Einführung in die Methoden der Künstlichen Intelligenz

Prädikatenlogik

PD Dr. David Sabel

SoSe 2014

Stand der Folien: 29. Mai 2023

Prädikatenlogik

In der Aussagenlogik nicht ausdrückbar:

- Beziehungen zwischen bestimmten Objekten
- Eigenschaften gelten für alle Objekte
- Existieren Objekte für die Eigenschaften gelten

Aber ausdrückbar in der **Prädikatenlogik**

Stufen

PL$_i$: Prädikatenlogik i.Stufe

- **PL**$_0$: Keine Quantoren erlaubt = Aussagenlogik
- **PL**$_1$: Quantifizieren über Individuen z.B. $\forall x.P(x)$
- **PL**$_2$: Quantifizieren über Beziehungen (Funktionssymbole) und Prädikate z.B. $\forall P.\forall f.\forall x.P(f(x))$
- **PL**$_3$: Quantifizieren über Eigenschaften von Eigenschaften
- ...

Wir betrachten nur: **PL**$_1$

PL_1

Wie in jeder Logik:

- **Syntax**: Syntaktische gültige Formeln (formale Sprache)
- **Semantik**: Bedeutung (Interpretation) der Formeln (Sätze,Erfüllbarkeit,...)

Definition ist in PL_1 etwas aufwändiger als in der Aussagenlogik

Syntax von PL_1: Signaturen

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

- $\mathcal{F}$ Menge der **Funktionssymbole**
- $\mathcal{P}$ Menge der **Prädikatensymbole**
- 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: mind. 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
  - $t_1, \ldots, t_n \in T(\Sigma, V)$
  - $f \in \mathcal{F}$ mit $arity(f) = n$

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

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 $arity(P) = n$,
  - $t_1, \ldots, t_n \in T(\Sigma, V)$

  dann $P(t_1, \ldots t_n) \in F(\Sigma, V)$
- 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 $F(\Sigma, V)$

PL$_1$: Abkürzungen

- $\forall x_1, \ldots, x_n : F = \forall x_1 : (\forall x_2 : (\ldots (\forall x_n.F)))$
- $\exists x_1, \ldots, x_n : F = \exists x_1 : (\exists x_2 : (\ldots (\exists x_n.F)))$
- Klammern lassen wir gem. 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 Funktionssysmbole f, g, h
- Prädikatensymbol P, Q, R, T

Beispiel

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

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

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

Terme $T(\Sigma, V) =$

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

Formeln $F(\Sigma, V) =$

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

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 = Menge der freien Variablen, 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: $\exists x.Q(x) \vee \forall x.P(x) = \exists x_1.Q(x_1) \vee \forall x_2.P(x_2)$

Aber: Dabei keine freien Variablen **einfangen**

$\forall x.P(z) \neq \forall z.P(z)$

Freie Variablen darf man **nicht** umbenennen!

Sprechweisen in PL_1

- **Atom:** Formel der Art $P(t_1, \ldots t_n)$, 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 $F = \forall x_1, \ldots, x_n : F'$ und F' ist eine Disjunktionen von Literalen.

Semantik: Interpretation

Beachte: Semantische Wahrheitswerte seien 0 und 1

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 $I_V : V \to D_S$

Semantik: Interpretation – Erweiterung auf Terme

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

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

Erweiterung der Interpretation auf Formeln

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

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

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

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

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

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

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

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

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

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

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 $I \models F$

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 (Tautologie, Satz)**, wenn sie von allen Interpretationen erfüllt wird
- **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.

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	$P \wedge \neg P$

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

  $I(\forall x : P(x)) = 1$ 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 $I \models \forall x.P(x)$

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

  $I(\forall x : P(x)) = 1$ 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 $I = (I_V, (\underbrace{\{0,1\}}_{D_S}), \mathcal{F}_S, \underbrace{\{P_S = \{0\}\}}_{\mathcal{P}_S})$

  $I(\forall x : P(x)) = 1$ 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?

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

Beispiel (2)

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

- $s = t$: $I(\{P(s), \neg P(s)\}) = I(s) \in P_S$ 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.

Semantische Folgerung

Definition

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

Beachte: Es gibt i.A. unendliche viele Modelle (aus praktischer Sicht, $\models$ so nicht entscheidbar)

Deduktionstheorem

Deduktionstheorem der PL$_1$

Für alle Formeln F und G gilt: $F \models G$ gdw. $F \Rightarrow G$ ist allgemeingültig (Tautologie).

Beweis durch Widerspruch

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

$$F \models G$$

gdw.

$\neg(F \Rightarrow G)$ ist unerfüllbar (widersprüchlich)

gdw.

$F \land \neg G$ ist unerfüllbar.

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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Es gibt **kein** Deduktionssystem, dass für jede PL_1-Formel nach endlicher Zeit entscheiden kann, ob dies ein Satz ist oder nicht.

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

D.h.: Bei beliebiger Formel F als Eingabe:
Ausgabe:

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

Normalformen von PL_1-Formeln

Ziel: Berechne **Klauselnormalform** d.h.

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

$$\forall x_1, \ldots, x_n.((L_{1,1} \vee \ldots \vee L_{1,n_1}) \wedge \ldots (L_{m,1} \vee \ldots \vee L_{m,n_m}))$$

mit $Li, j = P(t_1, \ldots, t_k)$ oder $\neg P(t_1, \ldots, t_k)$,
wobei P Prädikat, t Terme

Elementare Rechenregeln

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

Beachte ...

$$\forall x : (F \lor G) \quad \not\Leftrightarrow \quad (\forall x : F) \lor (\forall x : G)$$

Bsp.: $\forall x : (\mathsf{Frau}(x) \lor \mathsf{Mann}(x))$ vs. $(\forall x : \mathsf{Frau}(x)) \lor (\forall x : \mathsf{Mann}(x))$

$$\exists x : (F \land G) \quad \not\Leftrightarrow \quad (\exists x : F) \land (\exists x : G)$$

Bsp.: $\exists x : (\mathsf{Frau}(x) \land \mathsf{Mann}(x))$ vs. $(\exists x : \mathsf{Frau}(x)) \land (\exists x : \mathsf{Mann}(x))$

Pränexform / Negationsnormalform

PL$_1$-Formel F

- ist in **Pränexform** gdw. $F = Q_1x_1 : Q_2x_2, \ldots, Q_n : x_n(F')$, 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

Klauselnormalform

Wesentlicher Schritt: Entfernung der Existenzquantoren.

$\Rightarrow$ die sogenannte **Skolemisierung** (nach Thoralf Skolem)

Klauselnormalform

Wesentlicher Schritt: Entfernung der Existenzquantoren.

$\Rightarrow$ die sogenannte **Skolemisierung** (nach Thoralf Skolem)

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

Klauselnormalform

Wesentlicher Schritt: Entfernung der Existenzquantoren.

$\Rightarrow$ die sogenannte **Skolemisierung** (nach Thoralf Skolem)

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

Funktioniert noch nicht ganz, wenn $\forall$-Quantoren oben drüber sind!

Idee 2: In $\forall x_1, \ldots, x_n : \exists y.F[x_1, \ldots, x_n, y]$ ersetze y durch Funktion $f(x_1, \ldots, x_n)$

Skolemisierung

Notation:

- $G[x_1, \ldots x_n, y]$ sei Formel mit Vorkommen von $x_1, \ldots, x_n, y$
- $G[x_1, \ldots x_n, t]$ = Alle y durch t ersetzt

Theorem

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

Beweisskizze: (über Erfüllbarkeit)
$\Leftarrow$: klar

$\Rightarrow$: Es gibt I mit $I(F) = 1$.
Insbesondere für alle $d_1, \ldots, d_n \in D_S$ gibt es $e \in D_S$ mit $I(G[d_1, \ldots, d_n, e]) = 1$
Baue I': Wie I aber f_S so dass $f_S(d_1, \ldots, d_n) = e$.
Dann: $I'(F') = I'(G[d_1, \ldots, d_n, f_S(d_1, \ldots, d_n)]) = 1$

Beispiele (1)

$$\begin{aligned}
\exists x : P(x) \quad &\rightarrow \quad P(a) \\
\forall x : \exists y : Q(f(y,y),x,y) \quad &\rightarrow \quad \forall x : Q(f(g(x),g(x)),x,g(x)) \\
\forall x,y : \exists z : x+z=y \quad &\rightarrow \quad \forall x,y : x+h(x,y)=y.
\end{aligned}$$

Beispiele (2)

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

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

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

Allgemeinere Skolemisierung

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

Theorem

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 $x_1, \ldots, x_n$ mit $n \geq 0$. D.h. F ist von der Form $F[\exists y : G'[x_1, \ldots, x_n, y]]$.
Dann ist $F[G]$ (un-)erfüllbar gdw. $F[G'[x_1, \ldots, x_n, f(x_1, \ldots, x_n)]]$ (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 $\Leftrightarrow$ und $\Rightarrow$: $F \Leftrightarrow G \to F \Rightarrow G \land G \Rightarrow F$
   $F \Rightarrow G \to \neg F \lor G$
2. Negation ganz nach innen schieben:

$$\begin{array}{lcl}
\neg\neg F & \to & F \\
\neg(F \land G) & \to & \neg F \lor \neg G \\
\neg(F \lor G) & \to & \neg F \land \neg G \\
\neg\forall x : F & \to & \exists x : \neg F \\
\neg\exists x : F & \to & \forall x : \neg F
\end{array}$$

Transformation in CNF (2)

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

$$
\begin{array}{lcll}
\forall x : (F \wedge G) & \rightarrow & (\forall x : F) \wedge G & \text{falls } x \text{ nicht frei in } G \\
\forall x : (F \vee G) & \rightarrow & (\forall x : F) \vee G & \text{falls } x \text{ nicht frei in } G \\
\exists x : (F \wedge G) & \rightarrow & (\exists x : F) \wedge G & \text{falls } x \text{ nicht frei in } G \\
\exists x : (F \vee G) & \rightarrow & (\exists x : F) \vee G & \text{falls } x \text{ nicht frei in } G \\
\forall x : (F \wedge G) & \rightarrow & \forall x : F \wedge \forall x : G & \\
\exists x : (F \vee G) & \rightarrow & \exists x : F \vee \exists x : G &
\end{array}
$$

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 ("Ausmultiplikation"). $F \vee (G \wedge H) \rightarrow (F \vee G) \wedge (F \vee H)$
8. Allquantoren vor die Klauseln schieben und gebundene Variablen umbenennen, danach Allquantoren löschen

Transformation in CNF (2)

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

$$
\begin{array}{ll}
 & (L_{1,1} \vee \ldots \vee L_{1,n_1}) \\
\wedge & (L_{2,1} \vee \ldots \vee L_{2,n_2}) \\
\wedge & \\
\ldots & \\
\wedge & (L_{k,1} \vee \ldots \vee L_{k,n_k})
\end{array}
$$

oder in Mengenschreibweise:

$$
\begin{array}{l}
\{\{L_{1,1}, \ldots, L_{1,n_1}\}, \\
\ \{L_{2,1}, \ldots, L_{2,n_2}\}, \\
\ \ldots \\
\ \{L_{k,1}, \ldots, L_{1,n_k}\}\}
\end{array}
$$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:
   $\neg\exists y : \forall x : P(x) \iff Q(y)$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

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

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:
   $\neg\exists y : \forall x : (\neg P(x) \lor Q(y)) \land (Q(y) \Rightarrow P(x))$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:
   $\neg\exists y : \forall x : (\neg P(x) \lor Q(y)) \land (\neg Q(y) \lor P(x))$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

   $\neg\exists y : \forall x : (\neg P(x) \lor Q(y)) \land (\neg Q(y) \lor P(x))$
2. Negationen nach innen schieben:

   $\neg\exists y : \forall x : (\neg P(x) \lor Q(y)) \land (\neg Q(y) \lor P(x))$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:
   $\neg\exists y : \forall x : (\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x))$
2. Negationen nach innen schieben:
   $\forall y : \neg\forall x : (\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x))$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

   $\neg\exists y : \forall x : (\neg P(x) \lor Q(y)) \land (\neg Q(y) \lor P(x))$
2. Negationen nach innen schieben:

   $\forall y : \exists x : \neg((\neg P(x) \lor Q(y)) \land (\neg Q(y) \lor P(x)))$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

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

2. Negationen nach innen schieben:

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

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

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

2. Negationen nach innen schieben:

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

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

   $\neg\exists y : \forall x : (\neg P(x) \lor Q(y)) \land (\neg Q(y) \lor P(x))$
2. Negationen nach innen schieben:

   $\forall y : \exists x : (P(x) \land \neg Q(y)) \lor (Q(y) \land \neg P(x))$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

   $\neg\exists y : \forall x : (\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x))$
2. Negationen nach innen schieben:

   $\forall y : \exists x : (P(x) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(x))$
3. Skopus von Quantoren minimieren:

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

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:
   $\neg\exists y : \forall x : (\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x))$
2. Negationen nach innen schieben:
   $\forall y : \exists x : (P(x) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(x))$
3. Skopus von Quantoren minimieren:
   $\forall y : (\exists x : (P(x) \wedge \neg Q(y))) \vee (\exists x : (Q(y) \wedge \neg P(x)))$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:
   $\neg\exists y : \forall x : (\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x))$
2. Negationen nach innen schieben:
   $\forall y : \exists x : (P(x) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(x))$
3. Skopus von Quantoren minimieren:
   $\forall y : ((\exists x : P(x)) \wedge \neg Q(y)) \vee (\exists x : (Q(y) \wedge \neg P(x)))$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

   $\neg\exists y : \forall x : (\neg P(x) \lor Q(y)) \land (\neg Q(y) \lor P(x))$
2. Negationen nach innen schieben:

   $\forall y : \exists x : (P(x) \land \neg Q(y)) \lor (Q(y) \land \neg P(x))$
3. Skopus von Quantoren minimieren:

   $\forall y : ((\exists x : P(x)) \land \neg Q(y)) \lor (Q(y) \land \exists x : (\neg P(x)))$

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

   $\neg\exists y : \forall x : (\neg P(x) \vee Q(y)) \wedge (\neg Q(y) \vee P(x))$
2. Negationen nach innen schieben:

   $\forall y : \exists x : (P(x) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(x))$
3. Skopus von Quantoren minimieren:

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

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

Beispiel

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

1. Entfernen von $\Rightarrow$ und $\iff$:

   $\neg\exists y : \forall x : (\neg P(x) \lor Q(y)) \land (\neg Q(y) \lor P(x))$
2. Negationen nach innen schieben:

   $\forall y : \exists x : (P(x) \land \neg Q(y)) \lor (Q(y) \land \neg P(x))$
3. Skopus von Quantoren minimieren:

   $\forall y : ((\exists x : P(x)) \land \neg Q(y)) \lor (Q(y) \land \exists x : (\neg P(x)))$
4. Umbenennen:

   $\forall y : ((\exists x_1 : P(x_2)) \land \neg Q(y)) \lor (Q(y) \land \exists x_2 : (\neg P(x_2)))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:
   $\forall y : ((\exists x_1 : P(x_1)) \wedge \neg Q(y)) \vee (Q(y) \wedge \exists x_2 : (\neg P(x_2)))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:
   $\forall y : (P(f_1(y)) \land \neg Q(y)) \lor (Q(y) \land \exists x_2 : (\neg P(x_2)))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:
   $\forall y : (P(f_1(y)) \land \neg Q(y)) \lor (Q(y) \land \neg P(f_2(y)))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:
   $\forall y : (P(f_1(y)) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(f_2(y)))$
6. Allquantoren nach außen schieben
   $\forall y : (P(f_1(y)) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(f_2(y)))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:
   $\forall y : (P(f_1(y)) \land \neg Q(y)) \lor (Q(y) \land \neg P(f_2(y)))$
6. Allquantoren nach außen schieben
   $\forall y : (P(f_1(y)) \land \neg Q(y)) \lor (Q(y) \land \neg P(f_2(y)))$
7. Ausmultiplizieren
   $\forall y : (P(f_1(y)) \land \neg Q(y)) \lor (Q(y) \land \neg P(f_2(y)))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:
   $\forall y : (P(f_1(y)) \land \neg Q(y)) \lor (Q(y) \land \neg P(f_2(y)))$
6. Allquantoren nach außen schieben
   $\forall y : (P(f_1(y)) \land \neg Q(y)) \lor (Q(y) \land \neg P(f_2(y)))$
7. Ausmultiplizieren
   $\forall y.((P(f_1(y)) \lor Q(y)) \land (\neg Q(y) \lor Q(y)) \land (P(f_1(y)) \lor P(f_2(y))) \land (\neg Q(y) \lor P(f_2(y))))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:

   $\forall y : (P(f_1(y)) \land \neg Q(y)) \lor (Q(y) \land \neg P(f_2(y)))$

6. Allquantoren nach außen schieben

   $\forall y : (P(f_1(y)) \land \neg Q(y)) \lor (Q(y) \land \neg P(f_2(y)))$

7. Ausmultiplizieren

   $\forall y.((P(f_1(y)) \lor Q(y)) \land (\neg Q(y) \lor Q(y)) \land (P(f_1(y)) \lor P(f_2(y))) \land (\neg Q(y) \lor P(f_2(y))))$

8. Quantoren vor die Klauseln schieben, und umbenennen, und Quantoren löschen

   $\forall y.((P(f_1(y)) \lor Q(y)) \land (\neg Q(y) \lor Q(y)) \land (P(f_1(y)) \lor P(f_2(y))) \land (\neg Q(y) \lor P(f_2(y))))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:
   $\forall y : (P(f_1(y)) \wedge \neg Q(y)) \vee (Q(y) \wedge \neg P(f_2(y)))$
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
   $\forall y.((P(f_1(y)) \vee Q(y)) \wedge (\neg Q(y) \vee Q(y)) \wedge (P(f_1(y)) \vee P(f_2(y))) \wedge (\neg Q(y) \vee P(f_2(y))))$
8. Quantoren vor die Klauseln schieben, und umbenennen, und Quantoren löschen
   $\forall y.(P(f_1(y)) \vee Q(y)) \wedge \forall y.(\neg Q(y) \vee Q(y)) \wedge \forall y.(P(f_1(y)) \vee P(f_2(y))) \wedge \forall y.(\neg Q(y) \vee P(f_2(y)))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:

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

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

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

8. Quantoren vor die Klauseln schieben, und umbenennen, und Quantoren löschen

   $\forall y_1.(P(f_1(y_1)) \vee Q(y_1)) \wedge \forall y_2.(\neg Q(y_2) \vee Q(y_2)) \wedge \forall y_3.(P(f_1(y_3)) \vee P(f_2(y_3))) \wedge \forall y_4.(\neg Q(y_4) \vee P(f_2(y_4)))$

Beispiel (Forts.)

5. Entfernen der Existenzquantoren mittels Skolemisierung:

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

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

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

8. Quantoren vor die Klauseln schieben, und umbenennen, und Quantoren löschen

   $(P(f_1(y_1)) \vee Q(y_1)) \wedge (\neg Q(y_2) \vee Q(y_2)) \wedge (P(f_1(y_3)) \vee P(f_2(y_3))) \wedge (\neg Q(y_4) \vee P(f_2(y_4)))$

Resolution

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

Grundresolution

- Grundklauseln $\{L_1, \ldots, L_m\}$, 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:

$$\begin{array}{ll}
\textbf{Elternklausel 1:} & L, K_1, \ldots, K_m \\
\textbf{Elternklausel 2:} & \neg L, N_1, \ldots, N_n \\
\hline
\textbf{Resolvente:} & K_1, \ldots, K_m, N_1, \ldots, N_n
\end{array}$$

Beispiel

A1: $\text{Dieb(anton)} \lor \text{Dieb(ede)} \lor \text{Dieb(karl)}$
A2: $\text{Dieb(anton)} \Rightarrow (\text{Dieb(ede)} \lor \text{Dieb(karl)})$
A3: $\text{Dieb(karl)} \Rightarrow (\text{Dieb(ede)} \lor \text{Dieb(anton)})$
A4: $\text{Dieb(ede)} \Rightarrow (\neg\,\text{Dieb(anton)} \land \neg\,\text{Dieb(karl)})$
A5: $\neg\,\text{Dieb(anton)} \lor \neg\,\text{Dieb(karl)}$

Klauselform:

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

Beispiel: Resolution dazu

$\{\neg \text{Dieb(anton)}, \text{Dieb(ede)}, \text{Dieb(karl)}\}$

$\{\neg \text{Dieb(ede)}, \neg \text{Dieb(anton)}\}$

$\{\neg \text{Dieb(anton)}, \neg \text{Dieb(karl)}\}$

$\{\neg \text{Dieb(karl)}, \text{Dieb(ede)}, \text{Dieb(anton)}\}$

$\{\neg \text{Dieb(ede)}, \neg \text{Dieb(karl)}\}$

$\{\text{Dieb(anton)}, \text{Dieb(ede)}, \text{Dieb(karl)}$

Beispiel: Resolution dazu

{¬Dieb(anton),Dieb(ede),Dieb(karl)}

{¬Dieb(ede),¬Dieb(anton)}

{¬Dieb(anton),¬Dieb(karl)}

{¬Dieb(karl),Dieb(ede),Dieb(anton)}

{¬Dieb(ede),¬Dieb(karl)}

{Dieb(karl),¬Dieb(anton)}

{Dieb(anton),Dieb(ede),Dieb(karl)

Beispiel: Resolution dazu

{¬Dieb(anton),Dieb(ede),Dieb(karl)}

{¬Dieb(ede),¬Dieb(anton)}

{¬Dieb(anton),¬Dieb(karl)}

{¬Dieb(karl),Dieb(ede),Dieb(anton)}

{¬Dieb(ede),¬Dieb(karl)}

{Dieb(karl),¬Dieb(anton)}

{Dieb(anton),Dieb(ede),Dieb(karl)

{¬Dieb(anton)}

Beispiel: Resolution dazu

{¬Dieb(anton),Dieb(ede),Dieb(karl)}

{¬Dieb(ede),¬Dieb(anton)}

{¬Dieb(anton),¬Dieb(karl)}

{¬Dieb(karl),Dieb(ede),Dieb(anton)}

{¬Dieb(ede),¬Dieb(karl)}

{Dieb(anton),Dieb(ede),Dieb(karl)

{Dieb(karl),¬Dieb(anton)}

{¬Dieb(anton)}

{¬Dieb(karl),Dieb(ede)}

Beispiel: Resolution dazu

$\{\neg\text{Dieb(anton)},\text{Dieb(ede)},\text{Dieb(karl)}\}$

$\{\neg\text{Dieb(ede)},\neg\text{Dieb(anton)}\}$

$\{\neg\text{Dieb(anton)},\neg\text{Dieb(karl)}\}$

$\{\neg\text{Dieb(karl)},\text{Dieb(ede)},\text{Dieb(anton)}\}$

$\{\neg\text{Dieb(ede)},\neg\text{Dieb(karl)}\}$

$\{\text{Dieb(anton)},\text{Dieb(ede)},\text{Dieb(karl)}$

$\{\text{Dieb(karl)},\neg\text{Dieb(anton)}\}$

$\{\neg\text{Dieb(anton)}\}$

$\{\neg\text{Dieb(karl)},\text{Dieb(ede)}\}$

$\{\neg\text{Dieb(karl)}\}$

Beispiel: Resolution dazu

{¬Dieb(anton),Dieb(ede),Dieb(karl)}

{¬Dieb(ede),¬Dieb(anton)}

{¬Dieb(anton),¬Dieb(karl)}

{¬Dieb(karl),Dieb(ede),Dieb(anton)}

{¬Dieb(ede),¬Dieb(karl)}

{Dieb(karl),¬Dieb(anton)}

{Dieb(anton),Dieb(ede),Dieb(karl)

{¬Dieb(anton)}

{¬Dieb(karl),Dieb(ede)}

{¬Dieb(karl)}

{Dieb(anton),Dieb(ede)}

Beispiel: Resolution dazu

{¬Dieb(anton),Dieb(ede),Dieb(karl)}

{¬Dieb(ede),¬Dieb(anton)}

{¬Dieb(anton),¬Dieb(karl)}

{¬Dieb(karl),Dieb(ede),Dieb(anton)}

{¬Dieb(ede),¬Dieb(karl)}

{Dieb(karl),¬Dieb(anton)}

{Dieb(anton),Dieb(ede),Dieb(karl)

{¬Dieb(anton)}

{¬Dieb(karl),Dieb(ede)}

{¬Dieb(karl)}

{Dieb(anton),Dieb(ede)}

{Dieb(ede)}

Grundresolution

Satz

Die Grundresolution ist korrekt:

$$\begin{array}{lcl} C_1 & := & L, K_1, \ldots, K_m \\ C2 & :=: & \neg L, N_1, \ldots, N_n \\ \hline R & = & K_1, \ldots, K_m, N_1, \ldots, N_n \end{array}$$

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 $I(N_1 \vee \ldots \vee N_n) = 1$. 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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Theorem

Jede endliche unerfüllbare Grundklauselmenge läßt sich durch Resolution widerlegen.

Allgemeine Resolution

- Klauseln mit Variablen in den Termen.
- Idee: Erzeuge Grundklauseln durch **Substitution** und verwende dann Grundresolution
- Nur die Idee, man macht es später anders.

Substitution

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

$$\begin{array}{lcl} \sigma(x) & = & x, \text{ wenn } \sigma\ x \text{ nicht abbildet} \\ \sigma(f(t_1,\dots,t_n)) & = & f(\sigma(t_1),\dots,\sigma(t_n)) \end{array}$$

- Anwendung von Substitutionen auf Literale und Klauseln entsprechend
  $\sigma(\{L_1,\dots,L_n\}) = \{\sigma(L_1),\dots,\sigma(L_n)\}$

Beispiele

$$\sigma = \{x \mapsto a\} \qquad\qquad \begin{array}{l}\sigma(x) = a \\ \sigma(f(x,x)) = f(a,a)\end{array}$$

Beispiele

$$\sigma = \{x \mapsto a\} \qquad \begin{array}{l} \sigma(x) = a \\ \sigma(f(x,x)) = f(a,a) \end{array}$$

$$\sigma = \{x \mapsto g(x)\} \qquad \begin{array}{l} \sigma(x) = g(x) \\ \sigma(f(x,x)) = f(g(x),g(x)) \\ \sigma(\sigma(x)) = g(g(x)) \end{array}$$

Beispiele

$$\sigma = \{x \mapsto a\} \qquad \begin{array}{l}\sigma(x) = a \\ \sigma(f(x,x)) = f(a,a)\end{array}$$

$$\sigma = \{x \mapsto g(x)\} \qquad \begin{array}{l}\sigma(x) = g(x) \\ \sigma(f(x,x)) = f(g(x),g(x)) \\ \sigma(\sigma(x)) = g(g(x))\end{array}$$

$$\sigma = \{x \mapsto y, y \mapsto a\} \qquad \begin{array}{l}\sigma(x) = y \\ \sigma(\sigma(x)) = a \\ \sigma(f(x,y)) = f(y,a)\end{array}$$

Beispiele

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

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

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

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

Komposition

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

Beispiele:

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

Substitutionen ändern die Erfüllbarkeit nicht

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

Dann ist $\{K_1, \ldots, K_n\}$ genau dann erfüllbar, wenn $\{K_1, \ldots, K_n, \sigma(K_i)\}$ erfüllbar ist.

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

Resolution mit Unifikation

$$\begin{array}{ll} \text{Elternklausel 1:} & L, K_1, \dots, K_m \\ \text{Elternklausel 2:} & \neg L', N_1, \dots, N_n \\ \hline \text{Resolvente:} & \sigma(K_1, \dots, K_m, N_1, \dots, N_n) \end{array}$$

σ ist eine Substitution mit $\sigma(L) = \sigma(L')$

Da σ L und L' gleich macht, nenn man σ auch **Unifikator**

Eigenschaften:

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

Beispiel: Resolution reicht nicht aus

Der Friseur rasiert alle, die sich nicht selbst rasieren

Beispiel: Resolution reicht nicht aus

Der Friseur rasiert alle, die sich nicht selbst rasieren

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

Beispiel: Resolution reicht nicht aus

Der Friseur rasiert alle, die sich nicht selbst rasieren

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

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

Beispiel: Resolution reicht nicht aus

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:

$$\frac{\begin{array}{l}\{\text{Rasiert}(x, x), \text{Rasiert}(\text{friseur}, x)\} \\ \{\neg\text{Rasiert}(\text{friseur}, y), \neg\text{Rasiert}(y, y)\}\end{array}}{\{\text{Rasiert}(\text{friseur}, \text{friseur}), \neg\text{Rasiert}(\text{friseur}, \text{friseur})\}} \qquad \sigma = \{x \mapsto \text{friseur}, y \mapsto \text{friseur}\}$$

Beispiel: Resolution reicht nicht aus

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:

$$\begin{array}{l} \{\text{Rasiert}(x,x), \text{Rasiert}(\text{friseur}, x)\} \qquad \sigma = \{x \mapsto \text{friseur}, y \mapsto \text{friseur}\} \\ \{\neg\text{Rasiert}(\text{friseur}, y), \neg\text{Rasiert}(y,y)\} \\ \hline \{\text{Rasiert}(\text{friseur}, \text{friseur}), \neg\text{Rasiert}(\text{friseur}, \text{friseur})\} \end{array}$$

Jetzt erhält man keine weiteren Klauseln mehr!
aber: Die Formel ist widersprüchlich: Wer rasiert den Friseur?

Faktorisierung

Das Friseur-Beispiel zeigt:

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

Faktorisierung:

$$\begin{array}{ll} \text{Elternklausel:} & L, L', K_1, \ldots, K_m \\ \text{Faktor:} & \overline{\sigma(L, K_1, \ldots, K_m)} \end{array} \quad \sigma(L) = \sigma(L')$$

Friseur

$$
\begin{array}{l}
C1 : \{\text{Rasiert}(x,x), \text{Rasiert}(\text{friseur},x)\} \\
C2 : \{\neg\text{Rasiert}(\text{friseur},y), \neg\text{Rasiert}(y,y)\} \\
\hline
\text{Faktor von } C1 : F1 : \{\text{Rasiert}(\text{friseur},\text{friseur})\} \\
\text{Faktor von } C2 : F2 : \neg\{\text{Rasiert}(\text{friseur},\text{friseur})\} \\
\hline
\text{Resolvente von } F1 + F2 : \square
\end{array}
$$

Prädikatenlogischer Resolutionskalkül

Eingabe: Klauselmenge S

Regeln:

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

Abbruchbedingung: $\Box \in S$, dann widersprüchlich.

Beispiel

Transitivität der Teilmengenrelation: Prädikatsymbole $\subseteq$ und $\in$

- 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. $F_1 \implies F_2$ ist Tautologie.
- Daher zeigen wir $\neg(F_1 \implies F_2)$ ist widersprüchlich.
  (wir können daher mit $F_1 \wedge \neg F_2$ starten).

Umwandlung in Klauselform ergibt:

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: $\{\neg a \subseteq c\}$

$$\{x \subseteq y, \neg f(x,y) \in y\}$$

$$\{a \subseteq b\}$$

$$\{\neg x \subseteq y, \neg w \in x, w \in y\}$$

$$\{b \subseteq c\}$$

$$\{x \subseteq y, f(x,y) \in x\}$$

$$\{\neg a \subseteq c\}$$

$$\{x \subseteq y, \neg f(x,y) \in y\}$$

$$\{a \subseteq b\}$$

$$\{\neg w \in a, w \in b\}$$

$$\boxed{\sigma_1}$$

$$\boxed{\sigma_1}$$

$$\{\neg x \subseteq y, \neg w \in x, w \in y\}$$

$$\{b \subseteq c\}$$

$$\{x \subseteq y, f(x,y) \in x\}$$

$$\{\neg a \subseteq c\}$$

$$\sigma_1 \quad = \quad \{x \mapsto a, y \mapsto b\}$$

$\{x \subseteq y, \neg f(x,y) \in y\}$

$\{a \subseteq b\} \xrightarrow{\boxed{\sigma_1}} \{\neg w \in a, w \in b\}$

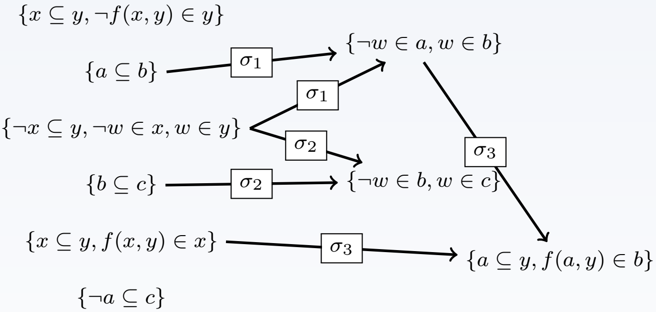
$\{\neg x \subseteq y, \neg w \in x, w \in y\}$

$\{b \subseteq c\} \xrightarrow{\boxed{\sigma_2}} \{\neg w \in b, w \in c\}$

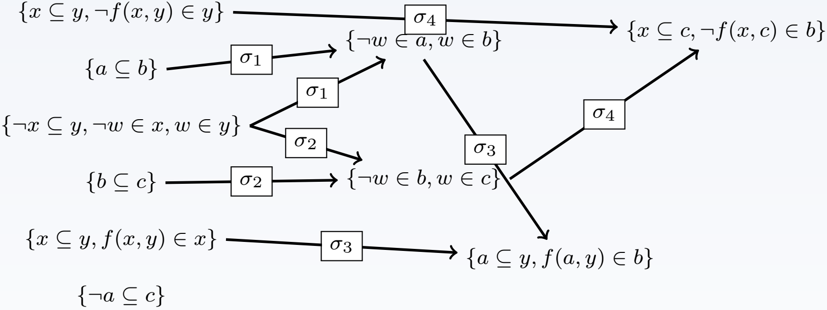
$\{x \subseteq y, f(x,y) \in x\}$

$\{\neg a \subseteq c\}$

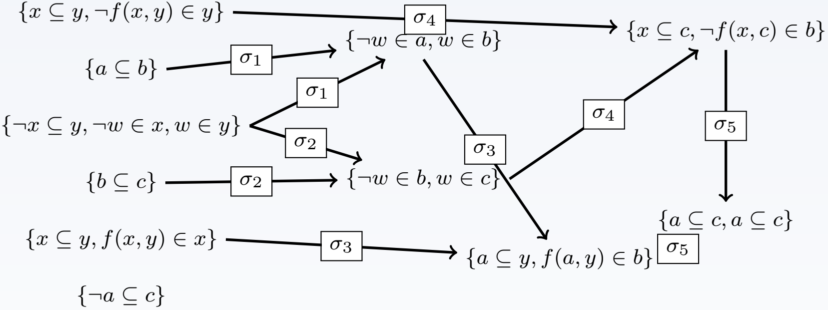
$$\begin{array}{lcl} \sigma_1 & = & \{x \mapsto a, y \mapsto b\} \\ \sigma_2 & = & \{x \mapsto b, y \mapsto c\} \end{array}$$



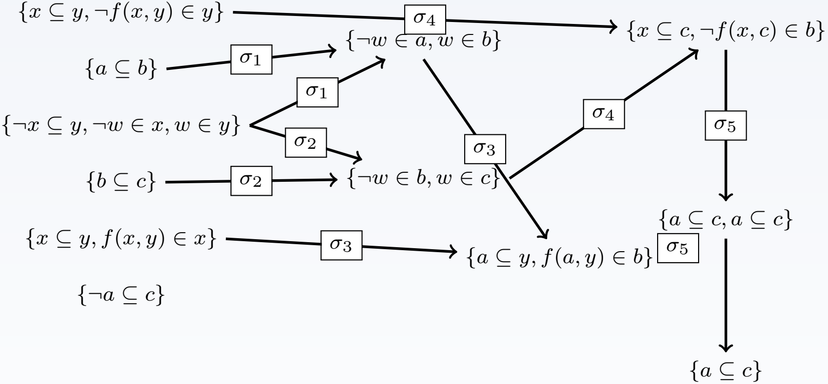
$$
\begin{aligned}
\sigma_1 &= \{x \mapsto a, y \mapsto b\} \\
\sigma_2 &= \{x \mapsto b, y \mapsto c\} \\
\sigma_3 &= \{x \mapsto a, w \mapsto f(a,y)\}
\end{aligned}
$$



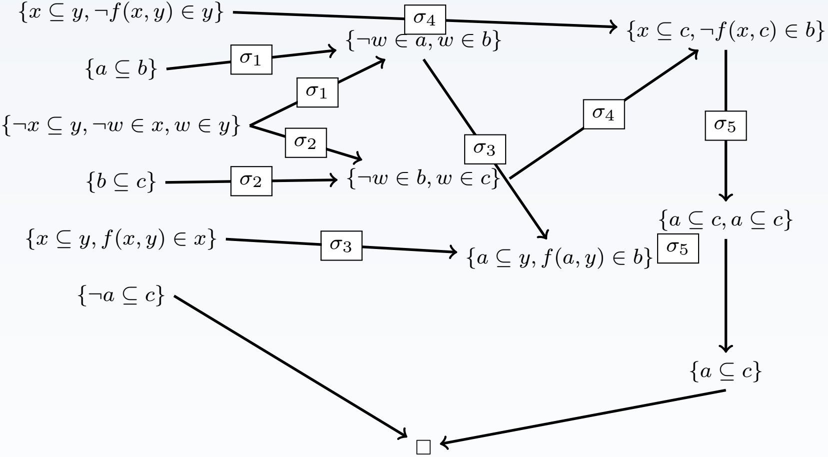
$$
\begin{array}{lcl}
\sigma_1 & = & \{x \mapsto a, y \mapsto b\} \\
\sigma_2 & = & \{x \mapsto b, y \mapsto c\} \\
\sigma_3 & = & \{x \mapsto a, w \mapsto f(a,y)\} \\
\sigma_4 & = & \{y \mapsto c, w \mapsto f(x,c)\}
\end{array}
$$



$$
\begin{array}{lcl}
\sigma_1 & = & \{x \mapsto a, y \mapsto b\} \\
\sigma_2 & = & \{x \mapsto b, y \mapsto c\} \\
\sigma_3 & = & \{x \mapsto a, w \mapsto f(a, y)\} \\
\sigma_4 & = & \{y \mapsto c, w \mapsto f(x, c)\} \\
\sigma_5 & = & \{x \mapsto a, y \mapsto c\}
\end{array}
$$



$$
\begin{aligned}
\sigma_1 &= \{x \mapsto a, y \mapsto b\} \\
\sigma_2 &= \{x \mapsto b, y \mapsto c\} \\
\sigma_3 &= \{x \mapsto a, w \mapsto f(a,y)\} \\
\sigma_4 &= \{y \mapsto c, w \mapsto f(x,c)\} \\
\sigma_5 &= \{x \mapsto a, y \mapsto c\}
\end{aligned}
$$



$$
\begin{aligned}
\sigma_1 &= \{x \mapsto a, y \mapsto b\}\\
\sigma_2 &= \{x \mapsto b, y \mapsto c\}\\
\sigma_3 &= \{x \mapsto a, w \mapsto f(a,y)\}\\
\sigma_4 &= \{y \mapsto c, w \mapsto f(x,c)\}\\
\sigma_5 &= \{x \mapsto a, y \mapsto c\}
\end{aligned}
$$

Unifikation

- Für die Resolution und Faktorisierung braucht man einen Unifikator
- Diesen kann man algorithmisch finden!
- Noch mehr: Man kann einen **allgemeinsten Unifikator** bestimmen
- Vorteil: Man braucht die spezielleren nicht zu betrachten

Allgemeinster Unifikator

(MGU = Most general unifier)

- Seien s, t Terme
- σ ist Unfikator 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 Unfikator ρ von s, t gibt es eine Substitution γ mit $\gamma\sigma = \rho$.

Beispiel

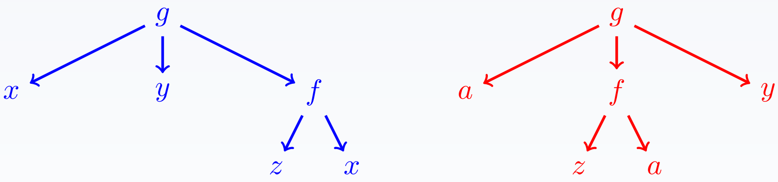
$$\frac{\begin{array}{l} P(x), Q(x) \\ \neg P(y), R(y) \end{array}}{Q(a), R(a)} \qquad \begin{array}{l} \sigma = \{x \mapsto a, y \mapsto a\} \\ \sigma \text{ ist ein Unifikator} \end{array}$$

$$\frac{\begin{array}{l} P(x), Q(x) \\ \neg P(y), R(y) \end{array}}{Q(y), R(y)} \qquad \begin{array}{l} \sigma = \{x \mapsto y\} \\ \sigma \text{ ist ein allgemeinster Unifikator} \end{array}$$

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

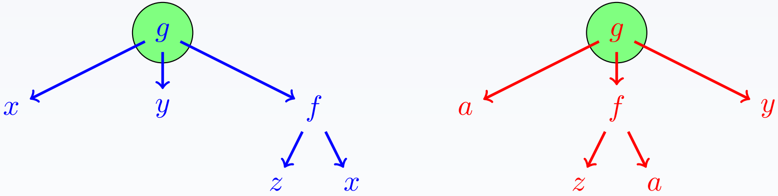
Beispiel zur Unifikation

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



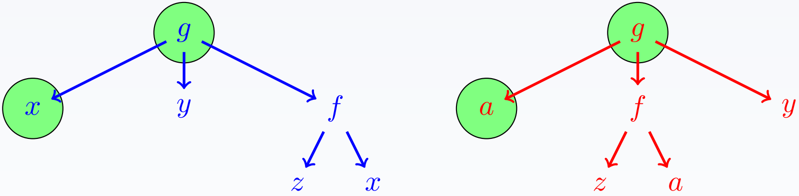
Beispiel zur Unifikation

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



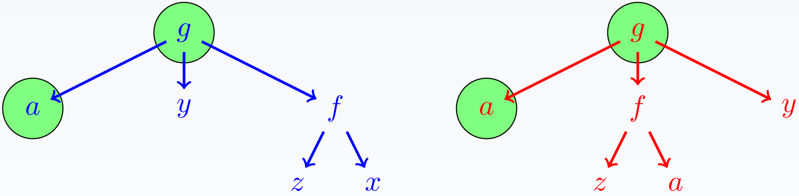
Beispiel zur Unifikation

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



Beispiel zur Unifikation

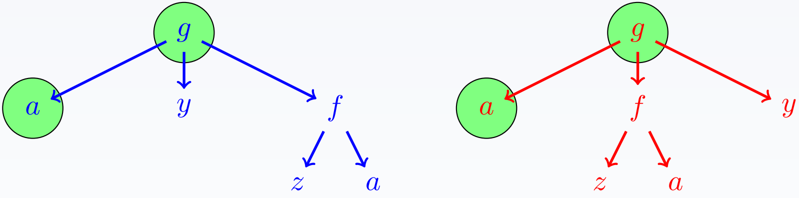
$$g(x, y, f(z, x)) \stackrel{?}{=} g(a, f(z, a), y)$$



- $x \mapsto a$

Beispiel zur Unifikation

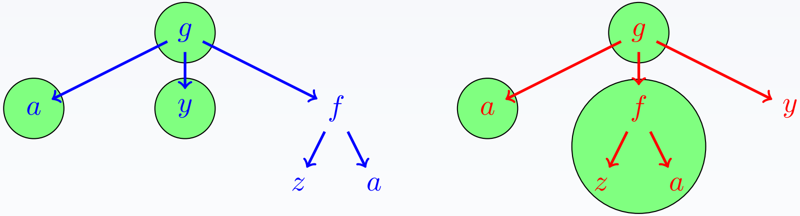
$$g(x, y, f(z, x)) \stackrel{?}{=} g(a, f(z, a), y)$$



- $x \mapsto a$

Beispiel zur Unifikation

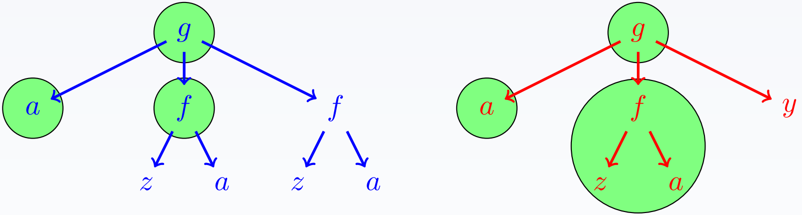
$$g(x, y, f(z, x)) \stackrel{?}{=} g(a, f(z, a), y)$$



- $x \mapsto a$

Beispiel zur Unifikation

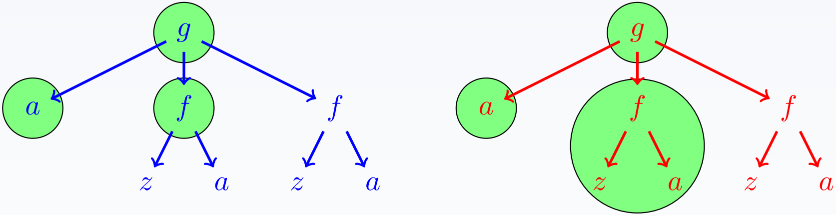
$$g(x, y, f(z, x)) \stackrel{?}{=} g(a, f(z, a), y)$$



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

Beispiel zur Unifikation

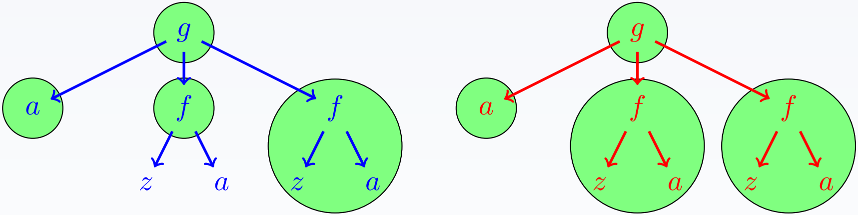
$$g(x, y, f(z, x)) \stackrel{?}{=} g(a, f(z, a), y)$$



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

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

   Dann: Gebe $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ als MGU aus.
2. Sonst: Wende eine der Unfikationsregeln 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

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

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

$$\frac{x \stackrel{?}{=} t, \Gamma}{x \stackrel{?}{=} t, \{x \mapsto t\}\Gamma} \quad x \in FV(\Gamma),\ x \not\in FV(t)$$ (Anwendung)

$$\frac{t \stackrel{?}{=} x, \Gamma}{x \stackrel{?}{=} t, \Gamma} \quad t \not\in V$$ (Orientierung)

Unifikationsregeln (2)

Abbruchbedingungen:

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

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

Beispiel

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

Beispiel

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

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

$$\to \{f(x,g(a,y)) \stackrel{?}{=} f(h(y),g(y,a)), g(x,h(y)) \stackrel{?}{=} g(z,z)\} \quad \text{(Dekomposition)}$$

Beispiel

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

$$\to \{f(x,g(a,y)) \stackrel{?}{=} f(h(y),g(y,a)), g(x,h(y)) \stackrel{?}{=} g(z,z)\} \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), g(a,y) \stackrel{?}{=} g(y,a), g(x,h(y)) = g(z,z) \qquad \text{(Dekomposition)}$$

Beispiel

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

$$\to \{f(x,g(a,y)) \stackrel{?}{=} f(h(y),g(y,a)), g(x,h(y)) \stackrel{?}{=} g(z,z)\} \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), g(a,y) \stackrel{?}{=} g(y,a), g(x,h(y)) = g(z,z) \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), a \stackrel{?}{=} y, y \stackrel{?}{=} a, g(x,h(y)) \stackrel{?}{=} g(z,z) \qquad \text{(Dekomposition)}$$

Beispiel

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

Beispiel

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

$$\to \{f(x,g(a,y)) \stackrel{?}{=} f(h(y),g(y,a)), g(x,h(y)) \stackrel{?}{=} g(z,z)\} \quad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), g(a,y) \stackrel{?}{=} g(y,a), g(x,h(y)) = g(z,z) \quad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), a \stackrel{?}{=} y, y \stackrel{?}{=} a, g(x,h(y)) \stackrel{?}{=} g(z,z) \quad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), y \stackrel{?}{=} a, g(x,h(y)) \stackrel{?}{=} g(z,z) \quad \text{(Orientierung)}$$

$$\to x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, g(x,h(a)) \stackrel{?}{=} g(z,z) \quad \text{(Anwendung, y)}$$

Beispiel

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

$$\to \{f(x,g(a,y)) \stackrel{?}{=} f(h(y),g(y,a)), g(x,h(y)) \stackrel{?}{=} g(z,z)\} \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), g(a,y) \stackrel{?}{=} g(y,a), g(x,h(y)) = g(z,z) \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), a \stackrel{?}{=} y, y \stackrel{?}{=} a, g(x,h(y)) \stackrel{?}{=} g(z,z) \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{=} h(y), y \stackrel{?}{=} a, g(x,h(y)) \stackrel{?}{=} g(z,z) \qquad \text{(Orientierung)}$$

$$\to x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, g(x,h(a)) \stackrel{?}{=} g(z,z) \qquad \text{(Anwendung, y)}$$

$$\to x \stackrel{?}{=} h(a), y \stackrel{?}{=} a, x \stackrel{?}{=} z, h(a) \stackrel{?}{=} z \qquad \text{(Dekomposition)}$$

Beispiel

$$\{k(f(x, g(a, y)), g(x, h(y)) \overset{?}{=} k(f(h(y), g(y, a)), g(z, z))\}$$

$$\to \{f(x, g(a, y)) \overset{?}{=} f(h(y), g(y, a)), g(x, h(y)) \overset{?}{=} g(z, z)\} \qquad \text{(Dekomposition)}$$

$$\to x \overset{?}{=} h(y), g(a, y) \overset{?}{=} g(y, a), g(x, h(y)) = g(z, z) \qquad \text{(Dekomposition)}$$

$$\to x \overset{?}{=} h(y), a \overset{?}{=} y, y \overset{?}{=} a, g(x, h(y)) \overset{?}{=} g(z, z) \qquad \text{(Dekomposition)}$$

$$\to x \overset{?}{=} h(y), y \overset{?}{=} a, g(x, h(y)) \overset{?}{=} g(z, z) \qquad \text{(Orientierung)}$$

$$\to x \overset{?}{=} h(a), y \overset{?}{=} a, g(x, h(a)) \overset{?}{=} g(z, z) \qquad \text{(Anwendung, y)}$$

$$\to x \overset{?}{=} h(a), y \overset{?}{=} a, x \overset{?}{=} z, h(a) \overset{?}{=} z \qquad \text{(Dekomposition)}$$

$$\to x \overset{?}{=} h(a), y \overset{?}{=} a, x \overset{?}{=} z, z \overset{?}{=} h(a) \qquad \text{(Orientierung)}$$

Beispiel

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

$$\{k(f(x,g(a,y)),g(x,h(y)) \stackrel{?}{\doteq} k(f(h(y),g(y,a)),g(z,z))\}$$

$$\to \{f(x,g(a,y)) \stackrel{?}{\doteq} f(h(y),g(y,a)), g(x,h(y)) \stackrel{?}{\doteq} g(z,z)\}$$ (Dekomposition)

$$\to x \stackrel{?}{\doteq} h(y), g(a,y) \stackrel{?}{\doteq} g(y,a), g(x,h(y)) = g(z,z)$$ (Dekomposition)

$$\to x \stackrel{?}{\doteq} h(y), a \stackrel{?}{\doteq} y, y \stackrel{?}{\doteq} a, g(x,h(y)) \stackrel{?}{\doteq} g(z,z)$$ (Dekomposition)

$$\to x \stackrel{?}{\doteq} h(y), y \stackrel{?}{\doteq} a, g(x,h(y)) \stackrel{?}{\doteq} g(z,z)$$ (Orientierung)

$$\to x \stackrel{?}{\doteq} h(a), y \stackrel{?}{\doteq} a, g(x,h(a)) \stackrel{?}{\doteq} g(z,z)$$ (Anwendung, y)

$$\to x \stackrel{?}{\doteq} h(a), y \stackrel{?}{\doteq} a, x \stackrel{?}{\doteq} z, h(a) \stackrel{?}{\doteq} z$$ (Dekomposition)

$$\to x \stackrel{?}{\doteq} h(a), y \stackrel{?}{\doteq} a, x \stackrel{?}{\doteq} z, z \stackrel{?}{\doteq} h(a)$$ (Orientierung)

$$\to x \stackrel{?}{\doteq} h(a), y \stackrel{?}{\doteq} a, z \stackrel{?}{\doteq} h(a)$$ (Anwendung,z)

Beispiel

$$\{k(f(x,g(a,y)),g(x,h(y)) \stackrel{?}{\doteq} k(f(h(y),g(y,a)),g(z,z))\}$$

$$\to \{f(x,g(a,y)) \stackrel{?}{\doteq} f(h(y),g(y,a)), g(x,h(y)) \stackrel{?}{\doteq} g(z,z)\} \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{\doteq} h(y), g(a,y) \stackrel{?}{\doteq} g(y,a), g(x,h(y)) = g(z,z) \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{\doteq} h(y), a \stackrel{?}{\doteq} y, y \stackrel{?}{\doteq} a, g(x,h(y)) \stackrel{?}{\doteq} g(z,z) \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{\doteq} h(y), y \stackrel{?}{\doteq} a, g(x,h(y)) \stackrel{?}{\doteq} g(z,z) \qquad \text{(Orientierung)}$$

$$\to x \stackrel{?}{\doteq} h(a), y \stackrel{?}{\doteq} a, g(x,h(a)) \stackrel{?}{\doteq} g(z,z) \qquad \text{(Anwendung, y)}$$

$$\to x \stackrel{?}{\doteq} h(a), y \stackrel{?}{\doteq} a, x \stackrel{?}{\doteq} z, h(a) \stackrel{?}{\doteq} z \qquad \text{(Dekomposition)}$$

$$\to x \stackrel{?}{\doteq} h(a), y \stackrel{?}{\doteq} a, x \stackrel{?}{\doteq} z, z \stackrel{?}{\doteq} h(a) \qquad \text{(Orientierung)}$$

$$\to x \stackrel{?}{\doteq} h(a), y \stackrel{?}{\doteq} a, z \stackrel{?}{\doteq} h(a) \qquad \text{(Anwendung,}z\text{)}$$

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, falls er nicht abbricht, genau einen allgemeinsten Unifikator.

- In dieser Form: MGU kann exponentiell groß werden:
- $\{x_1 \stackrel{?}{=} f(x_2, x_2), x_2 \stackrel{?}{=} f(x_3, x_3), \ldots, x_{n-1} \stackrel{?}{=} f(x_n, x_n), x_n \stackrel{?}{=} a\}$
- MGU: $\{x_1 \mapsto f(f(f(f \ldots f(f(a,a), f(a,a)) \ldots))), \ldots\}$
- Aber es gibt polynomielle Varianten (mit Sharing)
- Komplexität allgemein: $\mathcal{P}$-complete

Zurück zum Resolutionskalkül

Resolution:

$$\begin{array}{ll} \text{Elternklausel 1:} & L, K_1, \ldots, K_m \\ \text{Elternklausel 2:} & \neg L', N_1, \ldots, N_n \\ \hline \text{Resolvente:} & \sigma(K_1, \ldots, K_m, N_1, \ldots, N_n) \end{array} \quad \begin{array}{l} \sigma \text{ ist eine Substitution} \\ \text{mit } \sigma(L) = \sigma(L') \end{array}$$

Faktorisierung:

$$\begin{array}{ll} \text{Elternklausel:} & L, L', K_1, \ldots, K_m \\ \hline \text{Faktor:} & \sigma(L, K_1, \ldots, K_m) \end{array} \quad \sigma(L) = \sigma(L')$$

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) ist korrekt und widerlegungsvollständig.

(Beweis erfordert noch ein Lifting-Lemma: Grundresolution $\to$ 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 Klauselmenge, 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: Löschregel für isolierte Literale

ISOL: Löschregel 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 Löschregel 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

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

Beispiel

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

- 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 $\mathcal{C}$.

Beachte: **zusätzliche Bedingung** E hat mind. soviele Literale wie D

Grund: Faktorisierung

$$\underbrace{\{L, L', L_1, \ldots, L_n\}}_{D} \rightarrow \underbrace{\{\sigma(L_1), \ldots, \sigma(L_n)\}}_{E}$$

- 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
- 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 enthält Literale L und $\neg L$

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 99% aller erzeugten Klauseln subsumierte Klauseln sind.

Lineare Resolution

- Eingeschränkte (spezielle Variante) der Resolution
- Eingabe: Klauselmenge mit **Zentralklausel** und **Seitenklauseln**
- Erste Resolution: Zentralklausel + eine Seitenklausel
- Danach: Jede Resolution verwendet die zuletzt erhaltene Resolvente
- D.h. danach wird die Resolvente zur nächsten Zentralklausel

Beispiel: Lineare Resolution

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

$\{P, Q\}$ $\quad$ $\{P, \neg Q\}$ $\quad$ $\{\neg P, \neg Q\}$ $\quad$ $\{\neg P, Q\}$

Beispiel: Lineare Resolution

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

$$\begin{tikzcd}
\{P, Q\} \arrow[d] & \{P, \neg Q\} \arrow[dl] & \{\neg P, \neg Q\} & \{\neg P, Q\} \\
\{P\}
\end{tikzcd}$$

Beispiel: Lineare Resolution

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

$$\begin{tikzcd}
\{P, Q\} \arrow[d] & \{P, \neg Q\} \arrow[dl] & \{\neg P, \neg Q\} \arrow[ddll] & \{\neg P, Q\} \\
\{P\} \arrow[d] & & & \\
\{\neg Q\} & & &
\end{tikzcd}$$

Beispiel: Lineare Resolution

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

$$
\begin{tikzcd}
\{P,Q\} \arrow[d] & \{P,\neg Q\} \arrow[dl] & \{\neg P,\neg Q\} \arrow[ddll] & \{\neg P,Q\} \arrow[dddlll] \\
\{P\} \arrow[d] & & & \\
\{\neg Q\} \arrow[d] & & & \\
\{\neg P\} & & &
\end{tikzcd}
$$

Beispiel: Lineare Resolution

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

$$\begin{tikzcd}
\{P,Q\} \arrow[d] & \{P,\neg Q\} \arrow[dl] & \{\neg P,\neg Q\} \arrow[ddll] & \{\neg P,Q\} \arrow[dddlll] \\
\{P\} \arrow[d] \arrow[dddd, bend right] & & & \\
\{\neg Q\} \arrow[d] & & & \\
\{\neg P\} \arrow[d] & & & \\
\square & & &
\end{tikzcd}$$

Lineare Resolution: Eigenschaften

- Lineare Resolution + Faktorisierung ist widerlegungsvollständig
- Bei Hornklauseln: Faktorisierung nicht notwendig

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.
- Definitite Einsklauseln (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 zu erst 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 immer: 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 Hornklauselmenge korrekt und widerlegungsvollständig
- Aber so ist es noch **nichtdeterministisch**: Wahl der Seitenklausel als Elternklausel
- Dieser Nichtdeterminismus ist nicht: don't-care!
- Mögliche Abhilfe: Breitensuche (erhält Vollständigkeit)
- Aber: Sehr Platzintensiv

Wir betrachten SLD-Resolution im Kapitel „Prolog“ nochmal!