Algorithmische Verfahren für die Aussagenlogik

Algorithmische Verfahren für die Aussagenlogik

Wir betrachten Verfahren, die bei gegebener endlicher Menge $\Sigma \subseteq F$ und $A \in F$ entscheiden, ob $\Sigma \models A$ gilt.

Die bisher betrachteten Verfahren prüfen alle Belegungen der in den Formeln vorkommenden Variablen oder zählen die Theoreme eines geeigneten deduktiven Systems auf. Dies ist sehr aufwendig.

Nutze Erfüllbarkeitsprüfung/Satisfiability Check SAT:

$$\Sigma \models A \quad \text{gdw.} \quad \Sigma \cup \{\neg A\} \text{ unerfüllbar.}$$

Die Komplexität von Erfüllbarkeit bleibt weiterhin groß:

$$\mathrm{SAT} \text{ ist NP-vollständig.}$$

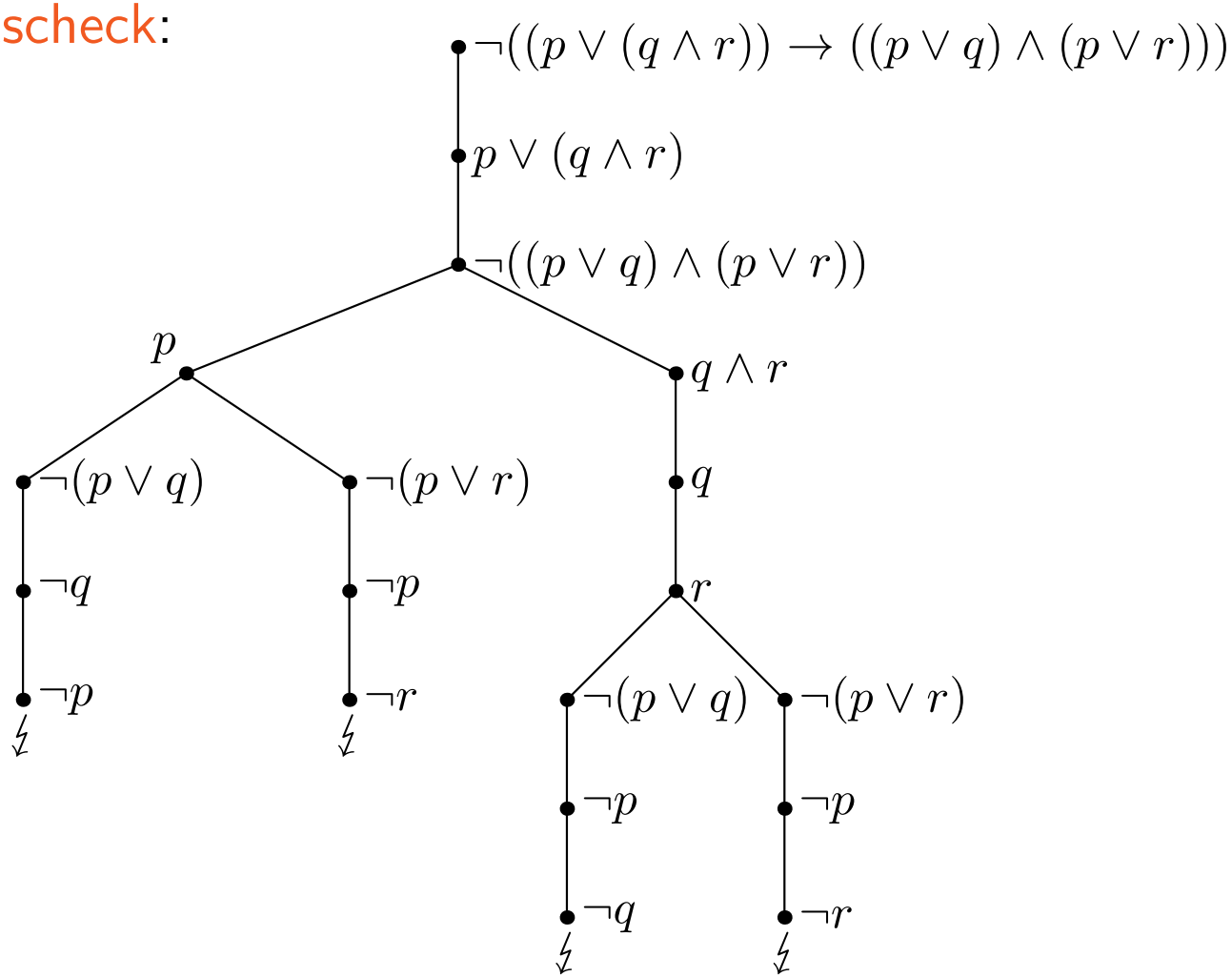
Suche nach Verfahren, die bei üblichen Eingaben schneller sind als die Brute-Force-Methode. Wir betrachten drei Verfahren:

Semantische Tableaus Davis-Putnam Resolution

Semantische Tableaus: Beispiel

Zeige, dass $\neg((p \vee (q \wedge r)) \rightarrow ((p \vee q) \wedge (p \vee r)))$ unerfüllbar ist.

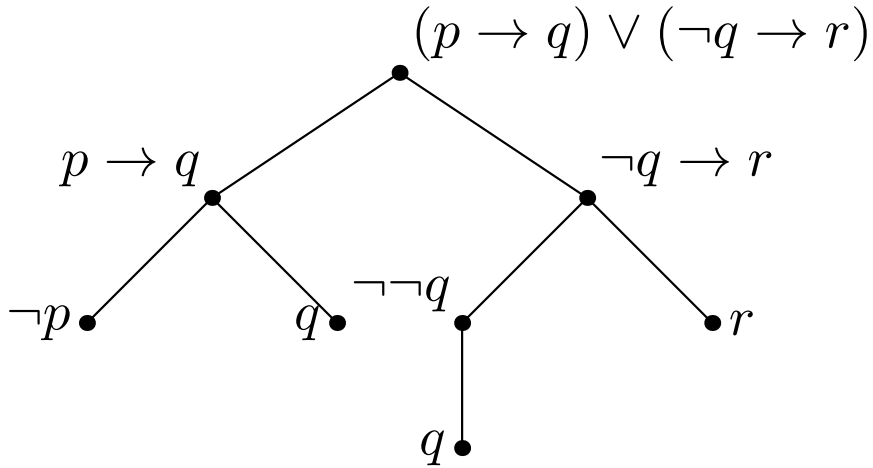
Erfüllbarkeitscheck:



Da alle Äste zu Widersprüchen führen, ist die Formel nicht erfüllbar.

Semantische Tableaus: Beispiel (Fort.)

Bestimme alle Belegungen, die $A \equiv (p \to q) \vee (\neg q \to r)$ erfüllen:



Demnach ist $\{\psi : F \to \mathbb{B} \mid \psi(p) = 0 \text{ oder } \psi(q) = 1 \text{ oder } \psi(r) = 1\}$ die Menge aller Belegungen, die A erfüllen.

An den Blättern lässt sich auch eine logisch äquivalente DNF ablesen, nämlich $\neg p \vee q \vee r$.

Intuition zu Tableaus

Ein Ast in einem Tableau ist ein Pfad von der Wurzel zu einem Blatt.

Die erfüllenden Belegungen der Wurzelformel sind die Vereinigung der erfüllenden Belegungen aller Äste.

- Für jede erfüllende Belegungen der Wurzel gibt es einen Ast in dem Tableau, so dass die Belegung alle Formeln auf dem Ast erfüllt.
- Umgekehrt bestimmt jeder erfüllbare Ast erfüllende Belegungen der Wurzelformel.

Trick: Sind Formeln maximal entfaltet (Tableau ist vollständig, s.u.), sind erfüllende Beleungen bzw. Widersprüche unmittelbar ersichtlich.

Definition von Tableaus

Zwei Arten von Formeln: β-Formeln führen zu Verzweigungen, α-Formeln führen nicht zu Verzweigungen:

- α-Formeln mit Komponenten α_1 und α_2 führen zu Folgeknoten mit Markierungen α_1 und α_2:

α	$\neg\neg A$	$A_1 \wedge A_2$	$\neg(A_1 \vee A_2)$	$\neg(A_1 \to A_2)$
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 führen zu Verzweigungen mit Knotenmarkierungen β_1 und β_2:

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

Beachte: Jede Formel in $F =_{def} F_{\{\neg,\vee,\wedge,\to\}}$ ist ein Literal (p oder $\neg p$ mit $p \in V$), eine α- oder β-Formel und genau von einem dieser drei Typen.

Definition von Tableaus (Fort.)

Definition 3.1 (Tableau)

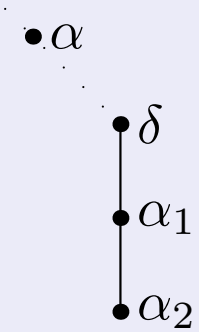
Tableaus sind binäre Bäume, deren Knoten mit Formeln aus F markiert sind. Die Menge der Tableaus T_A für $A \in F$ ist induktiv definiert durch:

(a) Wenn τ_A aus einem mit A beschrifteten Knoten besteht, gilt $\tau_A \in T_A$:

$$\bullet A$$

(b) Ist $\tau \in T_A$ und b ein mit δ markiertes Blatt von τ, dann lässt sich τ wie folgt zu einem Tableau $\tau' \in T_A$ erweitern:

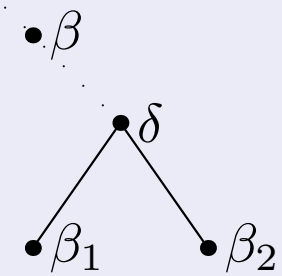
(α) Gibt es auf dem Ast zu b einen Knoten, der mit der α-Formel α markiert ist, hänge an b zwei aufeinander folgende Knoten an, die mit α_1 und α_2 markiert sind:



Definition von Tableaus (Fort.)

Definition 3.1 (Tableau (Fort.))

(β) Gibt es auf dem Ast zu b einen Knoten, der mit der β-Formel β markiert ist, hänge an b zwei Geschwisterknoten an, die mit β_1 und β_2 markiert sind:



Zur Vereinfachung der Darstellung identifizieren wir im Folgenden häufig die Äste in einem $\tau \in T_A$ mit der Formelmenge $\Theta \subseteq F$ ihrer Markierungen.

Eigenschaften von Tableaus I: Semantik

Lemma 3.2

Sei $A \in F$ eine Formel und $\tau \in T_A$ ein Tableau für A. Dann gilt

$$A \text{ ist erfüllbar} \quad \text{gdw.} \quad \exists \text{ Ast } \Theta \in \tau : \Theta \text{ ist erfüllbar.}$$

Das Lemma folgt aus einer stärkeren Aussage. Für jede Belegung ψ gilt:

$$\psi \text{ erfüllt } A \quad \text{gdw.} \quad \exists \text{ Ast } \Theta \in \tau : \psi \text{ erfüllt } \Theta.$$

Die erfüllenden Belegungen der Äste sind also genau die erfüllenden Belegungen der Wurzelformel.

Tableaus sind nicht eindeutig, aber Lemma 3.2 hat folgende Konsequenz: Entweder hat jedes Tableau $\tau \in T_A$ einen erfüllbaren Ast oder keines.

Vollständige und abgeschlossene Tableaus

Der Begriff der Erfüllbarkeit von Ästen ist semantischer Natur.
Das Ziel von Tableaus ist, Erfüllbarkeit von Formeln automatisch zu prüfen.

Definition 3.3

- Eine Formelmenge $\Theta \subseteq F$ heißt **vollständig**, falls für alle α-Formeln $\alpha \in \Theta$ auch $\{\alpha_1, \alpha_2\} \subseteq \Theta$ und für alle β-Formeln $\beta \in \Theta$ auch $\beta_1 \in \Theta$ oder $\beta_2 \in \Theta$.
- Ein Tableau τ heißt **vollständig**, falls jeder Ast $\Theta \in \tau$ vollständig ist.
- Eine Formelmenge Θ heißt **abgeschlossen**, falls es eine Formel $B \in F$ gibt mit $\{B, \neg B\} \subseteq \Theta$. Sonst heißt die Menge **offen**.
- Ein Tableau τ heißt **abgeschlossen**, wenn jeder Ast $\Theta \in \tau$ abgeschlossen ist.

Jedes Tableau kann zu einem vollständigen Tableau fortgesetzt werden.

Eigenschaften von Tableaus II: Syntax

Lemma 3.4 (Hintikka)

Sei $\Theta \subseteq F$ vollständig. Dann gilt: Θ ist erfüllbar gdw. Θ ist offen.

Abgeschlossene Mengen sind per Definition unerfüllbar.

Für die Rückrichtung sei Θ eine vollständige und offene Menge. Definiere

$$\psi(p) := \begin{cases} 0 & \neg p \in \Theta \\ 1 & \text{sonst.} \end{cases}$$

Belegung ψ ist wohldefiniert.

Zeige mit Noetherscher Induktion nach der Länge der Formeln, dass $\mathcal{B}_\psi(A) = 1$ für alle $A \in \Theta$.

Korrektheit und Vollständigkeit von Tableaus

Satz 3.5

Eine Formel $A \in F$ ist unerfüllbar gdw. es ein abgeschlossenes Tableau $\tau \in T_A$ gibt.

Auch hier gilt: es gibt ein abgeschlossenes Tableau für A gdw. alle vollständigen Tableaus für A abgeschlossen sind.

Die Tableaumethode geht auf Evert Willem Beth (1908 — 1964) zurück.

Vollständige und offene Formelmengen sind Hintikka-Mengen, nach Jaakko Hintikka (*1929). Das Lemma von Hintikka zeigt, dass sie erfüllbar sind.

Korrektheit und Vollständigkeit von Tableaus (Fort.)

Beweis (von Satz 3.5)

Sei A nicht erfüllbar.
Jedes Tableau kann zu einem vollständigen Tableau fortgesetzt werden.
Also gibt es zu A ein vollständiges Tableau $\tau \in T_A$.
Mit Lemma 3.2 sind alle Äste $\Theta \in \tau$ nicht erfüllbar.
Mit Lemma 3.4 sind alle Äste $\Theta \in \tau$ abgeschlossen.
Also gibt es ein abgeschlossenes Tableau $\tau \in T_A$.

Für die Rückrichtung sei $\tau \in T_A$ abgeschlossen.
Abgeschlossene Äste sind unerfüllbar.
Mit Lemma 3.2 ist Formel A unerfüllbar. ■

Tableaus für Formelmengen

Sei $\Sigma \subseteq F$ eine ggf. unendliche Formelmenge.
Die Menge T_Σ der **Tableaus für Σ** ist definiert wie zuvor, nur dass

- die Konstruktion mit einer Formel $A \in \Sigma$ beginnt und
- es als weiterer Konstruktionsschritt erlaubt ist, $\sigma \in \Sigma$ an ein Blatt anzuhängen.

Tableau $\tau \in T_\Sigma$ heißt vollständig, wenn zusätzlich zu den vorherigen Bedingungen jeder Ast $\Theta \in \tau$ die Menge Σ enthält, also $\Sigma \subseteq \Theta$.

Tableaus für Formelmengen (Fort.)

Lemma 3.6

Seien $\Sigma \subseteq F$ und $\tau \in T_\Sigma$ mit $\Sigma \subseteq \Theta$ für jeden Ast $\Theta \in \tau$. Dann gilt:

$$\Sigma \text{ ist erfüllbar} \quad \text{gdw.} \quad \exists \text{ Ast } \Theta \in \tau : \Theta \text{ ist erfüllbar.}$$

Satz 3.7

Eine Formelmenge $\Sigma \subseteq F$ ist unerfüllbar gdw. T_Σ ein abgeschlossenes Tableau enthält.

Der alte Beweis funktioniert noch immer, modulo folgender Änderungen:

- Lemma 3.2 ist durch Lemma 3.6 zu ersetzen.
- Für die Vollständigkeit ist folgendes Lemma notwendig.

Lemma 3.8

Für jede Formelmenge $\Sigma \subseteq F$ existiert ein vollständiges Tableau $\tau \in T_\Sigma$.

Systematische Tableaukonstruktion

Beweis von Lemma 3.8 Sei Σ unendlich. Es wird eine nicht-terminierende Methode angegeben, die eine Folge von Tableaus

$$\tau_0 \subseteq \tau_1 \subseteq \ldots \quad \text{konstruiert mit} \quad \tau := \bigcup_{i \in \mathbb{N}} \tau_i \ \textbf{vollständig}.$$

Da $\Sigma \subseteq F$, ist Σ abzählbar, also $\Sigma = \{A_0, A_1, \ldots\}$.

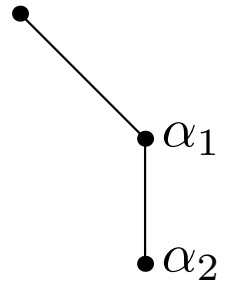
Nutze eine FIFO-Worklist $WL := \emptyset$ zur Speicherung von Knoten.
Nutze ferner einen Zähler $j := 0$, um Σ zu durchlaufen.

- $\tau_0 := \tau_{A_0}$. Ist A_0 kein Literal, push den Knoten von A_0 auf WL.
- τ_{n+1} entsteht aus τ_n wie folgt.

  Falls $WL \neq \emptyset$, pop WL. Sei der Knoten mit $Y \in F$ beschriftet.

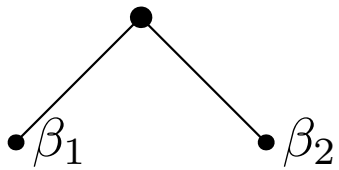
Systematische Tableaukonstruktion (Fort.)

- Ist Y eine α-Formel, erweitere jeden Ast, der durch den Knoten von Y geht, um die Teilformeln α_1 und α_2:



  Falls α_1 bzw. α_2 keine Literale sind, füge alle neuen mit α_1 bzw. α_2 beschrifteten Knoten der Worklist hinzu.

- Ist Y eine β-Formel, erweitere jeden Ast, der durch Y geht, um



  Falls die Teilformeln β_1, β_2 keine Literale sind, füge die entsprechenden Knoten der Worklist hinzu.

Systematische Tableaukonstruktion (Fort.)

Falls $WL = \emptyset$, inkrementiere j und wähle $Y := A_j$.

- Hänge mit Y beschriftete Knoten an alle Äste an.
  Sofern Y kein Literal ist, füge die Knoten der Worklist hinzu.

Systematische Tableaukonstruktion (Fort.)

Behauptung: τ ist vollständig.
Genauer: Jeder Ast $\Theta \in \tau$ ist vollständig und enthält Σ.

Beweis (Skizze):
Sei $\alpha \in \Theta$ eine α-Formel.
Dann ist sie bei der Erstellung in die Worklist aufgenommen worden.
Wegen der FIFO-Reihenfolge, wurde sie irgendwann bearbeitet.
Also sind $\{\alpha_1, \alpha_2\} \subseteq \Theta$.

Betrachte $A_i \in \Sigma$.
Irgendwann ist $j = i$ geworden.
Angenommen, das wäre nicht der Fall.
Dann gab es einen Index, bei dem die Worklist nie geleert wurde.
Das muss falsch sein.
Mit der Entnahme einer Formel $A \in F$ sind zwar endlich viele Formeln der Worklist hinzugefügt worden, die waren aber alle kleiner.

Übung: Warum folgt Terminierung?

Entscheidbarkeit und Semi-Entscheidbarkeit

Um aus der systematischen Tableaukonstruktion Semi-Entscheidbarkeit für Unerfüllbarkeit abzuleiten, passe das Verfahren wie folgt an:

Füge keine Knoten an abgeschlossene Äste an.

Lemma 3.9

(1) Die systematische Tableaukonstruktion terminiert für $\Sigma \subseteq F$ endlich.

(2) Sei $\Sigma \subseteq F$ unendlich und nicht erfüllbar. Dann terminiert die modifizierte Tableaukonstruktion mit einem abgeschlossenen Tableau.

Beachte: Der Kompaktheitssatz folgt aus der zweiten Aussage.
Ist Σ nicht erfüllbar, enthält T_Σ ein endliches, abgeschlossenes Tableau.
Also ist eine endliche Teilmenge von Σ nicht erfüllbar.

Entscheidbarkeit und Semi-Entscheidbarkeit (Fort.)

Lemma 3.10 (König)

Sei T ein unendlicher Baum mit endlichem Ausgangsgrad.
Dann gibt es einen unendlichen Pfad in T.

Zeige Lemma 3.9(2):

Im Fall der Terminierung ist das resultierende Tableau abgeschlossen.
Abgeschlossenheit ist nämlich die einzige Bedingung zur Terminierung.

Es bleibt Terminierung zu zeigen.
Angenommen das modifizierte Verfahren terminiert nicht.
Dann wird ein unendliches Tableau τ konstruiert.
Da das Tableau endlichen Ausgangsgrad hat, gibt es mit Königs Lemma einen unendlichen Pfad $\Theta \in \tau$.
Wie in Lemma 3.8 enthält der Pfad Σ, ist vollständig und offen.
Mit Hintikkas Lemma ist Θ erfüllbar.
Damit ist auch Σ erfüllbar. Widerspruch.

Entscheidbarkeit und Semi-Entscheidbarkeit (Fort.)

Bemerkung 3.11

- *Das Tableauverfahren ist ein Semi-Entscheidungsverfahren für Unerfüllbarkeit aufzählbarer Formelmengen $\Sigma \subseteq F$.*
- *Das Tableauverfahren ist eine Entscheidungsprozedur für Erfüllbarkeit endlicher Formelmengen $\Sigma \subseteq F$.*

Für die Entscheidbarkeit ist zu beachten, dass bei einer aufzählbaren Menge das Hinzufügen einer Formel $A_i \in \Sigma$ zu einem Tableau effektiv ist. Ebenso ist der Test auf Abgeschlossenheit entscheidbar.

Normalformen

Vorteile:
Die einfachere Gestalt der Normalform lässt spezielle Algorithmen zur Lösung bestimmter Fragestellungen zu.

Die Transformation sollte nicht zu teuer sein, sonst würde sich der Aufwand nicht lohnen.

Beispiele:

- Aus einer DNF lassen sich alle erfüllenden Belegungen direkt ablesen.
- Aus einer minimalen DNF lassen sich leicht Schaltnetze (mit UND-, ODER-, NEG-Gattern) herleiten.
- Die systematische Tableau-Konstruktion erlaubt es, diese Normalformen aus einem vollständigen Tableau abzulesen.

Normalformen (Fort.)

Transformiert werden kann in eine

- logisch äquivalente Formel: $A \models\!\!\models T(A)$
- erfüllbarkeitsäquivalente Formel: 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 $\neg, \vee, \wedge$

Negationsnormalform

Formel $A \in F$ ist in **Negationsnormalform (NNF)**, falls jede Negation direkt vor einer Variablen steht und keine zwei Negation einander folgen.

Definition 3.12 (NNF)

Die Menge der Formeln in **NNF** ist induktiv definiert durch

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

Lemma 3.13

Zu jeder Formel $A \in F_{\{\neg,\wedge,\vee,\rightarrow,\leftrightarrow\}}$ gibt es $B \in F_{\{\neg,\wedge,\vee\}}$ in NNF mit $A \models\!\!\!\models B$ und $|B| \in O(|A|)$.

Konjunktive Normalform

Definition 3.14 (Klausel)

Eine Formel $A \equiv (L_1 \vee ... \vee L_n)$ mit Literalen $L_1, \ldots, L_n$ heißt **Klausel**.

- Sind alle Literale 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 $k \in \mathbb{N}$ Literale enthält.

  1-Klauseln werden auch Unit-Klauseln genannt.

Eine Formel $A \equiv (A_1 \wedge ... \wedge A_m)$ ist in **KNF**, falls A eine Konjunktion von Klauseln $A_1, \ldots, A_m$ ist.

- Handelt es sich um k-Klauseln, so ist A in k-**KNF**.

Konjunktive Normalform (Fort.)

Beispiel 3.15

$A \equiv (p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee q) \wedge (\neg p \vee \neg q)$ ist in 2-KNF.

Betrachtet man Klauseln als Mengen von Literalen, so lassen sich Formeln in KNF als Mengen von Literalmengen darstellen.
Am Beispiel A:

$$\{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}, \{\neg p, \neg q\}\}.$$

Lemma 3.16

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

Die Schranke ist **strikt**:
Es gibt eine Folge von Formeln $(A_n)_{n\in\mathbb{N}}$ mit $|A_n| \leq 2n$, für die jede logisch äquivalente Formel B_n in KNF mindestens die Länge 2^n besitzt.

Disjunktive Normalform (Fort.)

Definition 3.17 (DNF)

Eine Formel $A \in F$ ist in **DNF**, falls A eine Disjunktion von Konjunktionen von Literalen ist:

$$A \equiv (A_1 \vee \ldots \vee A_m) \quad \text{mit} \quad A_i \equiv (L_1^i \wedge \ldots \wedge L_{n_i}^i).$$

Definition 3.18 (Duale Formel)

Die **duale Formel** $d(A)$ einer Formel $A \in F$ ist definiert durch:

$$\begin{aligned} d(p) &\equiv p \quad \text{für } p \in V \\ d(\neg A) &\equiv \neg d(A) \\ d(B \vee C) &\equiv d(B) \wedge d(C) \\ d(B \wedge C) &\equiv d(B) \vee d(C). \end{aligned}$$

Zusammenhänge zwischen den Normalformen

Lemma 3.19

Für jede Formel $A \in F$ gilt:

(1) *Ist A in KNF, dann ist NNF($\neg A$) in DNF.*

(2) *Ist A in KNF, so ist $d(A)$ in DNF und umgekehrt.*

Lemma 3.20

Für jede Formel $A \in F$ gilt:

(1) *Setzt man $\varphi(p) := 1 - \psi(p)$, so gilt $\mathcal{B}_\varphi(d(A)) = 1 - \mathcal{B}_\psi(A)$.*

(2) *A ist Tautologie gdw. $d(A)$ ein Widerspruch ist.*

(3) *A ist erfüllbar gdw. $d(A)$ keine Tautologie ist.*

Davis-Putnam-Algorithmen

Idee: Reduziere Erfüllbarkeit für eine Formel mit $n \in \mathbb{N}$ Variablen auf das Erfüllbarkeitsproblem für Formeln mit **maximal** $n - 1$ Variablen.

Ansatz: Suche nach einer erfüllenden Belegung durch iterative Auswahl der Werte einzelner Variablen — **Bottom-Up-Verfahren**.

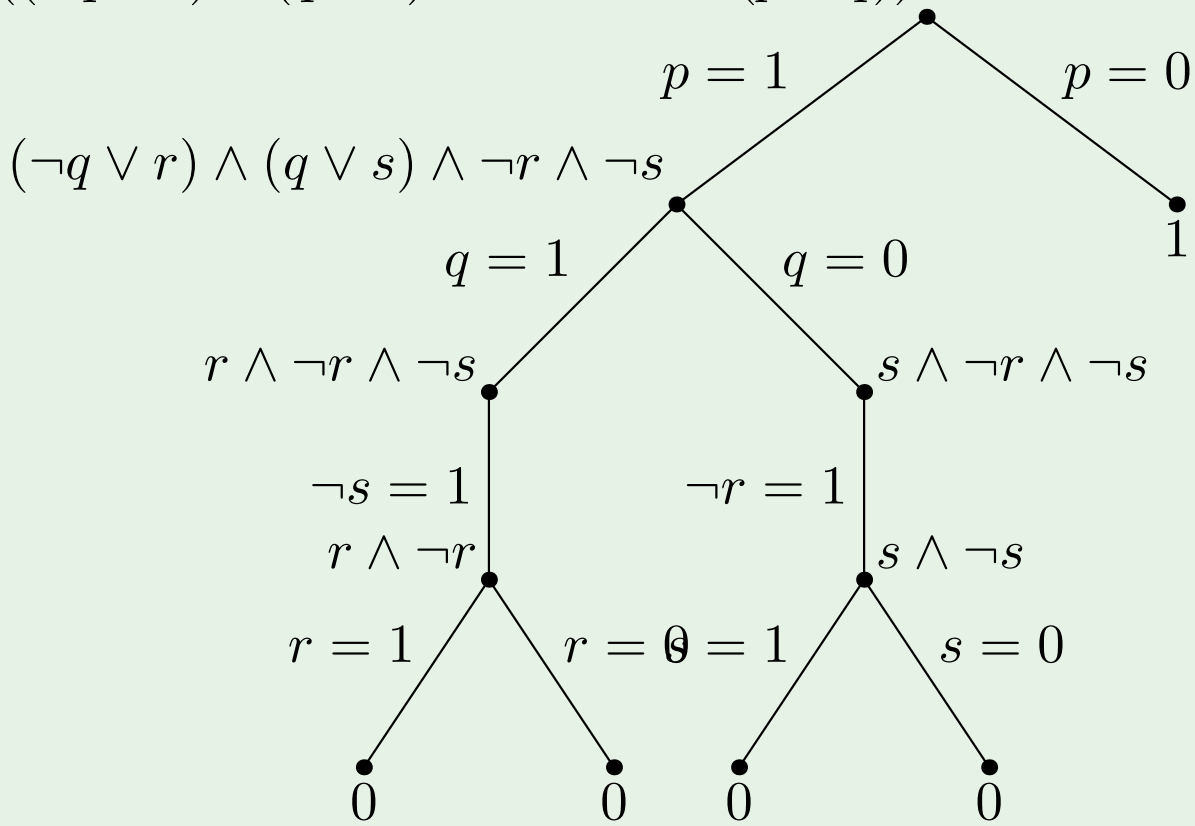
Algorithmen, die mit dieser Idee, Heuristiken und weiteren Verfeinerungen arbeiten, werden als Davis-Putnam-Algorithmen bezeichnet, nach Martin Davis (*1928) und Hilary Putnam (*1926).

Voraussetzung: Formel in **NNF** über $\neg, \wedge, \vee$.

Davis-Putnam-Algorithmen (Fort.)

Beispiel 3.21 (Darstellung der Abarbeitung als Baum)

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



Substitution

Definition 3.22 (Substitution)

Sei Formel $A \in F$ in NNF und $p \in V$.
Definiere $A[p/1]$ als Ergebnis des folgenden Ersetzungsprozesses:

(1) Ersetze in A jedes Vorkommen von p durch 1.

(2) Ersetze Teilformeln gemäß folgender Regeln so lange wie möglich:

- $\neg 1$ durch 0 und $\neg 0$ durch 1
- $B \wedge 1$, $B \vee 0$, $1 \wedge B$ und $0 \vee B$ durch B
- $B \vee 1$ und $1 \vee B$ durch 1
- $B \wedge 0$ und $0 \wedge B$ durch 0

Analog ist $A[p/0]$ definiert, wobei p durch 0 ersetzt wird.

$A[p/i]$ heißt auch die Kürzung von A mit $p = i$, $i \in \mathbb{B}$.

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

Substitution (Fort.)

Lemma 3.23

$A[p/1]$ und $A[p/0]$ sind wohldefiniert.

Formel $A[p/i]$ mit $i \in \mathbb{B}$ ist:

- *eine Formel in NNF bzw. KNF, wenn A diese Form hatte, oder*
- *die leere Formel, $A[p/i] = 1$, die als wahr interpretiert wird, oder*
- *die leere Klausel, $A[p/i] = 0$, die als falsch interpretiert.*

Variable $p \in V$ kommt nicht mehr in $A[p/i]$ vor.

Beispiel 3.24

Für A in KNF und Literal L gilt:
$A[L/1]$ entsteht durch Streichen aller Klauseln in A, die Literal L enthalten, und durch Streichen aller Vorkommen von $\neg L$ in den anderen Klauseln.

Korrektheit von Davis-Putnam-Algorithmen

Lemma 3.25

Eine Formel A in NNF ist erfüllbar gdw. $A[p/1] = 1$ oder $A[p/0] = 1$ oder eine der Formeln $A[p/1], A[p/0]$ erfüllbar ist.

Das Lemma folgt aus der Tatsache, dass für jede Belegung ψ gilt

$$\mathcal{B}_\psi(A) = \mathcal{B}_\psi(A[p/i]), \quad \text{wobei } i = \psi(p)$$

Durch Testen der Formeln $A[p/1]$ und $A[p/0]$, die nun $p \in V$ nicht mehr enthalten, kann rekursiv die Erfüllbarkeit von A entschieden werden.

Regelbasierte Definition von Davis-Putnam

Ausgehend von den Kürzungsregeln beschreiben wir im Folgenden einen Davis-Putnam-Algorithmus für Formeln in KNF.

Definition 3.26 (Regeln für Formeln in **KNF**)

Pure-Literal Regel: Kommt eine Variable $p \in V$ in Formel A nur positiv oder nur negativ vor, belege p mit 1 bzw. p mit 0 und kürze die Formel.

A ist erfüllbarkeitsäquivalent mit $A[p/1]$ bzw. $A[p/0]$.

Splitting-Regel: Kommt eine Variable $p \in V$ sowohl positiv als auch negativ in A vor, bilde die zwei Kürzungen $A[p/1]$ und $A[p/0]$.

A ist erfüllbar gdw. bei einer der Kürzungen der Wert 1 oder eine erfüllbare Formel auftritt.

Regelbasierte Definition von Davis-Putnam (Fort.)

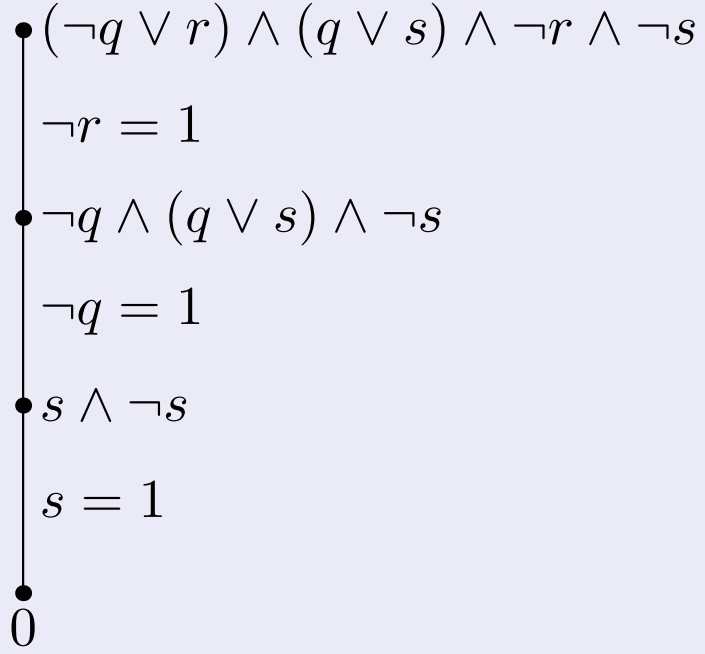
Definition 3.27 (Regeln für Formeln in **KNF**)

Unit-Regel:

Sei A in KNF und enthalte eine Unit-Klausel $A_i \equiv L$.

Bilde $A[L/1]$:

A erfüllbar gdw. $A[L/1]$ erfüllbar.



Regelbasierte Definition von Davis-Putnam (Fort.)

Klausel A_1 **subsumiert** Klausel A_2, in Zeichen $A_1 \subseteq A_2$, falls jedes Literal aus A_1 auch in A_2 auftritt.

Aus der Erfüllbarkeit einer Klausel A_1 folgt sofort die Erfüllbarkeit aller Klauseln A_2, die sie subsummiert: $A_1 \subseteq A_2$.

Definition 3.27 (Regeln für Formeln in **KNF** (Fort.))

Subsumption-Rule: Sei A in KNF. Streiche aus A alle Klauseln, die von anderen subsumiert werden: Funktion Subsumption_Reduce(A).

Streiche dabei auch tautologische Klauseln, die p und $\neg p$ für ein $p \in V$ enthalten.

> *Da Klauseln konjunktiv verknüpft sind, sind nur die zu berücksichtigen, die von keiner anderen subsumiert werden.*

procedure DPA — Davis-Putnam-Algorithmus

Eingabe: A in KNF
Ausgabe: Boolscher Wert für Erfüllbarkeit $\{0,1\}$

begin
if $A \in \{0,1\}$ **then return** A;
p:=**pure**(A,s);
//*liefert Variable 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 Variable in* A
if DPA($A[p/1]) = 1$ **then return** 1;
return DPA(A[p/0]);
end

Auswahlkriterien für die Splitting-Regel

- Wähle die erste in der Formel vorkommende Variable.
- Wähle die Variable, die am häufigsten vorkommt.
- Wähle die Variable mit $\sum_{p \text{ in } A_i} |A_i|$ minimal.
- Wähle die Variable, die in den kürzesten Klauseln am häufigsten vorkommt.
- Berechne die Anzahl der positiven und negativen Vorkommen in den kürzesten Klauseln und wähle die Variable mit der größten Differenz.
- Weitere Heuristiken in Implementierungen vorhanden.

Resolution

Idee: Aus Klauseln $(A \vee L)$ und $(B \vee \neg L)$ wird die neue Klausel $(A \vee B)$ erzeugt, denn

$$(A \vee L) \wedge (B \vee \neg L) \models\!\models (A \vee L) \wedge (B \vee \neg L) \wedge (A \vee B).$$

Sei L ein Literal; dann bezeichnen wir im Folgenden mit $\neg L$ das Literal $\neg p$, falls $L \equiv p$, und p, falls $L \equiv \neg p$, $p \in V$.

Ziel: Leere Klausel $\sqcup$ erzeugen, um **Unerfüllbarkeit** zu zeigen.

Resolution arbeitet auf Formeln in **KNF**. Dabei ist es günstig, Klauseln als Mengen darzustellen:

$$(p \vee \neg q \vee p) \quad \text{dargestellt als} \quad \{p, \neg q\}.$$

Resolution geht zurück auf John Alan Robinson (*1928).

Resolution (Fort.)

Definition 3.28 (Resolvente)

Seien K_1, K_2 Klauseln und L ein Literal mit $L \in K_1$ und $\neg L \in K_2$. Dann ist

$$R \equiv (K_1 \setminus \{L\}) \cup (K_2 \setminus \{\neg L\})$$

die **Resolvente von K_1 und K_2 nach L**.

Beachte: Die Resolvente kann die leere Klausel $\sqcup$ sein.

Das Hinzufügen von Resolventen führt zu äquivalenten Formeln.

Lemma 3.29

Sei A in KNF und R Resolvente zweier Klauseln aus A. Dann gilt

$$A \models\!\!\models A \cup \{R\}.$$

Resolution (Fort.)

Definition 3.30 (Herleitungen)

Seien A in KNF und K Klausel. Eine Folge $K_1, \ldots, K_n$ von Klauseln mit $K_n \equiv K$ ist eine **Herleitung von K aus A**, $A \underset{Res}{\vdash} K$, falls für $1 \leq k \leq n$:

$$K_k \in A \text{ oder } K_k \text{ ist eine Resolvente zweier } K_i, K_j \text{ mit } i, j < k.$$

Lemma 3.31

Als Kalkül ist Resolution korrekt aber **nicht vollständig**:

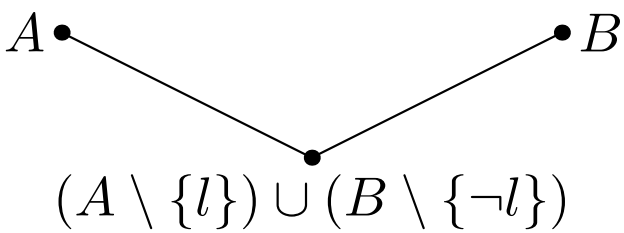
$$A \underset{Res}{\vdash} K \text{ impliziert } A \models K. \text{ Die Umkehrung gilt nicht.}$$

Satz 3.32 (Korrektheit und Widerlegungsvollständigkeit, Robinson)

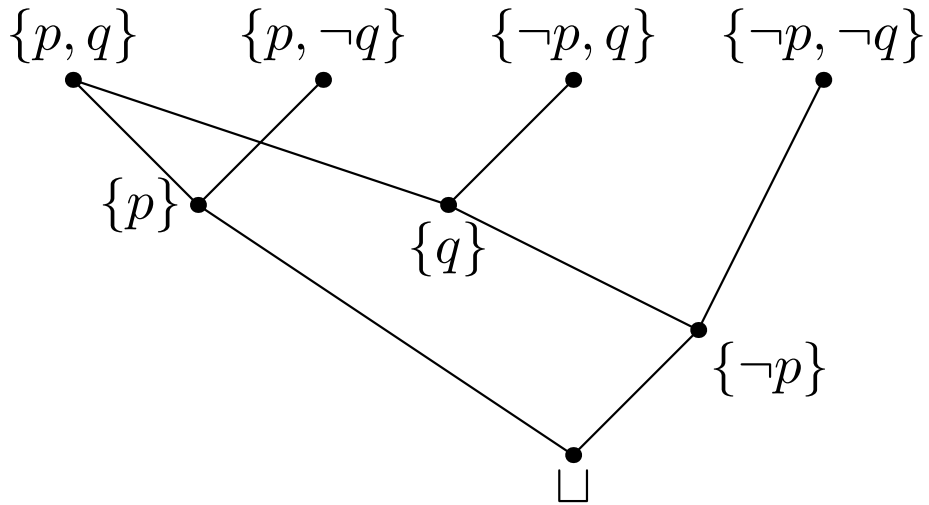
Eine Formel A in KNF ist unerfüllbar gdw. $A \underset{Res}{\vdash} \square$.

Darstellung

Darstellung einer Resolvente zweier Klauseln A, B nach L:



Darstellung von Herleitungen als gerichteter, azyklischer Graph (DAG):



Resolventenmethode: Heuristiken

Starke Herleitungen: Sei A in KNF und unerfüllbar.
Dann gibt es eine Herleitung $K_1, \ldots K_n \equiv \sqcup$, so dass

1. in der Herleitung keine Klausel mehrfach auftritt,
2. in der Herleitung keine Tautologie auftritt,
3. in der Herleitung keine schon subsumierte Klausel auftritt:
   Es gibt keine K_i, K_j mit $i < j$ und $K_i \subseteq K_j$.

Resolventenmethode: Heuristiken (Fort.)

- Stufenstrategie (Resolutionsabschluss)
  (Alle erfüllenden Belegungen)
- Stützmengenrestriktion
  (Set-of-Support, Unit-Klauseln bevorzugen)
- P-(N-)Resolution
- Lineare Resolution (SL-Resolution, PROLOG-Inferenzmaschine).

Resolventenmethode: Stufenstrategie

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. $\{\neg s, \neg t\}$ (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)	

Erhalte die erfüllende Belegung

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