Kapitel 08:
Prolog: Theoretische Grundlagen

Grundlage:

Schöning; Kreuzer/Kühling ("KK");
Inf 8 (Beckstein u.a.)

Friedrich-Alexander-Universität Erlangen-Nürnberg

Department Informatik

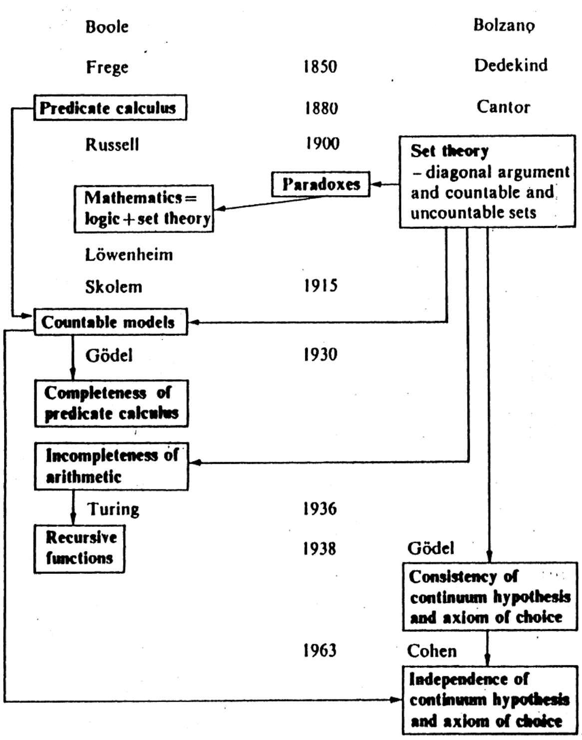
Ziel

- Automatisierung des logischen Schließens
  - Definition von Normalformen für Formeln und Transformationsregeln
  - Implementation der Schlußregel(n)
  - Beweisverfahren:
    - durch Widerlegung („Refutation")
    - Konstruktiv: Tableaux
      (s. Modellierungskapitel, Beschreibungslogik)
- Logikprogrammierung: Logische Sprache als Basis einer Programmiersprache – Prolog
- Literaturhinweis: Artikel „Automated Reasoning" in der *Stanford Encyclopedia of Philosophy* (on-line)

Inhalt

- Normalformen
- Hornformeln; Erfüllbarkeitstest
- Endlichkeitssatz
- Aussagenlogische Resolution
- Quantorenlogische Normalformen
- Herbrand-Theorie
- Unifikation
- Quantorenlogische Resolution

Metamathematik im 20. Jh.



© Crossley

Normalformen

- **Definition**:
  Ein **Literal** ist eine atomare Formel (Primformel) oder die Negation einer atomaren Formel (**positives** bzw. **negatives Literal**).

  Eine Formel F ist in **konjunktiver Normalform** (KNF, CNF), wenn sie eine Konjunktion von Adjunktionen von Literalen ist:

$$F = \left(\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} L_{i,j}\right)\right)$$

  Analog: "**dis"junktive Normalform** (DNF).

- **Satz**: [anknüpfend an die klassischen Äquivalenzen]
  Für jede Formel F gibt es eine (klassisch) äquivalente Formel in **KNF** (und eine äquivalente Formel in **DNF**).

  Beweis: Induktiv über den Formelaufbau von F; s. Schöning S. 28f.

Normalformen

- Mit dem Induktionsbeweis korrespondiert ein rekursiver Algorithmus zur Herstellung von KNF-Formeln:

Gegeben sei eine Formel F

Führe die folgenden Schritte durch:

0) Eliminiere $\rightarrow$ und $\leftrightarrow$ mittels ihrer Definition.

1) Ersetze in F jede Teilformel der Form $\neg\neg G$ durch G.

2) Ersetze in F jede Teilformel der Form $\neg(G \wedge H)$ durch $(\neg G \vee \neg H)$. Entsteht hierdurch eine Teilformel der Form $\neg\neg K$, so wende Schritt (1) an.

Normalformen

3) Ersetze in F jede Teilformel $\neg(G \vee H)$ durch $(\neg G \wedge \neg H)$. Entsteht hierdurch eine Teilformel der Form $\neg\neg K$, so wende Schritt (1) an.
4) Wiederhole die Schritte (2) und (3) so oft wie möglich.
5) Ersetze in F jede Teilformel der Form $(G \vee (H \wedge I))$ durch $((G \vee H) \wedge (G \vee I))$.
6) Ersetze in F jede Teilformel der Form $((G \wedge H) \vee I)$ durch $((G \vee I) \wedge (H \vee I))$.
7) Wiederhole die Schritte (5) und (6) so oft wie möglich.

Die resultierende Formel ist dann in KNF.

Hornformeln

- **Definition**:
  Eine Formel F ist eine **Hornformel**, falls F in KNF ist und jedes Adjunktionsglied in F höchstens ein positives Literal enthält.
- Beispiel:
  $F = (A \vee \neg B) \wedge (\neg C \vee \neg A \vee D) \wedge (\neg A \vee \neg B) \wedge D \wedge \neg E$

  ... als Konjunktion von Subjunktionen:

  $F \equiv (B \to A) \wedge (C \wedge A \to D) \wedge (A \wedge B \to \bot) \wedge (T \to D) \wedge (E \to \bot)$
- Gesucht: Algorithmischer Test für die Gültigkeit (Erfüllbarkeit) einer Formel.

  Klassisch genügt die Beschränkung auf Unerfüllbarkeitstests, denn F ist gültig gdw. $\neg F$ unerfüllbar ist.

  Grundsätzlich mit Wahrheitswertetafeln durchführbar, doch für Hornformeln gibt es einen sehr effizienten Erfüllbarkeitstest.

Erfüllbarkeitstest für Hornformeln

- **Algorithmus**:
  - Eingabe: Hornformel F
  - Versehe jedes Vorkommen A einer atomaren Formel in F mit einer Markierung, falls es in F eine Teilformel der Form $(T \rightarrow A)$ gibt;
  - **while** [es gibt in F eine Teilformel G der Form $(A_1 \wedge \ldots \wedge A_n \rightarrow B)$ oder $(A_1 \wedge \ldots \wedge A_n \rightarrow \perp)$, $n \geq 1$, wobei $A_1, \ldots, A_n$ bereits markiert sind und B noch nicht markiert ist] **do**
    if G hat die erste Form **then** (markiere jedes Vorkommen von B)
    else gib "unerfüllbar" aus und stoppe;
  - Gib "erfüllbar" aus und stoppe. Die erfüllende Belegung wird hierbei durch die Markierung gegeben.
- **Satz**: Der Markierungsalgorithmus ist korrekt und stoppt immer nach höchstens n Markierungsschritten (n : Anzahl der atomaren Formeln in F).

  Beweis: Schöning S. 33f. (Anmerkung: Konstruiert das kleinste Modell!)

Endlichkeitssatz ("Compactness Theorem")

- **Satz**:
  Eine Menge **M** von Formeln ist erfüllbar genau dann, wenn jede der endlichen Teilmengen von **M** erfüllbar ist.

  Beweis: Schöning S. 35f.
- Anmerkungen:
  - Beweisrichtung v.l.n.r. ist trivial, denn jedes Modell für **M** ist auch ein Modell für jede beliebige Teilmenge von **M**.
  - V.r.n.l. muss unter der Annahme, dass jede Teilmenge von **M** ein Modell besitzt, aus diesen ein Modell für **M** konstruiert werden.

Endlichkeitssatz ("Compactness Theorem")

- Dieser Beweis ist nicht konstruktiv – es wird zwar ein Modell konstruiert, allerdings unter Verwendung einer Annahme, die nicht algorithmisch in endlicher Zeit nachprüfbar ist (unendlich viele Indizes...).
- Der Endlichkeitssatz wird noch eine wichtige Rolle spielen in folgender Variante:

  Eine (evtl. unendliche) Formelmenge M ist unerfüllbar gdw.
  bereits eine endliche Teilmenge $M' \subset M$ unerfüllbar ist.

Aussagenlogische Resolution

- Gesucht wird ein Kalkül, mit dessen Hilfe die **Unerfüllbarkeit** einer endlichen Menge **aussagenlogischer** Formeln bewiesen werden kann.

  Begründung:
- Zur Überprüfung, ob F eine Tautologie darstellt, kann man $\neg F$ auf Unerfüllbarkeit testen.
- Eine Formel G **folgt** aus einer Formelmenge $\{F_1, ..., F_k\}$ klassisch gdw. $F_1 \wedge ... \wedge F_k \wedge \neg G$ unerfüllbar ist:
  Widerlegungsverfahren („Refutation").
- Die Schlussregel dieses Kalküls heißt **Resolutionsregel**.
- Die Definition eines solchen Kalküls ist nur sinnvoll, wenn man seine **Korrektheit** und **Vollständigkeit** zeigen kann.

Exkurs: Unerfüllbarkeit und logische Folgerung

$\Gamma \cup \{\neg F\}$ ist unerfüllbar gdw. $\Gamma \models F$

Modelltheoretische Argumentation:

"$\leftarrow$"

Wenn $\Delta \models \varphi$, dann sind alle Modelle für Δ auch Modelle für φ. Damit ist keine dieser Interpretationen Modell für $\neg\varphi$ und damit ist $\Delta \cup \neg\varphi$ unerfüllbar.

"$\rightarrow$"

Sei nun $\Delta \cup \neg\varphi$ unerfüllbar, aber Δ erfüllbar. Sei I die Interpretation, die Δ erfüllt. I erfüllt nicht $\neg\varphi$, denn sonst wäre $\Delta \cup \neg\varphi$ erfüllbar. Damit folgt, dass I φ erfüllt. (Eine Interpretation muss entweder φ oder $\neg\varphi$ erfüllen.). Da dies für willkürliche Modelle I für Δ gilt, gilt es für alle I, die Δ erfüllen. Damit sind alle Modelle für Δ auch Modelle für φ und φ ist logische Konsequenz von Δ.

Resolution (aussagenlogisch)

- Voraussetzung für die Anwendung der Resolution ist, dass die Formeln in KNF vorliegen.
- Sei $F = (L_{1,1} \vee \ldots \vee L_{1,n_1}) \wedge \ldots \wedge (L_{k,1} \vee \ldots \vee L_{k,n_k})$, wobei die $L_{i,j}$ Literale sind.
- Eine für die Anwendung der Resolution vorteilhafte Darstellung der Formeln in KNF ist die in **Mengen** von sog. **Klauseln**:
  $F = \{\{L_{1,1} \ldots L_{1,n_1}\}, \ldots \{L_{k,1} \ldots L_{k,n_k}\}\}$.
- Jedes Element von F, das selbst eine Menge von Literalen ist, heißt **Klausel**; die Klauseln entsprechen den Konjunktionsgliedern.
- Durch die Mengennotation werden Vereinfachungen, die sich aus Assoziativität, Kommutativität oder Idempotenz ergeben, direkt unterstützt.
- Abbildung von Formeln auf Mengen ist nicht eineindeutig.

Resolution (aussagenlogisch)

- **Definition**:

  a) Seien K_1, K_2 und K_3 Klauseln. Die Klausel K_3 heißt eine **Resolvente** von K_1 und K_2, wenn es

     ein Literal L gibt, so dass die folgenden beiden Bedingungen erfüllt sind:

     1) Es gilt $L \in K_1$ und $\neg L \in K_2$. Ist hierbei $L = \neg A$ ein negatives Literal, so sei $\neg L = A$.

     2) $K_3 = (K_1 \setminus \{L\}) \cup (K_2 \setminus \{\neg L\})$

  b) Ist K_3 eine Resolvente von K_1 und K_2, so verwenden wir nebenstehende grafische Darstellung:

$$\begin{array}{ccc} K_1 & & K_2 \\ & \diagdown \quad \diagup & \\ & K_3 & \end{array}$$

  c) Dabei ist die leere Klausel $K_3 = \emptyset$ zulässig. Sie ergibt sich z.B. als Resolvente von $K_1 = \{L\}$ und $K_2 = \{\neg L\}$.

     Eine Klauselmenge, die $\emptyset$ enthält, wird als **unerfüllbar** bezeichnet.

Resolution (aussagenlogisch)

- **Definition**:

  a) Für jede Klauselmenge K setzen wir

     $Res^1(K) = K \cup \{R \mid R \text{ Resolvente zweier Klauseln in } K\}$.

  b) Für $n \geq 2$ sei $Res^n(K) = Res(Res^{n-1}(K))$.

     Für $n = 0$ sei $Res^0(K) = K$. Wir nennen $Res^n(K)$ die Menge der Resolventen **n-ter Stufe** von K.

  c) Schließlich setzen wir $Res^\infty(K) = \bigcup_{n \geq 0} Res^n(K)$.

- **Satz**: (Resolutionssatz der Aussagenlogik)

  Eine Formel F ist genau dann unerfüllbar, wenn $\emptyset \in Res^\infty(K(F))$ gilt.

  Beweis: Korrektheit und Vollständigkeit, s. Schöning, S. 43–45

Resolution (aussagenlogisch)

- **Algorithmus**: Erfüllbarkeitstest für aussagenlogische Formeln

  Gegeben sei eine aussagenlogische Formel F in KNF.

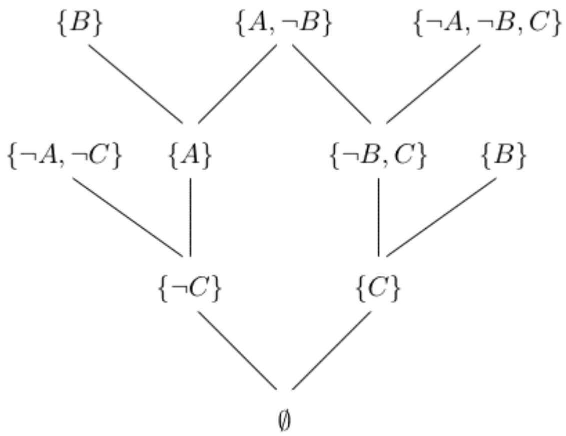
  1) Bilde die Klauselmenge K(F).
  2) Für n = 1, 2, 3, . . . berechne Res^n (K(F)) solange, bis

     $\emptyset \in Res^n(K(F))$ oder $Res^n(K(F)) = Res^{n-1}(K(F))$ gilt.
  3) Im ersten Fall gib "F ist unerfüllbar" aus,
     im zweiten Fall gib "F ist erfüllbar" aus und stoppe.

- **Definition**:
  Eine **Deduktion** (**Herleitung**) der leeren Klausel aus einer Klauselmenge F ist eine Folge $K_1, \ldots, K_m$ von Klauseln mit $K_m = \emptyset$ und jedes K_i (i = 1,...,m) ist Element aus F oder kann aus Klauseln K_a, K_b mit a, b < i resolviert werden.

Resolution (aussagenlogisch)

- Beispiel: **K** = {{B}, {A, ¬B}, {¬A, ¬B, C}, {¬A, ¬C}}
- **Resolutionsgraph**:



Eine Klausel heißt **Hornklausel**, wenn sie höchstens **ein positives Literal** enthält.

Die Klauselmenge **K** (Horn!) kann als Logikprogramm interpretiert werden:

a) Gegeben ist B
b) Es gilt $B \rightarrow A$ und $(A \wedge B) \rightarrow C$
c) Anfrage: Folgt $A \wedge C$?

Antwort: Ja, denn K ist unerfüllbar.

© KK, 3.6

Resolutionsstrategien

- Das Ableiten der leeren Klausel aus einer Klauselmenge kann als **Suchproblem** aufgefasst werden.

  Die bekanntesten **Suchstrategien** im Resolutionskalkül sind:

  - "Linear Resolution": Die Resolvente ist Elternklausel im nächsten Schritt
  - "Input Resolution": Eine Elternklausel ist eine Eingabeklausel
  - "Unit Resolution": Eine Elternklausel enthält nur ein Literal
  - "Set of Support Resolution": Eine Elternklausel hat einen Vorfahren in der Stützmenge
  - "Ordered Resolution"
  - Hyperresolution
  - "SLD Resolution": Linear resolution with Selection function restricted to Definite clauses
- Nicht alle Strategien sind (widerlegungs-)vollständig!
- Die SLD-Strategie ist Grundlage für die Beweiser in Logikprogrammiersystemen wie Prolog.

Einheitsresolution für Hornformeln

- Für Hornformeln (**nicht** jedoch beliebige KNF-Formeln) ist die folgende Einschränkung des Resolutionskalküls vollständig:

  Es darf nur dann eine Resolvente gebildet werden, wenn mindestens eine der beiden Klauseln nur aus einem **einzigen** Literal besteht.
- **Beweis**: Schöning, Übung 35

- Anmerkung: Bei solchen Resolutionen können immer nur kürzere Klauseln entstehen, daher lässt sich daraus ein effizienter Algorithmus für Hornformeln ableiten – ähnlich effizient wie der o.g. Markierungsalgorithmus.

Lineare Resolution

- **Definition**:

Sei F eine aussagenlogische Formel in KNF.

a) Die leere Klausel $\emptyset$ ist aus K(F) **linear resolvierbar**, falls es eine Klausel $K_0 \in$ K(F) gibt und eine Folge von Klauseln $K_1, \ldots, K_n$ mit folgenden Eigenschaften:

a1) Für i = 1, . . . , n gilt $K_{i-1}\ B_{i-1}\ K_{i,}$

wobei die **Seitenklausel** B_{i-1} entweder ein Element von K(F) ist oder $B_{i-1} = K_j$ mit $j < i - 1$.

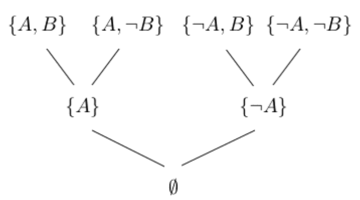
a2) $K_n = \emptyset$.

b) Eine Folge von Klauseln $K_0, \ldots, K_n$ wie in a) heißt eine **lineare Resolution** von F.

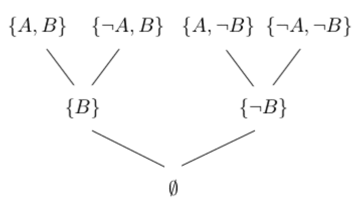
Lineare Resolution

- Beispiel: Sei K = {{A, B}, {A, ¬B}, {¬A, B}, {¬A, ¬B}}

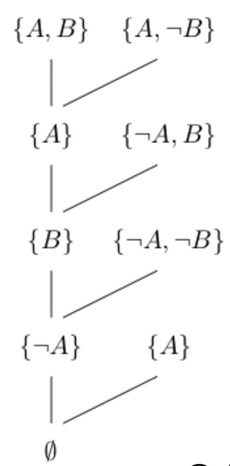
Unerfüllbarkeitstest nach Resolutionssatz (2 Mögl.)



oder



Lineare Resolution



Input-Resolution

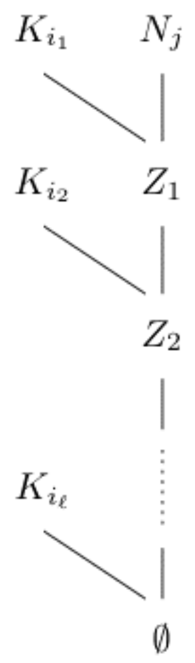
- **Definition**:
  Sei F eine aussagenlogische Formel in KNF.
  Eine **Input-Resolution** von F ist eine lineare Resolution von F, bei der in jedem Schritt als Seitenklausel eine der Klauseln aus K(F) verwendet wird.
- Bemerkung:
  Nicht jede unerfüllbare aussagenlogische Formel in KNF besitzt eine Input-Resolution;
  u.a. die im letzten Beispiel.

SLD-Resolution

- **Definition**:
  Sei F eine Hornformel.
  Eine **SLD-Resolution** ("Linear resolution with Selection function restricted to Definite clauses") ist eine Input-Resolution der folgenden Form:
  a) K_0 ist eine negative Klausel, die sog. **Zielklausel**.
  b) Bei jedem Resolutionsschritt ist eine der Elternklauseln eine nicht-negative Hornklausel, also eine **Programmklausel**.
- Anmerkungen:
  - Eine SLD-Resolution benötigt immer auch eine **Auswahlfunktion**, die bei jedem Schritt angibt, welches Literal als nächstes zu resolvieren ist.
    Diese ist bei der Logik-Programmierung zu spezifizieren.

SLD-Resolution

- Eine SLD-Resolution hat die folgende Gestalt:
  Sei K = $\{K_1, ..., K_n, N_1, ..., N_m\}$ unerfüllbar, wobei die K_i die Programm- und die N_j die negativen Klauseln sind. Dann gibt es $j \in \{1,...,m\}$ und $i_1, ..., i_l \in \{1, ..., n\}$ mit



  Die Zwischenresultate $Z_1, ..., Z_{l-1}$ können dabei nur negative Klauseln sein, da sie die Resolventen einer negativen und einer Programmklausel sind.

Vollständigkeit

- **Satz**:
  1. Die lineare Resolution ist vollständig,
     d.h. für jede unerfüllbare Klauselmenge K gibt es eine Klausel $k \in$ K, so dass die leere Klausel durch lineare Resolution aus K, basierend auf k, herleitbar ist.
  2. Die SLD-Resolution ist vollständig für Hornformeln.

  Beweis: s. Kreuzer/Kühling, S. 49–51

- Wie ist die Resolution auf die **Quantorenlogik** zu **übertragen**?
  - Wie soll entschieden werden, wann zwei Klauseln miteinander über ein komplementäres Literalpaar resolviert werden können?
  - Die beiden fraglichen Literale können ja Variablen enthalten, die eine "Gleichwertigkeit" der Literale verschleiern.

Quantorenlogische Normalformen

- Zur Erinnerung: Die **Sprache der Quantorenlogik**
  - Variablen, Funktionssymbole inklusive Konstanten, Prädikaten-/ Relationssymbole,
  - Terme: Variablen und Funktionen,
  - Formeln: Prädikate, Junktoren, Quantoren.
- Ersetzungs-**Satz**:
  Seien F_1 und F_2 zwei äquivalente Formeln und sei G eine weitere Formel, in der F_1 als Teilformel vorkommt.
  Sei G´ die Formel, die man aus G mittels Ersetzung von F_1 durch F_2 erhält. Dann gilt $G \equiv G'$.

  Beweisskizze:
  Teilformelinduktion über G ; Fallunterscheidung nach den Junktoren und Quantoren.

Substitutionen

- **Definition**:
  Sei F eine Formel, x eine in F vorkommende Variable und t ein Term. Ersetzt man jedes freie Vorkommen von x in einer Teilformel von F durch t, so heißt die entstehende Formel **F[x/t]** das Ergebnis der **Substitution** von x durch t.
- **Satz**:
  Sei F eine quantorenlogische Formel der Form **Qx.G**, wobei Q der Existenz- oder Allquantor ist und G die Variable x nur frei enthalte. Sei y eine Variable, die in G nicht vorkommt.

  Dann gilt: $\mathbf{F \equiv Qy.G[x/y]}$.
  Man sagt, dass die neue Formel durch **gebundene Umbenennung** aus F entsteht.

  Beweis (modelltheoretisch): Zu jeder zu F passenden Struktur A kann man $\mathbf{I_A(y) = I_A(x)}$ definieren und erhält eine zur neuen Formel passende Struktur mit demselben Wahrheitswert.

Substitutionen

- **Satz**:
  Sei F eine quantorenlogische Formel. Dann erhält man durch geeignete gebundene Umbenennungen eine äquivalente Formel G, in der **keine** Variable sowohl gebunden als auch in einer Teilformel frei vorkommt.
  Ferner erreicht man, dass hinter allen Quantoren von G paarweise verschiedene Variablen stehen.
  Eine solche äquivalente Formel G heißt in **bereinigter Form** oder die bereinigte Form von F .
  Beweis: Wiederholte Anwendung des vorhergehenden Satzes.
- **Definition**:
  Eine quantorenlogische Formel F heißt in **Pränexform** oder eine **Pränexformel**, wenn sie die Form

$$Q_1 x_1\ Q_2 x_2\ \dots\ Q_n x_n.\ G$$

  hat, wobei die Q_i Quantoren, die x_i Variablen sind und G eine Formel ist, in der **kein** Quantor (mehr) vorkommt.

Transformation in bereinigte pränexe Skolemform

- **Satz**:
  Für jede Formel F gibt es eine äquivalente (und bereinigte) Formel G in Pränexform.
  Beweis: Teilformelinduktion/Fallunterscheidung, s. Schöning, S. 63; resultiert in Algorithmus BPF(F). [Details s.a. KK, S. 69f.]
- **Satz**: Sei F eine quantorenlogische Formel der Form

$$F = \bigwedge_{y1} \dots \bigwedge_{yn} \bigvee_{z} G$$

  mit G in BPF, $n \geq 0$ und f ein neues n-stelliges Funktionssymbol.
  Dann kann sie in eine (erfüllungs-) äquivalente **Skolemformel** überführt werden durch wiederholte Anwendung von

$$F \risingdotseq \bigwedge_{y1} \dots \bigwedge_{yn} \bigvee_{z} G[z / f(y_1, \dots, y_n)]$$

  bis alle Existenzquantoren eliminiert sind.
  Beweis (modelltheoretisch): s. Schöning, S. 65f.

Pränexe skolemisierte konjunktive Normalform

- Bereinige geg. Formel F durch systematische Umbenennung der gebundenen Variablen $\Rightarrow F_1$, äquivalent.
- Seien $y_1, \ldots, y_n$ die in F bzw. F_1 vorkommenden freien Variablen. Ersetze F_1 durch $F_2 = \bigvee_{y1} \ldots \bigvee_{yn} F_1$ erfüllungsäquivalent.
- Stelle eine zu F_2 äquivalente Formel F_3 in Pränexform her.
- Eliminiere die vorkommenden Existenzquantoren durch Übergang zur Skolemformel F_4 , erfüllungsäquivalent zu F_3.
- Forme die Matrix von F_4 um in KNF $\Rightarrow F_5$
- Notiere F_5 als Klauselmenge.

Thoralf Skolem
1887–1963

Beispiel 1: Pränexe skolemisierte KNF

(Schöning, S. 70 verbessert)

$F = \neg\bigvee_x (P(x, z) \wedge \bigwedge_y Q(x, f(y)) \vee \bigwedge_y P(g(x, y), z))$

- Bereinigte Form durch Umbenennung von y zu w

$F_1 = \neg\bigvee_x (P(x, z) \wedge \bigwedge_y Q(x, f(y)) \vee \bigwedge_w P(g(x, w), z))$

- Frei vorkommendes z existentiell binden

$F_2 = \bigvee_z (\neg\bigvee_x (P(x, z) \wedge \bigwedge_y Q(x, f(y)) \vee \bigwedge_w P(g(x, w), z)))$

- Umformung in Pränexform (z.B.)

$F_3 = \bigvee_z \bigwedge_x \bigvee_y \bigwedge_w (\neg(P(x, z) \wedge \neg Q(x, f(y)) \vee P(g(x, w), z)))$

- Skolemisierung: Konstante a für z, Funktion h(x) für y

$F_4 = \bigwedge_x \bigwedge_w (\neg(P(x, a) \wedge \neg Q(x, f(h(x))) \vee P(g(x, w), a)))$

- Umformung der Matrix in KNF

$F_5 = \bigwedge_x \bigwedge_w (\neg(P(x, a) \vee P(g(x, w), a)) \wedge (\neg Q(x, f(h(x))) \vee P(g(x, w), a)))$

$\{\{\neg P(x, a), P(g(x, w), a)\}, \{\neg Q(x, f(h(x))), P(g(x, w), a)\}\}$

Beispiel 2: Pränexe skolemisierte KNF

- Hierbei werden die Negationen sogleich "nach innen" verschoben (Ziel: positive und negative Literale)

$\bigwedge_x (P(x) \rightarrow \bigvee_y ((Q(x,y) \rightarrow P(x)) \wedge \bigwedge_z (Q(y,z) \rightarrow P(x))))$

1. $\bigwedge_x (\neg P(x) \vee \bigvee_y ((\neg Q(x, y) \vee P(x)) \wedge \bigwedge_z (\neg Q(y, z) \vee P(x))))$
2. $\bigwedge_x (\neg P(x) \vee (\neg Q(x, f(x)) \vee P(x)) \wedge \bigwedge_z (\neg Q(f(x), z) \vee P(x)))$
3. $\bigwedge_x \bigwedge_z ((\neg P(x) \vee \neg Q(x,f(x)) \vee P(x)) \wedge (\neg P(x) \vee \neg Q(f(x),z) \vee P(x)))$
4. $(\neg P(x) \vee \neg Q(x, f(x)) \vee P(x)) \wedge (\neg P(x) \vee \neg Q(f(x), z) \vee P(x))$
5. $\{\neg P(x) \vee \neg Q(x, f(x)) \vee P(x), \neg P(x) \vee \neg Q(f(x), z) \vee P(x)\}$

   zwei Klauseln

Unentscheidbarkeit der Quantorenlogik

- Satz (Alonzo Church, 1903–1995):
  Das Gültigkeitsproblem (Erfüllbarkeitsproblem) der Quantorenlogik ist unentscheidbar.
  (D.h., es gibt keinen Algorithmus, der für eine beliebige quantorenlogische Formel in endlich vielen Schritten entscheidet, ob sie erfüllbar ist oder nicht.)
  Zum Beweis: ... Berechenbarkeitstheorie
- Dennoch kann zu einer gegebenen quantorenlogischen Formel systematisch nach einem Modell suchen, in dem sie gilt.
  - Welche Techniken sind geeignet?
  - Wie findet man eine möglichst kleine Grundmenge, so dass man ein Modell ansetzen kann?
  - Wenn eine Formel unerfüllbar ist, kann man dies in endlich vielen Schritten nachweisen? (**Semi-Entscheidbarkeit** der Quantorenlogik – Algorithmus von Gilmore
    [Schöning, S. 83] ⇒ Grundresolutionsalgorithmus)

Herbrand-Theorie

- **Definition**:
  Sei F eine geschlossene quantorenlogische Skolemformel, d.h.
  1) F enthält keine freien Variablen,
  2) F ist in BPF,
  3) F enthält keine Existenzquantoren.

  Dann sei D(F) die Menge aller variablenfreien Terme, die aus den in F vorkommenden Funktionssymbolen und Konstanten gebildet werden können. Enthält F keine Konstanten, so wählen wir eine zusätzliche Konstante a.

  Mit anderen Worten, die Menge D(F) ist induktiv wie folgt definiert:

  a) Alle in F enthaltenen Konstanten sind in D(F). Enthält F keine Konstanten, so sei eine Konstante a in D(F).

  b) Ist f ein in F vorkommendes, k-stelliges Funktionssymbol und sind $t_1, \ldots, t_k \in D(F)$ Terme, so sei $f(t_1, \ldots, t_k) \in D(F)$.

  Die Menge D(F) heißt das **Herbrand-Universum** von F

Herbrand-Theorie

- Beispiel:

x	y	R(a, x, f(x), y)
a	a	R(a, a, f(a), a)
a	f(a)	R(a, a, f(a), f(a))
f(a)	a	R(a, f(a), f(f(a)), a)
f(a)	f(a)	R(a, f(a), f(f(a)), f(a))
a	f(f(a))	R(a, a, f(a), f(f(a)))
...	...	...

Jacques Herbrand
1903-1931

- **Definition**:
  Sei F eine geschlossene Skolemformel. Eine zu F passende Struktur $\alpha = (U, \varphi)$ heißt eine **Herbrand-Struktur** für F, falls gilt

  a) U = D(F).

  b) Ist f ein in F vorkommendes k-stelliges Funktionssymbol und sind $t_1, \ldots, t_k \in D(f)$, so gilt $\varphi(f)\,(\alpha(t_1), \ldots, \alpha(t_k)) = f(t_1, \ldots, t_k)$.

  Man beachte dabei, dass letztere Vorschrift wohldefiniert ist, denn $\varphi(f)$ ist eine Abbildung $\varphi(f) : U^k \to U$ und es gilt $\alpha(t_i) \in U$.

Herbrand-Theorie

- **Definition**:
  Sei F eine geschlossene Skolemformel. Eine Herbrand-Struktur α für F heißt ein **Herbrand-Modell** für F, falls $\alpha \models F$
- **Satz**:
  Sei F eine geschlossene Skolemformel. Dann ist F genau dann erfüllbar, wenn F ein Herbrand-Modell besitzt.

  Zum Beweis: v.r.n.l. trivial. V.l.n.r.: Konstruktion eines korrespondierenden Herbrand-Modells; Induktion.
- Folgerung: **Satz** von Löwenheim–Skolem
  Jede erfüllbare Formel F der Quantorenlogik besitzt ein abzählbares Modell (Modell mit abzählb. Grundmenge)

  Beweis: F $\Rightarrow$ erfüllungs-äquivalente Skolemformel G. G besitzt Herbrand-Modell mit abz. Grundmenge D(G)

Leopold Löwenheim
1878–1957

Herbrand-Theorie

Kurt Gödel
1906-1978

- **Definition**:
  Sei F eine geschlossene Skolemformel, also von der Form
  $F = \bigwedge_{x1} \ldots \bigwedge_{xn} F^*$ mit der Matrixformel F^* von F.
  Die **Herbrand-Expansion** von F ist definiert als

  $$E(F) \rightleftharpoons \{F^* [x_1 / t_1] \ldots [x_n / t_n] \mid t_1, \ldots, t_n \in D(F)\}$$

  - Die Formeln in E(F) entstehen also, indem die Terme in D(F) in jeder möglichen Weise für die Variablen in F^* substituiert werden.
- **Satz von Gödel–Herbrand–Skolem**:
  Für jede Skolemformel F gilt: F ist erfüllbar genau dann, wenn E(F) (im aussagenlogischen Sinn) erfüllbar ist.
  Beweis: z.z.: F besitzt ein Herbrand-Modell gdw. E(F) erfüllbar ist.
- **Satz von Herbrand**:
  Eine Skolemformel F ist unerfüllbar gdw. es eine endliche Teilmenge von E(F) gibt, die (im aussagenlog. Sinn) unerfüllbar ist.
  Beweis: Satz von G–H–S kombiniert mit dem **Endlichkeitssatz**.

Grundresolutionsalgorithmus

- Aus dem Satz von Herbrand ergibt sich direkt ein Algorithmus zum Nachweis der Unerfüllbarkeit einer quantorenlogischen Formel G:
  1. Ersetze G durch eine erfüllungsäquivalente geschlossene Skolemformel F.
  2. Schreibe F in der Form F = $\bigwedge_{x1} \dots \bigwedge_{xn}$ F* und bringe F* in KNF, resultierend in H.
  3. Setze M := ∅ und i := 0.
  4. i := i + 1. Berechne H_i von E(H) und füge es zu M hinzu.
  5. Betrachte M als Menge aussagenlogischer Formeln und prüfe mithilfe des aussagenlogischen Resolutionskalküls, ob M unerfüllbar ist. Falls ja, gib "M ist unerfüllbar" aus und stoppe. Falls nein: **goto** 4.
- Bem.: Ist G erfüllbar, so ist dies eine Unendlichschleife!
  Semi-Entscheidbarkeit

Unifikation

- Die Suche nach **Grundinstanzen** – alle freien Variablen sind durch variablenfreie Terme ersetzt –, die schließlich zu einer Resolutionsherleitung der leeren Klausel führt, ist mit dem systematischen (und vorausschauenden) Durchprobieren aller **Grundsubstitutionen** sehr ineffizient.
- Quantorenlogische Resolution nach J. Alan Robinson (1930–):
  - Quantorenlogische Resolventen werden aus quantorenlogischen Klauseln erzeugt, wobei jeder Resolutionsschritt mit einer geeigneten Substitution einhergeht, die gewisse Literale in den Ausgangsklauseln zueinander komplementär macht.
  - Beispiel: Es genügt, in den beiden Klauseln

    $$\{P(x), \neg Q(g(x))\}, \{\neg P(f(y))\}$$

    die Substitution [x/f(y)] durchzuführen, um die Resolvente

    $\{\neg Q(g(f(y)))\}$ zu erhalten! …Idee des **allgemeinsten Unifikators**

Definition: Unifikation

Eine **Substitution** θ ist eine Abbildung, die endlich vielen Variablen $X_1, \ldots, X_n$ (evtl. variablenbehaftete) Terme $t_1, \ldots, t_n$ zuordnet:

$$\theta = \{X_1 \rightarrow t_1, \ldots, X_n \rightarrow t_n\}$$

Zwei Ausdrücke E und F sind voneinander **Varianten**, falls es Substitutionen θ und ϕ gibt mit:

$$E = F\theta \text{ und } F = E\phi$$

Eine Substitution θ ist **Unifikator** für zwei Ausdrücke E1 und E2, falls gilt:

$$E_1\theta = E_2\theta$$

E1 und E2 heißen dann **unifizierbar**.

Definition: Unifikation

Ein Unifikator θ für E1 und E2 ist **Variante eines zweiten Unifikators** p für E1 und E2, falls es eine Substitution ϕ gibt, derart dass gilt

$$\theta = p\phi$$

Ein **allgemeinster Unifikator** für E1 und E2 ist ein Unifikator, von dem jeder andere Unifikator für E1 und E2 eine Variante ist.

Unifikationsalgorithmus

Ein **allgemeinster Unifikator** θ für zwei Ausdrücke E_1 und E_2 lässt sich — sofern diese unifizierbar sind — **effektiv** auffinden:

1. $A_1 := E_1$, $A_2 := E_2$, $\Theta := \emptyset$
2. Gilt $A_1 = A_2$?

   ja: Fertig: Θ ist ein allgemeinster Unifikator von E_1 und E_2!

   nein: Enthält die Abweichungsmenge $d(A_1, A_2)$ eine Variable υ und einen Ausdruck t, in dem υ nicht auftritt (**occur-check**)?

   ja: $\theta := \{\upsilon \to t\}$

   $A_i := \theta(A_i)$ $(i = 1,2)$

   $\Theta := \theta \circ \Theta$

   weiter mit Schritt 2

   nein: Fehlschlag: E_1 und E_2 sind nicht unifizierbar!

Unifikationsalgorithmus

Dabei ist $d(E_1, E_2)$ die Abweichungsmenge der Ausdrücke E_1 und E_2, die wie folgt bestimmte Menge $\{t_1, t_2\}$:

1. Die beiden Ausdrücke werden simultan von links nach rechts durchsucht, bis die erste Stelle angetroffen wird, an der E_1 und E_2 unterschiedliche Symbole aufweisen.
2. t_i ist der Unterausdruck von E_i $(i = 1,2)$, der an dieser Stelle beginnt.

Unifikationsalgorithmus

- Der **occur-check** überprüft,
  - ob während der Berechnung des potentiellen Unifikators bei der Substitution eines Terms t für eine Variable X die Variable X im Term t vorkommt;
  - Falls ja, schlägt die Unifikation fehl.
- Unifiziere

$$t_1 = g(\underline{h(U, V)}, f(U)) \quad \text{und} \quad t_2 = g(\underline{X}, f(X))$$

Substitution $X \rightarrow h(U, V)$ liefert

$$t_1 = g(h(U, V), f(\underline{U})) \quad \text{und} \quad t_2 = g(h(U, V), f(\underline{h(U, V)}))$$

Aber hier führt der occur-check zur vorzeitigen Terminierung mit "nicht unifizierbar"

Unifikation: Beispiel

$t_1 = f(\underline{Y}, h(g(X,X), k(Y)))$ und
$t_2 = f(\underline{g(U,V)},h(Z,k(Z)))$

1. Hauptfunktionssymbole von t_1 und t_2 stimmen überein (und ihre Stelligkeit auch).
2. erste Argumentstellen von f in t_1 und t_2 (occur-check fällt negativ aus); $Y \rightarrow g(U, V)$ ergibt:

   $t_1 = f(g(U, V), h(\underline{g(X,X)}, k(g(U, V))))$ und
   $t_2 = f(g(U,V),h(\underline{Z},k(Z)))$

3. zweite Argumentstellen von f in t_1 und t_2 haben identische Funktionssymbole; $Z \rightarrow g(X,X)$ liefert:

   $t_1 = f(g(U, V), h(g(X, X), k(g(\underline{U}, V))))$ und
   $t_2 = f(g(U, V), h(g(X, X), k(g(\underline{X}, X))))$

Unifikation: Beispiel

4. schließlich resultiert $U \rightarrow X$

$$t_1 = f(g(X, V), h(g(X, X), k(g(X, \underline{V})))) \text{ und}$$
$$t_2 = f(g(X, V), h(g(X, X), k(g(X, \underline{X}))))$$

gefolgt von $X \rightarrow V$ in:

$$t_1 = f(g(V, V), h(g(V, V), k(g(V, V)))) = t_2$$

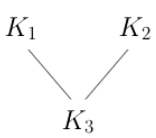
damit ist ein allgemeinster Unifikator:

$$p = \{Y \rightarrow g(V, V), Z \rightarrow g(V, V), U \rightarrow V, X \rightarrow V\}$$

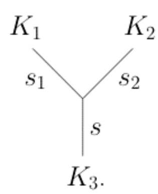
Quantorenlogische Resolution

- **Definition**:
  Seien K_1, K_2, K_3 quantorenlogische Klauseln. K_3 heißt **(quantorenlogische) Resolvente** von K_1 und K_2 , wenn gilt:
  1) Es gibt Variablenumbenennungen s_1 und s_2, so dass $\tilde{K}_1 = s_1(K_1)$ und $\tilde{K}_2 = s_2(K_2)$ keine gemeinsamen Variablennamen besitzen.
  1) Es gibt Literale $L_1, ..., L_k$ in $\tilde{K}_1$ und Literale $L'_1, ..., L'_m$ in $\tilde{K}_2$, so dass die Menge $L = \{\neg L_1, ..., \neg L_k, L'_1, ..., L'_m\}$ unifizierbar ist. Sei s der allgemeinste Unifikator von L.
  2) Es gilt $K_3 = s\,((\tilde{K}_1 \setminus \{L_1, ..., L_k\}) \cup (\tilde{K}_2 \setminus \{L'_1, ..., L'_m\}))$.

In diesem Fall schreiben wir:

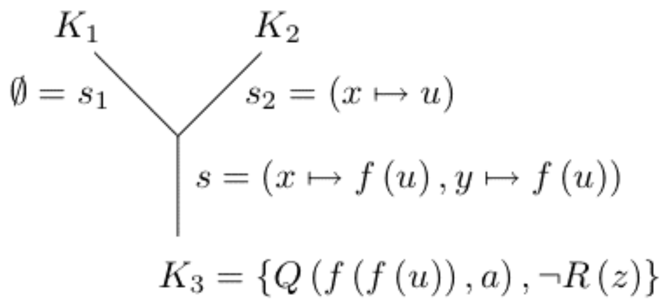


oder



Quantorenlogische Resolution: Beispiel

- Sei $K_1 = \{P(x), Q(f(x), a)\}$, $K_2 = \{\neg P(f(x)), \neg P(y), \neg R(z)\}$
- Dann gilt



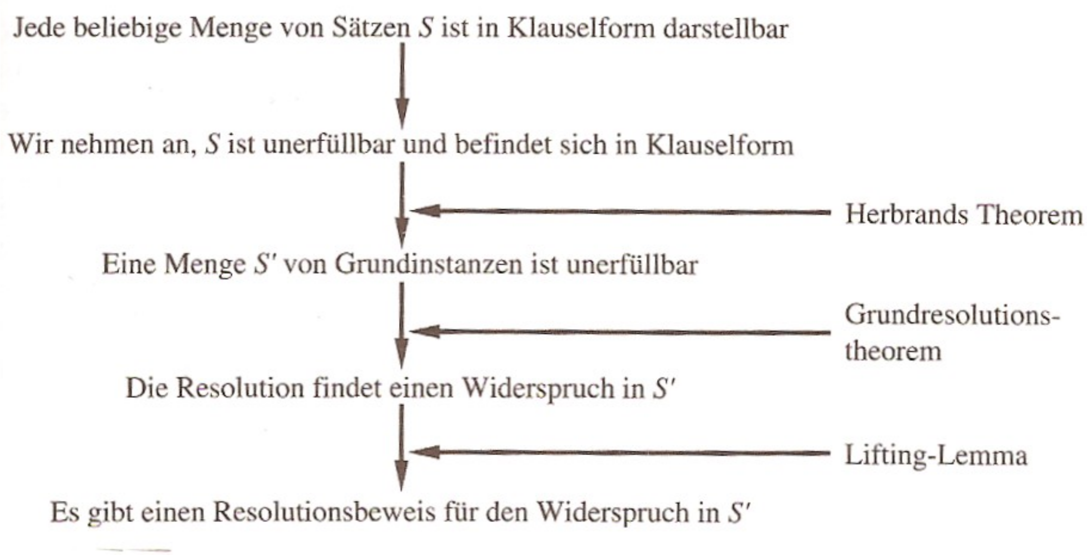
auch: [x/u]

$\tilde{K}_1 = s_1(K_1) = K_1$, $\tilde{K}_2 = s_2(K_2) = \{\neg P(f(u)), \neg P(y), \neg R(z)\}$

Resolutionssatz der Quantorenlogik

- **Lifting-Lemma**:
  Gegeben seien zwei quantorenlogische Klauseln K1, K2 und zwei Grundsubstitutionen s1, s2. Sei $\tilde{K}_1 = s_1(K_1)$ und $\tilde{K}_2 = s_2(K_2)$
  Gibt es eine (aussagenlogische) Resolvente $\tilde{K}_3$ von $\tilde{K}_1$ und $\tilde{K}_2$ so gibt es auch eine quantorenlogische Resolvente K_3 von K_1 und K_2, so dass $\tilde{K}_3$ aus K_3 durch eine Grundsubstitution entsteht.
  Beweis: Schöning, S. 97f.
- **Resolutionssatz der Quantorenlogik**:
  Sei F eine quantorenlogische Formel und F* die Matrixklauselform von F. F ist unerfüllbar gdw. $\emptyset \in Res^{\infty}(F^*)$
  Beweis: Schöning, S. 99f.

Struktur des Vollständigkeitsbeweises der quantorenlogischen Resolution



Resolution auf Logikprogrammen

- Das im aussagenlogischen Teil Gesagte zu **Resolutionsstrategien** (Folie 19) gilt hier analog!
- Sei W eine Formel mit den freien Variablen $X_1, \ldots, X_n$
  - $\bigwedge_{x_1} \ldots \bigwedge_{x_n}$ W heißt der **universelle Abschluss**,
  - $\bigvee_{x_1} \ldots \bigvee_{x_n}$ W heißt der **existentielle Abschluss**.
  - Kurzschreibweise: $\forall W$ und $\exists W$

Resolution auf Logikprogrammen

- **Anfragen** an ein Logikprogramm von der Form

  $? \; A_1 \wedge \ldots \wedge A_n$

  werden interpretiert als **Zielklauseln**:

  $\neg\exists(A_1 \wedge \ldots \wedge A_n)$

  Für ein gegebenes Logikprogramm P versucht man, mittels Resolution zu zeigen:

  $P \cup \{\neg\exists(A_1 \wedge \ldots \wedge A_n)\} \vdash \square$

Resolution auf Logikprogrammen

- **Satz**:

  Korrekt berechnete Antworten
  Jede erfolgreiche Abarbeitung der Anfrage $A_1 \wedge \ldots \wedge A_n$ an das Logikprogramm P liefert eine Substitution θ^* mit:

  $$P \models \forall((A_1 \wedge \ldots \wedge A_n)\, \theta^*)$$

  Die Antwortsubstitution θ wird durch Komposition der Unifikatoren in der Ableitung zu $P \cup \{\neg\exists(A_1 \wedge \ldots \wedge A_n)\} \vdash \square$ konstruiert.

Subjunktive Lesart der Resolution

- **Subjunktive Normalform**: Ausgehend von der KNF wandle Adjunktionen in Subjunktionen um.
- Die Resolution erscheint dann als Transitivität der Subjunktion.
- Beispiel: Gegeben sei folgende Datenbasis

Konjunktive NF	Subjunktive NF
$\neg P(w) \vee Q(w)$	$P(w) \rightarrow Q(w)$
$P(x) \vee R(x)$	$True \rightarrow P(x) \vee R(x)$
$\neg Q(y) \vee S(y)$	$Q(y) \rightarrow S(y)$
$\neg R(z) \vee S(z)$	$R(z) \rightarrow S(z)$

Daraus soll S(A) abgeleitet werden. Also:

Hinzufügen von $\neg S(A)$ – in subjunktiver NF: $S(A) \rightarrow False$ – und Anwendung der Resolution, bis ein Widerspruch (leere Klausel) entsteht.

Subjunktive Lesart der Resolution