Algorithmischer Aufbau der Aussagenlogik

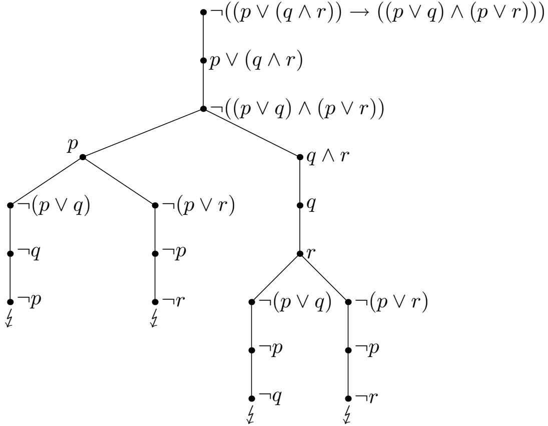
In diesem Abschnitt betrachten wir Verfahren die bei gegebener endlichen Menge Σ und A-Form A entscheiden ob $\Sigma \models A$ gilt. Die bisher betrachteten Verfahren prüfen alle Belegungen der in den Formeln vorkommenden Variablen oder zählen effektiv die Theoreme eines geeigneten deduktiven Systems 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:
$\Sigma \models A \quad \mathrm{gdw} \quad \{\Sigma, \neg A\}$ unerfüllbar:
Semantische Tableaux   Davis-Putnam   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:

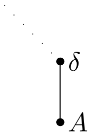
   $$\bullet A$$

   (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.)

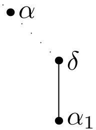
Graphisch:



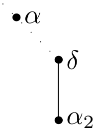
Formalisierung (Fort.)

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.

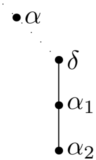
   Graphisch:



oder

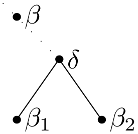


   In der Praxis werden jedoch an δ nacheinander die Knoten für beide Komponenten hinzugefügt:



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

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

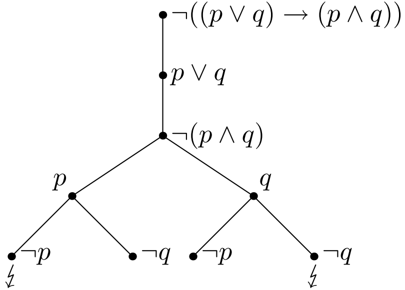
$$\neg(A \to (B \to A))$$
$$A$$
$$\neg(B \to A)$$
$$B$$
$$\neg A$$
$$\lightning$$

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

$$\neg(\neg(p \wedge q) \to (\neg p \vee \neg q))$$
$$\neg(p \wedge q)$$
$$\neg(\neg p \vee \neg q)$$
$$\neg\neg p$$
$$\neg\neg q$$
$$p$$
$$q$$
$$\neg p \qquad\qquad \neg q$$
$$\lightning \qquad\qquad \lightning$$

Beispiel 2.5

$\vdash_\tau (p \vee q) \to (p \wedge q)$ gilt <u>nicht</u>:

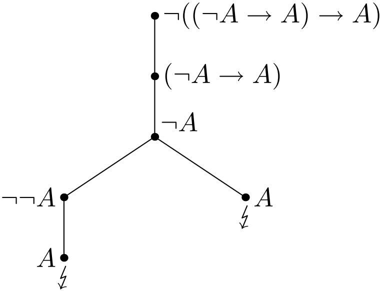


Es gibt Belegungen, die $\neg((p \vee q) \to (p \wedge q))$ erfüllen, nämlich φ mit $\varphi(p) = 1$ und $\varphi(q) = 0$ und φ' mit $\varphi'(p) = 0$ und $\varphi'(q) = 1$. Also gilt nicht $\vdash_\tau (p \vee q) \to (p \wedge q)$.

Beispiele (Fort.)

Beispiel 2.6

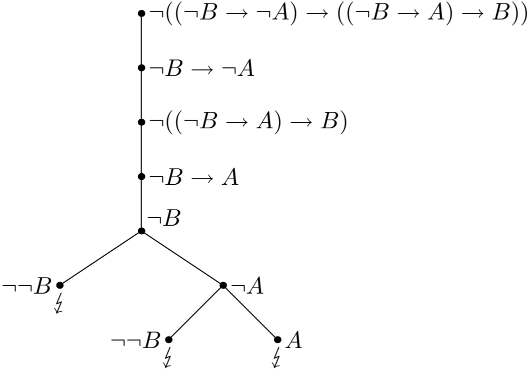
$\vdash_\tau (\neg A \rightarrow A) \rightarrow A$



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:

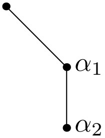
1. Γ *ist nicht erfüllbar* gdw τ_Γ *enthält ein abgeschlossenes Tableau.*
2. *Äquivalent sind*
   - $\Gamma \models A$ *(oder* $\Gamma \vdash A$*)*
   - $\tau_{\{\Gamma, \neg A\}}$ *enthält ein abgeschlossenes Tableau.*
3. *Äquivalent sind*
   - $\models A$ *(oder* $\vdash A$*)*
   - $\tau_{\neg A}$ *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 $\Gamma = \{A_1, A_2, \dots\}$.
  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!

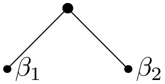
     τ_{n+1} entsteht aus τ_n wie folgt:
  3. Ist Y die erste unmarkierte α-Formel in τ_n, durch die ein offener Ast geht, so markiere Y und erweitere jeden offenen Ast, der durch Y geht, um die Teilformeln α_1 und α_2 von Y.



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*

$$\begin{array}{l} \bullet A_1 \\ | \\ \bullet A_2 \\ | \\ \bullet A_3 \\ \vdots \\ \bullet A_{n-1} \\ | \\ \bullet A_n \end{array}$$

als Anfangstableau.

Folgerungen (Fort.)

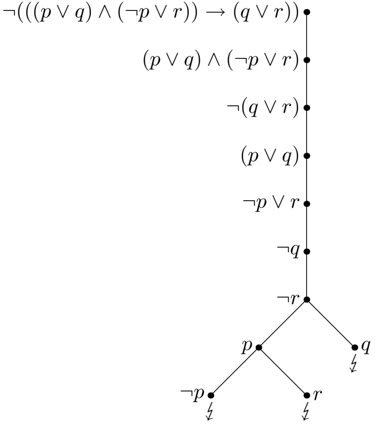
3.

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

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

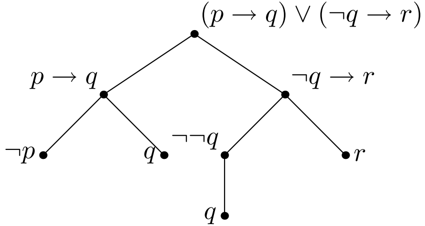
Folgerungen (Fort.)

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. $A \models\!\!\!\!\!\!=\!\!| \; T(A)$ 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 $A \in F(\{\neg, \wedge, \vee, \rightarrow\})$ 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 $A \equiv (L_1 \vee \dots \vee L_n)$ mit Literalen L_i für $i = 1, \dots, n$ 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.
  $A \equiv (A_1 \wedge \dots \wedge A_m)$ mit Klauseln A_i für $i = 1, \dots, m$.
- Sind die A_i k-Klauseln, so ist A in k-**KNF**.

Normalformen (Fort.)

Beispiel 2.16

$A \equiv (p \lor q) \land (p \lor \neg q) \land (\neg p \lor q) \land (\neg p \lor \neg q)$ 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 $A \in F(\{\neg, \land, \lor\})$ 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. $A \equiv (A_1 \vee \ldots \vee A_i)$ mit $A_i \equiv (L_{i1} \wedge \ldots \wedge L_{im_i})$.

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-Putnam-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 in NNF, $p \in \mathbb{V}$ 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“ ($\sqcup, \Box$)

  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. Ist φ eine Bewertung mit $\varphi(p) = i$, so gilt $\varphi(A) = \varphi(A[p/i])$.
2. $A \wedge p \models\mathrel{\mkern-3mu}\Dashv A[p/1] \wedge p$ $\qquad$ $A \wedge \neg p \models\mathrel{\mkern-3mu}\Dashv A[p/0] \wedge \neg p$
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**)

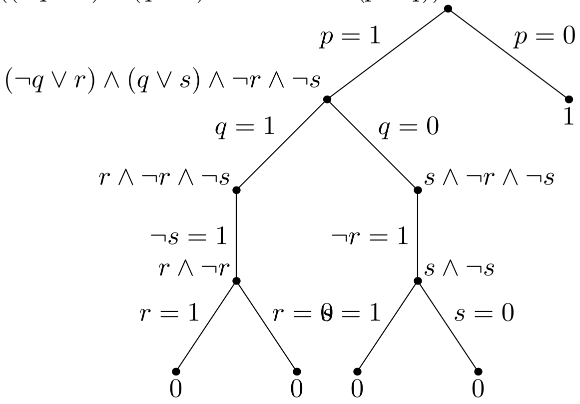
1. **Pure-Literal Regel** Kommt ein Atom $p \in \mathbb{V}$ in einer A-Form A nur positiv oder nur negativ vor, so können wir p mit 1 bzw. 0 belegen und die Formel dementsprechend kürzen.
   - $\hookrightarrow$ (Es gilt $A[p/0] \models A[p/1]$ bzw. $A[p/1] \models A[p/0]$), genauer A erfüllungsäquivalent $A[p/1]$ 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-Putnam-Algorithmen** bezeichnet.

Beispiel 2.23 (Darstellung der Abarbeitung als Baum)

$A \equiv \neg p \vee ((\neg q \vee r) \wedge (q \vee s) \wedge \neg r \wedge \neg s \wedge (p \vee q))$



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.

$(\neg q \lor r) \land (q \lor s) \land \neg r \land \neg s$

$\neg r = 1$

$\neg q \land (q \lor s) \land \neg s$

$\neg q = 1$

$s \land \neg s$

$s = 1$

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:: $SR(A)$.
   Streiche insbesondere tautologische Klauseln (solche die p und $\neg p$ für ein p enthalten).
   - Da in KNF alle Klauseln konjunktiv verknüpft sind, braucht man nur diejenigen zu berücksichtigen, die von keiner anderen subsumiert werden.

```
procedure Davis/Putnam
//Eingabe: A in KNF//
//Ausgabe: Boolscher Wert für Erfüllbarkeit (1,0)//
begin
if A ∈ {0,1} then return(A);
p:=pure(A,s);
//liefert Atom und Belegung, falls nur positiv oder nur negativ
   vorkommt sonst  null//
if p ≠ null then return(DPA(A[p/s]));
p:=unit(A,s);   //Unit Klausel mit Belegung sonst null//
if p ≠ 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 $\neg l \equiv \neg p$ $\qquad$ $l \equiv \neg p$ 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 $$\frac{A \qquad , \qquad B}{Res_l(A, B) \equiv (A \backslash \{l\}) \cup (B \backslash \{\neg l\})}$$ (Resolutionsregel)

Eigenschaften der Resolvente

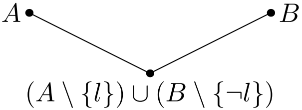
Beachte:

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

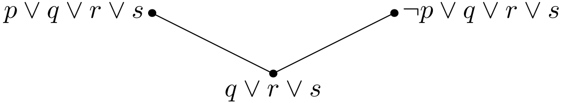
Darstellung - Beispiele

Beispiel 2.26

Darstellung für Klauseln A, B, die über l resolvieren

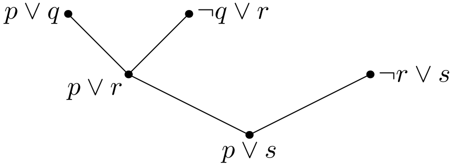


a) Formel $F \equiv \{(p \vee q \vee r \vee s), (\neg p \vee q \vee r \vee s)\}$

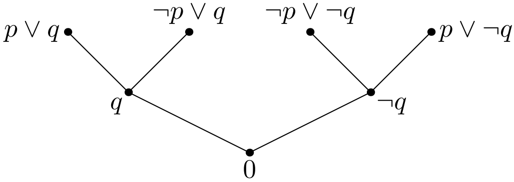


Beispiele

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

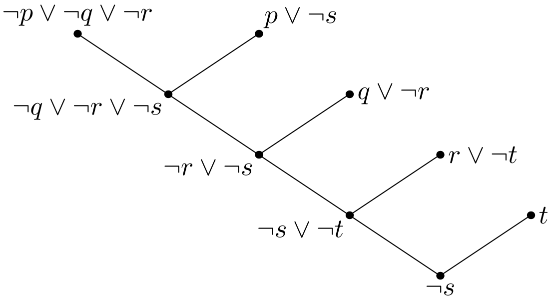


c) $F \equiv \{(p \vee q), (\neg p \vee q), (p \vee \neg q), (\neg p \vee \neg q)\}$



Beispiele

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



Ableitungen im Resolutionskalkül

Definition 2.27 (Herleitungen (Ableitungen))

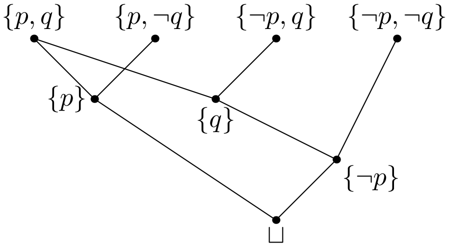
Sei $A \equiv \{C_1, \ldots, C_n\}$ eine Formel in KNF und C ein Klausel. Eine Folge $D_1, \ldots, D_k$ von Klauseln ist eine **Herleitung** der Klausel C aus A. Wenn $C \equiv D_k$ und für alle j mit $1 \leq j \leq k$ Klauseln $E, F \in A \cup \{D_1, \ldots, D_{j-1}\}$ existieren mit $E, F \underset{Res}{\vdash}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

- 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 $A \underset{Res}{\overset{+}{\vdash}} C$, *so* $A \models C$
2. *Der Resolutionskalkül ist nicht vollständig.*
   Es gibt A in KNF, C Klausel mit $A \models C$ *aber nicht* $A \underset{Res}{\overset{+}{\vdash}} C$
3. *Der Resolutionskalkül ist widerlegungsvollständig.*
   A in KNF, A widerspruchsvoll (unerfüllbar), so $A \underset{Res}{\overset{+}{\vdash}} \sqcup$

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 $C_1, \dots C_n \equiv \sqcup$ 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)	$\vdots$
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)	

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