Resolution und Prolog

Prof. Dr. Rudolf Kruse

University of Magdeburg
Faculty of Computer Science
Magdeburg, Germany
rudolf.kruse@cs.uni-magdeburg.de

NEURO FUZZY

Resolventenprinzip und Prolog

- Betrachtung formaler Ansätze zur Wissensmodellierung und Problemlösung
- Einsatzgebiete z.B. automatische Beweiser, Beschreibungen in der Robotik
- Im Folgenden u.a.:
  - Aussagenkalkül
    - Resolutionsregel / Resolventenverfahren
    - Horn-Klauseln
  - Prädikatenkalkül
    - Unifikation
    - Resolution

Aussagenkalkül (propositional calculus)

Begriffsbildung

- Atome: **Bat_OK, Liftable, Moves, ...**
- Literale **sind entweder Atome (positive Literale) oder negierte Atome (negative Literale): Bat_OK, ¬Bat_OK, Liftable, ¬Liftable, Moves, ¬Moves, ...**
- Klauseln **sind Disjunktionen von Literalen (als Abkürzung von** $L_1 \vee L_2 \vee ... \vee L_n$ **wird** $\{L_1,...,L_n\}$ **verwendet):**
  - **(Bat_OK ∧ Liftable) ⇒ Moves**
  - **äquivalent zu: ¬Bat_OK ∨ ¬Liftable ∨ Moves**
  - **äquivalent zu: {¬Bat_OK, ¬Liftable, Moves}**
- **NIL äq { } äq F (Wert immer falsch)**

Satz (Resolutionsregel)

- **C_1 und C_2 seien Klauseln und λ ein Atom. Dann kann man aus $\{\lambda\} \cup C_1$ und $\{\neg \lambda\} \cup C_2$ schließen, dass $C_1 \cup C_2$ gilt. $C_1 \cup C_2$ heißt Resolvente.**
- **Beweis:**

λ	C_1	C_2	$\{\lambda\} \cup C_1$	$\{\neg\lambda\} \cup C_2$	$C_1 \cup C_2$	$(\{\lambda\} \cup C_1) \wedge (\{\neg\lambda\} \cup C_2) \Rightarrow (C_1 \cup C_2)$
w	w	w	w	w	w	w
w	w	f	w	f	w	w
w	f	w	w	w	w	w
w	f	f	w	f	f	w
f	w	w	w	w	w	w
f	w	f	w	w	w	w
f	f	w	f	w	w	w
f	f	f	f	w	f	w

Beispiele

a) $\dfrac{R \vee P, \neg P \vee Q}{R \vee Q}$ **steht für** $(R \vee P, \neg P \vee Q) = R \veebar Q$

manchmal $(R \vee P, \neg P \vee Q) \vdash_{res} R \vee Q$

Dieses Prinzip heißt wegen auch Verkettung. $(\neg R \Rightarrow P \wedge P \Rightarrow Q) \Rightarrow \neg R \Rightarrow Q$

b) $\dfrac{R, \neg R \vee P}{P}$ **, dieses Beweisschema ist der Modus Ponens** $(R \wedge (R \Rightarrow P)) \Rightarrow P$

c) $\dfrac{P \vee Q \vee R \vee S, \neg P \vee Q \vee W}{Q \vee R \vee S \vee W}$ **, Q taucht nur einmal auf**

d)

$$\frac{P \vee \overset{\downarrow}{Q} \vee \neg R, P \vee W \vee \neg \overset{\downarrow}{Q} \vee R}{P \vee \neg R \vee R \vee W}, \quad \frac{P \vee Q \vee \neg \overset{\downarrow}{R}, P \vee W \vee \neg Q \vee \overset{\downarrow}{R}}{P \vee Q \vee W \vee \neg Q},$$

$P \vee W$ ist keine Resolvente

Korrektheit und Vollständigkeit

Φ: Formelmenge

φ: einzelne Formel

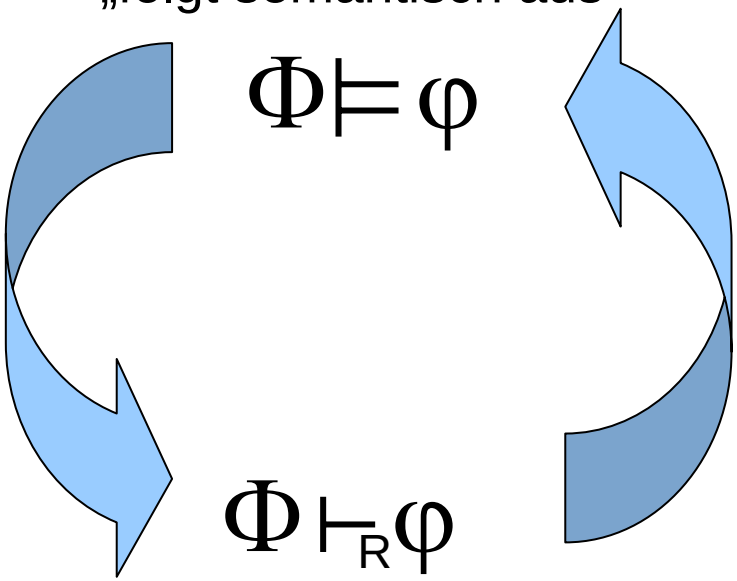
„folgt aus“ oder
„folgt semantisch aus“

$\Phi \models \varphi$

Für alle Belegungen der Variablen für die Φ gilt, gilt auch φ. (Wahrheitstafel!)

Vollständigkeit

(Für Resolution im Folgenden)



Korrektheit

(siehe oben)

$\Phi \vdash_R \varphi$

„ist ableitbar aus“ oder
„folgt syntaktisch aus“
(mit der Regelmenge R)

Syntaktische Manipulation der Formeln mit gegebenen Schlussregeln R

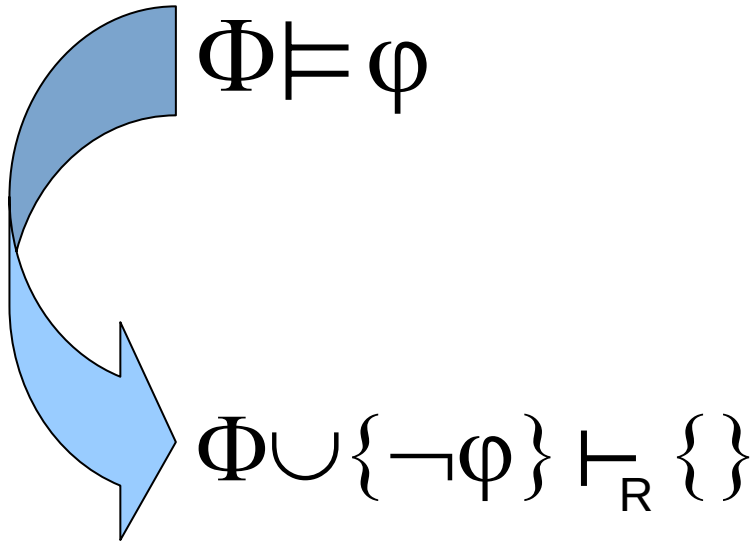
Bemerkung

Resolution ist korrekt, aber Resolution ist nicht **vollständig:**
$P \wedge R \models P \vee R$ **jedoch kann aus** $\{P, R\}$ **nichts mit dem Resolutionsverfahren geschlossen werden.**

Ausweg: **Wir zeigen, dass** $\{P, R\}$ **und** $\neg\{P \vee R\}$ **widersprüchlich sind (das kann man mit dem Resolutions-verfahren leicht zeigen):**
$\neg(P \vee R)$ äq $\neg P \wedge \neg R$ **, also ist die Klauselmenge insgesamt** $\{\neg P, \neg R, P, R\}$ **. Mit** $res(\neg P, P) = \text{NIL} = \text{F} = \{\ \}$ **erhält man jedoch eine leere Klauselmenge und somit F.**
Also muss das Gegenteil $P \wedge R \models P \vee R$ **gelten.**

Widerlegungsvollständigkeit

Widerlegungs-
vollständigkeit



- **Die Resolutionsregel ist nicht vollständig, aber widerlegungsvollständig (Beweis sehr komplex).**
- **Durch die Umformung vergrößert man die Menge der möglichen Schlüsse.**
- **Deshalb werden meist Widerspruchsbeweise geführt.**

Umwandlung in Klauseln

Satz. **Jede Aussage (well formed formula, wff) kann man in eine äquivalente Konjunktion von Klauseln umformen.**

Anmerkung: Resolutionsregel führt nicht aus der Menge der Klauseln heraus!

Beispiel. **(typische Vorgehensweise)**

Gegeben: $\neg(P \to Q) \vee (R \to P)$

a) Implikation beseitigen: $\neg(\neg P \vee Q) \vee (\neg R \vee P)$

b) Negation zu den Atomen bringen: $(P \wedge \neg Q) \vee (\neg R \vee P)$

(*de Morgansche Regeln*, ¬¬,...)

c) Konjunktive Normalform erzeugen:

$(P \vee \neg R) \wedge (\neg Q \vee \neg R \vee P)$ **(*Distr.*, *Assoz.*,...)**

d) Klauselform erzeugen:

$$\{(P \vee \neg R), (\neg Q \vee \neg R \vee P)\}$$

Beweis mit Resolutionsverfahren

Seien Δ eine Menge von Aussagen und ω eine Aussage.

Frage: **Gilt Δ ⊨ω (d.h. folgt ω aus Δ)?**

Beweisprozedur:

Schritt 1: **Konvertiere Δ in (konjunktive) Menge von Klauseln**

Schritt 2: **Konvertiere ¬ω in Klauselform**

Schritt 3: **Vereinige die Klauselmengen aus Schritt 1 und Schritt 2 in der Klauselmenge Γ**

Schritt 4: **Wende das Resolutionsprinzip auf die Klauseln in Γ an und füge die Ergebnisse zu Γ hinzu, bis die leere Klausel (der Widerspruch) erzeugt ist oder keine neuen Klauseln erzeugt werden können.**

Beweis mit Resolventenverfahren

Anmerkungen:

Das Verfahren ist (widerlegungs-)vollständig:
Gilt $\Delta \models \omega$, dann wird die leere Klausel auch erzeugt.

Das Verfahren ist auch entscheidbar:
Ist $|\Delta| < \infty$ und gilt $\Delta \models \omega$ <u>nicht</u>, so terminiert das Verfahren.

Beispiel

Δ = **{BAT_OK, ¬MOVES, BAT_OK ∧ LIFTABLE ⇒ MOVES}**

ω = **¬LIFTABLE**

Wissen: Falls die Batterie des Roboters OK ist und der Block nicht zu schwer ist, dann kann der Arm des Roboters den Block hochheben.

Frage: $\Delta \models \omega$ **?**

a) Semantische Lösung: **Mit Wahrheitstabellen nachweisen**

b) Syntaktische Lösung: **Mit der obigen Beweisprozedur:**

Klauselmengen:

Δ = **{BAT_OK, ¬MOVES,**

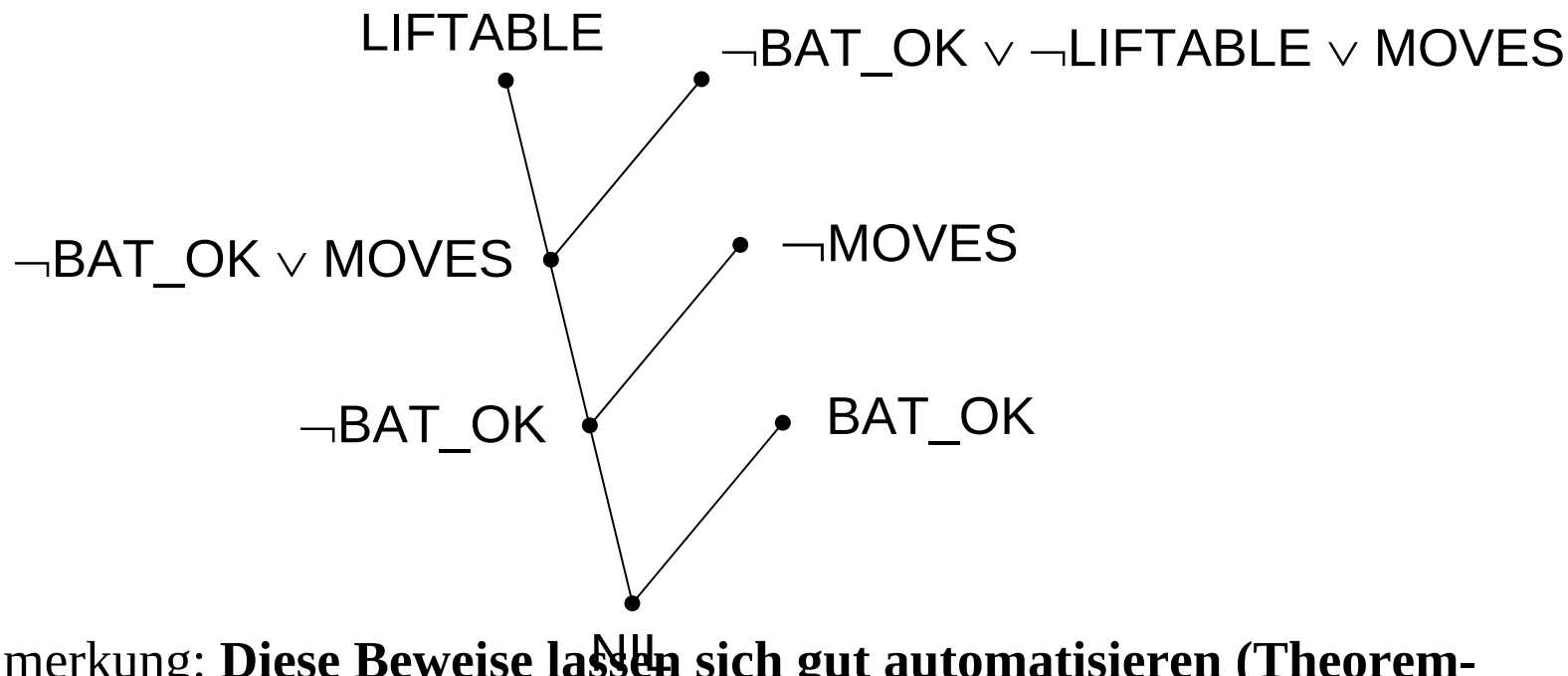
¬BAT_OK ∨ ¬LIFTABLE ∨ MOVES}

$\neg\omega$ = **{LIFTABLE}**

$\Gamma = \Delta \cup \neg\omega$

Beispiel

Widerspruchsbeweis durch Resolventenbildung:



Anmerkung: **Diese Beweise lassen sich gut automatisieren (Theorem-Beweiser, Maschinelles Beweisen) – siehe Internet für Programme.**

Suchstrategien in Widerlegungsbäumen

Problem:

Welche Resolvente wird zuerst gebildet um NIL schnell zu finden?

a) **Tiefensuche; Breitensuche; Klauselmengen der Klauseln mit nur einem Literal bevorzugen (unit clause strategy)**

b) **Verfeinerte Strategien wie set of support, linear input, ancestering filtering werden in Theorembeweisern für effiziente Implementierung verwendet**

Horn-Klauseln

Definition. **Eine Horn-Klausel ist eine Klausel, die höchstens ein positives Literal enthält.**

Beispiel.

a) $P, \neg P \vee Q, \neg P \vee \neg Q \vee R, \neg P \vee \neg R$

b) Es gibt 3 Typen von Horn-Klauseln:

$$\rightarrow P \qquad \text{(Fakt)}$$

$$P_1 \wedge P_2 \wedge P_3 \rightarrow Q \qquad \text{(Regel)}$$

$$P_1 \wedge P_2 \rightarrow \qquad \text{(Ziel, Anfrage)}$$

Bemerkung. **Es gibt schnelle Deduktionsalgorithmen (in linearer Zeit) für Horn-Klauseln, deswegen werden (falls möglich) Horn Klauseln verwendet. (Horn-Klauseln haben keine Disjunktionen positiver Literale!)**

Prädikatenkalkül (predicate calculus)

Anmerkung: **Das Prädikatenkalkül ermöglicht es Individuen zu definieren (zu benennen) und ihnen Eigenschaften zuzuweisen. Dies ist im Aussagenkalkül nicht möglich.**

Variable

Begriffsbildung

Terme: Vater_von(John); times(4, plus(3, 6)); Rolf; x; ...

Formeln:

Atomare Formel (Atom):

greater_than(7, 2); P(A,B,C,D); Q(x); …

Zusammengesetzte Formel:

greater_than(7, 2) ∧ less_than(14, 4) ∨
¬brother(John, Sam) ∨ P

Formeln mit Quantoren:

$$\forall x \left(P(x) \rightarrow Q(x)\right), \exists x \, \forall y \left(R(x, y)\right)$$

Prädikatenkalkül

Klauseln (Beispiel):

Im Folgenden ist die Funktion . eine Abbildung I × I → I, wobei I Menge der Individuen ist (*append operator*).

$$\big(\text{liste}([]) \wedge \forall X\ \forall L : \big(\text{liste}(L) \Rightarrow \text{liste}(.(X, L))\big)\big) \Rightarrow \text{liste}(.(a, []))$$

Kann man auch als Programm schreiben:

$\Rightarrow \text{liste}([])$ (Fakten)

$\text{liste}(L) \Rightarrow \text{liste}(.(X, L))$ (Regeln)

$\text{liste}(.(a, [])) \Rightarrow$ (Anfragen)

(Die Formel ist allgemeingültig, das Programm beweist die Allgemeingültigkeit.)

Prädikatenkalkül

Resolventen:

Werden wie im Aussagenkalkül gebildet. Eine Besonderheit ist die geeignete Substitution von freien Variablen, damit die Resolventenbildung möglich ist:

$$P(f(y), A) \vee Q(B, C), \quad \neg P(x, A) \vee R(x, C) \vee S(A, B)$$

Ersetze *x* durch *f*(*y*), dann erhält man als Resolvente

$$Q(B, C) \vee R(f(y), C) \vee S(A, B)$$

Unifikation

Unifikation nennt man das Finden einer geeigneten Substitution um Atome „gleich“ zu machen und somit die Resolution anwenden zu können.

Beispiele von Substitutionen:

Klausel	Substitution	Ergebnis (Substitutionsinstanz)
$P[X, f(y), B]$	$s_1 = \{z \mid x, w \mid y\}$	$P[z, f(w), B]$
		(alphabetische Variante)
$P[X, f(y), B]$	$s_2 = \{A \mid y\}$	$P[x, f(A), B]$
$P[X, f(y), B]$	$s_3 = \{g(z) \mid x, A \mid y\}$	$P[g(z), f(A), B]$
$P[X, f(y), B]$	$s_4 = \{C \mid x, A \mid y\}$	$P[C, f(A), B]$
		(Grundinstanz)

<u>Anmerkung:</u> Sind mehrere Substitutionen in einer Menge zusammengefasst werden alle gleichzeitig ausgeführt.

Unifikation

- **Ist ω ein Ausdruck und s eine Substitution, so ist ωs eine Substitutionsinstanz, z. B.** $P\left(x, f(y), B\right)s_1 = P\left(Z, f(w), B\right)$
- **Substitutionen können auch hintereinander geschaltet werden:**

  Sei $\omega = P(x, y),\ \ s_1 = \left\{f(y)|x\right\},\ \ s_2 = \left\{A|y\right\}$

  dann erhält man:

$$(\omega\ s_1)s_2 = \left[P(f(y), y)\right]s_2 = P(f(A), A)$$

$$= \left[P(x,y)\right]\left\{f\left(A\right)|x,A|y\right\} = \omega(s_1 s_2)$$

$$\omega(s_1 s_2) = P(f(A), A) \neq P(f(y), A) = \omega\left(s_2 s_1\right)$$

- **<u>Anmerkung:</u> Bei der Hintereinanderschaltung von Substitutionen wird die rechts stehende Substitution zunächst auf die links stehende angewandt. Es gilt Assoziativität.**

Unifizierbarkeit

Definition: **Eine Menge {ω_1,... ,ω_n} von Ausdrücken heißt unifizierbar (verschmelzbar), wenn es eine Substitution *s* gibt, so dass $\omega_1 s = \omega_2 s = ... = \omega_n s$ gilt. *s* heißt dann Unifizierer der Menge.**

Die Menge $\{P(x, f(y), B), P(x, f(B), B)\}$ **kann man z.B. mit der Substitution** $s = \{A \mid x, B \mid y\}$ **verschmelzen. Sie ist jedoch nicht die „einfachste“ Substitution,** $s' = \{B \mid y\}$ **ist einfacher.**

Unifizierbarkeit

Definition: **Es sei $A = \{w_1,...,w_n\}$ eine Menge von Ausdrücken. Dann gilt:**

***g* ist der allgemeinste Unifizierer von A $\Leftrightarrow$**

(i) ***g* ist Unifizierer von A**

(ii) **Ist *s* Unifizierer von A, dann gibt es eine Substitution *s'* mit $As = Ags'$.**

Die von dem allgemeinsten Verschmelzer produzierte Instanz ist bis auf alphabetische Varianten eindeutig.

Unifizierbarkeit

Anmerkung: **Es gibt eine ganze Reihe von Algorithmen, um den allgemeinsten Verschmelzer zu finden. Der Algorithmus aus [Chang und Lee, 1973] nutzt Ausdrücke in Listenstruktur**

$$\neg P(x, f(A, y)) \text{ entspricht } (\neg\ P\ x\ (f\ A\ y))$$

sowie die Idee der Abweichungsmenge (disagreement set).

Beispiel:

{{*A*,*z*},{*y*,*B*}} ist die Abweichungsmenge. Bei einer Implementierung bestimmt man in der Regel die Abweichungen nacheinander, d.h. die erste bestimmte Abweichung ist {*A*,*z*} und die Substitution somit {*A*|*z*}.

$$\{(\neg\ P\ x\ (\ f\ \ A\ y\ )\ ),\ (\neg\ P\ x\ (\ f\ \ z\ \ B\ )\ )\}$$

Verschmelzungsalgorithmus

(Γ Menge von Ausdrücken in Listenstruktur)

(I) $k := 0, \Gamma_k := \Gamma, s_k := \varepsilon$ **(ε leere Substitution)**

(II) **Enthält Γ_k nur identische Ausdrücke (d.h. als Menge nur einen Term), so ist s_k allgemeinster Unifikator; Ende**

(III) **Man bestimme die Abweichungsmenge D_k von Γ_k**

(IV) **Falls es eine Variable v_k in D_k und einen Term t_k in D_k gibt, so dass v_k nicht in t_k vorkommt, so setze mit (V) fort. Andernfalls ist Γ nicht verschmelzbar.**

(V) $s_{k+1} := s_k \{t_k \mid v_k\}; \Gamma_{k+1} := \Gamma_k \{t_k \mid v_k\}$ $\left(\Gamma_{k+1} = \Gamma s_{k+1}\right)$

(VI) **k:=k+1**

(VII) **Fahre mit (II) fort**

Verschmelzungsalgorithmus

Beispiel:

a)

$$\Gamma = \{p(a, x, f(g(y)), p(z, f(z), f(v))\}$$

$$s_0 = \varepsilon,\ \ s_1 = \{a \mid z\},\ \ \Gamma_1 = \{p(a, x, f(g(y)), p(a, f(a), f(v))\}$$

$$s_2 = \{a \mid z, f(a) \mid x\},\ \ \Gamma_2 = \{p(a, f(a), f(g(y)), p(a, f(a), f(v))\}$$

$$s_3 = \{a \mid z, f(a) \mid x, g(y) \mid v\},\ \ \Gamma_3 = \{p(a, f(a), f(g(y))\}$$

b) **ist nicht unifizierbar!**

$$\Gamma = \{q(f(a), g(x)), q(y, y)\}$$

Resolution im Prädikatenkalkül

- **γ_1, γ_2 seien zwei Klauseln (Mengen von Literalen). Gibt es ein Atom φ (Relationskonstante der Ordnung *n*, die von *n* Termen gefolgt wird und durch Kommata getrennt sind) in γ_1 und ein Literal $\neg\psi$ in γ_2 derart, dass φ und ψ den allgemeinsten Verschmelzer μ besitzen, so haben γ_1, γ_2 die Resolvente**

$$\rho = \left[(\gamma_1 - \{\varphi\}) \cup (\gamma_2 - \{\neg\psi\})\right]\mu$$

- **Dieses Verfahren ist korrekt und widerlegungsvollständig.**
- **Um Variablenkollisionen zu vermeiden, benennen wir die Variablen vorher so um, dass die Variablen in den Klauseln verschieden sind.**

Resolution im Prädikatenkalkül

Beispiele:

a) $\{P(x), Q(x,y)\}$ und $\{\neg P(A), R(B,z)\}$

ergeben als Resolvente:

$\{Q(A,y), R(B,z)\}$

b) $\{P(x,x), Q(x), R(x)\}$ und $\{\neg P(A,z), \neg Q(B)\}$

ergeben als Resolventen

$\{Q(A), R(A), \neg Q(B)\}$ oder $\{P(B,B), R(B), \neg P(A,z)\}$

Überführung in Klauselform

Überführung von Formeln in Klauselform:

Schritt 1: $\Rightarrow$, $\Leftrightarrow$ eliminieren

$F \Rightarrow G$ entspricht $\neg F \vee G$

$F \Leftrightarrow G$ entspricht $F \Rightarrow G \wedge G \Rightarrow F$

Schritt 2: Gültigkeitsbereiche der Negationszeichen verkleinern**.**

Beispiel:

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

$\neg(\forall X : F)$ entspricht $\exists X : \neg F$

Überführung in Klauselform

Schritt 3: Quantoren nach vorne ziehen

Im Folgenden:

Z komme in *A*(*X*) und *B*(*X*) nicht vor
X komme in *E* nicht vor
$Q, Q_1, Q_2 \in \{\exists,\forall\}$

Mögliche Umformungen:

$(QX : A(X)) \wedge E$ äquivalent $QX : (A(X) \wedge E)$

$(QX : A(X)) \vee E$ äquivalent $QX : (A(X) \vee E)$

$\forall X : A(X) \wedge \forall X : B(X)$ äquivalent $\forall X : (A(X) \wedge B(X))$

$\exists X : A(X) \vee \exists X : B(X)$ äquivalent $\exists X : (A(X) \vee B(X))$

$Q_1(X) : A(X) \wedge Q_2(X) : B(X)$ äquivalent $Q_1 X : Q_2 Z : (A(X) \wedge B(Z))$

$Q_1(X) : A(X) \vee Q_2(X) : B(X)$ äquivalent $Q_1 X : Q_2 Z : (A(X) \vee B(Z))$

Überführung in Klauselform

Schritt 4: Den quantorenfreien Ausdruck als Konjunktion von Disjunktionen von Literalen darstellen:

$$(A \wedge B) \vee C \text{ äquivalent } (A \vee C) \wedge (B \vee C)$$
$$A \vee (B \wedge C) \text{ äquivalent } (A \vee B) \wedge (A \vee C)$$

Überführung in Klauselform

Beispiel zu Schritt 4:

$$\forall X : \forall Y : \left(\left(\exists Z : \left(p\left(X,Z\right) \vee p\left(Y,Z\right)\right)\right) \Rightarrow \left(\exists Z : q\left(X,Y,Z\right)\right)\right)$$

äquivalent

$$\forall X : \forall Y : \left(\neg\left(\exists Z : \left(p\left(X,Z\right) \vee p\left(Y,Z\right)\right)\right) \vee \exists Z : q\left(X,Y,Z\right)\right)$$

äquivalent

$$\forall X : \forall Y : \left(\left(\forall Z : \left(\neg p\left(X,Z\right) \wedge \neg p\left(Y,Z\right)\right) \vee \exists Z : q\left(X,Y,Z\right)\right)\right)$$

äquivalent

$$\forall X : \forall Y : \forall Z : \exists U : \left(\left(\neg p\left(X,Z\right) \wedge \neg p\left(Y,Z\right)\right) \vee q\left(X,Y,U\right)\right)$$

äquivalent

$$\forall X : \forall Y : \forall Z : \exists U : \left(\left(\neg p\left(X,Z\right) \vee q\left(X,Y,U\right)\right) \wedge \left(\neg p\left(X,Z\right) \vee q\left(X,Y,U\right)\right)\right)$$

Überführung in Klauselform

Schritt 5: Elimination der Existenzquantoren:

Q sei der am weitesten links stehende Existenzquantor in A:

- **Wenn vor Q kein Allquantor steht, streiche (QX): aus A und ersetze X durch Konstante, die nicht in A vorkommt**
- **Sind Q_1X_1:, ..., Q_mX_m: alle Allquantoren die links von Q stehen, so wählt man ein Funktionssymbol f, das nicht in A vorkommt, definiert f als m-stellige Funktion, und ersetzt jedes Vorkommen von X durch $f(X_1,..., X_m)$ und streicht (QX): im Ausdruck A.**

  <u>Anmerkung:</u> f wird auch Skolemfunktion genannt. Die erhaltene Form nennt sich auch Skolem-Normalform.

Überführung in Klauselform

Schritt 6: Darstellung als Klauselmenge

$$\forall X : \left(A(X) \wedge B(X)\right) \text{ äquivalent } \left(\forall X : A(X)\right) \wedge \left(\forall Y : B(Y)\right)$$

Beispiel: **(siehe nächste Folie)**

Satz. ***A*** **Aussage,** ***S*** **Klauselform von** ***A*** **:**

a) ***A*** **unerfüllbar** $\Leftrightarrow$ ***S*** **unerfüllbar**

b) ***A*** **und** ***S*** **nicht notwendig äquivalent**

<u>Anmerkung:</u> Fehlende Äquivalenz liegt an Skolem-Normalform:

Beispiel: $P(a) \vee P(b) \models \exists x\, P(x)$**, aber i.A. gilt nicht** $P(a) \vee P(b) \models P(s)$ **(Skolem-Konstante** ***s*****).**

Überführung in Klauselform

Beispiel zu Schritt 6: **(Fortsetzung von Beisp. zu Schritt 4)**

Schritt 5 folgend wird *U* zunächst durch *f*(*X*,*Y*,*Z*) ersetzt:

$$\forall X:\forall Y:\forall Z:\left(\neg p\left(X,Z\right)\vee q\left(X,Y,f\left(X,Y,Z\right)\right)\right)\wedge \left(\neg p\left(Y,Z\right)\vee q\left(X,Y,f\left(X,Y,Z\right)\right)\right)$$

äquivalent

$$\forall X_1:\forall Y_1:\forall Z_1:\left(\neg p\left(X_1,Z_1\right)\vee q\left(X_1,Y_1,f\left(X_1,Y_1,Z_1\right)\right)\right)\wedge \forall X_2:\forall Y_2:\forall Z_2:\left(\neg p\left(Y_2,Z_2\right)\vee q\left(X_2,Y_2,f\left(X_2,Y_2,Z_2\right)\right)\right)$$

äquivalent

$$\left\{\neg p\left(X_1,Z_1\right)\vee q\left(X_1,Y_1,f\left(X_1,Y_1,Z_1\right)\right),\neg p\left(Y_2,Z_2\right)\vee q\left(X_2,Y_2,f\left(X_2,Y_2,Z_2\right)\right)\right\}$$

äquivalent

$$p\left(X_1,Z_1\right)\Rightarrow q\left(X_1,Y_1,f\left(X_1,Y_1,Z_1\right)\right.$$

$$p\left(Y_2,Z_2\right)\Rightarrow q\left(X_2,Y_2,f\left(X_2,Y_2,Z_2\right)\right.$$

Resolution in Theorem-Beweisern

Es werden Widerspruchsbeweise geführt.

Beispiel:

a)

$\text{package}(x) \mathrel{\hat{=}} x \text{ ist Paket} \quad \mathrel{\hat{=}} P(x)$

$\text{inroom}(x, y) \mathrel{\hat{=}} x \text{ ist ein Raum } y \quad \mathrel{\hat{=}} I(x,y)$

$\text{smaller}(x, y) \mathrel{\hat{=}} x \text{ ist kleiner als } y \quad \mathrel{\hat{=}} S(x,y)$

b) Alle Pakete in Raum 27 sind kleiner als alle Pakete in Raum 28

$\mathrel{\hat{=}}$ **$\forall x\, \forall y$: ((package(x) $\wedge$ (package(y) $\wedge$ inroom(x,27) $\wedge$ inroom(y,28)) $\Rightarrow$ smaller(x,y)**

$$\mathrel{\hat{=}} \neg P(x) \vee \neg P(y) \vee \neg I(x,27) \vee \neg I(y,28) \vee S(x,y)$$

Resolution in Theorem-Beweisern

Beispiel (Fortsetzung):

c) Jedes Paket in Raum 27 ist kleiner als eines der Pakete in Raum 29:

$$\hat{=} \forall x : \exists y : \left(P(x) \wedge P(y) \wedge I(x,27) \wedge I(y,29) \Rightarrow S(x,y)\right)$$

Anmerkung: Vorsicht bei linguistischen Mehrdeutigkeiten, es könnte auch gemeint gewesen sein:

$$\exists y : \forall x \left(P(x) \wedge P(y) \wedge I(x,27) \wedge I(y,29) \Rightarrow S(x,y)\right)$$

Der Vorteil des PKI ist, dass es den Nutzer zwingt, solche Mehrdeutigkeiten aufzulösen.

Resolution in Theorem-Beweisern

Beispiel (Fortsetzung):

d) Paket *A* ist in Raum 27 oder in Raum 28, Paket *B* ist in Raum 27, Paket *B* ist nicht kleiner als Paket *A*:

$$P(A),\ P(B),\ I(A,27) \vee I(A,28),\ I(B,27),\ \neg S(B,A)$$

e) Kann man aus den Eigenschaften b) und d) folgern, dass Paket *A* in Raum 27 ist?

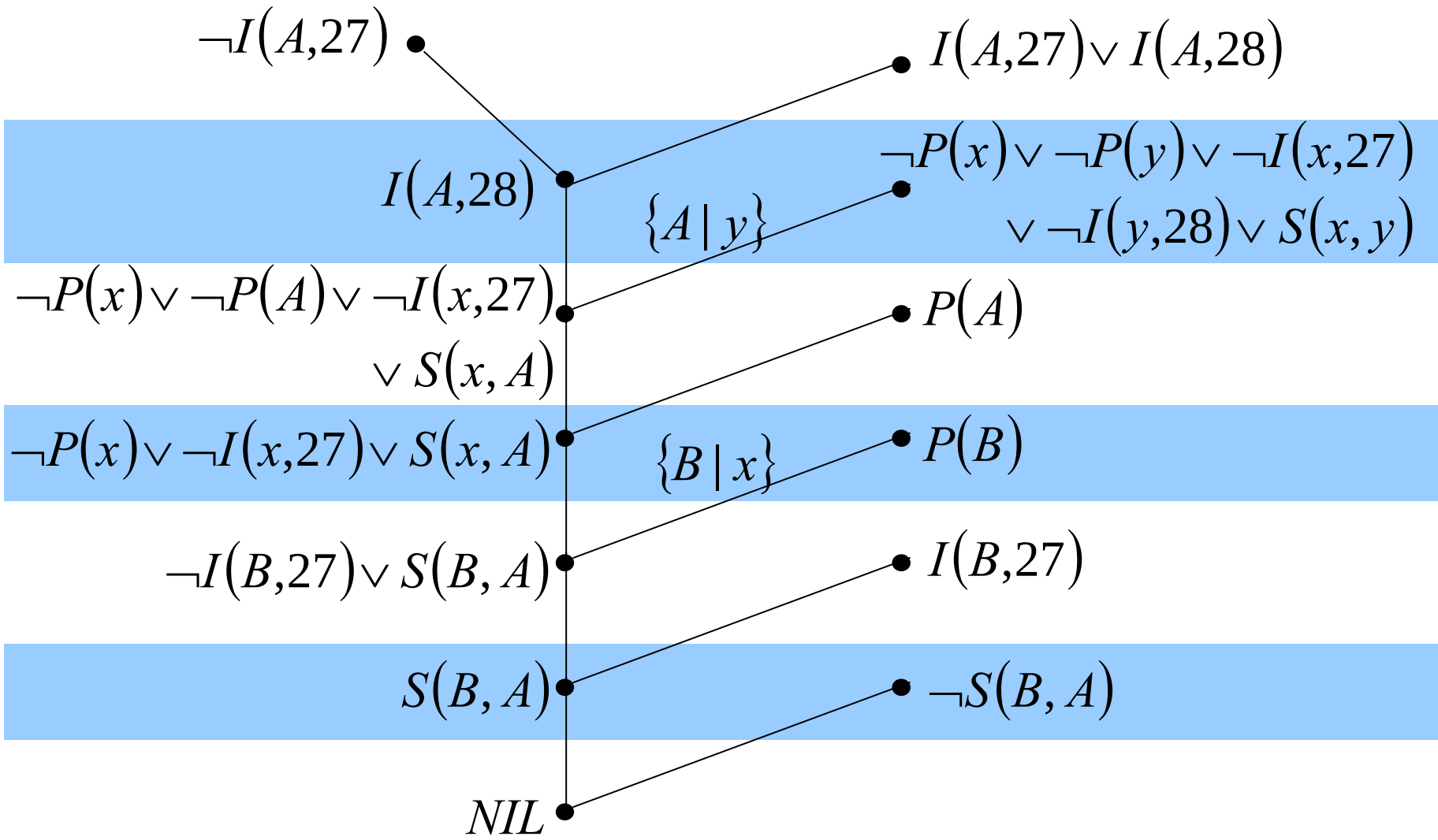
Ja:

$$\Delta = \{\neg P(x) \vee \neg P(y) \vee \neg I(x,27) \vee \neg I(y,28) \vee S(x,y),$$
$$P(A),\ P(B),$$
$$I(A,27) \vee I(A,28),\ I(B,27),$$
$$\neg S(B,A)\}$$

$$w = I(A,27)$$

$\Delta \cup \{\neg w\}$ wird zum Widerspruch geführt :

Resolution in Theorem-Beweisern



Widerlegung durch Resolution

Resolution in Theorem-Beweisern

Beispiel (Fortsetzung):

f) Kann man aus c) und d) herausfinden, in welchem Raum A ist?

Anfrage: $\exists u : I(A,u)$

Lösungsidee:

Es wird ein Antwortliteral *Ans*(*u*) eingeführt und dem Negat der Anfrage beigefügt:

$$\neg I(A,u) \vee Ans(u)$$

Dann wird die Resolution solange durchgeführt bis nur noch das Antwortliteral übrig bleibt.

Resolution in Theorem-Beweisern