# *Automatisches Beweisen*

## *– Kap 2.4 Prädikatenlogik –*


Prof. Dr. Wolfgang Küchlin

*Dipl.-Inform., Dr. sc. techn. (ETH)*

**Arbeitsbereich Symbolisches Rechnen**
**Wilhelm-Schickard-Institut für Informatik**
**Fakultät für Informations- und Kognitionswissenschaften**

**Universität Tübingen**

**Steinbeis Transferzentrum**
**Objekt- und Internet-Technologien (OIT)**

**Wolfgang.Kuechlin@uni-tuebingen.de**
**http://www-sr.informatik.uni-tuebingen.de**



SR

# Prädikatenlogik (PL1)



SR

# *Prädikatenlogik (PL1)*

- Sprache der Mathematik

- Neu im Vergleich zur Aussagenlogik
  - Funktions- und Relationssymbole *(predicate symbols)*
  - Existenz- und All-Quantoren
  - Atomare Formeln werden ersetzt durch Relationen (Prädikate) über Termen
  - Terme bezeichnen Individuen explizit
  - Es lassen sich unbeschränkt viele Terme bauen (z. Bsp. 0, f(0), f(f(0)), f(f(f(0))), …)



# *Syntax der Prädikatenlogik (1)*

- Elemente der Sprache:
  - Menge von (Individuen-) Variablen $\mathcal{V} = \{v_0, v_1, \ldots\}$
  - aussagenlogische Konstante(n) und Junktoren
  - Funktionssymbole: $\mathcal{F} = \{f, g, h, \ldots\}$ (Stelligkeit $\geq 0$)
  - Prädikatssymbole: $\mathcal{P} = \{R, S, T, \ldots\}$ (Stelligkeit $\geq 0$)

    (auch Relationssymbole genannt, dann stattd. $\mathcal{R} = \{R, S, T, \ldots\}$)
  - Quantoren: $\forall$, $\exists$
  - Hilfssymbole: Klammern, Komma



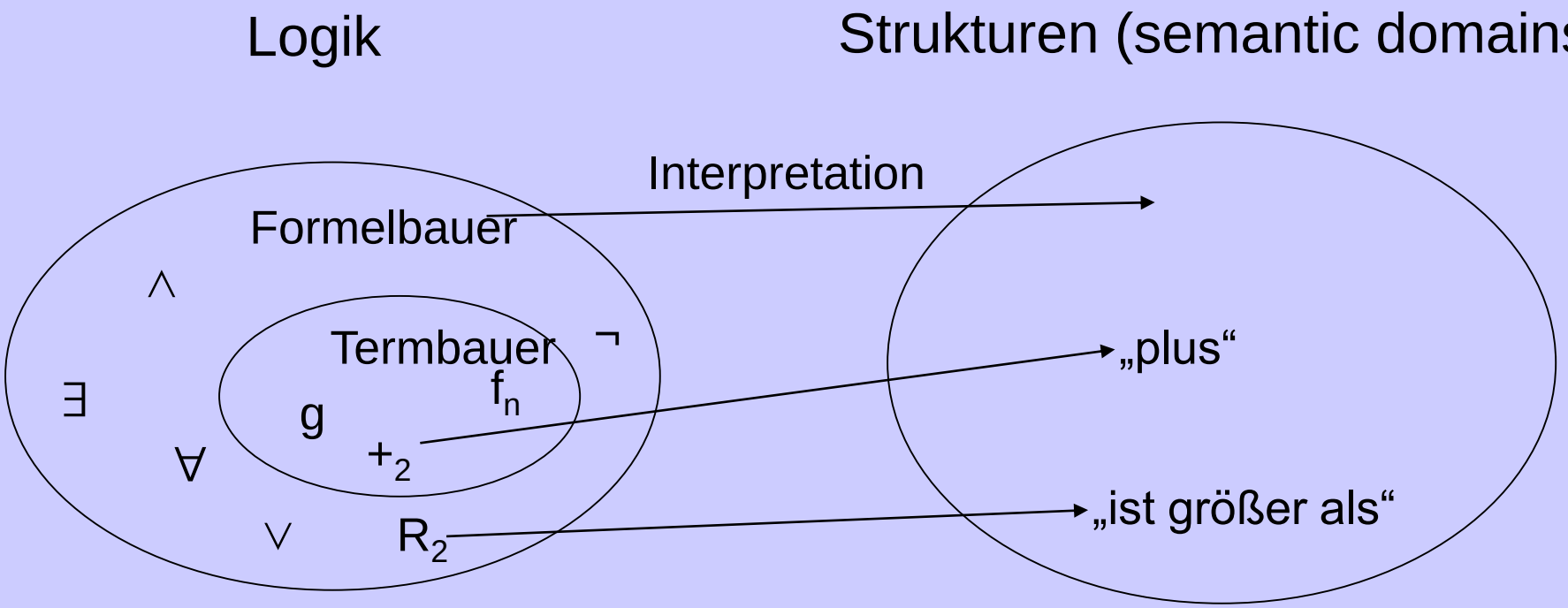
# *Syntax der Prädikatenlogik (2)*

- Terme T($\mathcal{V}$, $\mathcal{F}$): die kleinste Menge mit
  - $\mathcal{V} \subseteq$ T($\mathcal{V}$, $\mathcal{F}$)
  - Falls f$\in \mathcal{F}$ (mit Stelligkeit n) und $t_1$, …, $t_n \in$ T($\mathcal{V}$, $\mathcal{F}$), so auch $ft_1 \ldots t_n \in$ T($\mathcal{V}$, $\mathcal{F}$).
- Beispiel:
  - $\mathcal{V}$ = {x, y, z}
  - $\mathcal{F}$ = {c, f, g} (c 0-stellig, f 2-stellig, g 1-stellig, besser lesbar in Schreibweise $c_0$, $f_2$, $g_1$)
  - Terme: fxfgyc  oder gffcgxgz
  - Erweiterung mit Klammern: f(x, f(g(y), c)) oder g(f(f(c,g(x)),g(z)))
  - Achtung: Variablen stehen für Individuen, nicht mehr für Aussagen (anders als in der AL).

# *Syntax der Prädikatenlogik (3)*

- **Relationssymbole** $\mathcal{R}$ bezeichnen Relationen (Bool-wertige Funktionen)
- **Formeln** $\Phi(\mathcal{V},\mathcal{F},\mathcal{R})$: Definiert als kleinste Menge, so dass (schreibe $\Phi$ anstelle von $\Phi(\mathcal{V},\mathcal{F},\mathcal{R})$ )
  - $\bot \in \Phi$
  - Falls $R \in \mathcal{R}$(Stelligkeit n) und $t_1, \ldots, t_n \in T(\mathcal{V},\mathcal{F})$, so ist $Rt_1 \ldots t_n \in \Phi$.
    - dieses sind zusammen mit $\bot$ die **atomaren Formeln**
  - Falls $F,G \in \Phi$, so auch $(F \vee G) \in \Phi$, $(F \wedge G) \in \Phi$ und $\neg F \in \Phi$.
  - Falls $x \in \mathcal{V}$ und $F \in \Phi$, so auch $\exists x F \in \Phi$ und $\forall x F \in \Phi$.
    - Quantoren erstrecken sich auf die *ganze* folgende Formel: $\forall x\, P(x) \wedge Q(x)$.
    - Zur Lesbarkeit schreiben wir Klammern: $\forall x(P(x) \wedge Q(x))$ oder $\forall x[P(x) \wedge Q(x)]$
  - Beispiel: $\mathcal{V} = \{x, y\}$, $\mathcal{F} = \{f_2, g_1\}$, $\mathcal{R} = \{P_2, Q_2\}$
    Formel: $\forall x \exists y(Pxy \wedge Qfxgxy)$. Atomare Formel: $Qfxgxy$

# *Semantik der Prädikatenlogik*




Formelbauer: Symbole, mit denen Formeln gebaut werden
Termbauer: Symbole, mit denen Terme gebaut werden

# *Semantik der Prädikatenlogik (2)*

- Eine passende Semantic Domain benötigt Funktionen und Prädikate (Relationen) passend zu den Funktionssymbolen $\mathcal{F}$ und den Relationssymbolen $\mathcal{R}$.

- **$(\mathcal{F}, \mathcal{R})$-Struktur**:

  Tupel $(A,\mu)$, mit Universum $A \neq \{\}$ und Funktion

  $\mu$ (*meaning function*), die jedem $f_n \in \mathcal{F}$ eine

  n-stellige Funktion und jedem $R_m \in \mathcal{R}$ eine

  m-stellige Relation auf A zuweist.

# *Semantik der Prädikatenlogik (3)*

- **Variablenbelegung**:

  $\mathcal{V}$ eine Variablenmenge, (A,$\mu$) eine ($\mathcal{F}$, $\mathcal{R}$)-Struktur

  Eine Variablenbelegung $\beta$ ist eine Abbildung $\beta$: $\mathcal{V} \rightarrow$A.

- Notation: $\beta$[x/a]

  die an Stelle x auf a abgeänderte Funktion $\beta$

$$\beta[x/a](y) = \begin{cases} a & \text{falls } y = x, \\ \beta(y) & \text{sonst} \end{cases}$$

# *Semantik der Prädikatenlogik (4)*

- **Interpretation**

  Tupel $\mathcal{I} = (\mathcal{A}, \beta)$, bestehend aus $(\mathcal{F}, \mathcal{R})$-Struktur $\mathcal{A} = (A, \mu)$ und Variablenbelegung $\beta: \mathcal{V} \rightarrow A$ . Mit einer Interpretation können symbolische Terme als Individuen in A interpretiert werden und Formeln als Wahrheitswerte.

  - mit $\mathcal{I}' = \mathcal{I}[x/a]$ bezeichnen wir die Interpretation $\mathcal{I}' = (\mathcal{A}, \beta[x/a])$

- **Interpretation eines Terms**

  Sei $\mathcal{I} = (\mathcal{A}, \beta)$ eine Interpretation. Für Terme t ist $\mathcal{I}(t)$ rekursiv definiert durch:

  - $\mathcal{I}(t) = \beta(t)$ falls $t \in \mathcal{V}$.
  - $\mathcal{I}(ft_1 \ldots t_n) = \mu[f](\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n))$.

# *Beispiel zur Interpretation eines Terms*

- A=($\mathbb{N}$, $\mu$) mit
  - $\mu(f)$: $\mathbb{N}$ x $\mathbb{N}$ → $\mathbb{N}$: (x,y) ↦ x+y
  - $\mu(g)$: $\mathbb{N}$ → $\mathbb{N}$ : x ↦ x+1
  - $\mu(P) \subseteq \mathbb{N}^2$: (x,y) ∈ $\mu(P)$ gdw x=y
  - $\mu(Q) \subseteq \mathbb{N}^2$ : (x,y) ∈ $\mu(Q)$ gdw x<y
- $\beta(x)=2$

$$I(\text{fxgx}) = \mu[f](I(x), I(gx)) = \beta(x)+\mu[g](I(x))$$

$$= 2+(2+1) = 5$$

# *Semantik der Prädikatenlogik (5)*

- **Erfüllbarkeitsrelation**

  Sei $\mathcal{I}=(\mathcal{A}, \beta)$ Interpretation, F Formel.

  $\models_{\mathcal{I}} F$ definiert durch:

  - $\not\models_{\mathcal{I}} \bot$
  - $\models_{\mathcal{I}} Rt_1 \ldots t_n$ gdw. $(\mathcal{I}(t_1),\ldots,\mathcal{I}(t_n)) \in \mu(R)$
  - $\models_{\mathcal{I}} F \vee G$ gdw. $\models_{\mathcal{I}} F$ oder $\models_{\mathcal{I}} G$
  - $\models_{\mathcal{I}} F \wedge G$ gdw. $\models_{\mathcal{I}} F$ und $\models_{\mathcal{I}} G$
  - $\models_{\mathcal{I}} \neg F$ gdw. $\not\models_{\mathcal{I}} F$
  - $\models_{\mathcal{I}} \forall x F$ gdw. $\models_{\mathcal{I}[x/a]} F$ für alle $a \in \mathcal{A}$
  - $\models_{\mathcal{I}} \exists x F$ gdw. es gibt ein $a \in \mathcal{A}$ mit $\models_{\mathcal{I}[x/a]} F$

# *Sprechweisen*

- Für $\models_I$ F sagen wir
  - $I$ erfüllt F *(satisfies* F*, validates* F*)*
  - F gilt unter $I$ *(is valid under* $I$*)*
  - F ist wahr unter $I$ *(is valid under* $I$*)*
  - $I$ ist ein Modell von F ($I$ models F)
- Existiert ein $I$, so dass $\models_I$ F, so heißt F erfüllbar.
- Gilt $\models_I$ F für alle $I$, so heißt F allgemeingültig, $\models$F
- G $\models_I$ F bedeutet: Falls $\models_I$ G, dann auch $\models_I$ F
- Folgendes ist möglich:
  - $\not\models$ F (im allgemeinen), aber $\models_I$ F (im speziellen $I$)
  - G $\not\models$ F (im allgemeinen), aber G $\models_I$ F (im speziellen $I$)



# *Substitutionen (1)*

## ➢ Freie / gebundene Variablen

- Die Quantoren ∃ und ∀ binden Variablen.

- Fr(F) = Menge der freien Variablen von F
- Bd(F) = Menge der gebundenen Variablen von F

- Beispiel: F = ∀x (∃y Pxyz ∨ Qfu) ∧ ∃z Rax

  Fr(F) = {z, u, x}

  Bd(F) = {x, y, z}



# Substitutionen (2)

- Die simultane Substitution $t[x_1,\ldots,x_r / t_1,\ldots,t_r]$ bzw. $F[x_1,\ldots,x_r / t_1,\ldots,t_r]$ ist für Terme t, $t_1$, …, $t_r$, paarweise verschiedene Variablen $x_1$, …, $x_r$ und Formeln F rekursiv definiert.
  - Basis:
    - $x[x_1,\ldots,x_r/t_1,\ldots,t_r] = \begin{cases} x & \text{falls } x \notin \{x_1,\ldots,x_r\} \\ t_i & \text{falls } x = x_i \end{cases}$
    - $f_0[x_1,\ldots,x_r/t_1,\ldots,t_r] = f_0$
    - $f(y_1,\ldots,y_k)[x_1,\ldots,x_r/t_1,\ldots,t_r] = f(y_1[x_1,\ldots,x_r/t_1,\ldots,t_r], \ldots, y_k[x_1,\ldots,x_r/t_1,\ldots,t_r])$
    - $R(s_1,\ldots,s_k)[x_1,\ldots,x_r/t_1,\ldots,t_r] = R(s_1[x_1,\ldots,x_r/t_1,\ldots,t_r], \ldots, s_k[x_1,\ldots,x_r/t_1,\ldots,t_r])$
  - $(G \circ H)[x_1,\ldots,x_r/t_1,\ldots,t_r] = G[x_1,\ldots,x_r / t_1,\ldots,t_r] \circ H[x_1,\ldots,x_r / t_1,\ldots,t_r]$, für aussagenlog. Junktoren $\circ$ , analog für Negation $\neg$
  - $(QxF)[x,x_1,\ldots,x_r / t,t_1,\ldots,t_r] = Qu(F[x,x_1,\ldots,x_r / u,t_1,\ldots,t_r])$

    u neue Variable, die nicht in $t_1,\ldots,t_r$ und nicht in F vorkommt. Falls x nicht in $t_1,\ldots,t_r$ vorkommt kann u=x gewählt werden

    (beschränke die Subst. auf Fr(F) und benenne x in neue Variable u um, falls x in einem der Terme $t_i$ vorkommt)

# *Beispiele zur Substitution*

- fyz[y,z,u / z,x,y] = fzx
- ∀x Pxy [y/x] = ∀u(Pxy[x,y/u,x]) = ∀u(Puy[y/x]) = ∀u Pux
- (∃x Pxfyz)[x,z / u,fyy] = ∃x Pxfyfyy



# *Normalformen*

- Wie schon die AL-Resolution benötigt auch die PL-Resolution eine Formel in Normalform: Klausel-Form
- Eine geschlossene Formel ist in **Klausel-Form**, falls sie von der Bauart

$$Qx_1 \ldots x_n : M$$

ist. Hierbei ist $Qx_1 \ldots x_n$ ein **Präfix** aus allquantifizierten Variablen und M ist eine quantorfreie **Matrix** in konjunktiver Normalform.

- Satz (Skolem): Zu jeder geschlossenen Formel A existiert eine erfüllbarkeits-äquivalente Formel A* in Klausel-Form, also A* ≅ A.
- Wir beschränken uns im Folgenden auf geschlossene Formeln (ohne freie Variablen). Falls zunächst eine freie Variable vorkommt, wird diese gemäß unserer Intention quantifiziert.
  - Bsp: ∀x Pxy. Wollen wir beweisen, dass dies für alle y gilt, oder dass es ein y gibt, sodass dies gilt?



# *Normalformen*

- **Negationsnormalform** (NNF)

  Formel ist in NNF, wenn ¬ nur noch vor Relationssymbolen vorkommt (oder vor ⊥ falls T nicht in der Logik vorhanden).

- Algorithmus:
  - $\neg\forall x\, F \quad \equiv> \quad \exists x\, \neg F$
  - $\neg\exists x\, F \quad \equiv> \quad \forall x\, \neg F$
  - Aussagenlogische NNF-Transformationen



# *Normalformen*

- **Pränexe-Normalform** (PNF, *prenex normal form*)

  Formel ist in PNF, falls sie von der folgenden Form ist:

  $$Q_1 x_1 \ldots Q_n x_n \; F_0$$

- Algorithmus
  - $F \vee \exists x G \equiv> \exists y(F \vee G[x/y])$, wobei $y \notin Var(F) \cup Fr(G)$
  - $F \wedge \exists x G \equiv> \exists y(F \wedge G[x/y])$, wobei $y \notin Var(F) \cup Fr(G)$
  - $F \wedge \forall x G \equiv> \forall y(F \wedge G[x/y])$, wobei $y \notin Var(F) \cup Fr(G)$
  - $F \vee \forall x G \equiv> \forall y(F \vee G[x/y])$, wobei $y \notin Var(F) \cup Fr(G)$
- F ist in **Pränex-CNF (PCNF)**, falls $F_0$ in CNF
- Algorithmus: Distributivgesetz anwenden



# *Normalformen*

- **Skolem-Normalform** (SNF)

  Formel ist in SNF, wenn sie in PCNF ist und wenn ihr Präfix nur universelle Quantoren enthält.

- Algorithmus:
  - $\forall x_1 \ldots \forall x_k \exists x_{k+1} Q_{k+2} x_{k+2} \ldots Q_n x_n F_0 \;\; \cong >$
    $\forall x_1 \ldots \forall x_k Q_{k+2} x_{k+2} \ldots Q_n x_n (F_0[x_{k+1} / f x_1 \ldots x_k])$

    wobei f neues k-stelliges Funktionssymbol ist, das eine Skolem-Funktion bezeichnet, die zu jeder Kombination $x_1 \ldots x_k$ einen der für $x_{k+1}$ existierenden Werte auswählt.

# *Normalformen - Beispiel*

- $F = \exists x \forall y\, (Rxy \wedge \neg \exists z \forall u\, (Rzu))$
- NNF: $\exists x \forall y\, (Rxy \wedge \forall z \exists u\, (\neg Rzu))$
- PNF: $\exists x \forall y \forall z \exists u\, (Rxy \wedge \neg Rzu)$
- SNF: $\forall y \forall z (Rc_0 y \wedge \neg Rzf_2 yz)$



# *Normalformen*

- Einführung von Skolemfkt. erhält nur die Erfüllbarkeit
- Je nachdem, wie die Quantoren extrahiert wurden, bekommt man unterschiedliche Skolemfunktionen
  - NNF: $\exists x \forall y\ (Rxy \wedge \forall z \exists u\ \neg Rzu)$
  - PNF1: $\exists x \forall y \forall z \exists u\ (Rxy \wedge \neg Rzu)$
  - SNF1: $\forall y \forall z (Rc_0 y \wedge \neg Rzf_2 yz)$
  - NNF: $\exists x \forall y\ (Rxy) \wedge \forall z \exists u\ (\neg Rzu)$
  - PNF2: $\forall z \exists u \exists x \forall y\ (Rxy \wedge \neg Rzu)$
  - SNF2: $\forall z \forall y (Rf_1 zy \wedge \neg Rzg_1 z)$
- Auch die Reihenfolge der Skolemisierung hat Einfluss
  - $\exists x\ \exists y(G(f(x,y),1) \rightarrow \exists y(G(f(c_0,y),1) \rightarrow G(f(c_0,d_0),1)$
  - $\exists x\ \exists y(G(f(x,y),1) \rightarrow \exists x(G(f(x,g(x)),1) \rightarrow G(f(c_0\ ,\ g(c_0),1)$



# *Zusammenfassung: Klausel-Form*

- empfohlene Transformationsschritte zur Klausel-Form
  - Gebundene Variablen umbenennen (separieren)
  - Abgeleitete aussagenlog. Operatoren durch $\vee$, $\wedge$, $\neg$ ersetzen
  - NNF herstellen ($\neg$ nach innen schieben)
  - PCNF herstellen
  - $\exists$-Quantoren von außen nach innen eliminieren (Skolemfunktionen einführen)
- Am Beispiel
  - NNF: $\exists x \forall y\,(Rxy) \wedge \forall z \exists u\,(\neg Rzu)$
  - PCNF: $\forall z \exists u\ \exists x \forall y\,(Rxy \wedge \neg Rzu)$
  - Skolemisierung u: $\forall z\ \exists x \forall y\,(Rxy \wedge \neg Rzg_1 z)$
  - Skolemisierung x: SNF: $\forall z \forall y\,(Rh_1 zy \wedge \neg Rzg_1 z)$

# *Unifikation*

- AL-Resolution arbeitet auf komplementären Literalen
- Für PL-Resolution müssen komplementäre Literale i.A. durch **Unifikation** hergestellt werden.
- Beispiel: {{P(x, f(x,y))}, {¬P(g(c), f(z,c))}}
  - Zunächst keine komplementären Literale vorhanden
  - Wegen Klausel-Form sind x, y, z all-quantifiziert
  - Formel gilt also auch für x↦g(c), z↦g(c), y↦c, also im Spezialfall: {{P(g(c), f(g(c),c))}, {¬P(g(c), f(g(c),c))}}
  - Jetzt ist Resolution anwendbar und liefert □
- Der Unifikations-Algorithmus sucht eine *allgemeinste* Substitution, die einen gemeinsamen Spezialfall liefert



# *Unifikation*

- Hintergrund:
  syntaktisches Lösen von Termgleichungs-systemen

- Beispiel: {f(x,y)=z, g(y)=g(g(x))}
  y↦g(x), z↦f(x,g(x)) oder
  x↦a, y↦g(a), z↦f(a,g(a))

# *Unifikation*

- Substitutor / Substitution:

  Abbildung $\sigma: \mathcal{V} \rightarrow T(\mathcal{V}, \mathcal{F})$, mit $\sigma(x)=x$ für fast alle x

  - normalerweise in Postfix geschrieben: $x\sigma$
  - Grund-Substitution: Abbildung auf variablen-freie (Grund-)Terme

- Umbenennung:

  bijektiver Substitutor

- Separator von $K_1$ und $K_2$:

  Umbenennung $\xi$ mit $Fr(K_1\xi) \cap Fr(K_2) = \{\ \}$



# *Unifikation*

- Unifikator einer Literalmenge $\mathcal{L}$ :

  Substitutor $\sigma$ mit $\mathcal{L}\sigma$ einelementig

  ($\mathcal{L}$ heißt unifizierbar, falls es einen Unifikator gibt.)

- allgemeinster Unifikator (mgu) von $\mathcal{L}$:

  Unifikator $\mu$, so dass es für jeden anderen Unifikator $\nu$ einen Substitutor $\sigma$ gibt mit $\mu\sigma=\nu$

  (d.h. für alle x gilt: $\sigma(\mu(x))=\nu(x)$) )



# *Unifikation - Beispiele*

- {x, a} und {R(x,g(y)), R(g(a),g(a))} jeweils unifizierbar
- {R(y,y), R(g(x),x)} und { } jeweils nicht unifizierbar
- {P(f(x,y),g(y)), P(z,g(g(x)))} unifizierbar mit $\nu$={y↦g(a), z↦f(x,g(x)), x↦a}
- {R(x,g(y)), R(u,v)} unifzierbar mit
  $\mu$={x↦u, v↦g(y)} bzw. $\mu$‘={u↦x, v↦g(y)}



# *Unifikationsalgorithmus nach J.R. Robinson*

1. Falls in L verschiedene Prädikatssymbole auftauchen, STOP mit „L ist nicht unifizierbar.“
2. $i:=0$; $\mu_i:=id$;
3. Falls $L\mu_i$ einelementig, STOP mit „$\mu_i$ ist mgu“
4. Wähle $F_1$, $F_2$ aus $L\mu_i$ mit $F_1 \neq F_2$. Seien $s_1$ und $s_2$, die ersten unterschiedlichen Symbole.

   Falls $s_1$ und $s_2$ Funktionssymbole, STOP mit „L ist nicht unifzierbar.“
5. Falls $s_1$ Variable, bestimme Term t in $F_2$, der an Position von $s_2$ beginnt. // Falls $s_2$ Variable, entsprechendes mit $s_2$ und $F_1$
6. Falls $s_1 \in t$, *// (occurrence check)*

   STOP mit „L ist nicht unifizierbar“
7. $\mu_{i+1}:=\mu_i\{s_1 \mapsto t\}$; $i:=i+1$;
8. Goto 3;

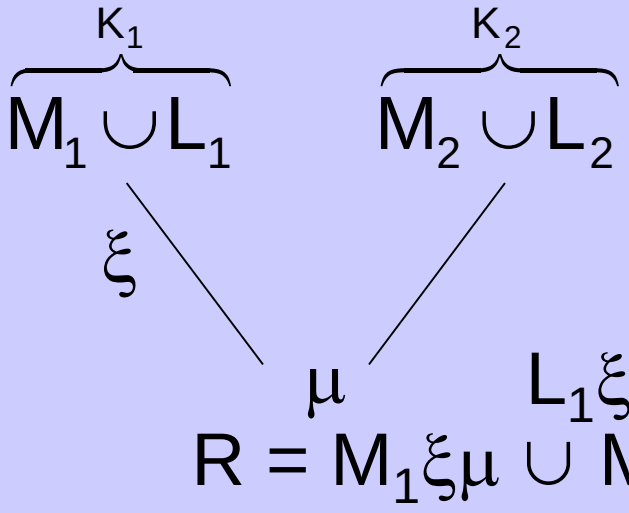
# *Unifikationsalgorithmus nach J.R. Robinson*

- Der Robinson Unifikations-Algorithmus liefert nicht unbedingt eine simultane Substitution
- Falls Variablen in der Datenstruktur eindeutig repräsentiert sind, also nur einmal pro Klausel (z.B. in einer Symboltabelle), und die Substitution dort mit Verweis (Zeiger) vermerkt wird, kann ein simultaner Substitutor abgelesen werden
  - An jedem Ort, an dem eine Variable vorkommt, wird auf den Eintrag in der Symboltabelle verwiesen und dort wird die Substitution vermerkt.
- Andernfalls muss jede neue Substitution auf die Terme der bereits vorhandenen Substitutionen angewendet werden.
- Beispiel von oben:
  - {P(f(x,y),g(y)), P(z,g(g(x)))} unifizierbar.
  - Robinson: $\nu=\{z\mapsto f(x,y),\ y\mapsto g(x)\}$
  - Die beiden Substitutionen sind nicht simultan sondern sequentiell (hintereinander) zu lesen
  - Simultaner Substitutor: $\sigma=\{z\mapsto f(x,g(x)),\ y\mapsto g(x)\}$
  - Es musste $\{y\mapsto g(x)\}$ auf f(x,y) angewendet werden



# *Prädikatenlogische binäre Resolution*

- Klauseln $K_1 = M_1 \cup L_1$ und $K_2 = M_2 \cup L_2$. (Mit Literalen $L_1$, $L_2$ )



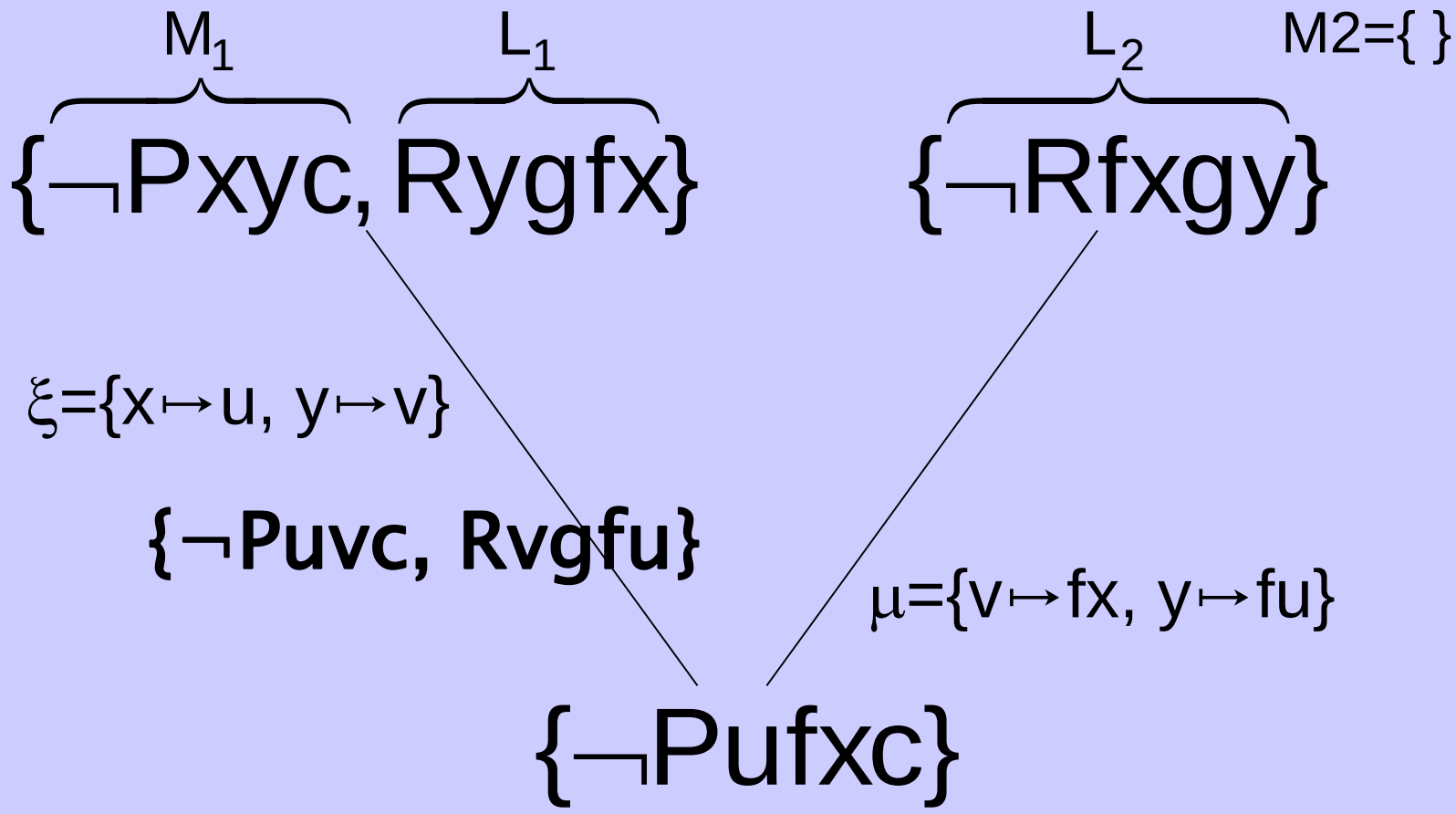

$\xi$ Separator von $K_1$ und $K_2$

$L_1\xi \cup \neg L_2$ unifizierbar mit mgu $\mu$

$R = M_1\xi\mu \cup M_2\mu$

- Dies ist die **binäre Resolution**

# *Beispiel PL1-Resolution (Spezialfall: binär)*



# *Allgemeine Prädikatenlogische Resolution*

- Die **allgemeine Resolution** benutzt statt Literalen $L_1$, $L_2$ zwei Literalmengen $m_1$, $m_2$, mit $m_1\sigma =: L_1$ und $m_2\sigma =: L_2$ und $L_1\xi\tau = \neg L_2\tau$
  - Die Mengen $m_1$ und $m_2$ unifizieren jeweils zu Literalen $L_1$ und $L_2$.
  - Des weiteren unifizieren $L_1\xi$ und $\neg L_2$
  - Wenn man vorab separiert ist $\sigma\tau=\mu$ insgesamt ein Unifikator für die Literalmengen (bis auf die Negationszeichen)
- Der erste Teil $m_1\sigma =: L_1$ und $m_2\sigma =: L_2$ heißt auch **Faktorisierung**
- Faktorisierung + binäre Resolution haben die gleiche Stärke wie Allgemeine Resolution



# *Beispiel PL1-Resolution (allgemein)*

- Im Allgemeinen ist binäre Resolution nicht ausreichend

Bsp.:
**{{¬ p(x,x), ¬ p(c,x)},**
**{ p(y,y), p(c,y)}}**

Die binären Resolventen sind alles Tautologien:

1. **{¬ p(c,y), p(c,y)} //res. 1.Lit. oben mit 1.Lit. unten**
2. **{¬ p(c,c), p(c,c)} //res. 1.Lit. oben mit 2.Lit. unten**
3. **{¬ p(c,c), p(c,c)} //res. 2.Lit. oben mit 1.Lit. unten**
4. **{¬ p(y,y), p(y,y)} //res. 2.Lit. oben mit 2.Lit. unten**

Es gibt aber einen Unifikator $\mu=\{x\mapsto c, y\mapsto c\}$, sodass
**{¬ p(x,x), ¬ p(c,x)}** $\mu$ **= {¬ p(c,c)} und**
**{ p(y,y), p(c,y)}** $\mu$ **= {p(c,c)}**
**mit Resolvente** $= \square$

# *Korrektheit und Vollständigkeit der PL-Resolution*

- Korrektheit:

  $F \vdash_{Res} C$ impliziert $F \models C$

  - relativ leicht zu zeigen. Beide Elternklauseln müssen in allen Modellen gelten. Da die komplementären Literale nicht gleichzeitig gelten können, muss die Resolvente gelten.

- Widerlegungsvollständigkeit:

  $F \models \bot$ impliziert $F \vdash_{Res} \square$.

  - Beweis ist aufwändig. Die Prädikatenlogischen Modelle sind komplex, da Funktions- und Prädikatssymbole durch beliebige Funktionen und Relationen passender Stelligkeit interpretiert werden können.
  - Man zeigt zuerst: Herbrand-Modelle sind allgemeinste Modelle
    - Es gibt ein Modell gdw. es gibt ein Herbrand Modell.
  - Man zeigt danach: Falls es kein Herbrand-Modell gibt, folgt $F \vdash_{Res} \square$.



# *Herbrand Universum HU(S)*

- Das **Herbrand Universum** HU(S) ist eine allgemeinste Domäne für eine Interpretation der Terme
- Sei S eine Menge von Klauseln in SNF.

  HU(S) ist wie folgt definiert
  - Alle Konstanten-Symbole von S sind in HU(S).
    - Gibt es ein solches nicht, sei ein beliebiges Symbol a in HU(S). Hierdurch beschränkt man sich o.B.d.A. auf nicht-leere Domänen.
  - Falls Terme $t_1, \ldots, t_n$ in HU(S), so auch $f(t_1,\ldots,t_n)$ in HU(S) für jedes n-stellige Funktionssymbol f in $\mathcal{F}$.
  - Der Term fab als Zeichenreihe wird durch den Syntaxbaum f(a,b) interpretiert
  - HU(S) enthält Terme, die die Elemente der Interpretations-Domäne *(semantic domain elements)* darstellen. Falls ein Funktionssymbol (mit Stelligkeit > 0) existiert, ist HU(S) bereits unendlich.



# *Herbrand Basis HB(S)*

- Die **Herbrand Basis** HB(S) enthält alle Grund-Instanzen aller atomaren Formeln aus S. (Einsetzen des Herbrand-Universums in die Variablen.)
  - Manchmal auch: alle atomaren Formeln, die sich mit den Prädikatssymbolen und Termen aus HU(S) bilden lassen

- Beispiel:

  $S := \{P(x) \vee Q(y), \neg P(a), \neg Q(b)\}$

  $HU(S) = \{a, b\}$

  $HB(S) = \{P(a), P(b), Q(a), Q(b)\}$



# *Herbrand Interpretation*

- Eine minimale (allgemeine) Interpretation der Formel S
  - i.A. sind immer noch viele solche möglich
- Die symbolischen Terme werden durch die Elemente von HU(S) interpretiert.
- Interpretation der Funktionssymbole wird erweitert um (minimale) Interpretation der Prädikatssymbole, sodass die Elemente von HB(S) auf 1 oder 0 abgebildet werden
  - Jede Interpretation der Prädikatssymbole durch Relationen muss jeder atomaren Formel aus HB(S) einen Wert zuordnen
  - Hierfür gibt es viele Möglichkeiten, also gibt es auch viele Herbrand-Interpretationen. Diese unterscheiden sich aber nur in der Zuordnung von 1 bzw. 0 zu den Elementen von HB(S).

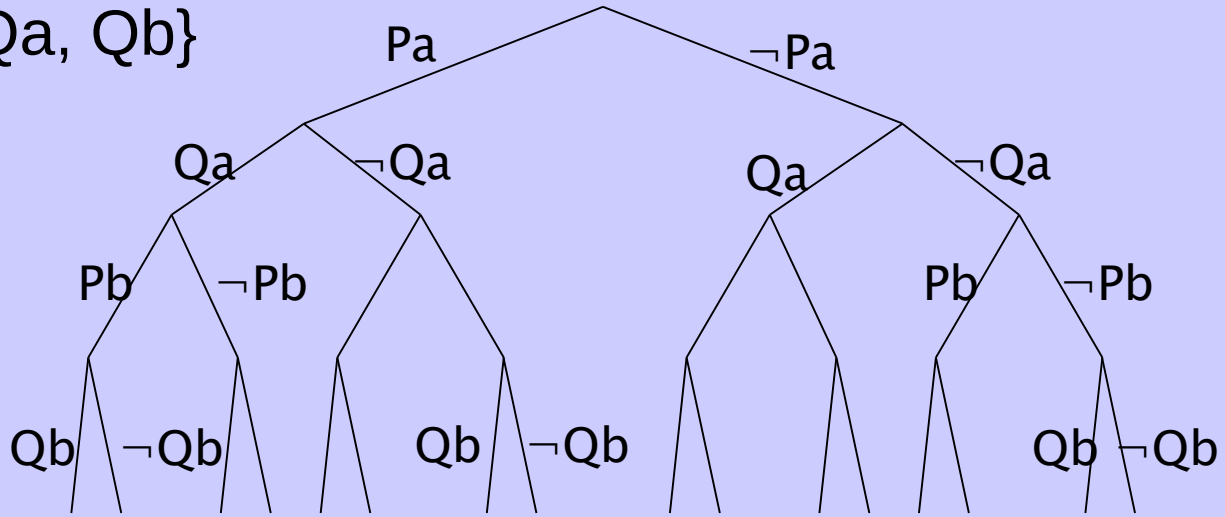


# *Semantische Bäume*

- Ein **Semantischer Baum** **ST(S)** repräsentiert alle möglichen Herbrand-Interpretationen von S.
  - Jeder von der Wurzel ausgehende Pfad ist eine mögliche H-Interpretation
  - Je nach Aufzählung von HB(S) ein anderer (äquivalenter) Baum
  - Im Allgemeinen ist der Baum unendlich
- Beispiel (endlich): S={{Px v Qy}, {¬Pa}, {¬Pb}},

  HB(S)={Pa, Pb, Qa, Qb}

  HU(S)={a, b}



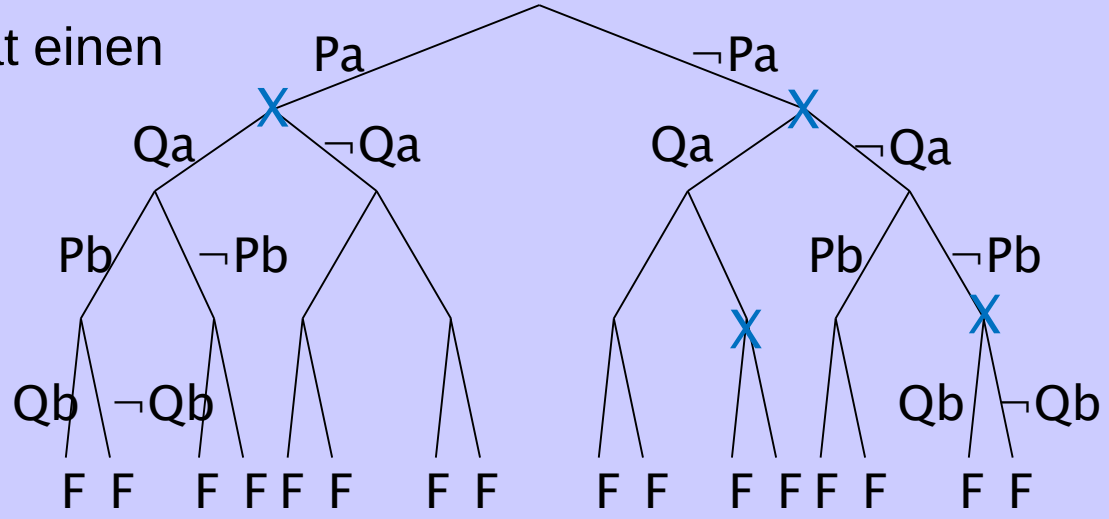
# *Herbrand-Modell*

- Eine Herbrand-Interpretation, durch die S erfüllt wird, ist ein Herbrand-Modell von S.
- Ein Herbrand Modell von S ist eindeutig charakterisiert durch diejenigen Elemente der Herbrand-Basis, deren Wert =1 ist.
- Eine Klauselmenge S hat genau dann ein Modell, wenn sie ein Herbrand-Modell hat.
  - „=>“: Das benötigte Herbrand-Modell besteht aus denjenigen Elementen von HB(S), die im Modell zu 1 evaluieren.
  - Es ist essentiell, dass S skolemisiert wurde, sodass die nötigen Elemente in HU(S) und damit in HB(S) vorhanden sind.
  - Korollar: kein Modell ⇔ kein Herbrand-Modell



# *Semantische Bäume und Herbrand-Modelle*

- Eine Interpretation (=Pfad) kann Modell sein oder nicht.
  - Sie ist ein Modell, wenn sie alle Klauseln in S erfüllt
  - Sie ist kein Modell, wenn sie (mind.) eine Klausel K falsifiziert.
    - Es gibt einen **Fehlerknoten** endlicher Tiefe mit einer **Fehlerklausel,** die durch die endliche Teilmenge von HB(S) von der Wurzel zum Fehlerknoten falsifiziert wird
    - Ein Pfad mit Fehlerknoten heißt **geschlossen**, sonst **offen**
    - Ein geschlossener Pfad hat einen höchsten Fehlerknoten




- Beispiel 1

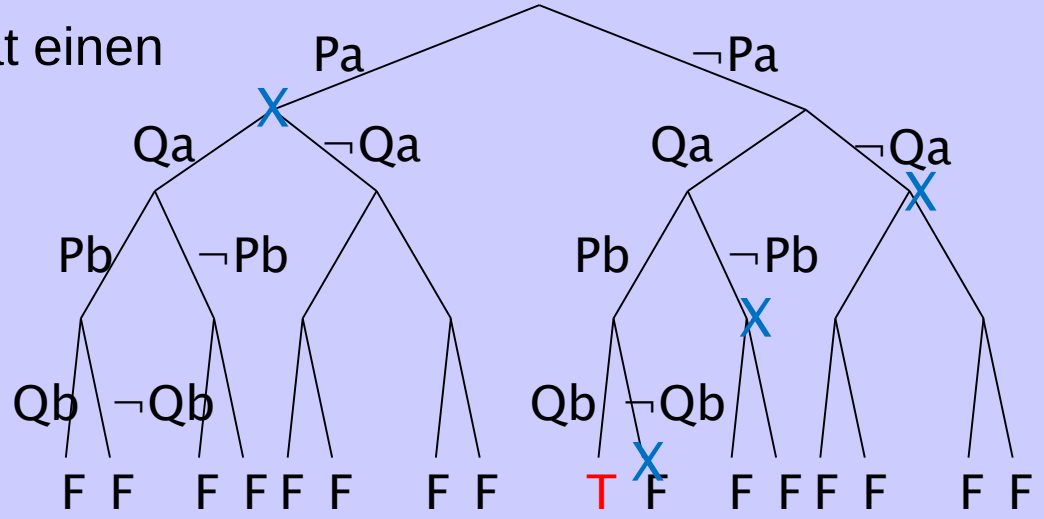
  S={{Px}, {Qy}, {¬Pa}, {Pb}},

  HB(S)={Pa, Pb, Qa, Qb}

  HU(S)={a, b}

# *Semantische Bäume und Herbrand-Modelle*

- Eine Interpretation (=Pfad) kann Modell sein oder nicht.
  - Sie ist ein Modell, wenn sie alle Klauseln in S erfüllt
  - Sie ist kein Modell, wenn sie (mind.) eine Klausel K falsifiziert.
    - Es gibt einen **Fehlerknoten** endlicher Tiefe mit einer **Fehlerklausel,** die durch die endliche Teilmenge von HB(S) von der Wurzel zum Fehlerknoten falsifiziert wird
    - Ein Pfad mit Fehlerknoten heißt **geschlossen**, sonst **offen**
    - Ein geschlossener Pfad hat einen höchsten Fehlerknoten




- Beispiel 2

  S={{Px v Qy}, {¬Pa}, {Pb}},

  HB(S)={Pa, Pb, Qa, Qb}

  HU(S)={a, b}

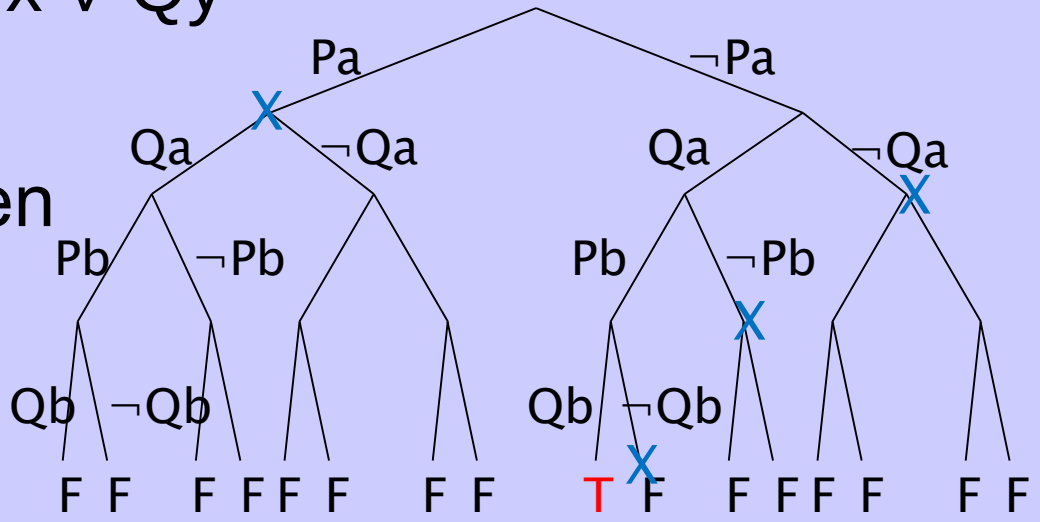
# *Einschub: Allquantifizierung*

- Es gilt: $\forall x Px \vee \forall y Qy \equiv \forall x \forall y\ (Px \vee Qy)$
- Beispiel 2: $S=\{\{Px \vee Qy\}, \{\neg Pa\}, \{Pb\}\}$,

  $HB(S)=\{Pa, Pb, Qa, Qb\}$, $HU(S)=\{a, b\}$

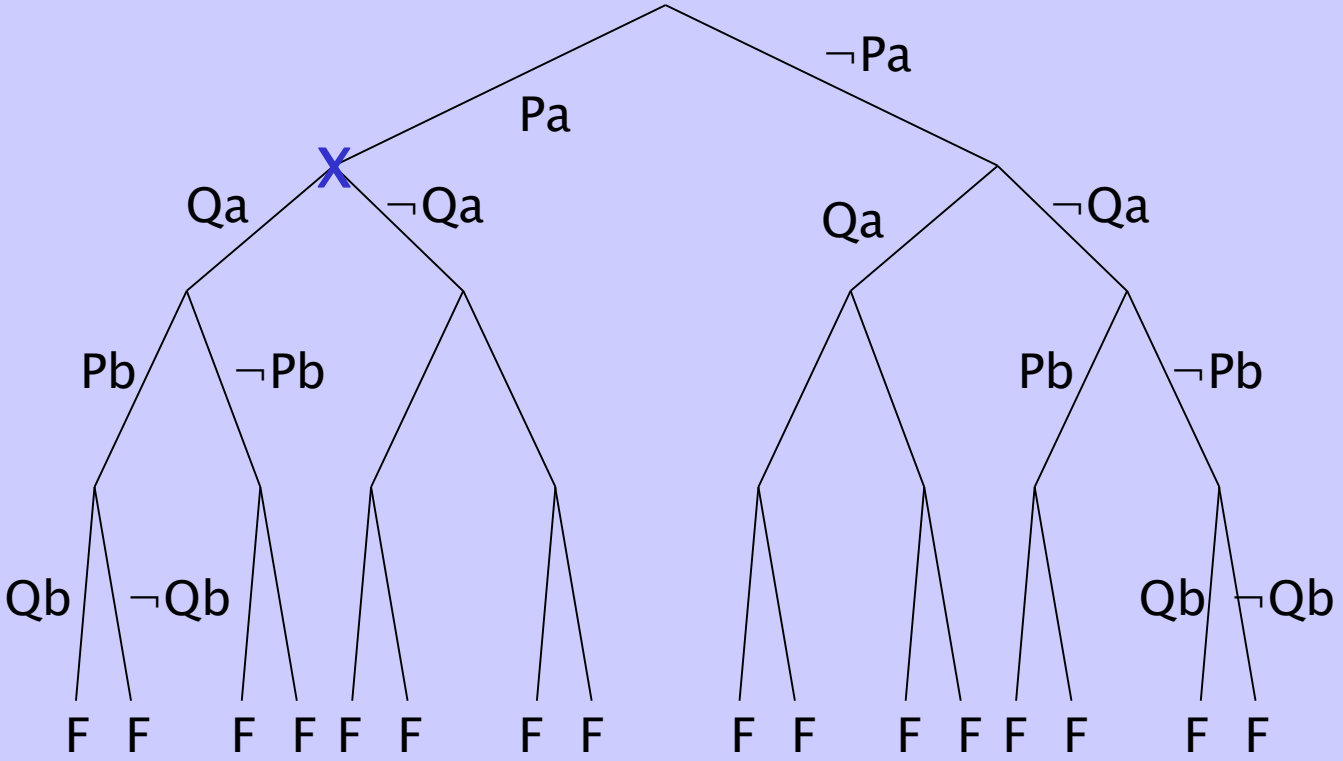
  $\forall x \forall y\ (Px \vee Qy) \equiv \forall y\ (Pa \vee Qy) \wedge \forall y\ (Pb \vee Qy)$

  $\equiv [(Pa \vee Qa) \wedge (Pa \vee Qb)] \wedge [(Pb \vee Qa) \wedge (Pb \vee Qb)]$
- Jede Kombination von $Px \vee Qy$ muss gelten
- $\forall x Px \vee \forall y Qy$ muss gelten
  - ansonsten Kombination der Gegenbeisp. möglich



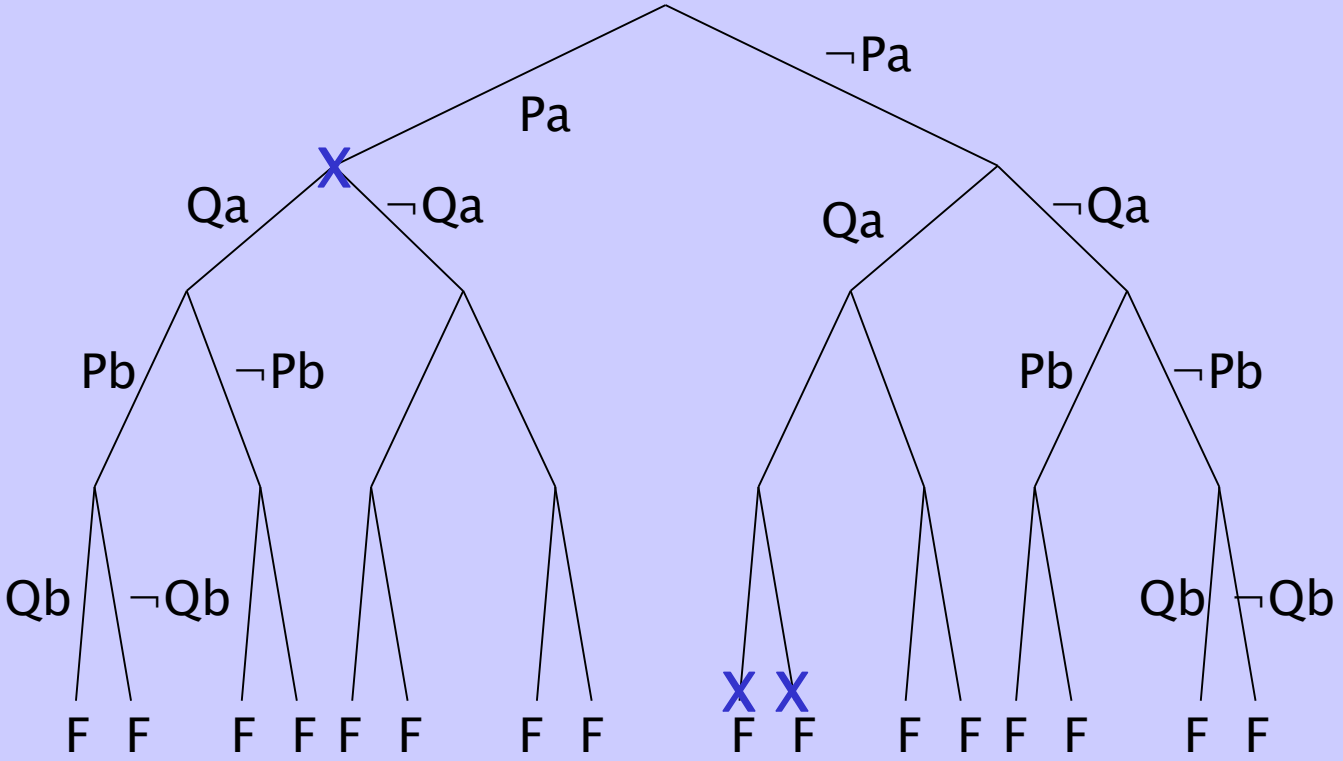
# *Semantische Bäume, Fehlerknoten*

- Bsp. 3: S={{Px ∨ Qy}, {¬Pa}, {¬Qb}}, HU(S)={a, b}, HB(S)={Pa, Pb, Qa, Qb}



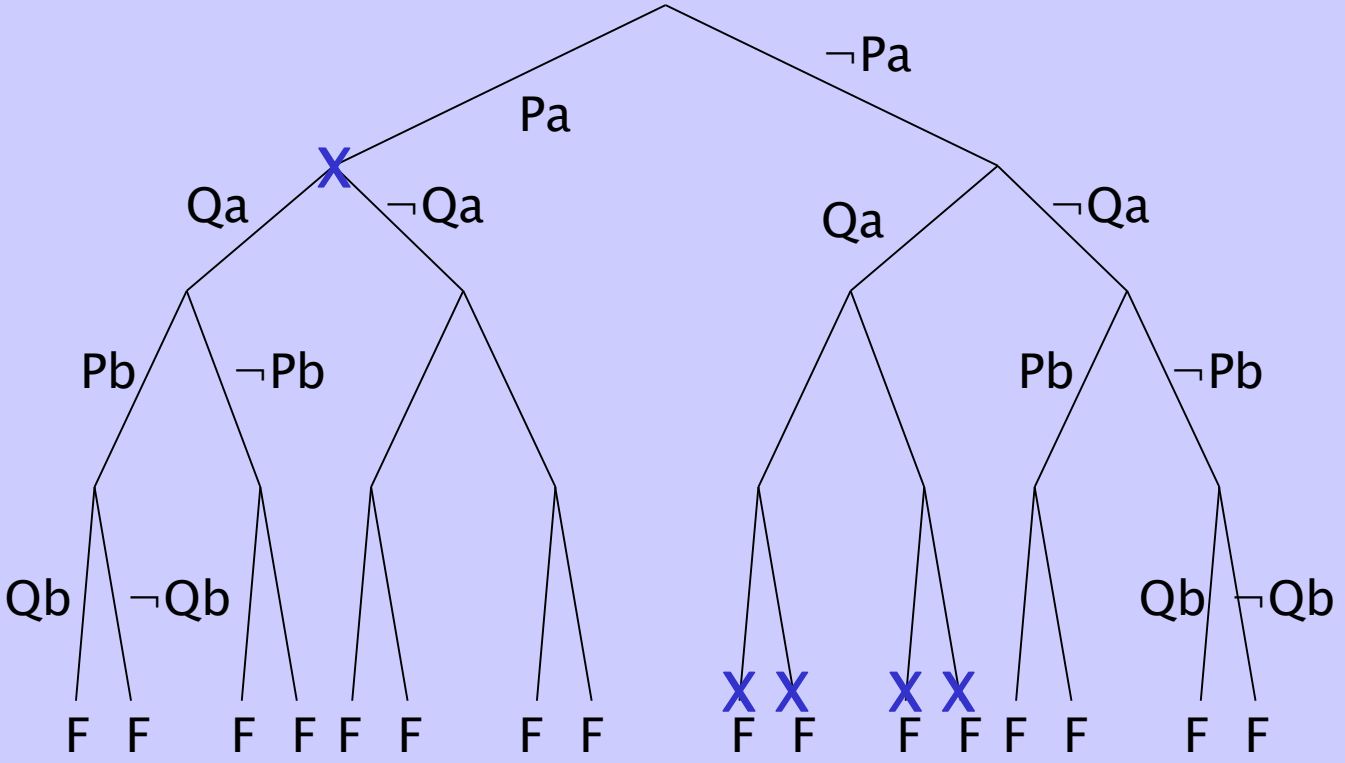
# *Semantische Bäume, Fehlerknoten*

- Bsp. 3: S={{Px ∨ Qy}, {¬Pa}, {¬Qb}}, HU(S)={a, b}, HB(S)={Pa, Pb, Qa, Qb}



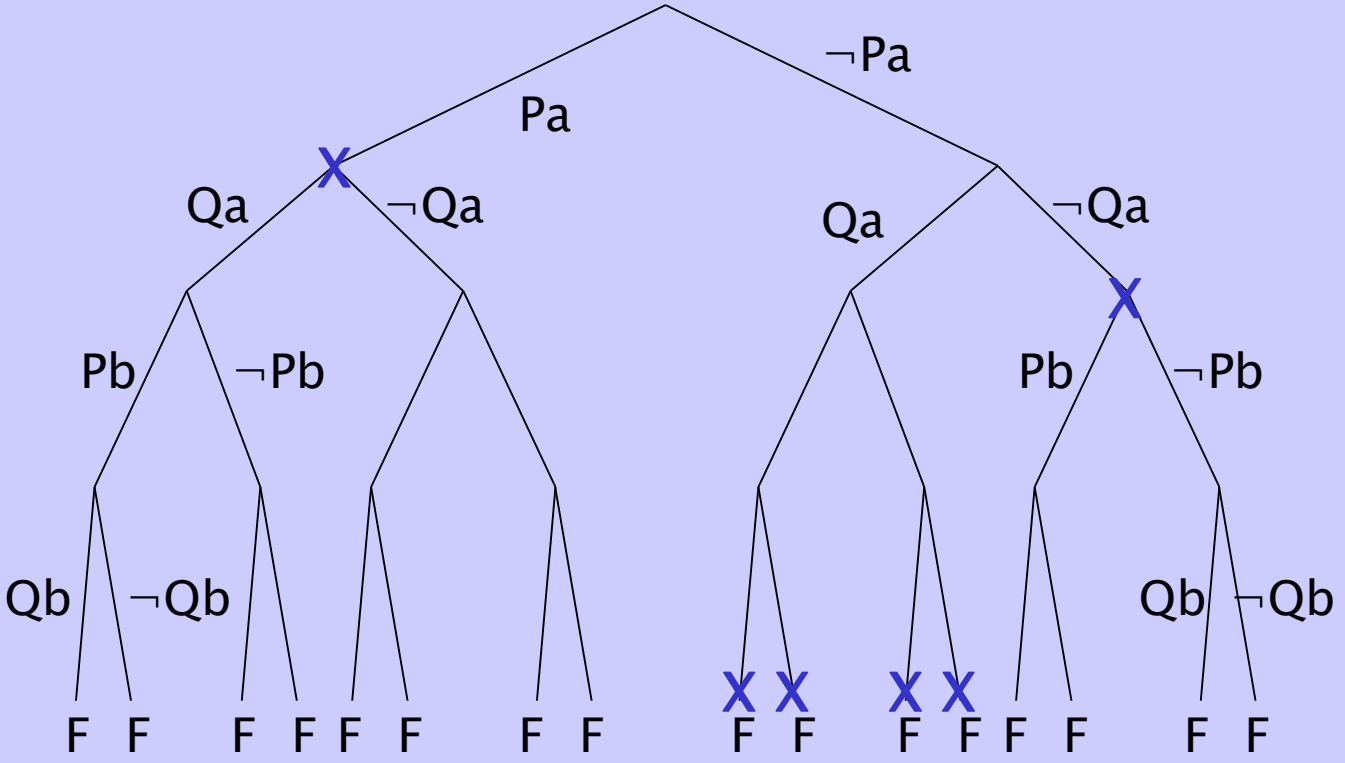
# *Semantische Bäume, Fehlerknoten*

- Bsp. 3: S={{Px ∨ Qy}, {¬Pa}, {¬Qb}}, HU(S)={a, b}, HB(S)={Pa, Pb, Qa, Qb}



# *Semantische Bäume, Fehlerknoten*

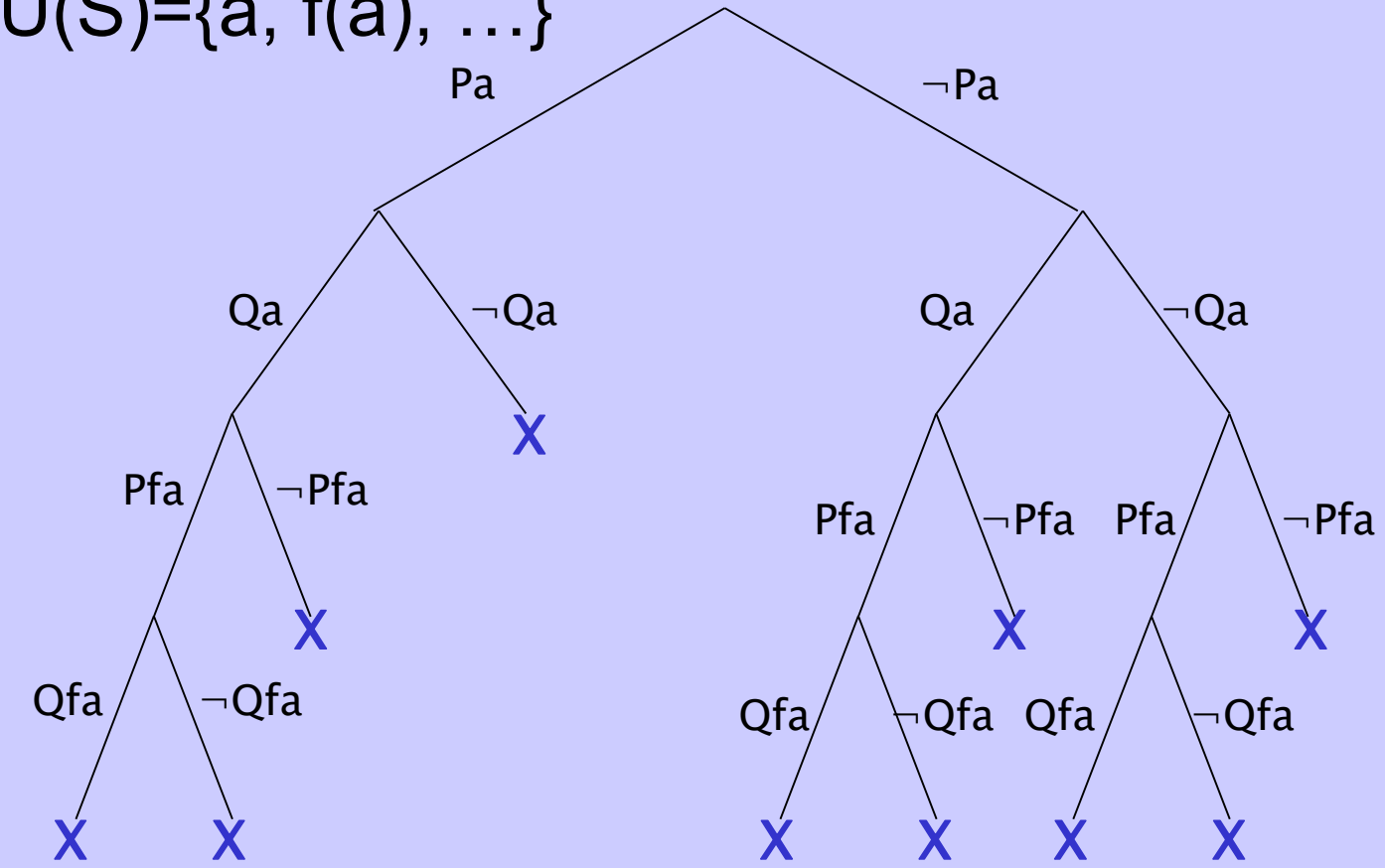
- Bsp. 3: S={{Px ∨ Qy}, {¬Pa}, {¬Qb}}, HU(S)={a, b}, HB(S)={Pa, Pb, Qa, Qb}



# *Semantische Bäume, Fehlerknoten*

- Bsp. 4: S={{¬Px∨Qx}, {Pfy}, {¬Qfz}}, ST(S) ist unendlich

  HU(S)={a, f(a), …}



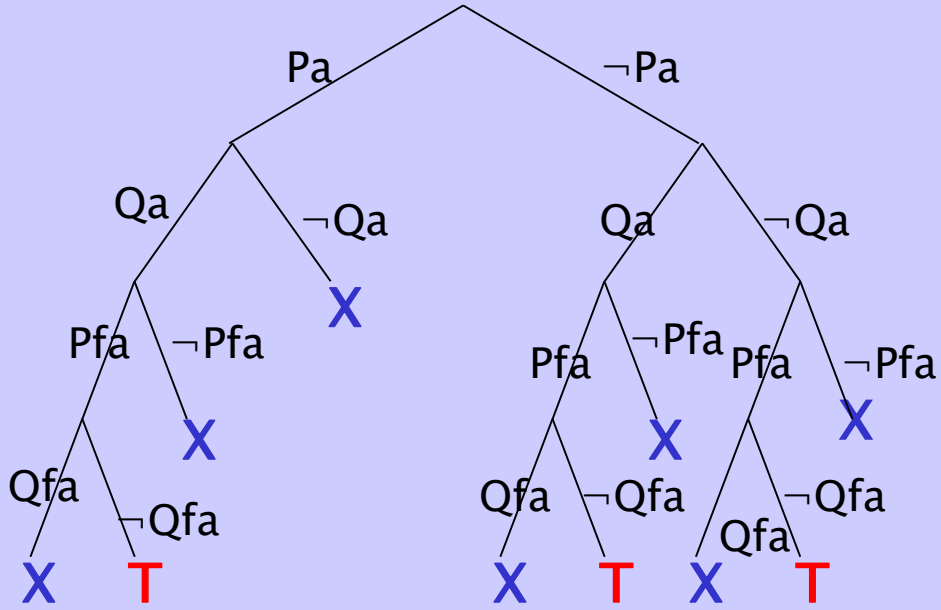
# *Satz von Herbrand*

- Ein semantischer Baum heißt **geschlossen**, wenn alle seine Pfade geschlossen sind
  - Jeder Pfad hat einen Fehlerknoten (in endlicher Tiefe)
- S ist unerfüllbar gdw. ST(S) ist geschlossen
  - Ein Pfad ohne Fehlerknoten liefert ein Herbrand-Modell.
  - Zu jedem Modell gibt es ein Herbrand-Modell auf offenem Pfad.
- Satz (Herbrand): S ist unerfüllbar gdw. eine endl. Menge von Grund-Instanzen von Klauseln in S ist unerfüllbar.
  - Die Grundinstanzen ( = ohne Variablen) ergeben sich aus den Elementen von HB(S) oberhalb der (höchsten) Fehlerknoten
  - **Korollar**: PL-Unerfüllbarkeit rein aussagenlogisch beweisbar
    - Bem.: Das war die ursprüngliche Motivation von Davis-Putnam !



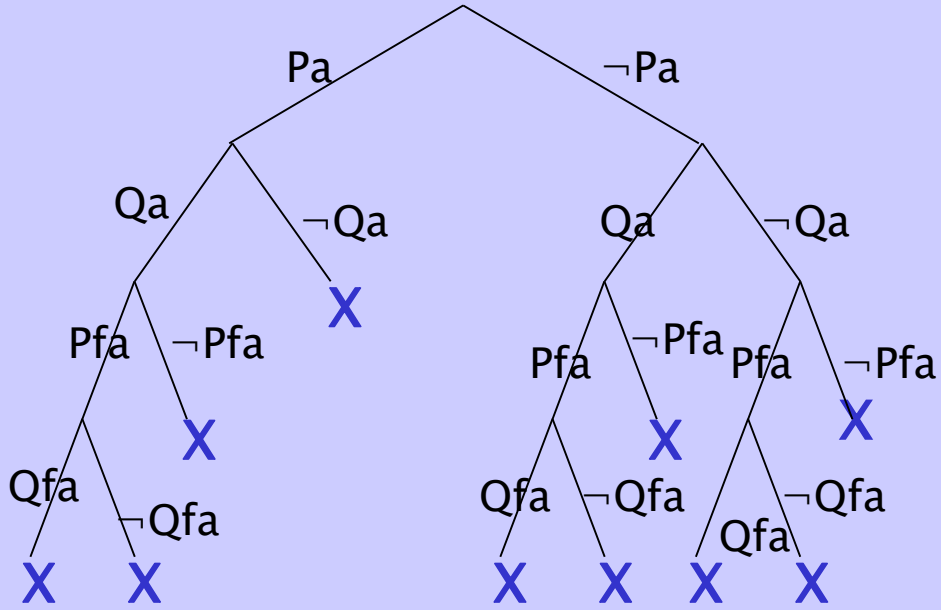
# *Semantische Bäume, Fehlerknoten*

- Bsp. 4: S={{¬Px ∨ Qx}, {Pfy}, {¬Qfz}},  HU(S)={a, f(a), …}
- Stufe 1:  {{¬Pa ∨ Qa}, {Pfa}, {¬Qfa}} konsistent (erfüllbar)



# *Semantische Bäume, Fehlerknoten*

- Bsp. 4: S={{¬Px∨Qx}, {Pfy}, {¬Qfz}},  HU(S)={a, f(a), …}
- Stufe 1:  {{¬Pa∨Qa}, {Pfa}, {¬Qfa}} konsistent (erfüllbar)
- Stufe 2: Stufe 1 ∪ {{¬Pfa∨Qfa}, {Pffa}, {¬Qffa}} inkonsistent !



# *Semantische Bäume, Fehlerknoten*

- Bsp. 4: S={{¬Px∨Qx}, {Pfy}, {¬Qfz}},  HU(S)={a, f(a), …}
- Stufe 1:  {{¬Pa∨Qa}, {Pfa}, {¬Qfa}} konsistent (erfüllbar)
- Stufe 2: Stufe 1 ∪ {{¬Pfa∨Qfa}, {Pffa}, {¬Qffa}} inkonsistent !
- Dies war der Ansatz von Davis-Putnam:
  - Stufe 1 ∪ Stufe 2 = {{¬Pa∨Qa}, {¬Pfa∨Qfa}, {Pfa}, {¬Qfa}, {Pffa}, {¬Qffa}}
  - Die Atome sind ohne Prädikatenlogische Variablen, können also als Aussagenlogische „Variablen“ gelesen werden
  - Lese Pfa nicht als P(f(a)) sondern als Symbol „Pfa“
  - Unit Propagation von Pfa und ¬Qfa gibt {{¬Pa∨Qa}, { }, {Pfa}, {¬Qfa}, {Pffa}, {¬Qffa}}, also UNSAT!



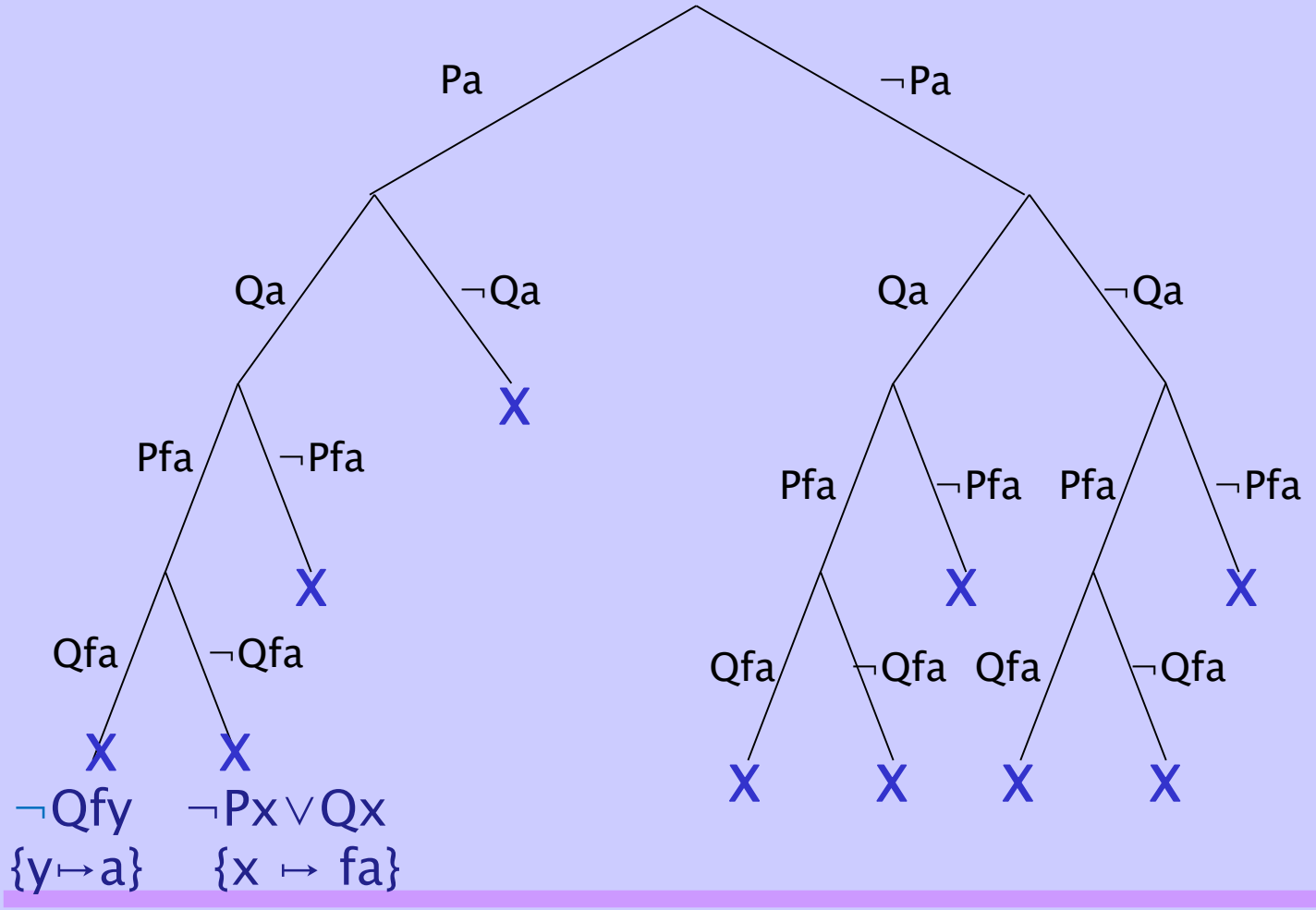
# *Semantische Bäume und Resolution*

- Ein **Inferenzknoten** ist ein Knoten im Semantischen Baum, dessen beide Kinder Fehlerknoten sind
- Satz: Jeder geschlossene Semantische Baum hat einen Inferenzknoten.
  - Ansonsten gäbe es einen unendlichen nicht geschlossenen Pfad und damit ein Modell
- Satz: Sei n ein Inferenzknoten mit Kindern $n_1$ und $n_2$. Seien $G_1$ und $G_2$ die beiden Grund-Fehlerklauseln an $n_1$ und $n_2$. Dann gibt es eine AL-Resolvente G zwischen $G_1$ und $G_2$ und diese schlägt an n fehl (oder oberhalb von n)
  - $G_1$ und $G_2$ enthalten komplementäre Literale, da sie nur wegen der komplementären Elemente von HB(S) an $n_1$ und $n_2$ fehlschlagen



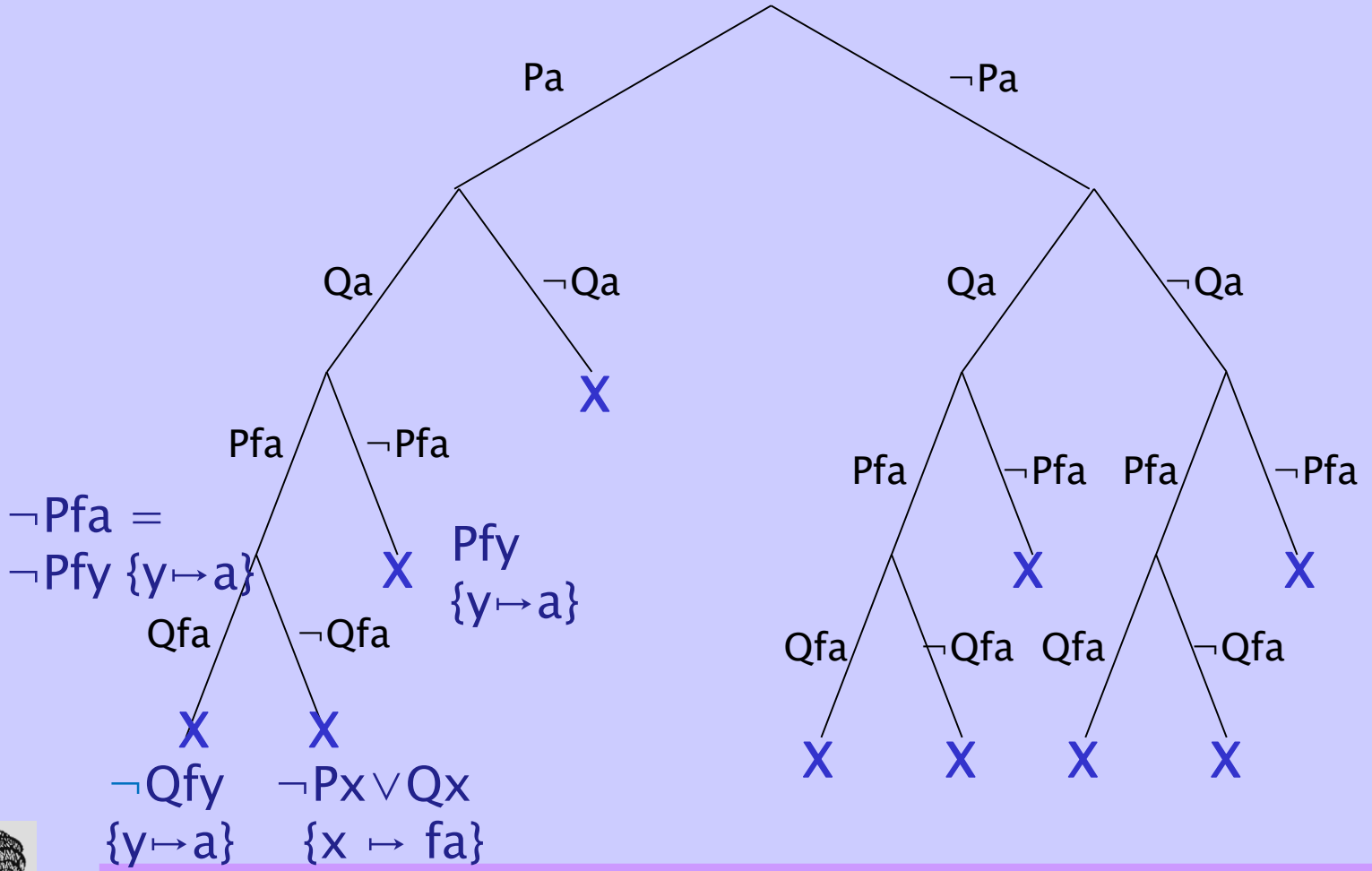
# Semantische Bäume, Inferenzknoten

- S={{¬Px∨Qx}, {Pfy}, {¬Qfy}}



# Semantische Bäume, Inferenzknoten

- S={{¬Px∨Qx}, {Pfy}, {¬Qfy}}



# *Widerlegungsvollständigkeit der Resolution*

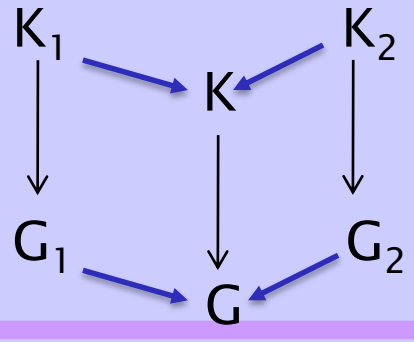
- Sei T ein geschlossener semantischer Baum von S; sei n ein Inferenzknoten mit Kindern $n_1$ und $n_2$ und Grund-Fehlerklauseln $G_1$ und $G_2$.
- Es gibt eine Grund-Resolvente (AL-Resolvente) G, die bei n (oder schon oberhalb n) falsifiziert wird.
- Nach endlich vielen AL-Resolutionsschritten werden alle Inferenzknoten zu Fehlerknoten, inklusive der Wurzel. Dort schlägt die leere Klausel fehl.
- Theorem (Herbrand): Eine PL-Formel S ist beweisbar gdw. eine Formel aus Grundinstanzen von S ist aussagenlogisch beweisbar



# *Widerlegungsvollständigkeit der Resolution*

- Jetzt noch zu zeigen: die PL-Resolution mit Unifikation „deckt die Grundresolution vollständig ab“. Nach endl. vielen PL-Resolutionen wird die leere Klausel erzeugt.
- **Lifting Lemma** (Robinson): Seien $G_1$ und $G_2$ Grundinstanzen der Klauseln $K_1$ und $K_2$, und sei G eine (Grund-)Resolvente von $G_1$ und $G_2$. Dann gibt es auch eine allgemeine Resolvente K von $K_1$ und $K_2$,

  und G ist eine Grundinstanz von K.



# *Widerlegungsvollständigkeit der Resolution*

- Nach dem Satz von Herbrand kann man die Grund-Instanzen von S enumerieren (mittels Enumeration von HB(S)) und jeweils auf Erfüllbarkeit prüfen (z.B. mittels AL-Resolution oder mittels DPLL)
- Nach dem Lifting Lemma genügt es, statt vieler AL-Resolventen „summarisch“ die PL-Resolventen direkt auf der PL-Formel zu bilden.
- Vollständigkeits-Theorem (Robinson): Falls S unerfüllbar ist, kann auf S mittels PL-Resolution in endlicher Zeit die leere Klausel abgeleitet werden.
  - Prädikatenlogik ist nur semi-entscheidbar!
    - Falls S erfüllbar ist werden i.A. unendlich viele Resolventen erzeugt



# *Beispiel: Gruppentheorie*

- Beispiel aus Davis-Logemann-Loveland (C.ACM 5 (1962))

  In einer Gruppe ist ein Links-Inverses auch ein Rechts-Inverses
- Axiome:
  1. e*x = x
  2. I(x)*x = e
  3. (x*y)*z = w  => x*(y*z) = w
  4. x*(y*z) = w  => (x*y)*z = w
- Theorem
  - x*I(x) = e



# *Beispiel: Gruppentheorie*

- Um Beweise in =-Theorie zu vermeiden,
  und Funktionssymbol * zu vermeiden, das das HU aufbläht,
  schreibe P(x,y,z) für x*y = z
- Axiome:
  1. P(e,x,x) statt: e*x = x
  2. P(I(x),x,e) statt: I(x)*x = e
  3. ¬P(x,y,u) ∨ ¬P(u,z,w) ∨ ¬P(y,z,v) ∨ P(x,v,w) statt:(x*y)*z=w =>x*(y*z)=w
     // (x*y = u) ∧ (u*z = w) ∧ (y*z = v) => (x*v=w)
  4. ¬P(y,z,v) ∨ ¬P(x,v,w) ∨ ¬P(x,y,u) ∨ P(u,z,w) statt:x*(y*z)=w =>(x*y)*z=w
- Theorem
  - P(x,I(x),e), bzw negiert und Skolemisiert: ¬P(s,I(s),e) statt: x*I(x) = e