Prädikatenlogik

Syntax

Semantik

Formeln, Modelle, Tautologien und Anwendungen

Entscheidbarkeitsfragen

Kalküle und Tableaux

Normalformen und Resolution

Pränexe Normalform

Eine PL1-Formel ist in **PNF** (Pränexer Normalform), wenn „alle **Quantoren vorne** stehen.“

Formale Def.?

Normalformensatz (sogar für „bereinigte PNF“)

Zu jeder PL1-Formel gibt es mindestens eine äquivalente **PNF**-Formel, in der **zusätzlich**

- keine Variable sowohl gebunden als auch frei vorkommt und
- alle Quantorvariablen unterschiedlich sind.

} **bereinigt**

Begründung konstruktiv: **Umformungsalgorithmus BPNF**

- Bereinigung
- auf Junktorenbasis $\neg, \wedge, \vee$ bringen
- Umformung: Quantoren vorziehen

Bereinigung durch gebundene Umbenennungen

$P(x) \vee (\forall x (\exists x Q(x,x) \wedge R(x)) \vee S(x))$ ← **Beispiel**

1. Suche von links nach rechts eine noch nicht „erledigte“ Quantorvariable. Gibt es keine, ist die Bereinigung fertig.

   $P(x) \vee (\forall x (\exists x Q(x,x) \wedge R(x)) \vee S(x))$

2. Gibt es ein kollidierendes (erledigtes oder freies) Vorkommen derselben Variable ?

   JA, dann:

   - Benenne eine Quantorvariable der beiden sowie alle durch sie gebundenen Variablenvorkommen um zu einem neuen Variablennamen.
   - Markiere die unerledigte beider Vorkommenmengen als erledigt.

   $P(x) \vee (\forall y (\exists x Q(x,x) \wedge R(y)) \vee S(x))$

   NEIN, dann: […ist der Variablenname unproblematisch, also …]

   - Markiere die gefundene Quantorvariable sowie alle durch sie gebundenen Variablenvorkommen als erledigt.

3. Gehe nach 1. **…** $P(x) \vee (\forall y (\exists x Q(x,x) \wedge R(y)) \vee S(x))$ **usw.**

Äquivalente Umformung in bereinigte PNF (1)

Algorithmus BPNF (eine strikte Variante) **Beispiele**

1. Bereinigung durch gebundene Umbenennungen *(wie gerade gesehen)*
   $\forall x P(x) \vee \exists x Q(x) \Rightarrow \forall x P(x) \vee \exists y Q(y)$, $\forall x \exists x P(x) \Rightarrow \forall x \exists y P(y)$
2. $\rightarrow$ und $\leftrightarrow$ durch $\neg \wedge \vee$ ersetzen *vgl. Aussagenlogik* $\varphi \rightarrow \psi \Rightarrow \neg\varphi \vee \psi$
3. Alle Negationen hinter die Quantoren ziehen:
   a) $\neg\forall x\varphi \Rightarrow \exists x \neg\varphi$ $\neg\forall x \neg P(x,y) \Rightarrow \exists x \neg\neg P(x,y)$
   b) $\neg\exists x\varphi) \Rightarrow \forall x \neg\varphi$, und ggf.* …
   c) $\neg(\varphi \wedge/\vee \psi) \Rightarrow (\neg\varphi \vee/\wedge \neg\psi)$ ← *von $\wedge$ nach $\vee$ oder umgekehrt*
   d) $\neg\neg\varphi \Rightarrow \varphi$ $\exists x \neg\neg P(x,y) \Rightarrow \exists x\, P(x,y)$
4. Alle Quantifikationen in der vorliegenden Reihenfolge vor die ganze Formel ziehen *(durch leere Scope-Erweiterungen)*
   $\forall x P(x) \vee \exists y \neg Q(y) \Rightarrow \forall x\, (P(x) \vee \exists y \neg Q(y))$
   $\Rightarrow \forall x \exists y\, (P(x) \vee \neg Q(y))$

Satz: BPNF **terminiert** und ist **korrekt**. ← Ü53a ← Ü53b

*) (3.c/d): wenn Quantor in φ oder ψ / in φ. (3.d) ist entbehrlich und beschleunigt nur.

„Skolemisierung“

Eine PL1-Formel ist in **Skolem-Form**, wenn sie in **BPNF** ist und **kein** ∃ enthält.

Normalerweise gibt es zu einer PL1-Formel φ **nicht unbedingt** irgendeine **äquivalente** Formel in Skolem-Form – z.B. gibt es keine zu $\forall x \exists y P(x,y)$.

Aber ...

z.B. im Falle $\forall x \exists y P(x,y)$, zu jedem x können wir aus allen zu x „passenden“ y eines willkürlich herausgreifen und als $f(x)$ dem x zuordnen
→ $\forall x P(x,f(x))$ – weg ist der Existenzquantor! Doch was haben wir erhalten?

erfüllbarkeitsäquivalent

Satz: Skolem-Form

(1) Zu jeder PL1-Formel φ existiert eine Formel ψ in Skolem-Form, die **genau dann erfüllbar ist, wenn φ erfüllbar ist**.

(2) Solch eine erfüllbarkeitsäquivalente Formel ψ erhält man in endlich vielen Schritten durch den **Algorithmus Skol**.

Skolemisierungs-Algorithmus

Algorithmus Skol:

Überführung einer **PNF**-PL1-Formel φ (Input) in eine (normalerweise ***nicht*** zu φ ***äquivalente***) Formel ψ (Output) in Skolem-Form, **die genau dann erfüllbar ist, wenn φ erfüllbar ist**.

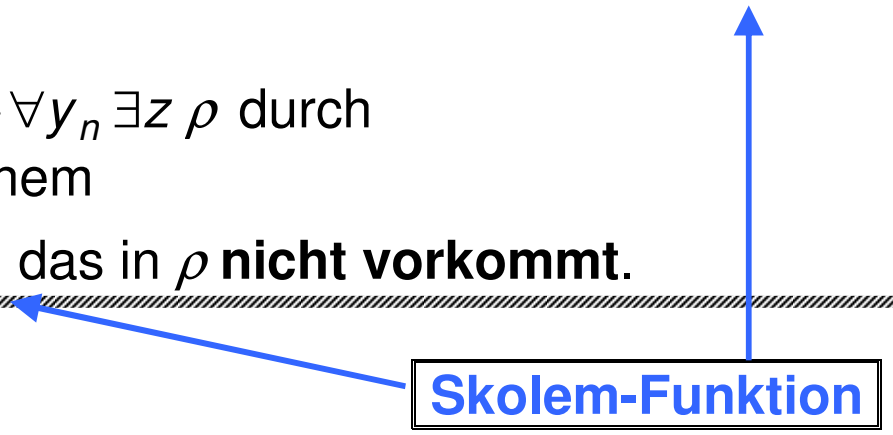
Wende BPNF an.
Wenn nun kein Existenzquantor vorkommt: STOP.

Wiederhole, solange die Formel einen Existenzquantor enthält:

Sie hat die Form $\exists y\ \rho$ oder $\forall y_1 \cdots \forall y_n \exists z \rho$.

– Im ersten Fall ersetze $\exists y\ \rho$ durch $\rho_{[y/c]}$ mit einer Konstante ***c***, die in ρ **nicht vorkommt**.

– Im zweiten Fall ersetze $\forall y_1 \cdots \forall y_n \exists z\ \rho$ durch $\forall y_1 \cdots \forall y_n\ \rho_{[z/f(y_1,\ldots,y_n)]}$ mit einem n-stelligen Funktionssymbol ***f***, das in ρ **nicht vorkommt**.



Skolemisierungs-Beispiel

$\exists x\ \forall y\ \forall z\ \exists u\ \forall v\ \exists w\ P(x,y,z,u,v,w)$

→ $\exists \boldsymbol{x}\ \forall y\ \forall z\ \exists u\ \forall v\ \exists w\ P(\boldsymbol{x},y,z,u,v,w)$

→ $\forall \boldsymbol{y}\ \forall \boldsymbol{z}\ \exists \boldsymbol{u}\ \forall v\ \exists w\ P(\boldsymbol{a},y,z,\boldsymbol{u},v,w)$

→ $\forall \boldsymbol{y}\ \forall \boldsymbol{z}\ \forall \boldsymbol{v}\ \exists \boldsymbol{w}\ P(a,y,z,\boldsymbol{f(y,z)},v,\boldsymbol{w})$

→ $\forall \boldsymbol{y}\ \forall \boldsymbol{z}\ \forall \boldsymbol{v}\ P(a,y,z,f(y,z),v,\ \boldsymbol{g(y,z,v)})$

$\forall y\ \forall z\ \forall v\ P(a,y,z,f(y,z),v,\ g(y,z,v))$

Ü54

Zwei Fakten zur Äquivalenz bezüglich Erfüllbarkeit und Allgemeingültigkeit (1)

Lemma: Jede PL1-Formel φ mit (mindestens) einer freien Variablen y_i ist **erfüllbarkeitsäquivalent** zu der Formel $\exists y_i\ \varphi$.

Grund: Die Erfüllbarkeit
sowohl von φ mit der freien Variablen y_i
als auch von $\exists y_i\ \varphi$
bedeutet die Existenz einer passenden Struktur und die Existenz eines passenden Wertes für y_i, die φ wahr machen.

Mehrfach angewendet, auf **alle** y_i → **Erfüllbarkeitsäquivalenz zu (per ∃) geschlossener Formel**

Korollar Ex.-Abschluss: Jede PL1-Formel φ mit den freien Variablen $y_1,\cdots,y_n$ ist erfüllbarkeitsäquivalent zu der geschlossenen Formel $\exists y_1 \cdots \exists y_n\ \varphi$.

Zwei Fakten zur Äquivalenz bezüglich Erfüllbarkeit und Allgemeingültigkeit (2)

Analog: **allgemeingültigkeitsäquivalent**

Lemma: Jede PL1-Formel φ mit (mindestens) einer freien Variablen y_i ist genau dann **allgemeingültig**, wenn die Formel $\forall y_i\ \varphi$ allgemeingültig ist.

Grund: Die Allgemeingültigkeit
sowohl von φ mit der freien Variablen y_i
als auch von $\forall y_i\ \varphi$
bedeutet dass in allen passenden Strukturen
alle Werte für y_i die Formel φ wahr machen.

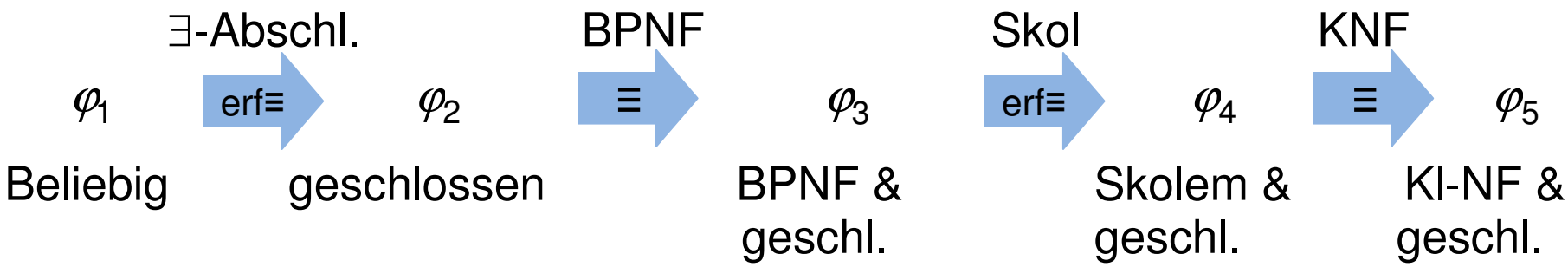
Mehrfach angewendet auf alle y_i → **Allgemeingültigkeitsäquivalenz zu (per $\forall$) geschlossener Formel**

Korollar: Jede PL1-Formel φ mit den freien Variablen $y_1,\cdots,y_n$ ist allggk.-äquivalent zu der geschlossenen Formel $\forall y_1 \cdots \forall y_n\ \varphi$.

Klauselnormalform

Eine PL1-Formel in Skolem-Form ist in **Klausel-Normalform**,
wenn ihre **Matrix** (:= der Rest hinter den Quantoren) in **KNF** ist.

Erfüllbarkeitsäquivalente Klauselnormalform
Zu jeder PL1-Formel existiert
eine erfüllbarkeitsäquivalente Formel in Klauselnormalform.



Man lässt meist die (nun redundanten) Allquantoren weg
und schreibt φ_5 in $\{\ \}_{KNF}$-Form.

Klausel-NF Beispiel

Formel		$\forall y \exists z P(x,y,z) \rightarrow \neg \exists y \forall v [Q(x,v) \rightarrow R(x,y)]$
abgeschlossen.	→	$\exists x \{ \forall y \exists z P(x,y,z) \rightarrow \neg \exists y \forall v [Q(x,v) \rightarrow R(x,y)] \}$
bereinigt	→	$\exists x \{ \forall y \exists z P(x,y,z) \rightarrow \neg \exists u \forall v [Q(x,v) \rightarrow R(x,u)] \}$
$\rightarrow$ ersetzt	→	$\exists x \{ \neg \forall y \exists z P(x,y,z) \vee \neg \exists u \forall v [\neg Q(x,v) \vee R(x,u)] \}$
$\neg$ nach innen	→	$\exists x \{ \exists y \forall z \neg P(x,y,z) \vee \forall u \exists v [Q(x,v) \wedge \neg R(x,u)] \}$
pränex	→	$\exists x \exists y \forall z \forall u \exists v \{ \neg P(x,y,z) \vee [Q(x,v) \wedge \neg R(x,u)] \}$
skolemisiert	→	$\forall z \forall u \{ \neg P(a,b,z) \vee [Q(a,f(z,u)) \wedge \neg R(a,u)] \}$
Klausel-NF	→	$\forall z \forall u \{ [\neg P(a,b,z) \vee Q(a,f(z,u))] \wedge [\neg P(a,b,z) \vee \neg R(a,u)] \}$
als Mengen	→	$\{ \{\neg P(a,b,z), Q(a,f(z,u))\} , \{\neg P(a,b,z), \neg R(a,u)\} \}$

Ü55

Herbrand-Universum

WWW-Suchmaschinen finden ca. 200.000 Seiten mit [herbrand logic].
Raten Sie, über wie viele Jahre hinweg Jacques Herbrand publizierte!

Über welche Terme „kann man konkret reden“;
welche Objekte im Universum haben eine Bezeichnung, sind „bekannt“?

Konstanten,
Funktionen darauf anwenden,
auch wiederholt.

Für eine geschlossene Skolem-Formel φ ist das **Herbrand-Universum** $\mathrm{D}(\varphi)$ die wie folgt induktiv definierte Termemenge
Konstanten in φ gehören zu $\mathrm{D}(\varphi)$; falls es keine gibt, dann c (bzw. a_1).
f n-stelliges Fkt.-symbol in φ und $\tau_1,\ldots,\tau_n \in \mathrm{D}(\varphi) \Rightarrow f(\tau_1,\ldots,\tau_n) \in D(\varphi)$.

Beispiel

$$\mathrm{D}(\forall x\,(P(g(x),a) \rightarrow Q(f(x,b)))) = \{a, b, g(a), g(b), f(a,a), \ldots, g(f(a,g(b))), \ldots\}$$

Herbrand-Expansion

Beispiel:

$D(\forall x\,(P(g(x),a) \rightarrow Q(f(x,b)))) = \{a, b, g(a), g(b), f(a,a), \ldots, g(f(a,g(b))), \ldots\}$

Die Formel spricht über alle Objekte x,
insbesondere über alle benennbaren,
d.h. $P(g(x),a) \rightarrow Q(f(x,b))$ gilt einzeln
für alle Objekte x des Herbrand-Universums.

Für eine geschlossene Skolem-Formel ist die **Herbrand-Expansion**

von $\varphi = \forall x_1 \cdots \forall x_n Matrix(\varphi)$ die Formelmenge

$$E(\varphi) := \{Matrix(\varphi)_{[x_1,\ldots,x_n / \tau_1,\ldots,\tau_n]} \mid \tau_1,\ldots,\tau_n \in D(\varphi)\}$$

Beispiel:

$E(\forall x\, P(g(x),a) \rightarrow Q(f(x,b))) =$

$\{P(g(a),a) \rightarrow Q(f(a,b)),\quad P(g(b),a) \rightarrow Q(f(b,b)),\quad \ldots\}$

Ü56

Grundresolution

Algorithmus **GRes**

Sei φ eine geschlossene Formel in Klauselnormalform.

Sei $E(\varphi) = \{\rho_1, \rho_2 ...\}$ die Herbrand-Expansion (ab-/aufzählbar!).

$M := \emptyset$.

Für $i := 1, 2, ...$,so lange bis $\emptyset \in M$:

- $M :=$ Resolventenmenge$(M \cup \rho_i)$.

Satz von Herbrand: GRes ist **Halbentscheidungsverfahren**

Der Algorithmus GRes **terminiert** für die Formel φ genau dann, wenn φ **unerfüllbar** ist.

Das Verfahren ist i.a. sehr **langwierig**
– und bei Erfüllbarkeit ohnehin **fruchtlos**.

Wir werden es durch die Technik der **Unifikation** beschleunigen.

Grundresolution, Beispiel (1)

Die KI-NF-Formel

$\varphi = \forall x[P(x) \wedge P(g(a,b,c)) \wedge \neg P(g(c,f(c),f(f(c))))]$

ist **unerfüllbar**,

denn für sub = [x / $g(c,f(c),f(f(c)))$] ergibt sich
sowohl $P(g(c,f(c),f(f(c))))$ als auch $\neg P(g(c,f(c),f(f(c))))$.

Wie löst das eine einfach programmierte **GRes**?

Wie probiert sie systematisch für *x* das Herbrand-Universum und so die Herbrand-Expansion von φ durch?

0	a b c	#=3
1	f(a) f(b) f(c) g(aaa) g(aab) **...** g(ccc)	#=30
2	f(f(a)) **...** f(g(ccc)) g(aaf(a)) **...** g(g(ccc) g(ccc) g(ccc))	#=35910
3	**...**	#=ca $3 \cdot 10^{13}$
4	f(f(f(f(a)))) … f(g(cf(c)f(f(c)))) **...**	wievielter Term? ☹
5	usw. **...**	

Grundresolution, Beispiel (2)

Methodenskizze des Beispiels:

0 Funktionsanwendungen ergeben „nullte“ Teilliste – die Konstanten (bzw. *c*).

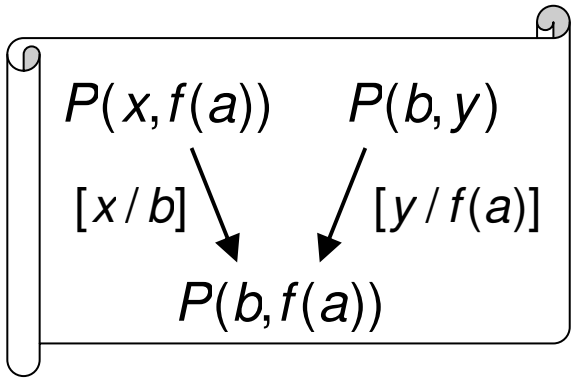
Für *i*=1,2,…

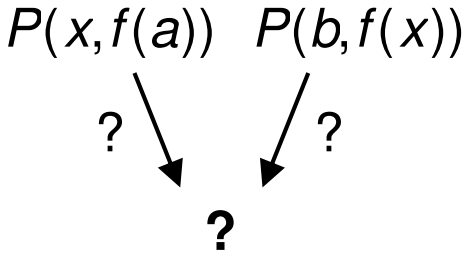
i-te Teilliste dazu, für ***maximal*** *i Funktionsanwendungen übereinander*,
d.h. für jede vorkommende Funktion systematisch:

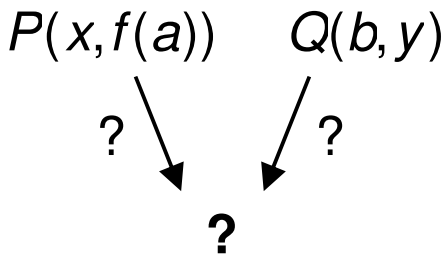
Anwenden auf alle möglichen Argumentkombinationen,
gebildet aus der bisherigen Liste
– mindestens ein Argument aus (*i*–1)-ter Teilliste.
Alle diese Terme zur *i*-ten Teilliste hinzuschreiben.

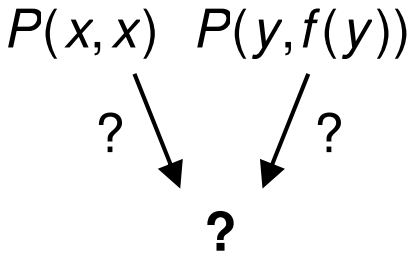
Unifikation, informell

Ziel: Variablen in mehreren atomaren Formeln so (konstant) **substituieren**, dass diese Formeln **identisch** werden









Ein **allgemeinster Unifikator** macht *keine überflüssigen Festlegungen* durch Konstanten; jeden Unifikator erhält man aus ihm durch weitere Substitution.

Beispiel: $[x,y/b,f(a)]$ ist der allgemeinste Unifikator von $P(x,f(a),z)$ und $P(b,y,z)$, er ergibt $P(b,f(a),z)$. $[x,y,z/b,f(a),c]$ ist ein weiterer Unifikator ($\rightarrow P(b,f(a),c)$).

Unifikationsalgorithmus

Geg.: $\mathcal{L} = \{L_1,...,L_n\}$, $n \geq 1$, alle L_i PL1-Literale
(d.h. atomare PL1-Formel oder Negation davon)

1. Literalarbeitsmenge $\mathcal{M} := \mathcal{L}$, Substitution $s := [\ /\ ]$ (leer)
2. IF $\mathcal{M}$ einelementig THEN **{Ausgabe := *s*; STOP}** ELSE continue
3. Bestimme $L_1, L_2 \in \mathcal{M}$ mit $L_1 \neq L_2$;
   z_1, z_2 seien die ersten in L_1, L_2 voneinander verschiedenen Zeichen.
4. IF weder z_1 noch z_2 Variablenname THEN **{Ausgabe := "geht nicht"; STOP}**
5. O.B.d.A. sei z_1 Variablenname; und t der Teilterm von L_2, der mit z_2 beginnt.
   IF z_1 kommt in t vor THEN **{Ausgabe := "geht nicht"; STOP}**
6. s := „s zuzüglich $[z_1 / t]$"; $\mathcal{M} := \{q_{[z_1 / t]} \mid q \in \mathcal{M}\}$; GOTO 2.

Achtung: eigentlich Zeichen**vorkommen**, nicht Zeichen

Der Algorithmus

- terminiert
- entscheidet, ob eine Literalemenge unifizierbar ist
- und gibt, wenn ja, den allgemeinsten Unifikator aus.

Unifikation – Beispiel

L = { P(f(x),w,f(a)), P(f(w),w,f(z)), P(f(g(z)),x,y) }

s=[/] M = { P(f(x),w,f(a)), P(f(w),w,f(z)), P(f(g(z)),x,y) }

P(f(x),w,f(a))
P(f(w),w,f(z))

s=[x/w] M = { P(f(w),w,f(a)), P(f(w),w,f(z)), P(f(g(z)),w,y) }

P(f(w),w,f(a))
P(f(w),w,f(z))

s=[x/w] [z/a] M = { P(f(w),w,f(a)), P(f(g(a)),w,y) }

P(f(w),w,f(a))
P(f(g(a)),w,y)

s=[x/w] [z/a] [w/g(a)] M = { P(f(g(a)),g(a),f(a)), P(f(g(a)),g(a),y) }

P(f(g(a)),g(a),f(a))
P(f(g(a)),g(a),y)

s=[x/w] [z/a] [w/g(a)] [y/f(a)] M = { P(f(g(a)),g(a),f(a)) }
↑ allgemeinster Unifikator ↑ ↑ das „Unifikat“ ↑

Ü57

PL1-Resolutionsschritt, informell

K_1 und K_2 seien **Klauseln** aus **PL1-Literalen** in Mengenform

sub_1 und sub_2 seien Variablen-Umbenennungen, so dass $K_{1\,sub1}$ und $K_{2\,sub2}$ **variablenfremd**.

Es gebe Literalmengen $\{L_1,\ldots,L_m\} \subseteq K_{1\,sub1}$ und $\{L'_1,\ldots,L'_n\} \subseteq K_{2\,sub2}$, so dass $\{L_1,\ldots,L_m,L'_1,\ldots,L'_n\}$ widersprüchlich unifizierbar mit allgemeinstem Unifikator sub.

Resolvente:

$[\ (K_{1\,sub1} \setminus \{L_1,\ldots,L_m\}) \cup (K_{2\,sub2} \setminus \{L'_1,\ldots,L'_n\})\ ]_{sub}$

Beispiel:

$K_1 = \{P(f(x)), \neg Q(z), P(z)\}$ $\qquad$ $K_2 = \{\neg P(x), R(g(x),a)\}$

$sub_1 = [\,]$ $\qquad$ $sub_2 = [x/y]$

$\{P(f(x)), \neg Q(z), P(z)\}$ $\qquad$ $\{\neg P(y), R(g(y),a)\}$

$sub = [y/f(x)]$ $\qquad$ $sub = [y/f(x)]$

$\{P(f(x)), \neg Q(z), P(z)\}$ $\qquad$ $\{\neg P(f(x)), R(g(f(x)),a)\}$

$\{\neg Q(z), P(z), R(g(f(x)),a)\}$

Es geht auch anders – mit denselben K_1 und K_2!

Vorgezogene Variablentrennung: ABKNF

Ohne die Variablentrennung der Klauseln würden uns einige Resolutionsmöglichkeiten verloren gehen – z.B. gleich im vorigen Beispiel!

Damit wir nicht bei jeder Suche nach Unifikationsmöglichkeiten die Klauseln erneut **variablenfremd** machen müssen, erledigen wir dies am besten einmal **vorab**, indem wir zur KNF-Formel eine (erfüllbarkeits-) äquivalente **auseinanderbenannte KNF-Formel** (**ABKNF**) erzeugen. Dann müssen wir für jeden Resolutionsschritt nur noch eine passende Unifikation suchen, **ohne umzubenennen**.

Begründung und Ergebnis im vorigen Beispiel:

$$\forall x \forall z\, [(P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge (\neg P(x) \vee R(g(x),a))]$$
$$\equiv\ \forall x \forall z\, (P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge \forall x \forall z\, (\neg P(x) \vee R(g(x),a))$$
$$\equiv\ \forall x \forall z\, (P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge \forall y \forall z\, (\neg P(y) \vee R(g(y),a))$$
$$\equiv\ \forall x \forall y \forall z\, (P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge \forall x \forall y \forall z\, (\neg P(y) \vee R(g(y),a))$$
$$\equiv\ \forall x \forall y \forall z\, [(P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge (\neg P(y) \vee R(g(y),a))]$$

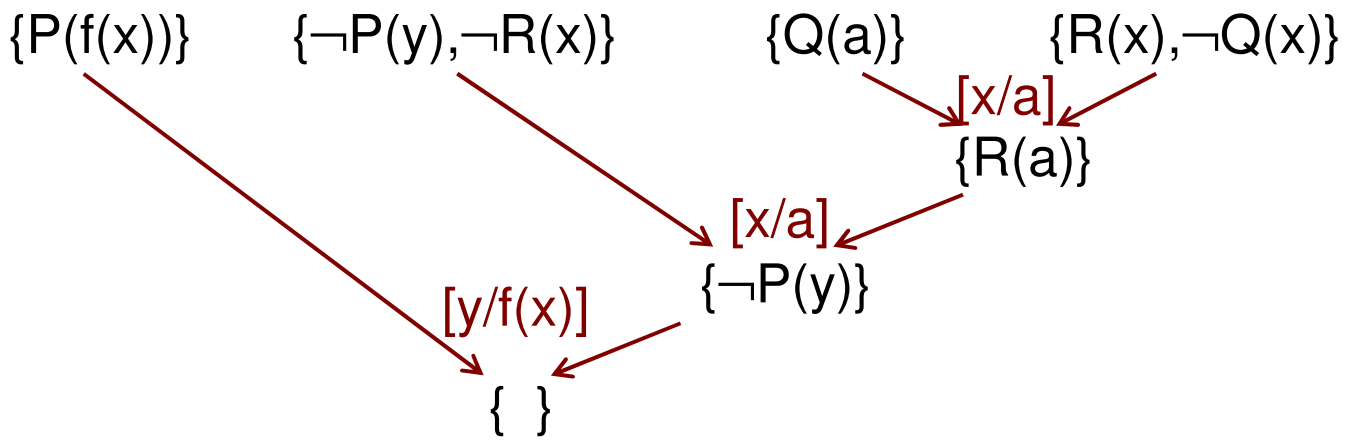
PL1-Resolution, Satz und Beispiel

Eine PL1-Formel ist genau dann **unerfüllbar**, wenn aus ihrer Klauselnormalform per PL1-**Resolution** die **leere Klausel** erzeugt werden kann.

Gegeben sei bereits in Kl-NF:

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

... unerfüllbar wegen ...



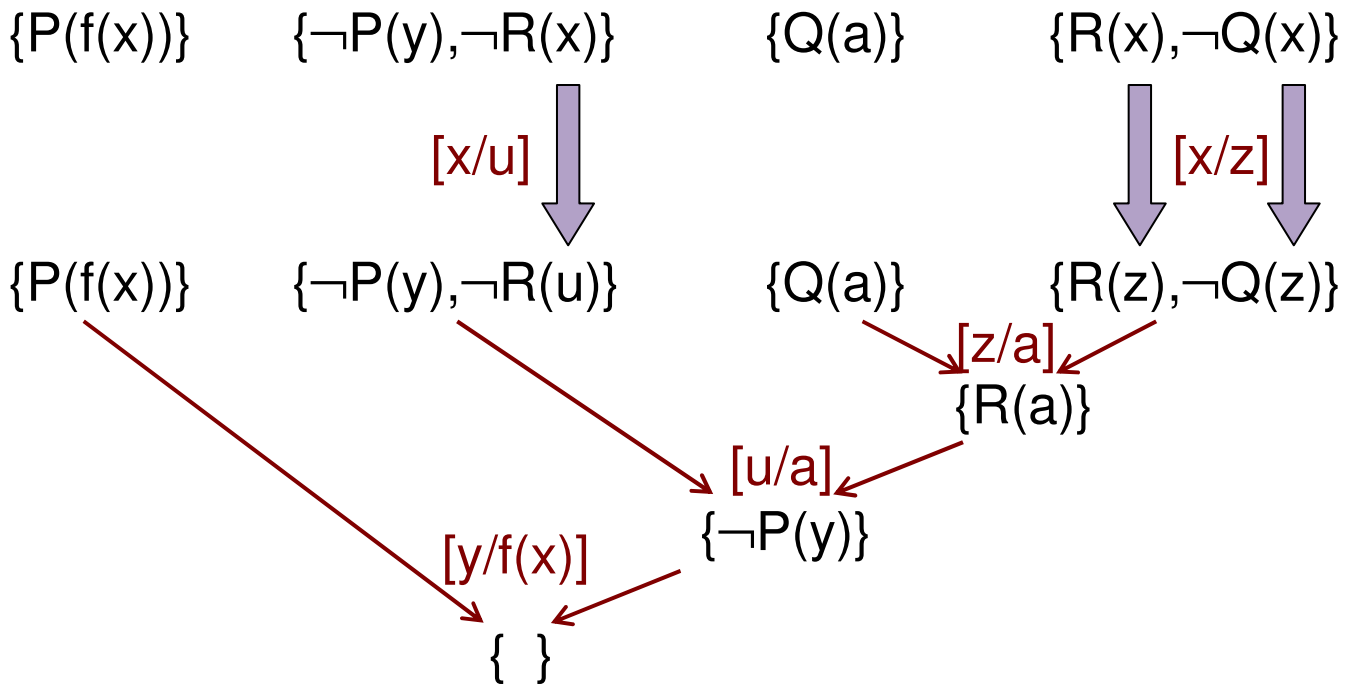
Und jetzt nochmal „richtig“:

Im vorigen Beispiel **fehlte** die **vorgezogene Variablentrennung**.

Aber wir hatten Glück: Die Resolution klappte – **zufällig** – sogar

- ohne Vorab-Variablentrennung bzw.
- nachträgliche Umbenennung!

Natürlich klappt sie dann erst recht **mit**:



Ü58

Ü59