Prädikatenlogik: Syntax

Signatur : Welche Zeichen gibt es?

Funktionssymbole

Prädikatensymbol (Eigenschaften)

Terme:

Variablen

$f(t_1, \ldots t_n)$ wenn t_i Terme und f Funktionssymbol

Formeln:

$P(t_1, \ldots t_n)$ wenn t_i Terme und P Prädikatensymbol *(Atom)*

$(\neg F), (F \vee G), (F \wedge G), (F \Rightarrow G), (F \Leftrightarrow G)$

$(\forall x : \ F)$

$(\exists x : \ F)$

Beispiele

$(\forall x : (\exists y : R(x, y)))$

$(\forall x : (\forall y : (\forall z : ((R(x, y) \wedge R(x, y)) \Rightarrow R(x, z)))))$

vereinfachte Schreibweise:
$\forall x, y, z : R(x, y) \wedge R(x, y) \Rightarrow R(x, z)$

x, y, z, sind Variablen; R ist ein Prädikatensymbol

Definitionen

Atom:	$P(t_1, \ldots t_n)$
Literal:	Ein Atom oder ein negiertes Atom
Grundterm:	Ein Term t ohne Variablen
Grundatom:	Ein Atom ohne Variablen
geschlossene Formel:	ohne freie Variablen
Klausel:	$\forall x_1, x_2, \ldots : F$
	F Disjunktion von Literalen
	und $\{x_1, x_2, \ldots\} = FV(F)$

Interpretationen

Interpretationen I sind analog wie in der Aussagenlogik

Es wird durch I noch zusätzlich festgelegt:

- Menge $D \neq \emptyset$ der möglichen Elemente
- Für Prädikatensymbole P:
  $I(R)$: $D \times \ldots \times D \rightarrow Bool$
- Für Funktionssymbole f:
  $I(f)$: $D \times \ldots \times D \rightarrow D$

Interpretationen: Berechnung

$H = f(t_1, \ldots t_n)$	dann $I\Big(f(t_1, \ldots t_n)\Big) = I(f)\Big(I(t_1), \ldots, I(t_n)\Big)$
$H = P(t_1, \ldots t_n)$	dann $I\Big(P(t_1, \ldots t_n)\Big) = I(P)\Big(I(t_1), \ldots, I(t_n)\Big)$
$H = \texttt{false}$	dann $I(H) = 0$
$H = \texttt{true}$	dann $I(H) = 1$
$H = \neg F$	dann $I(H) = 1$ falls $I(F) = 0$
$H = F \vee G$	dann $I(H) = 1$ falls $I(F) = 1$ oder $I(G) = 1$
$H = F \wedge G$,	dann $I(H) = 1$ falls $I(F) = 1$ und $I(G) = 1$
$H = F \Rightarrow G$	dann $I(H) = 1$ falls $I(F) = 0$ oder $I(G) = 1$
$H = F \Leftrightarrow G$	dann $I(H) = 1$ falls $I(F) = 1$ gdw. $I(G) = 1$
$H = \forall x : F$	dann $I(H) = 1$ falls **für alle** $a \in D : I[a/x](F) = 1$
$H = \exists x : F$	dann $I(H) = 1$ falls **für ein** $a \in D : I[a/x](F) = 1$

Modelle, Tautologien, Widerspruch etc.

$I \models F$: I macht F wahr

Eine (geschlossene) Formel F heißt:

• allgemeingültig (Tautologie, Satz)	gdw.	alle I machen F wahr
• erfüllbar	gdw.	ein I macht F wahr
• unerfüllbar (widersprüchlich)	gdw.	kein I macht F wahr
• falsifizierbar	gdw.	ein I macht F falsch

F ist allgemeingültig gdw. $\neg F$ unerfüllbar

Beispiele

allgemeingültig: $\forall x.Q(x) \vee \neg Q(x)$

unerfüllbar: $P \wedge \neg P$

erfüllbar, falsifizierbar: $\forall x.P(x)$

Semantische Folgerung

$F \models G$ gdw. für alle I: $I \models F$ impliziert $I \models G$

Deduktionstheorem:
Für alle Formeln F und G gilt:

$F \models G$ gdw. $F \Rightarrow G$ ist Tautologie

Folgerung:

$$F \models G$$
gdw.
$\neg(F \Rightarrow G)$ ist unerfüllbar (widersprüchlich)
gdw.
$F \wedge \neg G$ ist unerfüllbar.

Komplexität

Es ist **unentscheidbar**, ob eine geschlossene Formel eine Tautologie ist

(Halteproblem für Turingmaschinen)

Tautologien der Prädikatenlogik sind rekursiv aufzählbar

praktische Folgerung:

Deduktionssystem:

Wenn F Tautologie:	dann Antwort JA in endlicher Zeit
Wenn F keine Tautologie:	Antwort NEIN oder Nichtterminierung

Klauselnormalformen

Klauselnormalform (conjunctive normal form, CNF)

Konjunktion von Klauseln: $K_1 \wedge \ldots \wedge K_n$

Klausel: Disjunktion von Literalen mit All-Quantor-Präfix

Elementare Rechenregeln

Regeln der Aussagenlogik gelten auch für Formeln.

Regeln für Quantoren:

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

Transformation in CNF

Unter Erhaltung der Unerfüllbarkeit

1: Elimination von $\Leftrightarrow$ und $\Rightarrow$:

$$\begin{array}{lcl} F \Leftrightarrow G & \rightarrow & F \Rightarrow G \wedge G \Rightarrow F \\ F \Rightarrow G & \rightarrow & \neg F \vee G \end{array}$$

2. Negation ganz nach innen schieben:

$$\begin{array}{lcl} \neg\neg F & \rightarrow & F \\ \neg(F \wedge G) & \rightarrow & \neg F \vee \neg G \\ \neg(F \vee G) & \rightarrow & \neg F \wedge \neg G \\ \neg\forall x : F & \rightarrow & \exists x : \neg F \\ \neg\exists x : F & \rightarrow & \forall x : \neg F \end{array}$$

Transformation in CNF

3. Skopus von Quantoren minimieren:

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

4. Alle gebundenen Variablen umbenennen.

Transformation in CNF

5. $\exists$ eliminieren mittels Skolemisierung
   von außen nach innen.

   $\exists x : F[x] \;\rightarrow\; F[g(y_1, \ldots, y_n)]$

   wobei y_i die Variablen sind, die weiter außen
   unter einem Allquantor stehen.

   g neues Funktionssymbol
   pro Existenzquantor ein neues Funktionssymbol

Transformation in CNF

6. Allquantoren nach außen schieben

7. Distributivität, Assoziativität, Kommutativität von $\wedge, \vee$
   iterativ (mit richtiger Strategie) anwenden,
   um $\wedge$ nach außen zu schieben ("Ausmultiplikation").
   $F \vee (G \wedge H) \rightarrow (F \vee G) \wedge (F \vee H)$

   (Oder: Schnelle CNF-Erzeugung)

Resultat: CNF

Konjunktion von Disjunktionen (Klauseln) von Literalen:

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

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

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

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

Skolemisieren: $\forall y : (P(f(y)) \wedge \neg Q(y)) \ \vee \ (Q(y) \wedge \neg P(f(y)))$

Distrib.: $\forall y : (P(f(y)) \vee Q(y)) \ \wedge \ (P(f(y)) \vee \neg P(f(y)))$
$\wedge \ (\neg Q(y) \vee Q(y)) \ \wedge \ (\neg Q(y) \vee \neg P(f(y)))$

Klausel-Erz.+
Umbenennen: $(\forall y_1 : P(f(y_1)) \vee Q(y_1)) \wedge (\forall y_2 : P(f(y_2)) \vee \neg P(f(y_2)))$
$\wedge(\forall y_3 : \neg Q(y_3) \vee Q(y_3)) \wedge (\forall y_4 : \neg Q(y_4) \vee \neg P(f(y_4)))$

Grundresolution

Resolutionsschritt ist analog zur Resolution in Aussagenlogik:

$$\begin{array}{ll} \text{Elternklausel 1:} & L,\ A_1, \ldots, A_m \\ \text{Elternklausel 2:} & \neg L, B_1, \ldots, B_n \\ \text{Resolvente:} & \hline A_1, \ldots, A_m, B_1, \ldots, B_n \end{array}$$

Das Resolutionsverfahren vervollständigt Klauselmengen
(analog zur Aussagenlogik)

Erfolg, wenn leere Klausel erzeugt wurde.

Stopp, wenn keine neuen Klauseln erzeugbar sind: *Erfüllbar*

Grundresolution: Vollständigkeit

Satz Jede unerfüllbare Grundklauselmenge ist mit Grundresolution widerlegbar.

$\Longrightarrow$ Grundresolution ist ein Entscheidungsverfahren für Grundklauselmengen.

Das folgt direkt aus den Sätzen zur Aussagenlogik.

Allgemeine Resolution (Prädikatenlogik)

Man braucht: „komplementär machen“.

$P(x)$ und
$\neg P(y)$

$x \mapsto y$ ist eine passende **Substitution**

Resolution mit Unifikation

Elternklausel 1: $L,\ A_1, \ldots, A_m$ σ ist Substitution

Elternklausel 2: $\neg L', B_1, \ldots, B_n$ mit $\sigma(L) = \sigma(L')$

Resolvente: $\sigma(A_1, \ldots, A_m, B_1, \ldots, B_n)$

Beispiel: modus ponens

Historisches Beispiel:
Alle Menschen sind sterblich, Sokrates ist ein Mensch,
also ist Sokrates sterblich.

$M(So) \ \wedge \ (\forall y : (M(y) \Rightarrow S(y))) \ \Rightarrow \ S(So)$

Klauselmenge zur Negation: $\neg(M(So) \ \wedge \ (M(y) \Rightarrow S(y)) \ \Rightarrow \ S(So))$

$$\{M(So)\}$$
$$\{\neg M(y) \vee S(y)\}$$
$$\{\neg S(So)\}$$

1. Resolvente: $\{S(So)\}$ aus $\{M(So)\}$ und $\{\neg M(y) \vee S(y)\}$
   mit Substitution $y \mapsto So$

2. Resolvente: $\emptyset$ aus $\{S(So)\}$ und $\{\neg S(So)\}$

Extra Regel: Faktorisierung

Elternklausel: $\dfrac{L, L', K_1, \ldots, K_m}{\sigma(L) = \sigma(L')}$

Faktor: $\sigma(L, K_1, \ldots, K_m)$

z.B. $\{P(x) \vee P(y)\}$ hat $\{P(y)\}$ als Faktor mit Substitution $x \mapsto y$.

Resolutionskalkül

Der Resolutionskalkül transformiert Klauselmengen $\mathcal{S}$ wie folgt:

1. $\mathcal{S} \rightarrow S \cup \{R\}$, wobei R eine Resolvente

2. $\mathcal{S} \rightarrow S \cup \{F\}$, wobei F ein Faktor

Diese Transformationen werden iteriert (nicht-deterministisch)

Der Resolutionskalkül terminiert mit Erfolg, wenn $\emptyset \in \mathcal{S}$.
Antwort: die Eingabe ist widersprüchlich

Beispiel:

(Variante des Russelschen Widerspruchs)

das Beispiel zeigt auch, dass Faktorisierung notwendig ist.

Der Friseur rasiert alle, die sich nicht selbst rasieren

$$\forall x : \neg(\text{rasiert}(x, x)) \Leftrightarrow \text{rasiert}(\text{Friseur}, x)$$

Klauselmenge:

1. $\{\text{rasiert}(x, x), \text{rasiert}(\text{Friseur}, x)\}$,
2. $\{\neg\text{rasiert}(\text{Friseur}, y), \neg\text{rasiert}(y, y)\}$

Beispiel: Friseur Widerlegung

Faktorisierung von $\{\text{rasiert}(x, x), \text{rasiert}(\text{Friseur}, x)\}$ mit

$$\{x \mapsto \text{Friseur}\}$$

ergibt $\text{rasiert}(\text{Friseur}, \text{Friseur})$

Faktorisierung von $\{\neg\text{rasiert}(\text{Friseur}, y), \neg\text{rasiert}(y, y)\}$ mit

$$\{y \mapsto \text{Friseur}\}$$

ergibt : $\neg\text{rasiert}(\text{Friseur}, \text{Friseur})$

Resolution ergibt leere Klausel

Ohne Faktorisierung ist in diesem Beispiel
keine Resolutions-Widerlegung möglich.

Beispiel: Resolution

Beweis der Transitivität der Teilmengenrelation mit Resolution

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

Theorem: $\forall x, y, z : x \subseteq y \wedge y \subseteq z \Rightarrow x \subseteq z$

Klauselform von Axiom $\wedge \neg$Theorem:

H1:	$\neg x \subseteq y, \neg w \in x, w \in y$	($\Rightarrow$ Teil der Definition)
H2:	$x \subseteq y, f(x,y) \in x$	(zwei $\Leftarrow$ Teile der Definition,
H3:	$x \subseteq y, \neg f(x,y) \in y$	f ist die Skolem Funktion für w)
C1:	$a \subseteq b$	(drei Teile der negierten Behauptung,
C2:	$b \subseteq c$	a, b, c sind Skolem Konstanten für x, y, z)
C3:	$\neg a \subseteq c$	

Beispiel: Resolutionswiderlegung

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: $\neg a \subseteq c$

H1,1 & C1,	$\{x \mapsto a, y \mapsto b\}$	$\vdash$	R1: $\neg w \in a, w \in b$
H1,1 & C2,	$\{x \mapsto b, y \mapsto c\}$	$\vdash$	R2: $\neg w \in b, w \in c$
H2,2 & R1,1,	$\{x \mapsto a, w \mapsto f(a,y)\}$	$\vdash$	R3: $a \subseteq y, f(a,y) \in b$
H3,2 & R2,2,	$\{y \mapsto c, w \mapsto f(x,c)\}$	$\vdash$	R4: $x \subseteq c, \neg f(x,c) \in b$
R3,2 & R4,2,	$\{x \mapsto a, y \mapsto c\}$	$\vdash$	R5: $a \subseteq c, a \subseteq c$
R5 & (Faktorisierung)		$\vdash$	R6: $a \subseteq c$
R6 & C3		$\vdash$	R7: $\emptyset$

Unifikation

Resolutions- und Faktorisierungsregel verwenden Substitutionen, die zwei Atome syntaktisch gleich machen.
Diese Substitutionen nennt man **Unifikatoren**

Gesucht: **allgemeinste Unifikatoren**

Beispiele: Unifikator und allgemeinster Unifikator

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

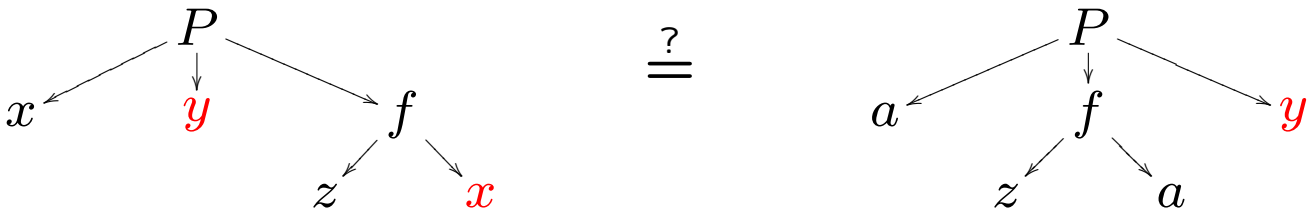
$\sigma = \{x \mapsto a, y \mapsto a\}$
σ ist Unifikator

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

$\sigma = \{x \mapsto y\}$
σ ist allgemeinster Unifikator

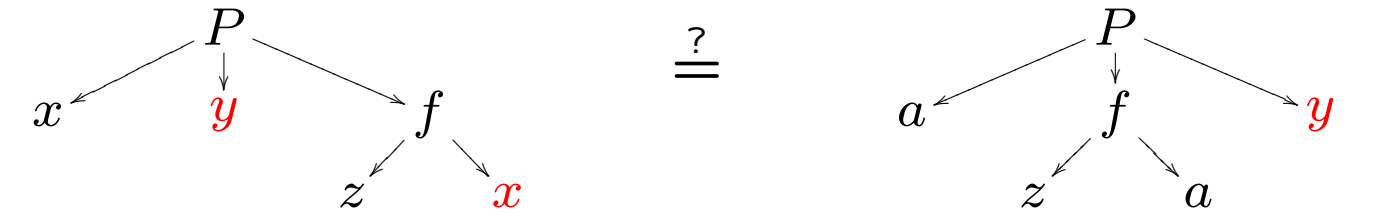
Beispiel zur Unifikation

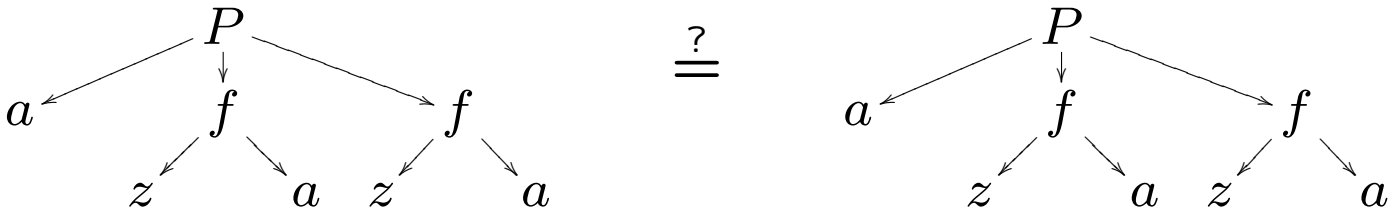
$$P(x, y, f(z, x) = P(a, f(z, a), y)$$



Beispiel zur Unifikation

$$P(x, y, f(z, x) = P(a, f(z, a), y)$$





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

Unifikationsalgorithmus:

Eingabe: zwei Terme oder Atome s und t:

Ausgabe: "nicht unifizierbar" oder einen allgemeinsten Unifikator:

Zustände: Menge Γ von Gleichungen $s \stackrel{?}{=} t$.

Initialzustand: $\Gamma_0 = \{s \stackrel{?}{=} t\}$.

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)

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

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

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

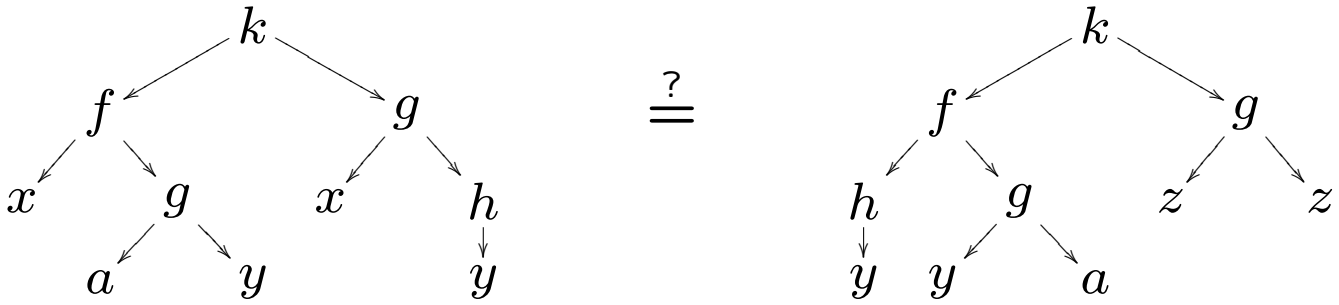
Abbruchbedingungen:

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

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

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

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

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

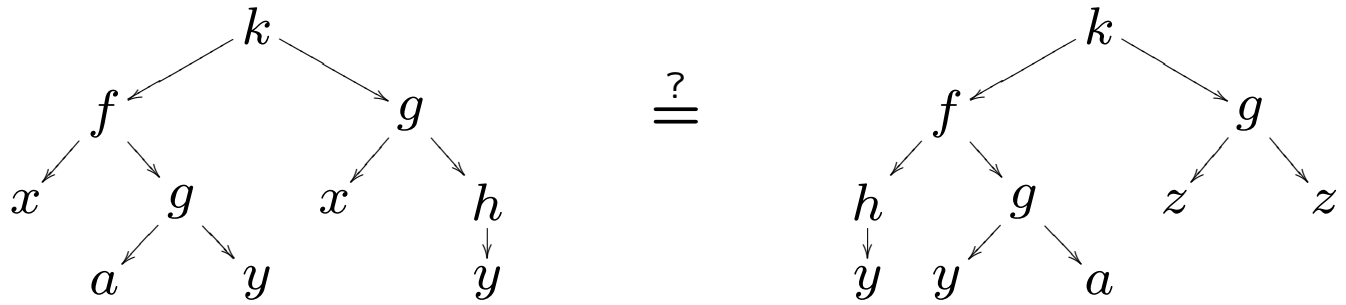
Beispiel

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

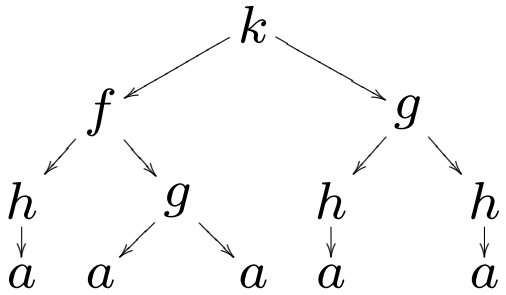
Unifizierte Terme: $k(f(h(a),g(a,a)),g(h(a),h(a)))$.

Beispiel

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



Unifizierter Term: $k(f(h(a),g(a,a)),g(h(a),h(a)))$.



Theoretische Eigenschaften: Unifikation

Satz

Der Unifikationsalgorithmus terminiert, ist korrekt und vollständig.
Er gibt genau einen allgemeinsten Unifikator aus, falls er nicht abbricht.
Der Unifikator ist eindeutig bis auf „Variablenumbenennung “
Der Algorithmus kann polynomiell, sogar in $O(n*\log(n))$, implementiert werden.

Folgerung: Unifikationsalgorithmus ist ausreichend zur Entscheidung der Unerfüllbarkeit einer Menge von 1-Klauseln

Vollständigkeit der Resolution

Begriffe:

$\sigma(C)$ ist Grundinstanz einer Klausel C,
wenn keine Variablen mehr in $\sigma(C)$ vorkommen.

Satz (Gödel-Herbrand-Skolem Theorem)

Zu jeder unerfüllbare Menge C von Klauseln gibt es eine endliche unerfüllbare Menge von Grundinstanzen von C.

Vollständigkeit der Resolution

Satz

Die Resolution mit Unifikation und Faktorisierung ist vollständig.

Dazu benötigt man:

- den Satz von Gödel-Herbrand-Skolem,
- die Vollständigkeit der Grundresolution
- Ein Lemma zum Lifting
  Zusammenhang zwischen allgemeinen Resolutionsschritten und Grundresolutionsschritten.
  Hier erkennt man: die Faktorisierung ist erforderlich

Eigenschaften der Resolution

Es gilt:

- i.a terminiert Resolution nicht
- Sätze der Prädikatenlogik sind unentscheidbar aber noch semi-entscheidbar
- Resolution ist nicht gut geeignet zum Vorwärtsschließen und nicht zum Berechnen von Modellen

Redundanz-Elimination bei Resolution : Löschregeln: Subsumtion, Tautologie und Isoliertheit

Es gibt folgende Situationen im Resolutionsbeweiser:

- Tautologische Klauseln werden hinzugefügt
  $\{P, \neg P, \ldots\}$
- Spezialisierungen von bereits vorhandenen Klauseln werden hinzugefügt
  Wenn $\{P(x), Q\}$ schon in M, vorhanden ist,
  dann sind
  $\{P(a), Q\}$ aber auch $\{P(y), Q, R\}$ Spezialisierungen
- neue allgemeine Klausel hergeleitet,
  die allgemeiner ist als schon vorhandene Klauseln

Isolierte Literale

Sei L ein Literal in der Klausel D, die in der Klauselmenge $\mathcal{M}$ ist.

L heißt *isoliert*, wenn kein L' in $\mathcal{M}$ vorkommt, so dass
L und L' verschiedenes Vorzeichen haben
und L und L' unifizierbar sind.

Der Isoliertheitstest kann in Zeit $O(n^3 \log(n))$ durchgeführt werden.

Löschregel für isolierte Literale:
Lösche Klauseln aus einer Klauselmenge, wenn diese isolierte Literale enthalten.

Isolierte Literale

- können schon am Anfang vorhanden sein.
- können durch die Anwendung von Redundanzlöschungen entstehen.

Beispiel

Gegeben: $\mathcal{M} :=$	$\{\{P(x)\}, \{\neg P(y), Q(y)\}\}$
Resolution ergibt:	$\{\{P(x)\}, \{\neg P(y), Q(y)\}, \{Q(z)\}\}$
Resolvente subsumiert Elternklausel:	$\{\{P(x)\}, \{Q(z)\}\}$

Jetzt sind alle Literale isoliert.

Isolierte Literale: Beispiel

Gegeben: $\mathcal{M} := \quad \{\{P(x,x)\}, \{\neg P(y,z), Q(y,z)\}, \{\neg Q(a,b)\}\}$

Resolution ergibt: $\{\{P(x,x)\}, \{\neg P(y,z), Q(y,z)\}, \{Q(z,z)\}, \{\neg Q(a,b)\}\}$

$\neg Q(z,z)$ ist jetzt isoliert.

Löschung bewirkt eine Schleife

Löschen isolierter Literale: Korrektheit und Vollständigkeit

Es gilt:

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

Argumentation:

Die isolierten Literale kommen auch in der Resolvente vor, deshalb können die entspr. Resolventen nicht mehr zur leeren Klausel führen

Löschen isolierter Literale: Beispiel

Betrachte die Klauselmenge

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

Klauselmenge ist unerfüllbar und hat keine isolierten Literale.
$C1 + C4$ ergibt die Resolvente $\{Q(a)\}$.
Resolvente hat ein isoliertes Literal: Sackgasse in der Suche.

Subsumtion

Die Klausel D subsumiert die Klausel E,
wenn es eine Substitution σ gibt, so dass $\sigma(D) \subseteq E$

Löschregel für subsumierte Klauseln:
Wenn Klausel D die Klausel E subsumiert
und E hat nicht weniger Literale als D,
dann lösche die Klausel E

Subsumtion: Beispiele

$\{P(x)\}$ subsumiert $\{P(a), P(b), Q(y)\}$.

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.

Vorsicht $\{\neg P(x), P(f(x))\}$ impliziert $\{\neg P(x), P(f(f(x))\}$,
aber subsumiert nicht.

Subsumtion: Komplexität

Der Test, ob eine Klausel C eine andere subsumiert ist $\mathcal{NP}$-vollständig.

Die Komplexität kommt von den Permutationsmöglichkeiten der Literale beim Subsumtionstest.

Subsumtion: Vorwärts- und Rückwärtsanwendung

Vorwärtsanwendung: Lösche Resolventen / Faktoren, die von Klauseln aus der Klauselmenge subsumiert werden.

Wichtige Löschregel in resolutionsbasierten Deduktionssystemen

Rückwärtsanwendung Verwende neueste Resolvente zum Löschen von vorhandenen Klauseln

nicht immer implementiert: kann ineffizient sein

Resolution mit Subsumtion: Vollständigkeit

Satz

Der Resolutionskalkül zusammen mit Subsumtions-Löschung ist widerlegungsvollständig.

Argumentation:

Wenn es eine Resolutionsherleitung der leeren Klausel gibt,
dann auch eine ohne die subsumierten Klauseln
Evtl. in der Herleitung eine Klauseln durch die subsumierende Klausel ersetzen.

Tautologie-Löschung

Eine Klausel D ist eine *Tautologie*, wenn D unter allen Interpretationen wahr ist.

Syntaktische Erkennung: D ist Tautologie, wenn zwei komplementäre Literale L, L' in D enthalten sind

Dieser Test ist in Zeit $O(n^3)$ durchführbar.

Löschregel: tautologische Klauseln kann man löschen.

Tautologielöschung: Beispiele

$\{Pa, \neg Pa\}$,

$\{Qa, P(f(x)), \neg P(f(x), Qb\}$

$\{Px, \neg Px\}$.

Keine Tautologien: $\{Px, \neg Pf(y)\}$
$\{\neg P(x, y), P(y, x)\}$.

Widerlegungsvollständigkeit

Satz

Der Resolutionskalkül zusammen mit
- Löschung subsumierter Klauseln,
- Löschung von Klauseln mit isolierten Literalen ist widerlegungs-
- Löschung von Tautologien

vollständig.

Die Davis-Putnam-Prozedur für aussagenlogische Klauselmengen hat als Regeln: Resolution, Subsumtionsregel, Isolationsregel aber auch noch Fallunterscheidung

Einige Weitere Einschränkung der Resolution

Set-of-Support
UR-Resolution
Unit-Resolution
Input-Resolution
Lineare-Resolution
SL-Resolution
SL-Resolution für Hornklauseln

Set-of-Support

Aufteilung der Klauselmenge in:

- Klauseln aus den Voraussetzungen (widerspruchsfrei)
- Klauseln aus dem negierten Theorem SOS-Menge

Set-of-Support = alle Klauseln, Resolventen, Faktoren, die eine Vorgängerklausel in SOS haben.
Resolution: nur erlaubt, wenn eine Elternklausel in SOS ist.
Faktorisierung: nur erlaubt, wenn die Klausel in SOS ist.

Beispiel

Klauselmenge:	$\{P, Q\}, \{\neg P, Q\}, \{P, \neg Q\}, \{\neg P, \neg Q\}$
SOS	$\{P, Q\}$
Rest-Klauselmenge:	$\{\neg P, Q\}, \{P, \neg Q\}, \{\neg P, \neg Q\}$
Resolventen:	$\{P, Q\}, \{\neg P, Q\} \vdash \{Q\}$
SOS danach:	$\{P, Q\}, \{Q\}$

Unit-Resolution

Bei jedem Resolutionsschritt
muss mindestens eine Elternklausel eine Unit sein.

Faktorisierung ist erlaubt.

Fakt: Nicht widerlegungsvollständig

Eingaberesolution

Def. Bei jedem Resolutionsschritt muss mindestens eine Elternklausel aus der Eingabemenge sein.

Eigenschaften: Eine Klauselmenge ist genau dann
mit Unit-Resolution widerlegbar,
wenn sie mit Inputresolution widerlegbar ist.
D.h. Eingabe und Unit-Resolution sind äquivalent

Lineare Resolution

Definition

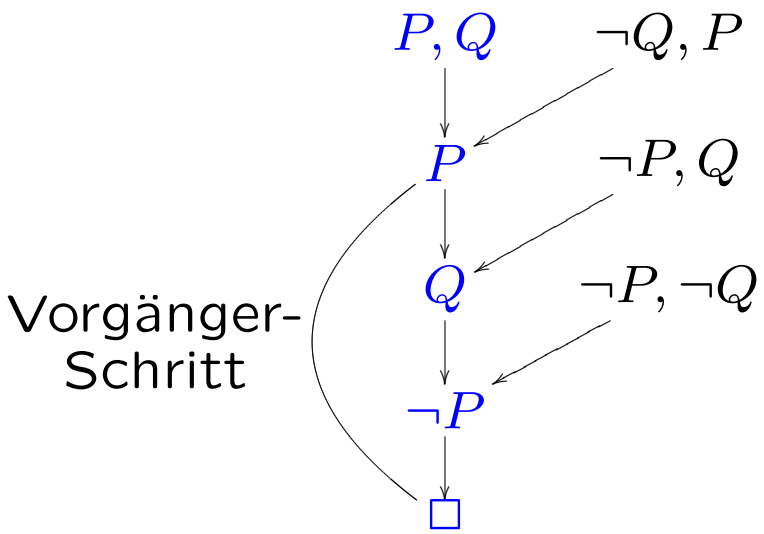
Initial:	Klauselmenge; eine Klausel wird als Zentralklausel Z_0 ausgewählt.
Iterationsschritt:	aktuelle Zentralklausel + andere Klausel ergibt neue Zentralklausel. Z_i + andere Klausel $\rightarrow$ Z_{i+1} implizite Faktorisierung (vor Resolution) ist erlaubt

Eigenschaft: Lineare Resolution ist widerlegungsvollständig

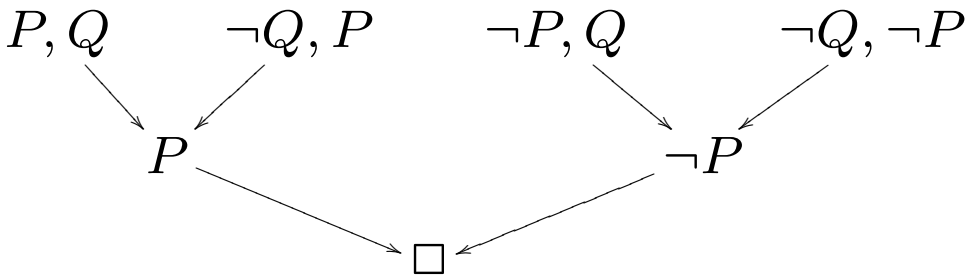
Lineare Resolution, Beispiel

Klauselmenge $\{P, Q\}, \{\neg P, Q\}, \{P, \neg Q\}, \{\neg P, \neg Q\}$ ist unerfüllbar:

lineare Resolution



allgemeine Resolution



Lineare Resolution, Beispiel

Faktorisierung ist notwendig:

$$\{\{P(x), P(y)\}, \{\neg P(x), \neg P(y)\}\}.$$

In diesem Beispiel:
Lineare Resolution ohne Faktorisierung
ergibt immer eine Zentralklausel der Länge zwei,

Hornklauselmengen

Definitionen

positives Literal	:=	Atom ; Literal ohne Negation
negatives Literal	:=	negiertes Atom
Hornklausel	:=	Klausel mit maximal einem positiven Literal.
definite Hornklausel	:=	Klausel mit genau einem positiven Literal.
negative Hornklausel, Anfrage	:=	Klausel ohne positive Literale.
Hornklauselmenge	:=	Klauselmenge nur aus Hornklauseln

Hornklauselmengen

Eigenschaften:

- definite Hornklauseln kann man schreiben als
  $L_1 \wedge \ldots \wedge L_n \Rightarrow L_{n+1}$
- Klauseln ohne positive Literale sind Hornklauseln
- Eine unerfüllbare Hornklauselmenge enthält
  mindestens eine positive Unitklausel.
  und mindestens eine negative Hornklausel
- Unerfüllbare Hornklauselmengen sind widerlegbar mittels
  Unit-Resolution.

SL-Resolution

SL-Resolution = Lineare Resolution plus Selektionsfunktion
D.h. Beschränkung der (für Resolution) erlaubten Literale

Wir betrachten ab jetzt:

SL-Resolution für Hornklauselmengen

Annahme: Hornklauseln haben Reihenfolge der negativen Literale:

SL-Resolution für Hornklauselmengen

Vorgehen der linearen Resolution:

- Menge von initialen (definiten) Horn Klauseln
- Zielklausel (nur negative Literale)
- Nächste Zielklausel (auch rein negativ)
  = Resolvente aus Zielklausel und einer initialen Klausel
- Selektionsfunktion:
  Das letzte Literal wird zuerst wegresolviert

Beispiel:

Zielklausel	$\neg L_1, \ldots \neg L_n$
initiale (passende Klausel):	$K_1, \neg K_2, \ldots \neg K_m$
	K_1 und L_n sind Resolutions-Literale
Resolvente:	$\neg L_1, \ldots \neg L_{n-1}, \neg K_2, \ldots \neg K_m$

SL-Resolution für Hornklauselmengen: Eigenschaften

Widerlegungsvollständig für Hornklauseln, auch ohne Faktorisierung

Die Gesamtsubstitution in die Anfrage bei Erfolg
kann man als Antwort deuten.

Beispiel SL-Resolution

SL-Resolution mit Horn Klauseln

	Prolog Notation:	Klausel Notation:
C1:	$A(x,y) \Leftarrow P(x,y)$	$A(x,y) \vee \neg P(x,y)$
C2:	$A(x,z) \Leftarrow P(x,y) \wedge A(y,z)$	$A(x,z) \vee \neg P(x,y) \vee \neg A(y,z)$
C3:	$P(a,b)$	
C4:	$P(b,c)$	
C5:	$P(c,d)$	

Theorem: $\exists v : A(a,v) \wedge P(v,d)$?
Negiertes Theorem: $\neg A(a,v) \vee \neg P(v,d)$

Suchraum für Selektionsfunktionen

Beispiele für Selektionsfunktionen sind:

1. Das am stärksten instanziierte Literal zuerst

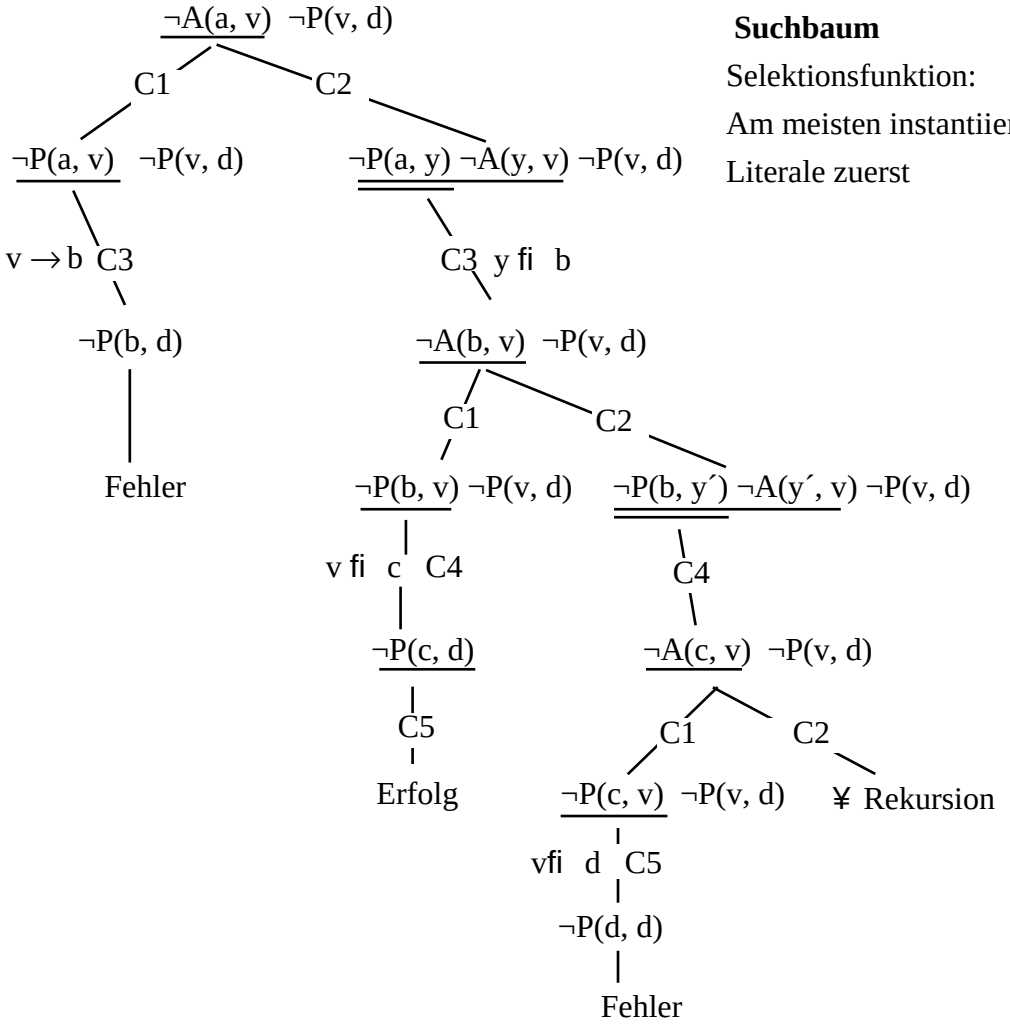
2. Das am wenigsten instanziierte Literal zuerst.

Die zuletzt eingeführten Literale, die zuerst wegresolviert werden, sind unterstrichen.

Suchstrategien: Tiefensuche mit Backtracking
Breitensuche zu speicher-aufwendig

SLD-Bäume: Beispiel

C1:	$A(x,y) \Leftarrow P(x,y)$	C4:	$P(b,c)$
C2:	$A(x,z) \Leftarrow P(x,y) \wedge A(y,z)$	C5:	$P(c,d)$
C3:	$P(a,b)$	Anfrage	$\neg A(a,v) \vee \neg P(v,d)$



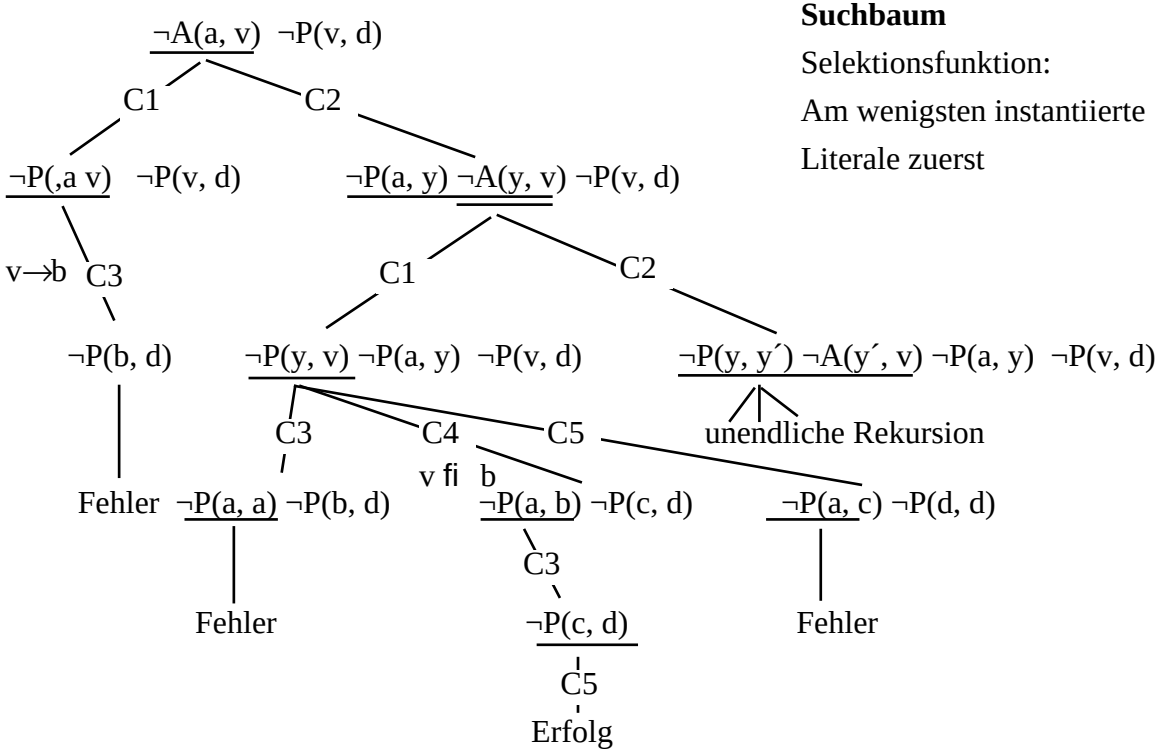
Suchbaum

Selektionsfunktion:

Am meisten instantiierte Literale zuerst

SLD-Bäume: Beispiel

C1:	$A(x,y) \Leftarrow P(x,y)$	C4:	$P(b,c)$
C2:	$A(x,z) \Leftarrow P(x,y) \wedge A(y,z)$	C5:	$P(c,d)$
C3:	$P(a,b)$	Anfrage	$\neg A(a,v) \vee \neg P(v,d)$



Suchbaum

Selektionsfunktion:

Am wenigsten instantiierte Literale zuerst