

Logikbasierte Systeme der KI;

Wissensverarbeitung

Prädikatenlogik

Prof. Dr. M. Schmidt-Schauß

SoSe 2024

Stand der Folien: 28. Mai 2024

Prädikatenlogik

In der Aussagenlogik nicht ausdrückbar:

- Beziehungen **zwischen** bestimmten Objekten
- Eigenschaft P **gilt** für (mindestens) ein Objekt
- Die Eigenschaft P **gilt** für **alle** Objekte
- **Existenz** von Objekten mit bestimmten Eigenschaften

Obige Aussagen sind ausdrückbar in der **Prädikatenlogik**

Quantifizierte Aussagenlogik (bzw. Quantifizierte Boolesche Logik)

mit \forall, \exists für aussagenlogische Variablen:

Beispiel: $\forall A. \exists B : A \Leftrightarrow B$

Aussagen teilweise ausdrückbar

(Verallgemeinerungs-) Stufen der Prädikatenlogik:

PL_i: Prädikatenlogik i .Stufe

- **PL₀**: Keine Quantoren erlaubt = Aussagenlogik
- **PL₁**: Quantifizieren über Individuen z.B. $\forall x.P(x)$
- **PL₂**: Quantifizieren über Beziehungen (Funktionssymbole) und Prädikate z.B. $\forall P.\forall f.\forall x.P(f(x))$
- **PL₃**: Quantifizieren über Eigenschaften von Eigenschaften
- ...

Wir betrachten nur: **PL₁**

PL₁

Wie in jeder Logik:

- **Syntax**: Syntaktische gültige Formeln (formale Sprache)
- **Semantik**: Bedeutung (Interpretation) der Formeln
Wahr / falsch und Abhängigkeit
(Sätze, Erfüllbarkeit, ...)

Definition ist in PL₁ etwas aufwändiger als in der Aussagenlogik

Beispiele für PL_1 -Formeln

$$\forall x : Planet(x) \Rightarrow (\exists y : Sonne(y) \wedge umkreist(x, y))$$

$$\forall x : Planet(x) \Rightarrow \neg(Sonne(x))$$

$$\exists(x) : Planet(x)$$

$$\forall x : Sonne(x) \Rightarrow umkreist(innersterPlanet(x), x)$$

$$\forall x : Sonne(x) \vee Planet(x)$$

Syntax von PL_1 : Signaturen

Signatur $\Sigma = (\mathcal{F}, \mathcal{P})$, wobei

- \mathcal{F} Menge der **Funktionssymbole**
- \mathcal{P} Menge der **Prädikatsymbole**
- die Mengen \mathcal{F} und \mathcal{P} sind disjunkt
- jedes $f \in \mathcal{F}$ und jedes $P \in \mathcal{P}$ hat eine **Stelligkeit** $arity(f) \geq 0$ bzw. $arity(P) \geq 0$

Konstanten: $f \in \mathcal{F}$ mit $arity(f) = 0$

Forderung: mindestens eine Konstante in \mathcal{F}

Syntax von PL_1 : Terme

Prädikatenlogische Terme $T(\Sigma, V)$

- V abzählbar unendliche Menge von Variablen
- $\Sigma = (\mathcal{F}, \mathcal{P})$ Signatur

$T(\Sigma, V)$ ist induktiv definiert durch

- für alle $x \in V$: $x \in T(\Sigma, V)$
- Wenn
 - $t_1, \dots, t_n \in T(\Sigma, V)$
 - $f \in \mathcal{F}$ mit $arity(f) = n$

dann $f(t_1, \dots, t_n) \in T(\Sigma, V)$

Beispiel: $f(g(x, a), y, z)$

(wenn $arity(f) = 3$, $arity(g) = 2$, $arity(a) = 0$ und $x, y, z \in V$)

 PL_1 : Syntax von Formeln

Formeln: $F(\Sigma, V)$ induktiv definiert über $\Sigma(\mathcal{F}, \mathcal{P})$, Variablen V

- **Prädikatenlogische Atome:** Wenn
 - $P \in \mathcal{P}$ mit $arity(P) = n$,
 - $t_1, \dots, t_n \in T(\Sigma, V)$
 dann $P(t_1, \dots, t_n) \in F(\Sigma, V)$
- **Komplexe Formeln:** Falls $F, G \in F(\Sigma, V)$, $x \in V$, dann auch:
 - $(\neg F)$
 - $(F \vee G)$
 - $(F \wedge G)$
 - $(F \Rightarrow G)$
 - $(F \Leftrightarrow G)$
 - $(\forall x : F)$
 - $(\exists x : F)$

in $F(\Sigma, V)$

PL₁: Abkürzungen

- $\forall x_1, \dots, x_n : F = \forall x_1 : (\forall x_2 : (\dots (\forall x_n . F)))$
- $\exists x_1, \dots, x_n : F = \exists x_1 : (\exists x_2 : (\dots (\exists x_n . F)))$
- Klammern lassen wir gemäß den üblichen Regeln weg
- Konstantensymbole: Schreibweise a statt $a()$
- Wir verwenden auch true und false

Notation

- Variablen u, v, w, x, y, z
- Konstanten a, b, c, d, e
- Mehrstellige Funktionssymbole f, g, h
- Prädikatensymbole P, Q, R, T

Beispiel

Signatur $\Sigma := (\underbrace{\{a, b, f, g\}}_{\mathcal{F}}, \underbrace{\{P, Q, R\}}_{\mathcal{P}})$ mit

$$\begin{aligned} \text{arity}(a) &= \text{arity}(b) = \text{arity}(P) = 0, \\ \text{arity}(f) &= \text{arity}(Q) = 1, \\ \text{arity}(g) &= \text{arity}(R) = 2 \end{aligned}$$

Variablen $V = \{x, y, z, \dots\}$

Terme $T(\Sigma, V) =$

$x, y, z, \dots,$
 $a, b, f(a), f(b), f(x), \dots$
 $g(a, a), g(a, b), g(a, f(a)), \dots$
 $f(f(a)), f(f(b)), \dots f(g(a, a)), \dots$
 $g(f(f(a)), a), \dots$
 \dots

Formeln $F(\Sigma, V) =$

$P, Q(a), Q(b), Q(x), \dots$
 $R(a, a), \dots R(a, b), \dots$
 $\neg P, \neg Q(a), \dots$
 $P \wedge Q(a), P \wedge \neg Q(a), \dots$
 $P \vee Q(a), P \vee \neg Q(a), \dots$
 $P \Rightarrow Q(a), P \Leftrightarrow \neg Q(a), \dots$
 $\forall x : Q(x), \dots$
 $\exists x : R(x, y) \Rightarrow Q(x), \dots$

Binder, Geltungsbereiche, Skopus

- $\forall x \dots$ und $\exists x \dots$ **binden** die Variable x
- **Geltungsbereich (Skopus, scope)** von x in $\forall x.F$ (bzw. $\exists x.F$) ist F
- Bindungsbereich reicht soweit wie möglich (Konvention)
- Bei mehreren Bindern: Der innerste Bindungsbereich zählt
Beispiel $\exists x.(Q(x) \vee \forall x.P(x))$
- Variablen außerhalb eines Skopus: **freie Variable**

$FV(\cdot) =$ Menge der freien Variablen von \dots , Beispiele:

- $FV(x) = FV(f(x)) = FV(g(x, g(x, a))) = \{x\}$
- $FV(P(x) \wedge Q(y)) = \{x, y\}$
- $FV(\exists x : R(x, y)) = \{y\}$.

Umbenennung

Gebundene Variablen darf man **umbenennen**

Beispiel: $\exists x.Q(x) \vee \forall x.P(x) \equiv \exists x_1.Q(x_1) \vee \forall x_2.P(x_2)$

Aber: Dabei keine freien Variablen **eingangen**

$\forall x.P(z) \not\equiv \forall z.P(z)$

Freie Variablen darf man **nicht** umbenennen!

Sprechweisen in PL_1

- **Atom:** Formel der Art $P(t_1, \dots, t_n)$, wobei $P \in \mathcal{P}$ und $t_i \in T(\Sigma, V)$
- **Literal:** Atom oder ein negiertes Atom (z.B. $P(a)$ und $\neg P(f(a, x))$)
- **Grundterm:** Term $t \in T(\Sigma, V)$ ohne Variablensymbole, d.h. $FV(t) = \emptyset$
- **Grundatom:** Atom F ohne Variablensymbole, d.h. $FV(F) = \emptyset$
- **geschlossene Formel:** Formel F ohne freie Variablen, d.h. $FV(F) = \emptyset$
- **Klausel:** geschlossene Formel F mit einem Quantorpräfix nur aus Allquantoren
d.h. $F = \forall x_1, \dots, x_n : F'$ und F' ist eine Disjunktion von Literalen.



Semantik: Interpretation



Logiker: (Alfred Tarski 1901-1983)

Eigenschaften / Vorteile einer denotationalen Semantik

- Idee: Man macht eine unabhängige formale Beschreibung einer Objekt-Welt passend zur Logik.
- Man kann abklären was wahre und was falsche Aussagen sind.
- Umformungsregeln kann man verifizieren



Semantik: Interpretation

Beachte: Semantische Wahrheitswerte sind 0 und 1 (falsch und wahr)

Interpretation

Eine **Interpretation** $I = (S, I_V)$ mit $S = (D_S, \mathcal{F}_S, \mathcal{P}_S)$ besteht aus:

- nichtleere Menge D_S (**Trägermenge**)
- \mathcal{F}_S = Interpretation jedes $f \in \mathcal{F}$ als $arity(f)$ -stellige **totale Funktion f_S über D_S**
- \mathcal{P}_S = Interpretation jedes $P \in \mathcal{P}$ als $arity(P)$ -stellige **Relation P_S über D**
(Ausnahme: 0-stelliges P : wird entweder als 0 oder 1 interpretiert)
- Variablenbelegung $I_V : V \rightarrow D_S$

Semantik: Beispiel



Sonne; Planet; umkreist, innersterPlanet

- nichtleere Menge
 $D_S = \{UnsreSonne, Merkur, Venus, Erde, Mars\}$
- Funktion $I(\text{innersterPlanet}) : UnsreSonne \rightarrow Merkur$
- Relationen: $I(\text{Sonne}) = \{UnsreSonne\}$.
 $I(\text{umkreist}) = \{(Merkur, UnsreSonne), (Venus, UnsreSonne), (Erde, UnsreSonne), (Mars, UnsreSonne)\}$
- usw.

Variablenbelegung: z.B. bei Formeln und Auswertung:
 x kann mit allen Objekten in D belegt werden:

$$\forall x : Sonne(x) \Rightarrow umkreist(innersterPlanet(x), x)$$

Semantik: Interpretation – Erweiterung auf Terme

Fortsetzung auf Terme: $I(t)$ ordnet Term t ein Element aus D_S zu:

- $I(f(t_1, \dots, t_n)) = f_S(I(t_1), \dots, I(t_n))$
- $I(x) = I_V(x)$

Erweiterung der Interpretation auf Formeln

Interpretation von Formeln (gegeben $I = ((D_S, \mathcal{F}_S, \mathcal{P}_S), I_V)$)

$$I(\text{false}) = 0 \text{ und } I(\text{true}) = 1$$

$$I(Q(t_1, \dots, t_n)) = \begin{cases} 1, & \text{falls } (I(t_1), \dots, I(t_n)) \in Q_S \\ 0, & \text{falls } (I(t_1), \dots, I(t_n)) \notin Q_S \end{cases}$$

$$I(\neg F) = \begin{cases} 1, & \text{falls } I(F) = 0 \\ 0, & \text{falls } I(F) = 1 \end{cases}$$

$$I(F \vee G) = \begin{cases} 1, & \text{falls } I(F) = 1 \text{ oder } I(G) = 1 \\ 0, & \text{sonst} \end{cases}$$

$$I(F \wedge G) = \begin{cases} 1, & \text{falls } I(F) = 1 \text{ und } I(G) = 1 \\ 0, & \text{sonst} \end{cases}$$

$$I(F \implies G) = \begin{cases} 1, & \text{falls } I(F) = 0 \text{ oder } I(G) = 1 \\ 0, & \text{sonst} \end{cases}$$

$$I(F \iff G) = \begin{cases} 1, & \text{falls } I(F) = I(G) \\ 0, & \text{sonst} \end{cases}$$

Erweiterung der Interpretation auf Formeln (2)

$$I(\forall x : F) = \begin{cases} 1, & \text{falls } I[a/x](F) = 1 \text{ für alle } a \in D_S \\ 0, & \text{sonst} \end{cases}$$

$$I(\exists x : F) = \begin{cases} 1, & \text{falls } I[a/x](F) = 1 \text{ für ein } a \in D_S \\ 0, & \text{sonst} \end{cases}$$

Notation dabei: $I[a/x] = \begin{cases} I(y), & \text{falls } y \neq x \\ a & \text{falls } y = x \end{cases}$

Termalgebra

Termalgebra

Für eine Signatur $\Sigma = (\mathcal{F}, \mathcal{P})$ und Menge von Variablen V sei F_T die Menge der Funktionen

$$F_T = \{f_\Sigma \mid f \in \mathcal{F}, \text{arity}(f) = n, f_\Sigma(t_1, \dots, t_n) := f(t_1, \dots, t_n)\}$$

Dann nennt man $(T(\Sigma, V), F_T)$ die **Termalgebra** über der Signatur Σ .

Semantik: Modelle, Tautologien, ...

Modell einer PL_1 -Formel F :

Interpretation I mit $I(F) = 1$

Schreibweise $I \models F$

Eine PL_1 -Formel heißt:

- **allgemeingültig**, wenn sie in allen Interpretationen gültig ist.
- **erfüllbar**, wenn sie von einer Interpretation erfüllt wird, d.h. es gibt I mit $I \models F$.
- **unerfüllbar (widersprüchlich)**, wenn sie von keiner Interpretation erfüllt wird.
- **falsifizierbar**, wenn sie in einer Interpretation falsch wird.

Tautologie, Satz: allgemeingültige, geschlossene PL_1 -Formel



Zusammenhänge

Auch in PL_1 gilt:

- Eine Formel F ist allgemeingültig gdw. $\neg F$ unerfüllbar
- Falls F nicht allgemeingültig ist: F ist erfüllbar gdw. $\neg F$ erfüllbar



Beispiele

Jede Interpretation macht 0-stelliges Prädikat P' wahr oder falsch:

allgemeingültig: $P' \vee \neg P'$

unerfüllbar: $P' \wedge \neg P'$

Formel $\forall x.P(x)$ erfüllbar und falsifizierbar:

- Sei $I = (I_V, (\underbrace{\{0, 1\}}_{D_S}, \mathcal{F}_S, \underbrace{\{P_S = \{0, 1\}\}}_{P_S}))$

$I(\forall x : P(x)) = 1$ gdw. $I[0/x](P(x)) = 1$ und $I[1/x](P(x)) = 1$

$I[0/x](P(x)) = I[0/x](x \in P_S) = 0 \in P_S = 1$

$I[1/x](P(x)) = I[1/x](x \in P_S) = 1 \in P_S = 1$

D.h. $I \models \forall x.P(x)$

- Sei $I = (I_V, (\underbrace{\{0, 1\}}_{D_S}, \mathcal{F}_S, \underbrace{\{P_S = \{0\}\}}_{P_S}))$

$I(\forall x : P(x)) = 1$ gdw. $I[0/x](P(x)) = 1$ und $I[1/x](P(x)) = 1$

$I[0/x](P(x)) = I[0/x](x \in P_S) = 0 \in P_S = 1$

$I[1/x](P(x)) = I[1/x](x \in P_S) = 1 \in P_S = 0$

D.h. $I(\forall x : P(x)) = 0$

Beispiel (2)

Für welche s, t ist die Klausel $\{P(s), \neg P(t)\}$ eine Tautologie?

- $s = t$: $I(\{P(s), \neg P(s)\}) = I(s \in P_S \text{ oder } I(s) \notin P_S) \Rightarrow \{P(s), \neg P(s)\}$ ist Tautologie

- $s \neq t$:

Verwende die Termalgebra, wobei die Konstanten sind:

alle Konstanten die in s, t vorkommen und Konstanten c_{x_i} für alle Variablen x_i , die in s, t vorkommen, also $\mathcal{F}_S := \{f_s = f\}$

Wähle P_S so dass $I(s) \in P_S$, aber $I(t) \notin P_S$. Das zeigt:

Niemals eine Tautologie.

Und: Domain D muss mehr als ein Element enthalten um ein Gegenbeispiel zu konstruieren.

Semantische Folgerung

Definition

$F \models G$ gdw. G gilt (ist wahr) in allen Modellen von F .

Sprechweise: G ist semantische Folgerung von F . oder:
 G ist Konsequenz von F

Beachte: Es gibt i.A. unendliche viele Modelle,
die i.a. unendlich viele Elemente haben
D.h. aus praktischer Sicht: \models kann so nicht geprüft werden.



Deduktionstheorem

Deduktionstheorem der PL_1

Für alle Formeln F und G gilt:

$F \vdash G$ gdw. $F \Rightarrow G$ ist allgemeingültig.



Beweis durch Widerspruch

Da F allgemeingültig ist genau dann wenn $\neg F$ unerfüllbar ist, folgt unmittelbar:

$$\begin{aligned} F \models G \\ \text{gdw.} \\ \neg(F \Rightarrow G) \text{ ist unerfüllbar (widersprüchlich)} \\ \text{gdw.} \\ F \wedge \neg G \text{ ist unerfüllbar.} \end{aligned}$$

Unentscheidbarkeit der Prädikatenlogik

Theorem

Es ist unentscheidbar, ob eine geschlossene Formel ein Satz der Prädikatenlogik ist.

Beweisidee: Kodiere jede Turingmaschine M als PL-Formel F_M , sodass F_M genau dann ein Satz ist, wenn M (bei leerem Band als Eingabe) hält.

Aber:

Theorem

Die Menge der Sätze der Prädikatenlogik ist rekursiv aufzählbar.

Konsequenzen

Unentscheidbarkeit:

Es gibt **keinen** Algorithmus (bzw. Deduktionssystem) der für jede geschlossene PL_1 -Formel nach endlicher Zeit entscheiden kann, ob dies ein Satz ist oder nicht.

Aufzählbarkeit:

Aber: Es gibt Algorithmen, die bei Tautologie T als Eingabe, nach endlicher Zeit terminieren, und mit Ausgabe: Ist Tautologie.

Man benutzt dazu die rekursive Aufzählbarkeit: Irgendwann wird die Tautologie T in der Aufzählung vorkommen.

D.h.: bei Eingabe: beliebige Formel F

- Ausgabe: "Ist Tautologie", dann ist der Beweis erbracht
- Abbruch nach gewisser Zeit (Timeout, oder Speicherüberlauf): Man weiß nichts.

Eine entscheidbare Formelklasse

Als Beispiel**geschlossene, quantorenfreie Formeln.**

Eine Formel, die keine Quantoren enthält und keine Variablen.

Einfaches Entscheidungsverfahren:

Jedes $P(s_1, \dots, s_n)$ wird einfach als aussagenlogische Variable interpretiert. Danach aussagenlogische Methoden.

$$\{P(f(a)) \wedge (P(f(a)) \implies P(g(b)))\} \vdash P(g(b))$$

aussagenlogisch, nach Erzeugen von Namen und Umbenennung:
 $\{A \wedge (A \implies B)\} \vdash B$

Dh.: Die Aussagenlogik ist Teil der Prädikatenlogik.

Es gibt weitere entscheidbare Klassen. . .

Vorbereitung der Deduktion: Normalformen von PL_1 -Formeln

Ziel: Berechne **Klauselnormalform** so dass die Unerfüllbarkeit erhalten bleibt.

Klauselnormalform:

- Konjunktion von Disjunktion von Literalen
- Quantoren nur ganz außen
- Nur Allquantoren
- Alles andere (insbes. \exists -Quantoren) wird „wegtransformiert“

Als Formel: $\forall x_1, \dots, x_n. ((L_{1,1} \vee \dots \vee L_{1,n_1})$
 $\wedge \dots \wedge$

$\forall x'_1, \dots, x'_n. (L_{m,1} \vee \dots \vee L_{m,n_m}))$

wobei $L_{i,j}$ von der Form $P(t_1, \dots, t_k)$ oder $\neg P(t_1, \dots, t_k)$,
 P Prädikat, t Terme

Elementare Rechenregeln

$$\begin{aligned} \neg \forall x : F &\Leftrightarrow \exists x : \neg F \\ \neg \exists x : F &\Leftrightarrow \forall x : \neg F \\ (\forall x : F) \wedge G &\Leftrightarrow \forall x : (F \wedge G) \quad \text{falls } x \text{ nicht frei in } G \\ (\forall x : F) \vee G &\Leftrightarrow \forall x : (F \vee G) \quad \text{falls } x \text{ nicht frei in } G \\ (\exists x : F) \wedge G &\Leftrightarrow \exists x : (F \wedge G) \quad \text{falls } x \text{ nicht frei in } G \\ (\exists x : F) \vee G &\Leftrightarrow \exists x : (F \vee G) \quad \text{falls } x \text{ nicht frei in } G \\ \forall x : F \wedge \forall x : G &\Leftrightarrow \forall x : (F \wedge G) \\ \exists x : F \vee \exists x : G &\Leftrightarrow \exists x : (F \vee G) \end{aligned}$$

Beachte ...

$$\forall x : (F \vee G) \not\equiv (\forall x : F) \vee (\forall x : G)$$

$$\text{Bsp.: } \forall x : (\text{Frau}(x) \vee \text{Mann}(x)) \quad \text{vs.} \quad (\forall x : \text{Frau}(x)) \vee (\forall x : \text{Mann}(x))$$

$$\exists x : (F \wedge G) \not\equiv (\exists x : F) \wedge (\exists x : G)$$

$$\text{Bsp.: } \exists x : (\text{Frau}(x) \wedge \text{Mann}(x)) \quad \text{vs.} \quad (\exists x : \text{Frau}(x)) \wedge (\exists x : \text{Mann}(x))$$

Klauselnormalform

Wesentlicher (neuer) Schritt: Entfernung der Existenzquantoren.

Sogenannte **Skolemisierung** (nach Thoralf Skolem)

Plan: In $\exists x : F[x]$ ersetze x durch eine neue Konstante a d.h. $\exists x : F[x] \rightarrow F[a]$ (alle x durch a ersetzen).

Funktioniert noch nicht ganz, wenn \forall -Quantoren außen sind!

Deshalb: In $\forall x_1, \dots, x_n : \exists y. F[x_1, \dots, x_n, y]$ ersetze y durch Funktion $f(x_1, \dots, x_n)$

Pränexform / Negationsnormalform

PL₁-Formel F

- ist in **Pränexform** gdw. $F = Q_1x_1 : Q_2x_2, \dots, Q_n : x_n(F')$, wobei $Q_i = \forall$ oder $Q_i = \exists$ und F' enthält keine Quantoren
- ist in **Negationsnormalform** gdw. F enthält weder \implies noch \iff und \neg steht nur vor Atomen

Herstellung dieser Normalformen mit den Rechenregeln möglich:

- Pränexform: Quantoren nach außen schieben, evtl. Umbenennung von Variablen nötig
- Negationsnormalform: \iff , \implies entfernen, dann Negationen nach innen schieben

Skolemisierung

Notation:

- $G[x_1, \dots, x_n, y]$ sei Formel mit Vorkommen von x_1, \dots, x_n, y
- $G[x_1, \dots, x_n, t]$ entspricht: statt y steht t in der Formel

Theorem

Eine Formel $F = \forall x_1 \dots x_n : \exists y : G[x_1, \dots, x_n, y]$ ist (un-)erfüllbar gdw. $F' = \forall x_1 \dots x_n : G[x_1, \dots, x_n, f(x_1, \dots, x_n)]$ (un-)erfüllbar ist, wobei f ein n -stelliges Funktionssymbol ist mit $n \geq 0$, das nicht in G vorkommt.

Beweisskizze: (semantisch: über Erfüllbarkeit)

\Leftarrow : klar

\Rightarrow : Es gibt I mit $I(F) = 1$.

Insbesondere für alle $d_1, \dots, d_n \in D_S$ gibt es $e \in D_S$ mit

$I(G[d_1, \dots, d_n, e]) = 1$

Baue I' : Wie I aber f_S so dass $f_S(d_1, \dots, d_n) = e$.

Dann: $I'(F') = I'(G[d_1, \dots, d_n, f_S(d_1, \dots, d_n)]) = 1$

Beispiele (1)

$$\exists x : P(x) \rightarrow P(a)$$

$$\forall x : \exists y : Q(f(y, y), x, y) \rightarrow \forall x : Q(f(g(x), g(x)), x, g(x))$$

$$\forall x, y : \exists z : x + z = y \rightarrow \forall x, y : x + h(x, y) = y.$$

Beispiele (2) Skolemisierung

$$\forall x : Planet(x) \Rightarrow (\exists y : Sonne(y) \wedge umkreist(x, y))$$

$$\forall x : Planet(x) \Rightarrow \neg(Sonne(x))$$

$$\exists(x) : Planet(x)$$

$$\forall x : Sonne(x) \Rightarrow umkreist(innersterPlanet(x), x)$$

$$\forall x : Sonne(x) \vee Planet(x)$$

Die erste und dritte Formel werden zu:

$$\forall x : Planet(x) \Rightarrow (Sonne(SonneVon(x)) \wedge umkreist(x, SonneVon(x)))$$

$$Planet(Erde)$$

Das erhält die Erfüllbarkeit und Unerfüllbarkeit,
Aber Signatur und Modelle sind verschieden

Beispiele (2)

Skolemisierung erhält i.A. nicht die Allgemeingültigkeit (Falsifizierbarkeit):

- $(\forall x : P(x)) \vee \neg(\forall x : P(x))$ ist eine Tautologie.
- $(\forall x : P(x)) \vee (\exists x : \neg P(x))$ ist äquivalent dazu.
- $\forall x : P(x) \vee \neg P(a)$ nach Skolemisierung.

I mit $D_S = \{a, b\}$, $P_S = \{a\}$ falsifiziert die letzte Formel!

Beispiele (3)

Andersherum gilt es:

Skolemisierung erhält die Unerfüllbarkeit:

- $(\forall x : P(x)) \wedge \neg(\forall x : P(x))$ ist ein Widerspruch
- $(\forall x : P(x)) \wedge (\exists x : \neg P(x))$ ist äquivalent dazu.
- $\forall x : P(x) \wedge \neg P(a)$ nach Skolemisierung.

Einsetzen von a für x zeigt: letzte Formel ist widersprüchlich.

Allgemeinere Skolemisierung

Unterschied: Existenz-Quantor an beliebiger Position p , aber nicht im Skopus eines anderen Existenzquantors, kein \neg , \implies , \iff oben drüber

Theorem [Gödel, Herbrand, Skolem]

Sei F eine geschlossene Formel, G eine existentiell quantifizierte Unterformel in F an einer Position p , Weiterhin sei G nur unter Allquantoren, Konjunktionen, und Disjunktionen. Die Allquantoren über G binden die Variablen x_1, \dots, x_n mit $n \geq 0$. D.h. F ist von der Form $F[\exists y : G'[x_1, \dots, x_n, y]]$.

Dann ist $F[G]$ (un-)erfüllbar gdw. $F[G'[x_1, \dots, x_n, f(x_1, \dots, x_n)]]$ (un-)erfüllbar ist, wobei f ein n -stelliges Funktionssymbol ist, das nicht in G vorkommt.

Transformation in CNF

- Unter Erhaltung der Un-(Erfüllbarkeit)!
- Eingabe: **geschlossene** Formel

Prozedur:

1. Elimination von \iff und \implies :

$$F \iff G \rightarrow F \implies G \wedge G \implies F$$

$$F \implies G \rightarrow \neg F \vee G$$

2. Negation ganz nach innen schieben:

$$\neg \neg F \rightarrow F$$

$$\neg(F \wedge G) \rightarrow \neg F \vee \neg G$$

$$\neg(F \vee G) \rightarrow \neg F \wedge \neg G$$

$$\neg \forall x : F \rightarrow \exists x : \neg F$$

$$\neg \exists x : F \rightarrow \forall x : \neg F$$

Transformation in CNF (2)

3. (Optimierung:) Skopus von Quantoren minimieren, d.h. Quantoren so weit wie möglich nach innen schieben

$$\forall x : (F \wedge G) \rightarrow (\forall x : F) \wedge G \quad \text{falls } x \text{ nicht frei in } G$$

$$\forall x : (F \vee G) \rightarrow (\forall x : F) \vee G \quad \text{falls } x \text{ nicht frei in } G$$

$$\exists x : (F \wedge G) \rightarrow (\exists x : F) \wedge G \quad \text{falls } x \text{ nicht frei in } G$$

$$\exists x : (F \vee G) \rightarrow (\exists x : F) \vee G \quad \text{falls } x \text{ nicht frei in } G$$

$$\forall x : (F \wedge G) \rightarrow \forall x : F \wedge \forall x : G$$

$$\exists x : (F \vee G) \rightarrow \exists x : F \vee \exists x : G$$

4. Alle gebundenen Variablen sind systematisch umzubenennen, um Namenskonflikte aufzulösen.

Transformation in CNF (3)

5. Existenzquantoren werden durch Skolemisierung eliminiert
6. Allquantoren nach außen schieben
7. Distributivität (und Assoziativität, Kommutativität) iterativ anwenden, um \wedge nach außen zu schieben um Klauselform zu erreichen ("Ausmultiplikation").

$$F \vee (G \wedge H) \rightarrow (F \vee G) \wedge (F \vee H)$$
- 7.b Alternativ: Tseitin-Transformation.
8. Allquantoren vor die Klauseln schieben und gebundene Variablen umbenennen.

Transformation in CNF (2)

Das Resultat dieser Prozedur ist eine Konjunktion von Disjunktionen (Klauseln) von Literalen:

$$\begin{aligned} & (L_{1,1} \vee \dots \vee L_{1,n_1}) \\ \wedge & (L_{2,1} \vee \dots \vee L_{2,n_2}) \\ \wedge & \\ \dots & \\ \wedge & (L_{k,1} \vee \dots \vee L_{k,n_k}) \end{aligned}$$

oder in Mengenschreibweise:

$$\begin{aligned} & \{ \{L_{1,1}, \dots, L_{1,n_1}\}, \\ & \{L_{2,1}, \dots, L_{2,n_2}\}, \\ & \dots \\ & \{L_{k,1}, \dots, L_{k,n_k}\} \} \end{aligned}$$

Jede Klausel ist implizit allquantifiziert (alle Variablen).

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:

$$\begin{aligned} \forall y : ((\exists x_1 : P(x_1)) \wedge \neg Q(y)) \vee (Q(y) \wedge \exists x_2 : (\neg P(x_2))) \forall y : \\ (P(f_1(y)) \wedge \neg Q(y)) \vee (Q(y) \wedge \exists x_2 : (\neg P(x_2))) \forall y : \\ (P(f_1(y)) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(f_2(y))) \end{aligned}$$

6. Allquantoren nach außen schieben

$$\forall y : (P(f_1(y)) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(f_2(y)))$$

7. Ausmultiplizieren zu Klauseln

$$\begin{aligned} \forall y : (P(f_1(y)) \wedge \neg Q(y)) \vee (Q(y) \wedge \\ \neg P(f_2(y))) \forall y : ((P(f_1(y)) \vee Q(y)) \wedge (\neg Q(y) \vee Q(y)) \wedge \\ (P(f_1(y)) \vee P(f_2(y))) \wedge (\neg Q(y) \vee P(f_2(y)))) \end{aligned}$$

8. Quantoren vor die Klauseln schieben, und umbenennen.

$$\begin{aligned} \forall y. ((P(f_1(y)) \vee Q(y)) \wedge (\neg Q(y) \vee Q(y)) \wedge (P(f_1(y)) \vee \\ P(f_2(y))) \wedge (\neg Q(y) \vee P(f_2(y)))) \forall y. (P(f_1(y)) \vee Q(y)) \wedge \\ \forall y. (\neg Q(y) \vee Q(y)) \wedge \forall y. (P(f_1(y)) \vee P(f_2(y))) \wedge \\ \forall y. (\neg Q(y) \vee P(f_2(y))) \forall y_1. (P(f_1(y_1)) \vee Q(y_1)) \wedge \\ \forall y_2. (\neg Q(y_2) \vee Q(y_2)) \wedge \forall y_3. (P(f_1(y_3)) \vee P(f_2(y_3))) \wedge \end{aligned}$$

$$Q(y_2)) \wedge (P(f_1(y_3)) \vee P(f_2(y_3))) \wedge (\neg Q(y_4) \vee P(f_2(y_4)))$$

Beispiel

Eingabe $\neg \exists y : \forall x : P(x) \iff Q(y)$

1. Entfernen von \Rightarrow und \iff :

$$\begin{aligned} \neg \exists y : \forall x : P(x) \iff Q(y) \iff \neg \exists y : \forall x : (P(x) \Rightarrow \\ Q(y)) \wedge (Q(y) \Rightarrow P(x)) \iff \neg \exists y : \forall x : (\neg P(x) \vee Q(y)) \wedge (Q(y) \Rightarrow \\ P(x)) \iff \neg \exists y : \forall x : (\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x)) \end{aligned}$$

2. Negationen nach innen schieben:

$$\begin{aligned} \neg \exists y : \forall x : (\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x)) \forall y : \neg \forall x : \\ (\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x)) \forall y : \exists x : \\ \neg((\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x))) \forall y : \exists x : \\ \neg(\neg P(x) \vee Q(y)) \vee \neg(\neg Q(y) \vee P(x)) \forall y : \exists x : (P(x) \wedge \neg Q(y)) \vee \\ \neg(\neg Q(y) \vee P(x)) \forall y : \exists x : (P(x) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(x)) \end{aligned}$$

3. Skopus von Quantoren minimieren:

$$\begin{aligned} \forall y : \exists x : (P(x) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(x)) \forall y : (\exists x : \\ (P(x) \wedge \neg Q(y))) \vee (\exists x : (Q(y) \wedge \neg P(x))) \forall y : ((\exists x : \\ P(x)) \wedge \neg Q(y)) \vee (\exists x : (Q(y) \wedge \neg P(x))) \forall y : ((\exists x : \\ P(x)) \wedge \neg Q(y)) \vee (Q(y) \wedge \exists x : (\neg P(x))) \end{aligned}$$

4. Umbenennen:

$$\forall y : ((\exists x : P(x)) \wedge \neg Q(y)) \vee (Q(y) \wedge \exists x : (\neg P(x))) \forall y :$$

Anmerkungen: Transformation in Klauselnormalform

- Analog zur Aussagenlogik: exponentielle Vergrößerung vermeidbar:
 - \iff -Elimination: durch Einführen von Abkürzungen
 - Ausmultiplizieren (Vermeidbar durch Tseitin- Transformation)
- Man sieht: Ursprüngliche Form / Idee der Formeln geht verloren

Resolution

- Resolutionskalkül für PL_1 (Alan Robinson)
- Versucht mit Resolution Widersprüchlichkeit nachzuweisen
- Eingabe: Prädikatenlogische Klauselmenge
- Kalkül erzeugt neue Klauseln
- Widerspruch = leere Klausel \square .
- Zunächst betrachten wir: **Grundresolution**
- Danach: **Allgemeine Resolution**

Grundresolution

- Grundklauseln $\{L_1, \dots, L_m\}$, d.h. L_i enthalten keine Variablen, nur Grundterme
- Z.B. $\{P(f(a)), \neg Q(g(h(b, c)))\}$ mit $a, b, c, f, g \in \mathcal{F}$

Grundresolution:

Elternklausel 1: L, K_1, \dots, K_m

Elternklausel 2: $\neg L, N_1, \dots, N_n$

Resolvente: $\frac{L, K_1, \dots, K_m, \neg L, N_1, \dots, N_n}{K_1, \dots, K_m, N_1, \dots, N_n}$

Grundresolution

Einige Anmerkungen.

- Wenn es keine Variablen in der Klauselmenge gibt, Dann kann die Klauselmenge direkt in eine aussagenlogische Klauselmenge übersetzt werden.
- Dann reicht der DPLL-Algorithmus aus um Widersprüchlichkeit zu entscheiden.

D.h. Grundresolution soll nur die Resolution im Spezialfall illustrieren.

Erinnerung: (Grund-)resolution ist korrekt:

Die Resolvente (neue Klausel) ist
logische Folgerung
aus der aktuellen Klauselmenge!

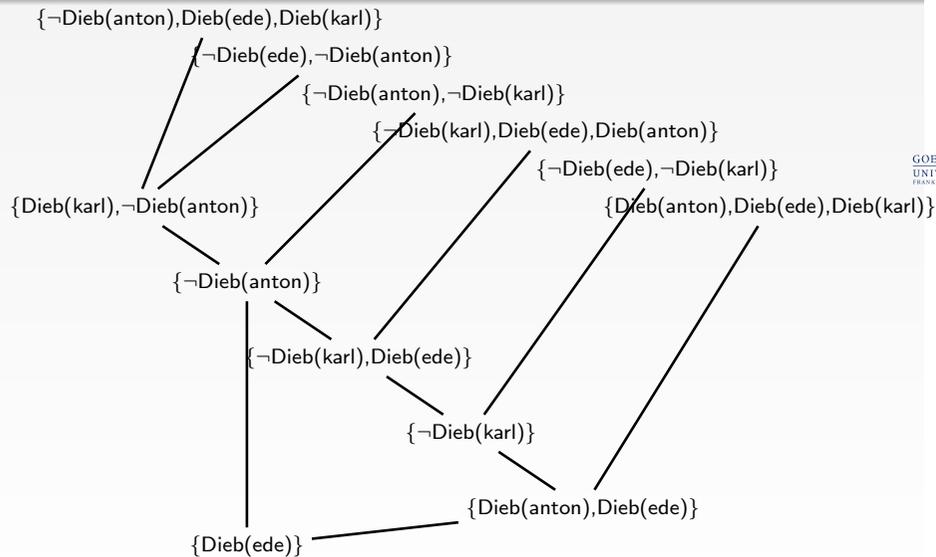
Beispiel

- A1: $\text{Dieb}(\text{anton}) \vee \text{Dieb}(\text{ede}) \vee \text{Dieb}(\text{karl})$
 A2: $\text{Dieb}(\text{anton}) \Rightarrow (\text{Dieb}(\text{ede}) \vee \text{Dieb}(\text{karl}))$
 A3: $\text{Dieb}(\text{karl}) \Rightarrow (\text{Dieb}(\text{ede}) \vee \text{Dieb}(\text{anton}))$
 A4: $\text{Dieb}(\text{ede}) \Rightarrow (\neg \text{Dieb}(\text{anton}) \wedge \neg \text{Dieb}(\text{karl}))$
 A5: $\neg \text{Dieb}(\text{anton}) \vee \neg \text{Dieb}(\text{karl})$

Klauselform:

- A1: $\text{Dieb}(\text{anton}), \text{Dieb}(\text{ede}), \text{Dieb}(\text{karl})$
 A2: $\neg \text{Dieb}(\text{anton}), \text{Dieb}(\text{ede}), \text{Dieb}(\text{karl})$
 A3: $\neg \text{Dieb}(\text{karl}), \text{Dieb}(\text{ede}), \text{Dieb}(\text{anton})$
 A4a: $\neg \text{Dieb}(\text{ede}), \neg \text{Dieb}(\text{anton})$
 A4b: $\neg \text{Dieb}(\text{ede}), \neg \text{Dieb}(\text{karl})$
 A5: $\neg \text{Dieb}(\text{anton}), \neg \text{Dieb}(\text{karl})$

Beispiel: Resolution dazu



Resolutions-Herleitung von Dieb(ede) ist korrekt!

Grundresolution

Satz

Die Grundresolution ist korrekt:

$$\begin{aligned} C_1 &:= L, K_1, \dots, K_m \\ C_2 &:= \frac{\neg L, N_1, \dots, N_n}{R} \\ R &= K_1, \dots, K_m, N_1, \dots, N_n \end{aligned}$$

Dann gilt $C_1 \wedge C_2 \models R$.

Beweis: Zeige: Wenn $I(C_1 \wedge C_2) = 1$ dann auch $I(R) = 1$.

Fall: $I(L) = 1$, dann ist $I(\neg(L)) = 0$ und $I(N_1 \vee \dots \vee N_n) = 1$.
D.h. für ein N_i gilt: $I(N_i) = 1$, und daher $I(R) = 1$

Fall: $I(L) = 0$. Analog muss für ein K_i gelten $I(K_i) = 1$ und daher $I(R) = 1$.

Widerlegungsvollständigkeit

Theorem

Jede endliche unerfüllbare Grundklauselmengem lässt sich durch Resolution widerlegen.

Allgemeine Resolution

Theorem

[Kompaktheit] (Satz von Gödel, Herbrand und Skolem)

Jede endliche unerfüllbare Klauselmengem (mit Variablen) hat eine endliche Menge M von Grundklauseln als Instanzen, so dass M unerfüllbar ist.

- Grundinstanzen durch Substitution bilden, dann Grundresolution versuchen?
(oder DPLL)
Nachteil: erforderliche Anzahl der Grundinstanzen unklar
Diese kann sehr groß sein!
- **Andere Variante:** Resolution auf Klauseln mit Variablen, **Allgemeine Resolution** s.u.

Substitution

- **Substitution** σ : **endliche** Abbildung von Variablen auf Terme
- Schreibweise: $\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$
- Erweiterung von σ auf Terme:

$$\sigma(x) = x, \text{ wenn } \sigma \text{ } x \text{ nicht abbildet}$$

$$\sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n))$$
- Anwendung von Substitutionen auf Literale und Klauseln entsprechend

$$\sigma(\{L_1, \dots, L_n\}) = \{\sigma(L_1), \dots, \sigma(L_n)\}$$

Beispiele

$$\begin{aligned} \sigma = \{x \mapsto a\} & \quad \sigma(x) = a \\ & \quad \sigma(f(x, x)) = f(a, a) \\ \sigma = \{x \mapsto g(x)\} & \quad \sigma(x) = g(x) \\ & \quad \sigma(f(x, x)) = f(g(x), g(x)) \\ & \quad \sigma(\sigma(x)) = g(g(x)) \\ \sigma = \{x \mapsto y, y \mapsto a\} & \quad \sigma(x) = y \\ & \quad \sigma(\sigma(x)) = a \\ & \quad \sigma(f(x, y)) = f(y, a) \\ \sigma = \{x \mapsto y, y \mapsto x\} & \quad \sigma(x) = y \\ & \quad \sigma(f(x, y)) = f(y, x) \end{aligned}$$

Komposition

Komposition von Substitutionen: für alle x : $(\sigma\tau)x := \sigma(\tau(x))$

Beispiele:

- $\{x \mapsto a\}\{y \mapsto b\} = \{x \mapsto a, y \mapsto b\}$
- $\{y \mapsto b\}\{x \mapsto f(y)\} = \{x \mapsto f(b), y \mapsto b\}$
- $\{x \mapsto b\}\{x \mapsto a\} = \{x \mapsto a\}$

Klauseln und Klauselmengen: Konventionen

Konventionen

- Jede Klausel hat nur All-Quantoren als Quantorenprefix
- Klauseln sind geschlossen (keine freien Variablen)
- Schreibweise:
 - Man lässt die Quantoren weg
 - Annahme: verschiedene Klauseln sind variablen-disjunkt

Zum Beispiel

Klausel, formal richtig: $\forall x, y, z : \neg P(f(x, g(y))) \vee Q(z)$

Klausel, Notation: $\neg P(f(x, g(y))), Q(z)$

Substitutionen ändern die Erfüllbarkeit nicht

Sei $\{K_1, \dots, K_n\}$ eine prädikatenlogische Klauselmenge und σ eine Substitution.

Dann ist $\{K_1, \dots, K_n\}$ genau dann erfüllbar, wenn $\{K_1, \dots, K_n, \sigma(K_i)\}$ erfüllbar ist. (Da Klauseln allquantifiziert)

- D.h. man könnte:
 - Erst substituieren, bis man Grundklauseln hat (alle Möglichkeiten oder ausreichend viele)
 - Dann Grundresolution anwenden
- Das sind aber zu viele Möglichkeiten. Eigentlich muss man Literale $P(t)$ und $\neg P(s')$ der Elternklauseln nur gleich machen (Grundterm nicht erforderlich)



Resolution mit Unifikation

Elternklausel 1: L, K_1, \dots, K_m σ ist eine Substitution
 Elternklausel 2: $\neg L', N_1, \dots, N_n$ mit $\sigma(L) = \sigma(L')$
 Resolvente: $\frac{}{\sigma(K_1, \dots, K_m, N_1, \dots, N_n)}$

Da σ die Literale L und L' gleich macht, nennt man σ auch **Unifikator**

Eigenschaften:

- Wenn $C \rightarrow C \cup \{R\}$ wobei R Resolvente, dann ist C erfüllbar gdw. $C \cup \{R\}$ erfüllbar.
- D.h. Resolution mit Unifikation ist korrekt.



Beispiel: Resolution alleine reicht nicht!

Der Friseur rasiert alle, die sich nicht selbst rasieren

$$\forall x : \neg(\text{Rasiert}(x, x)) \iff \text{Rasiert}(\text{friseur}, x)$$

CNF: $\{\{\text{Rasiert}(x, x), \text{Rasiert}(\text{friseur}, x)\}, \{\neg \text{Rasiert}(\text{friseur}, x), \neg \text{Rasiert}(x, x)\}\}$

Resolution mit Unifikation:

$$\frac{\{\text{Rasiert}(x, x), \text{Rasiert}(\text{friseur}, x)\} \quad \sigma = \{x \mapsto \text{friseur}, y \mapsto \text{friseur}\} \quad \{\neg \text{Rasiert}(\text{friseur}, y), \neg \text{Rasiert}(y, y)\}}{\{\text{Rasiert}(\text{friseur}, \text{friseur}), \neg \text{Rasiert}(\text{friseur}, \text{friseur})\}}$$

weitere Resolventen:

$\{\text{Rasiert}(\text{friseur}, x), \neg \text{Rasiert}(\text{friseur}, x)\}, \{\text{Rasiert}(x, x), \neg \text{Rasiert}(x, x)\},$

Jetzt erhält man keine weiteren Resolventen mehr!
 aber: Die Formel ist widersprüchlich.

Faktorisierung

Das Friseur-Beispiel zeigt:

- Allgemeine Resolution ist **nicht** widerlegungsvollständig!
- D.h. Widersprüche werden nicht immer gefunden.

Notwendig:

Faktorisierung (Schrumpfung):

Elternklausel: L, L', K_1, \dots, K_m $\sigma(L) = \sigma(L')$
 Faktor: $\frac{}{\sigma(L, K_1, \dots, K_m)}$

Faktorisierung ist nicht notwendig in einem aussagenlogischen Resolutions-Kalkül !

Friseur



$$\frac{C1 : \{ \text{Rasiert}(x, x), \text{Rasiert}(\text{friseur}, x) \}}{F1 : \{ \text{Rasiert}(\text{friseur}, \text{friseur}) \}}$$

$$\frac{C2 : \{ \neg \text{Rasiert}(\text{friseur}, y), \neg \text{Rasiert}(y, y) \}}{F2 : \neg \{ \text{Rasiert}(\text{friseur}, \text{friseur}) \}}$$

$$\text{Resolvente von } F1 + F2 : \square$$


Prädikatenlogischer Resolutionskalkül

Eingabe: Klauselmenge S

Folgende Regeln werden beliebig oft angewendet:

Regeln:

- ① $S \rightarrow S \cup \{R\}$, wobei R eine Resolvente von zwei (nicht notwendig verschiedenen) Klauseln aus S ist.
- ② $S \rightarrow S \cup \{F\}$, wobei F ein Faktor einer Klausel aus S ist.

Erfolg, wenn: $\square \in S$, dann widersprüchlich.

Beispiel



Transitivität der Teilmengenrelation:

Prädikatensymbole \subseteq und \in (Infix geschrieben)

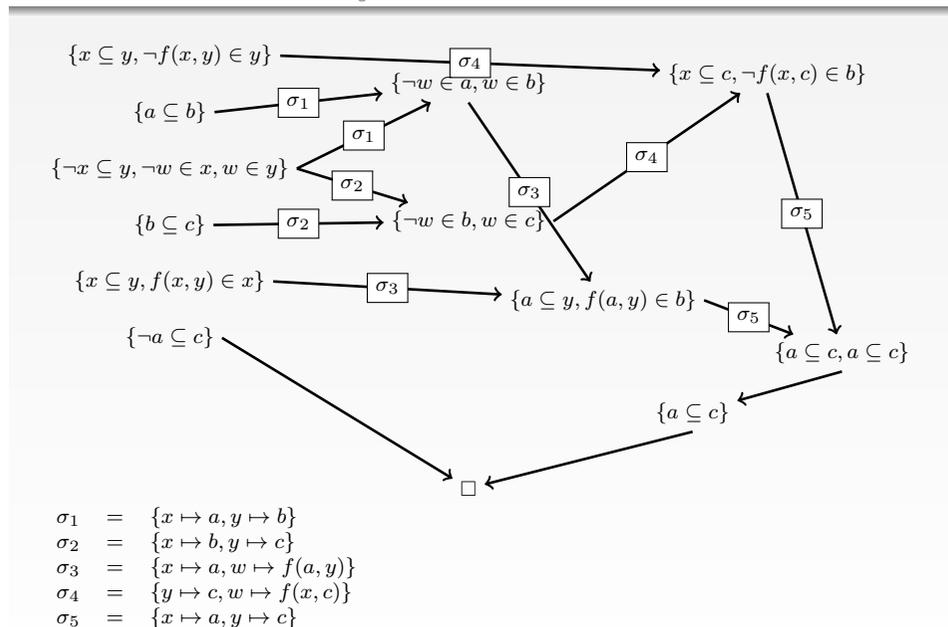
- Axiom: Definition von \subseteq unter Benutzung von \in

$$F_1 = \forall x, y : x \subseteq y \Leftrightarrow \forall w : w \in x \Rightarrow w \in y$$

- Folgerung: $F_2 = \forall x, y, z : x \subseteq y \wedge y \subseteq z \Rightarrow x \subseteq z$
- Wir wollen zeigen $F_1 \models F_2$ bzw. $F_1 \Rightarrow F_2$ ist Tautologie.
- Daher zeigen wir $\neg(F_1 \Rightarrow F_2)$ ist widersprüchlich. (wir können daher mit $F_1 \wedge \neg F_2$ starten).

Umwandlung in Klauselform ergibt 6 Klauseln:

- H1: $\{ \neg x \subseteq y, \neg w \in x, w \in y \}$
- H2: $\{ x \subseteq y, f(x, y) \in x \}$
- H3: $\{ x \subseteq y, \neg f(x, y) \in y \}$
- C1: $\{ a \subseteq b \}$
- C2: $\{ b \subseteq c \}$
- C3: $\{ \neg a \subseteq c \}$



Unifikation

- Für die Resolution und Faktorisierung braucht man Berechnung von Unifikatoren
- Man kann einen **allgemeinsten Unifikator** berechnen !!
- Vorteil: Man braucht die (i.a. unendlich vielen) spezielleren Unifikatoren nicht zu betrachten



Allgemeinster Unifikator

(MGU = Most general unifier)

- Seien s, t Terme
- σ ist **Unifikator** für s, t gdw. $\sigma(s) = \sigma(t)$
- σ ist **allgemeinster Unifikator** für s, t , gdw. σ ist ein Unifikator für s, t und für jeden anderen Unifikator ρ von s, t gibt es eine Substitution γ mit $\gamma\sigma = \rho$.
(eingeschränkt auf die Variablen der Eingabgleichung)



Beispiel

$$\frac{P(x), Q(x) \quad \neg P(y), R(y)}{Q(a), R(a)} \quad \sigma = \{x \mapsto a, y \mapsto a\}$$

σ ist ein Unifikator

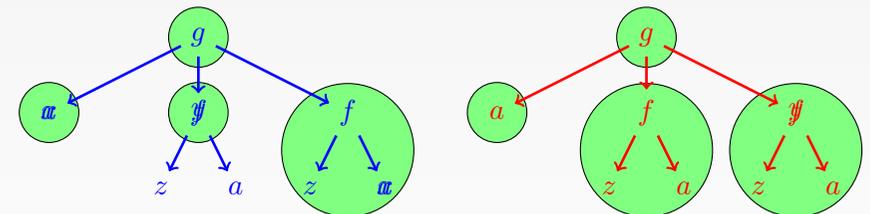
$$\frac{P(x), Q(x) \quad \neg P(y), R(y)}{Q(y), R(y)} \quad \sigma = \{x \mapsto y\}$$

σ ist ein allgemeinster Unifikator

- Allgemeinste Unifikatoren sind nicht (ganz) eindeutig:
auch $\{y \mapsto x\}$ ist ein MGU
- aber: ist allgemeinst bis auf Variablenumbenennung

Beispiel zur Unifikation

$$g(x, y, f(z, x)) \stackrel{?}{=} g(a, f(z, a), y)$$



- $x \mapsto a$
- $y \mapsto f(z, a)$

Unifikationsalgorithmus

Algorithmus Unifikationsalgorithmus U_1 **Datenstrukturen:** Γ Menge von Termgleichungen $\{s_i \stackrel{?}{=} t_i\}$ **Eingabe:** Wenn s, t unifiziert werden sollen, setze $\Gamma := \{s \stackrel{?}{=} t\}$ **Ausgabe:** Nicht unifizierbar, oder MGU für s, t **Algorithmus:**

- 1 Wenn $\Gamma = \{x_1 = t_1, \dots, x_n = t_n\}$, wobei
 - Alle x_i sind paarweise verschiedene Variablen,
 - kein x_i kommt in einem t_j vor

Dann: Gebe $\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ als MGU aus.

- 2 Sonst: Wende eine der Unifikationsregeln auf Γ an (im Anschluss)
 - Tritt dabei Fail auf, dann breche ab mit „Nicht unifizierbar“, sonst
 - Erhalte Γ' und mache mit $\Gamma := \Gamma'$ weiter mit Schritt 1.



Unifikationsregeln

Anwenden auf Menge Γ von Termgleichungen.

$$\frac{f(s_1, \dots, s_n) \stackrel{?}{=} f(t_1, \dots, t_n), \Gamma}{s_1 \stackrel{?}{=} t_1, \dots, s_n \stackrel{?}{=} t_n, \Gamma} \quad (\text{Dekomposition})$$

$$\frac{x \stackrel{?}{=} x, \Gamma}{\Gamma} \quad (\text{Löschregel})$$

$$\frac{x \stackrel{?}{=} t, \Gamma}{x \stackrel{?}{=} t, \{x \mapsto t\} \Gamma} \quad x \in FV(\Gamma), x \notin FV(t) \quad (\text{Anwendung})$$

$$\frac{t \stackrel{?}{=} x, \Gamma}{x \stackrel{?}{=} t, \Gamma} \quad t \notin V \quad (\text{Orientierung})$$



Unifikationsregeln (2)



Abbruchbedingungen:

$$\frac{f(\dots) \stackrel{?}{=} g(\dots), \Gamma}{\text{Fail}} \quad \text{wenn } f \neq g \quad (\text{Clash})$$

$$\frac{x \stackrel{?}{=} t, \Gamma}{\text{Fail}} \quad \text{wenn } x \in FV(t) \text{ und } t \neq x \quad (\text{occurs check Fehler})$$

Beispiel



$$\begin{aligned} & \{k(f(x, g(a, y)), g(x, h(y))) \stackrel{?}{=} k(f(h(y), g(y, a)), g(z, z))\} \\ \rightarrow & \{f(x, g(a, y)) \stackrel{?}{=} f(h(y), g(y, a)), g(x, h(y)) \stackrel{?}{=} g(z, z)\} & (\text{Dekomposition}) \\ \rightarrow & x \stackrel{?}{=} h(y), g(a, y) \stackrel{?}{=} g(y, a), g(x, h(y)) = g(z, z) & (\text{Dekomposition}) \\ \rightarrow & x \stackrel{?}{=} h(y), a \stackrel{?}{=} y, y \stackrel{?}{=} a, g(x, h(y)) \stackrel{?}{=} g(z, z) & (\text{Dekomposition}) \\ \rightarrow & x \stackrel{?}{=} h(y), y \stackrel{?}{=} a, g(x, h(y)) \stackrel{?}{=} g(z, z) & (\text{Orientierung}) \\ \rightarrow & x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, g(x, h(a)) \stackrel{?}{=} g(z, z) & (\text{Anwendung, } y=a) \\ \rightarrow & x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, x \stackrel{?}{=} z, h(a) \stackrel{?}{=} z & (\text{Dekomposition}) \\ \rightarrow & x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, x \stackrel{?}{=} z, z \stackrel{?}{=} h(a) & (\text{Orientierung}) \\ \rightarrow & x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, z \stackrel{?}{=} h(a) & (\text{Anw., } z = h(a)) \end{aligned}$$

MGU: $\{x \mapsto h(a), y \mapsto a, z \mapsto h(a)\}$ Unifizierte Terme: $k(f(h(a), g(a, a)), g(h(a), h(a)))$

Eigenschaften des Unifikationsalgorithmus

Theorem

Der Unifikationsalgorithmus terminiert, ist korrekt und vollständig. Er liefert, einen allgemeinsten Unifikator, wenn die Eingabe unifizierbar ist. Wenn die Eingabe nicht unifizierbar ist, bricht der Algorithmus ab.

- In dieser Form: MGU kann exponentiell groß werden:
- $\{x_1 \stackrel{?}{=} f(x_2, x_2), x_2 \stackrel{?}{=} f(x_3, x_3), \dots, x_{n-1} \stackrel{?}{=} f(x_n, x_n), x_n \stackrel{?}{=} a\}$
- MGU: $\{x_1 \mapsto f(f(f(f \dots f(f(a, a), f(a, a)) \dots))), \dots\}$
- Die üblichen Algorithmen (mit Sharing) haben Komplexität $O(n \log(n))$.
- Komplexität: \mathcal{P} -complete. D.h. nicht parallelisierbar.

Zurück zum Resolutionskalkül

Resolution:

Elternklausel 1: L, K_1, \dots, K_m σ ist eine Substitution
 Elternklausel 2: $\neg L', N_1, \dots, N_n$ mit $\sigma(L) = \sigma(L')$
 Resolvente: $\frac{\sigma(K_1, \dots, K_m, N_1, \dots, N_n)}$

Abhilfe durch Extraregel:

Faktorisierung:

Elternklausel: L, L', K_1, \dots, K_m $\sigma(L) = \sigma(L')$
 Faktor: $\frac{\sigma(L, K_1, \dots, K_m)}$

Verwende für σ einen berechneten MGU!

Eigenschaften des Resolutionskalküls

Gödel-Herbrand-Skolem Theorem

Zu jeder unerfüllbaren Menge C von Klauseln gibt es eine **endliche unerfüllbare** Menge von Grundinstanzen (Grundklauseln) von C .

Zusammen mit der Widerlegungsvollständigkeit der Grundresolution kann man folgern

Satz

Der prädikatenlogische Resolutionskalkül (mit Resolution und Faktorisierung (und Unifikation)) ist korrekt und widerlegungsvollständig.

(Beweis erfordert noch ein Lifting-Lemma: Grundresolution \rightarrow allgemeine Resolution)

Optimierungen: Löschregeln

Genau wie bei Aussagenlogischer Resolution gibt es Optimierungen.

- Klauseln mit isolierte Literalen
- Tautologische Klauseln
- Subsumierte Klauseln

Isolierte Literale

Isoliertes Literal

- Sei \mathcal{C} eine Klauselmeng, D eine Klausel in \mathcal{C} und L ein Literal in D .
- L heißt **isoliert**, wenn es keine Klausel $D' \neq D$ mit einem Literal L' in \mathcal{C} gibt, so dass L und L' verschiedenes Vorzeichen haben und L und L' unifizierbar sind.

ISOL: Löschrregel für isolierte Literale

ISOL: Löschrregel für isolierte Literale

Wenn D eine Klausel aus \mathcal{C} ist mit einem isolierten Literal, dann lösche die Klausel D aus \mathcal{C} .

Satz

Die Löschrregel für isolierte Literale kann zum Resolutionskalkül hinzugenommen werden, ohne die Widerlegungsvollständigkeit zu verlieren.

Beweis: Die leere Klausel kann nicht mit D hergeleitet werden, da es keinen Resolutionspartner gibt

Beispiel

$$\begin{aligned} C1 &: P(a) \\ C2 &: P(b) \\ C3 &: \neg Q(b) \\ C4 &: \neg P(x), Q(x) \end{aligned}$$

- Resolution $C1 + C4$ ergibt: R1: $\{Q(a)\}$
- $Q(a)$ ist isoliert, daher ist die neue Klausel eine Sackgasse
- D.h. $\{Q(a)\}$ löschen oder gar nicht erst erzeugen

Subsumtion

Subsumtion

Seien D und E Klauseln. D **subsumiert** die Klausel E wenn es eine Substitution σ gibt, so dass $\sigma(D) \subseteq E$

Löschen subsumierter Klauseln ist korrekt:
Jede Resolutionsableitung, die E benutzt, hätte auch D benutzen können

Löschregel

SUBS: Löschregel für subsumierte Klauseln

Wenn D und E Klauseln aus \mathcal{C} sind, D subsumiert E und E hat nicht weniger Literale als D , dann lösche die Klausel E aus \mathcal{C} .

!!: **zusätzliche Bedingung:** E hat mind. so viele Literale wie D
Grund: Faktorisierung

$$\underbrace{\{L, L', L_1, \dots, L_n\}}_D \rightarrow \underbrace{\{\sigma(L_1), \dots, \sigma(L_n)\}}_E$$

- Faktor E wird von der Elternklausel D subsumiert
- Zusätzliche Bedingung verhindert das Löschen

Subsumtion: Beispiele

- P subsumiert $\{P, S\}$.
- $\{Q(x), R(x)\}$ subsumiert $\{R(a), S, Q(a)\}$
- $\{E(a, x), E(x, a)\}$ subsumiert $\{E(a, a)\}$. D.h eine Klausel subsumiert einen ihren Faktoren. In diesem Fall wird nicht gelöscht.
- $\{\neg P(x), P(f(x))\}$ impliziert $\{\neg P(x), P(f(f(x)))\}$ aber subsumiert nicht.

SUBS: Eigenschaften

- Subsumierte Klauseln zu finden ist \mathcal{NP} -vollständig.
Bereits die Frage ob Klausel D die Klausel E subsumiert ist \mathcal{NP} -vollständig.
- Diese hohe Komplexität ist praktisch nicht relevant, da man die Suche einschränken kann.

Theorem

Der Resolutionskalkül zusammen mit der Löschung subsumierter Klauseln ist widerlegungsvollständig.

Tautologische Klauseln

Tautologische Klausel

Sei D eine Klausel. Wir sagen dass D eine **Tautologie** ist, wenn D in allen Interpretationen wahr ist.

Beispiele: $\{P(a), \neg P(a)\}$, $\{Q(a), P(f(x)), \neg P(f(x)), Q(b)\}$ oder $\{P(x), \neg P(x)\}$.

Syntaktisches Kriterium

Klausel c ist Tautologie
gdw.

Klausel C enthält Literale L und $\neg L$.
Komplexität ist polynomiell

TAUT: Löschregel

TAUT: Löschregel für tautologische Klauseln

Wenn D eine tautologische Klausel aus der Klauselmenge \mathcal{C} ist, dann lösche die Klausel D aus \mathcal{C} .



TAUT: Eigenschaften

Offensichtlich: Korrektheit

Theorem

Die Löschregel für tautologische Klauseln ist korrekt und erhält Widerlegungsvollständigkeit



Insgesamt: 3 Löschregeln

**Theorem**

Der Resolutionskalkül zusammen mit Löschung subsumierter Klauseln, Löschung von Klauseln mit isolierten Literalen und Löschung von Tautologien ist widerlegungsvollständig.

Praktisch sind diese Löschregeln unbedingt notwendig, da sehr viele erzeugte Klauseln redundant sind.

Resolution: Strategien



Es gibt viele Ansätze zu Strategien, um das Resolutionsverfahren erfolgreicher zu machen:

- ① Gewichtung von Klauseln / Literalen; und Bevorzugung minimaler Klauseln/Literale
- ② (prädikatenlogische) Modelle als Orientierung, wo der Widerspruch eher zu finden ist.
- ③ Gewichtung: Größe der Klauseln / Literale "Bessere bevorzugen!"
- ④ Einschränkungen der Resolutionsklauseln / -Literale

Ein Beweiser Projekt: SPASS



<https://www.mpi-inf.mpg.de/departments/automation-of-logic>

Es gibt auch Resolutionsbeweiser für Prädikatenlogik.

Weitere Beweis-Systeme
für andere Problemstellungen / Rahmenbedingungen



Ein Beweiser Projekt: SPASS



Ein Beispiel: **Hilbert Logik:**

Formeln werden als Terme dargestellt.

$derivable(F)$ bedeutet, dass F eine herleitbare Formel ist.

Es gibt Axiome und den Modus Ponens (als Formel)

Axiome:

$\forall X, Y : derivable(imp(X, imp(Y, X)))$

$\forall X, Y, Z : derivable(imp(imp(X, imp(Y, Z)), imp(imp(X, Y), imp(X, Z))))$

$\forall X, Y : derivable(X) \wedge derivable(imp(X, Y)) \implies derivable(Y)$

Die zu beweisende Vermutung:

$\forall X, Y, Z : derivable(imp(imp(X, Y), imp(imp(Y, Z), imp(X, Z))))$

Bspl zu Beweiser Projekt: SPASS

Ein Beispiel zu Hilbert Logik! Syntaktisch korrekte Eingabe:

```
begin_problem(hilbert02).

list_of_descriptions.
name({*Hilbert Kalkuel-Prop-logic*}).
author({*anonymous*}).
status(unsatisfiable).
description({*example from hilberts calculus for propositional logic*}).
end_of_list.

list_of_symbols.
functions[(imp,2)].
predicates[(derivable,1)].
end_of_list.

list_of_formulae(axioms).
formula(forall([X,Y],derivable(imp(X,imp(Y,X)))).
formula(forall([X,Y,Z],derivable(imp(imp(X,imp(Y,Z)),imp(imp(X,Y),imp(X,Z))))).
formula(forall([X,Y],implies(and(derivable(X),derivable(imp(X,Y))),derivable(Y)))).
end_of_list.

list_of_formulae(conjectures).
formula(forall([X,Y,Z],derivable(imp(imp(X,Y),imp(imp(Y,Z),imp(X,Z))))).
end_of_list.

list_of_settings(SPASS).
{* set_flag(PGiven,1). set_flag(PProblem,0). *}
end_of_list.

end_problem.
```

Bspl. zu Beweiser Projekt: SPASS



Ein Beispiel: **Hilbert Logik:** Berechnung bzw. Herleitung:

<https://webspass.spass-prover.org>

Protokoll in: hilbert02-antwort.txt

Statistik: 5209 Klauseln hergeleitet, nicht-redundante: 1675 Klauseln.

Zeit: weniger als 1 Sekunde.

Lineare Resolution

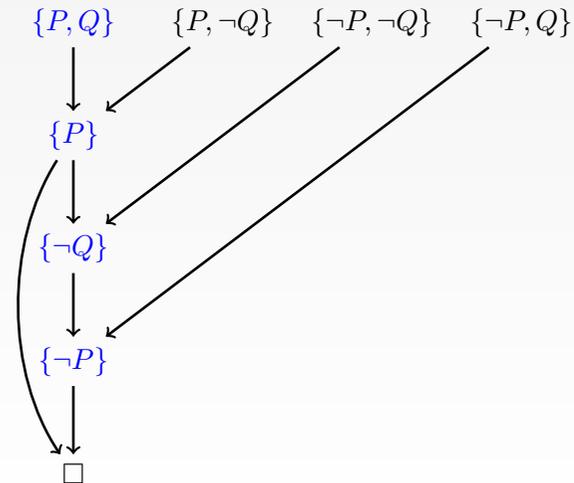
Ist eingeschränkte (spezielle Variante) der Resolution
 Gut implementierbar und gute Laufzeiteigenschaften

- Eingabe: Klauselmenge mit **Zentralklausel** und **Seitenklauseln**
- Erste Resolution: Zentralklausel + eine Seitenklausel
- Danach: Jede Resolution verwendet die zuletzt erhaltene Resolvente und eine andere Klausel
- Danach wird die Resolvente zur nächsten Zentralklausel
- (Faktorisierung ist auch dabei)



Beispiel: Lineare Resolution

Widerlege $\{\{P, Q\}, \{P, \neg Q\}, \{\neg P, Q\}, \{\neg P, \neg Q\}\}$ durch lineare Resolution mit der Zentralklausel $\{P, Q\}$:



Lineare Resolution: Eigenschaften

- Lineare Resolution + Faktorisierung zusammen sind **widerlegungsvollständig**
- Bei Hornklauseln: **Faktorisierung nicht notwendig**
- Intuition zu Hornklauseln:
Regeln: $(L_1 \wedge \dots \wedge L_n) \implies L_{n+1}$
Ziele/Anfragen: $\neg K_1 \vee \dots \vee \neg K_m$



Hornklauseln

Hornklauseln: Syntaktische eingeschränkte Klauseln

Verwendung in logischen Programmiersprachen, wie z.B. Prolog

Eine **Hornklausel** ist eine Klausel, die **höchstens ein positives Literal** enthält.

Eine Klauselmenge, die nur aus Hornklauseln besteht, nennt man **Hornklauselmenge**.

Beispiele:

- $\{\neg R(x), P(a), Q(f(y))\}$ ist **keine** Hornklausel
- $\{\neg R(f(x)), \neg P(g(x, a)), Q(y)\}$ ist eine Hornklausel
- $\{\neg R(g(y)), \neg P(h(b))\}$ ist eine Hornklausel

Hornklauseln (2)

Hornklauseln lassen sich weiter unterteilen:

- **Definite Klauseln** sind Klauseln mit **genau einem** positiven Literal.
- Definite Einklauseln (mit positivem Literal) werden auch als **Fakt** bezeichnet
- Klauseln, die **nur negative Literale** enthalten, nennt man auch ein **definites Ziel**.

Eine Menge von definiten Klauseln nennt man auch **definites Programm**.



SLD-Resolution

- Nur auf Hornklauselmengen definiert
- S = Selektionsfunktion
- L = Lineare Resolution
- D = Definite Klauseln



SLD-Resolution (2)

Verfahren

- Zentralklausel ist definites Ziel.
- Lineare Resolution mit dieser Zentralklausel
- Z.B. $\{\neg P(a), \neg Q(f(x)), \neg R(a, h(y))\}$
- Selektionsfunktion bestimmt deterministisch **welches Literal** aus der Zentralklausel als nächstes wegresolviert werden soll
- also $\neg P(a)$, $\neg Q(f(x))$, oder $\neg R(a, h(y))$

Es gilt: Die Reihenfolge der wegresolvierten Literale ist **don't care-Nichtdeterminismus**:

Wenn man Literal L_i zuerst wegresolviert und man findet die leere Klausel nicht, dann findet man sie auch nicht, wenn man zunächst Literal L_j wegresolviert



SLD-Resolution (3)

Selektionsfunktionen:

- das linkeste
- das am wenigsten instanziierte
- das am meisten instanziierte
- ...

SLD-Resolution (4)

Beachte:

- Ein definites Ziel als Zentralklausel (z.B. $\{\neg P(x, y)\}$)
- Eingabeklauseln sind definite Klauseln (alle genau ein positives Literal)
- Die lineare Resolution resolviert:
Eine Eingabeklausel mit der Zentralklausel
- **Es können nie zwei Resolventen als Elternklauseln verwendet werden!**
(Bei allgemeiner linearen Resolution ist das nicht der Fall)
- Grund: Jede Resolvente besteht ausschließlich aus negativen Literalen!

SLD-Resolution (5)

Eigenschaften:

- SLD-Resolution ist für Hornklauselmengen korrekt und widerlegungsvollständig
- Aber ist **nichtdeterministisch**: Wahl der Seitenklausel als Elternklausel
- Dieser Nichtdeterminismus ist nicht: don't-care!
- Mögliche Abhilfe: Breitensuche (erhält Vollständigkeit)
Aber: Sehr platzintensiv

SLD-Resolution (5)

Eigenschaften:

- SLD-Resolution berechnet für Hornklauselmengen Antworten auf Anfragen:
Die Gesamtsubstitution am Ende (bei einer Tiefensuche)
- Anfragen sind die negativen Klauseln.
- Eine **Antwort** ist eine Grund-Substitution σ (bzgl. einer Anfrage A), so dass die definiten Klauseln zusammen mit $\sigma(A)$ unerfüllbar sind.
- SLD-Resolution ist **vollständig in Bezug auf alle Antworten**:
- Es wird bei fairer Vorgehensweise zu jeder möglichen Antwort eine (evtl. allgemeinere) Antwort berechnet.