leq k$ Klauseln $E, F \in A \cup \{D_1, \ldots, D_{j-1}\}$ existieren mit $E, F \underset{Res}{\vdash} D_j$.

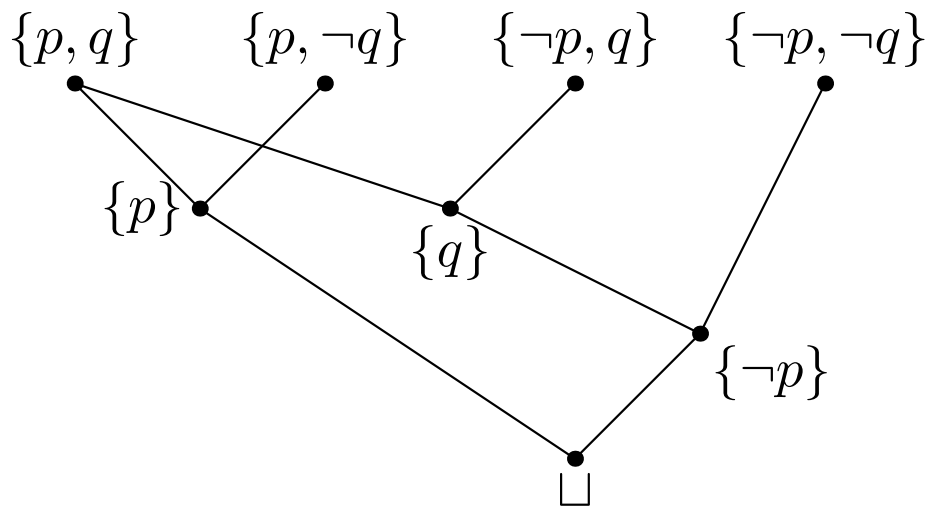
- C ist (mit der Resolventenregel oder im Resolutionskalkül) **herleitbar** aus A

  Schreibweise: $A \overset{+}{\underset{Res}{\vdash}} C$ (A Ausgangs-Klauseln)
- k ist die **Länge** der Herleitung.

- **Minimale Herleitungen** sind solche für die kein Schritt weggelassen werden kann.

↪ Gilt $A \overset{+}{\underset{Res}{\vdash}} C_1$ und $A \overset{+}{\underset{Res}{\vdash}} C_2$, so schreibe $A \overset{+}{\underset{Res}{\vdash}} C_1, C_2$.

- Darstellung von Herleitungen mit Hilfe von **DAG's**.



Korrektheit und Widerlegungsvollständigkeit

Satz 2.28

1. *Der Resolutionskalkül ist korrekt.*

   A in KNF, C Klausel dann $A \overset{+}{\underset{Res}{\vdash}} C$, *so* $A \models C$

2. *Der Resolutionskalkül ist nicht vollständig.*

   Es gibt A in KNF, C Klausel mit $A \models C$ *aber nicht* $A \overset{+}{\underset{Res}{\vdash}} C$

3. *Der Resolutionskalkül ist widerlegungsvollständig.*

   A in KNF, A widerspruchsvoll (unerfüllbar), so $A \overset{+}{\underset{Res}{\vdash}} \square$

Korrektheit und Widerlegungsvollständigkeit (Forts.)

Beweis:

1. $\surd$, 2. $A \equiv p$, $C \equiv p \vee q \rightsquigarrow$ Behauptung.

3. Induktion nach Länge der Formel:

Kürzeste Formel:$\{(p), (\neg p)\}$, dann $p, \neg p \underset{Res}{\vdash} \sqcup$.

Verwende dabei: A widerspruchsvoll, so auch $A[p/1]$ und $A[p/0]$ widerspruchsvoll.

- Sei A mit Länge $n+1$, A widerspruchsvoll. Es gibt ein Atom p in A das sowohl positiv als auch negativ vorkommt. Betrachte $A[p/1]$ und $A[\neg p/1]$, beide nicht erfüllbar. Angenommen nicht Wert 0.
- Nach Ind.Vor.: $A[p/1] \overset{+}{\underset{Res}{\vdash}} \sqcup$, $A[p/0] \overset{+}{\underset{Res}{\vdash}} \sqcup$.

Korrektheit und Widerlegungsvollständigkeit (Forts.)

- A in KNF. $A[p/1]$ entsteht durch Streichen der Klauseln, die p enthalten und durch Streichen von $\neg p$ aus Klauseln, die $\neg p$ enthalten.

$\hookrightarrow$ Fügt man in $A[p/1]$ die eliminierten Literale $\neg p$ und zu $A[\neg p/1]$ die Atome p wieder hinzu, so sind diese Formeln $A[p/1](\neg p)$ und $A[p/0](p)$ Teilformen von A.

Korrektheit und Widerlegungsvollständigkeit (Forts.)

- Dann aber entweder $A[p/1](\neg p) \overset{+}{\underset{Res}{\vdash}} \sqcup$ bzw. $A[\neg p/1](p) \overset{+}{\underset{Res}{\vdash}} \sqcup$ auch aus A herleitbar oder
  $A[p/1](\neg p) \overset{+}{\underset{Res}{\vdash}} \neg p$ und $A[\neg p/1](p) \overset{+}{\underset{Res}{\vdash}} p$.
  Dann aber $\neg p, p \underset{Res}{\vdash} \sqcup$ und somit $A \overset{+}{\underset{Res}{\vdash}} \sqcup$.
- Ergibt $A[p/1]$ oder $A[\neg p/1]$ den Wert 0, so enthält A eine Klausel p (falls $A[\neg p/1] = 0$) oder eine Klausel $\neg p$ (falls $A[p/1] = 0$). Dann analoger Schluss. ■

Lemma 2.29

Sei A in KNF, C eine Klausel, dann gilt $A \models C$ gdw *es gibt eine Teilklausel $C' \subseteq C$:*

$$A \overset{+}{\underset{Res}{\vdash}} C'$$

- *Ist A erfüllbar und C Primimplikant von A, dann gilt $A \overset{+}{\underset{Res}{\vdash}} C$.*

Resolventenmethode: Strategien/Heuristiken

- **Verfeinerungen der Methode**

Starke Herleitungen: A in KNF, widerspruchsvoll.

Dann gibt es eine Herleitung $C_1, \ldots C_n \equiv \sqcup$ mit

1. in der Herleitung tritt keine Klausel mehrfach auf,
2. in der Herleitung tritt keine Tautologie auf,
3. in der Herleitung tritt keine schon subsumierte Klausel auf: Es gibt kein C_i, C_j, $j < i$, $C_j \subseteq C_i$.

- **Strategien, Heuristiken**
  - Stufenstrategie (Resolutionsabschluss)
    (Alle erfüllende Bewertungen)
  - Stützmengenrestriktion
    (Set of Support, Unit-Klauseln bevorzugen)
  - P-N Resolution
  - Lineare Resolution (SL-Resolution)
    (PROLOG-Inferenzmaschine)

Beispiel: $A \equiv \{\{\neg p, \neg q, \neg r\}, \{p, \neg s\}, \{q, \neg r\}, \{r, \neg t\}, \{t\}\}$
Stufen:

0	1	2	3
1. $\{\neg p, \neg q, \neg r\}$	6. $\{\neg q, \neg r, \neg s\}$ (1,2)	11. $\{\neg r, \neg s\}$ (6,3)	21. $\{\neg s, \neg r\}$ (11,4)
2. $\{p, \neg s\}$	7. $\{\neg p, \neg r\}$ (1,3)	12. $\{\neg q, \neg s, \neg t\}$ (6,4)	22. $\{\neg s\}$ (11,10)
3. $\{q, \neg r\}$	8. $\{\neg p, \neg q, \neg t\}$ (1,4)	13. $\{\neg p, \neg t\}$ (7,4)	⋮
4. $\{r, \neg t\}$	9. $\{q, \neg t\}$ (3,4)	14. $\{\neg p, \neg r, \neg t\}$ (8,3)	
5. $\{t\}$	10. $\{r\}$ (4,5)	15. $\{\neg p, \neg q\}$ (8,5)	
		16. $\{q\}$ (10,3)	
		17. $\{\neg r, \neg s, \neg t\}$ (6,9)	
		18. $\{\neg q, \neg s\}$ (6,10)	
		19. $\{\neg p\}$ (7,10)	
		20. $\{\neg p, \neg t\}$ (8,9)	

$$\varphi(q) = 1, p(p) = 0, \varphi(s) = 0, \varphi(r) = 1, \varphi(t) = 1$$