Logische und funktionale Programmierung

Vorlesung 9: Resolution

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Wiederholung

- Prädikatenlogik: Signatur, Terme, Formeln
- Substitutionen
- Semantik der Prädikatenlogik: Interpretation, Erfüllbarkeit, Folgerbarkeit
- Pränex-Normalform, Skolem-Normalform, Herbrand-Struktur, Herbrand Expansion
- Algorithmus von Gilmore

Wiederholung

Resolution

- Aussagenlogische Resolution
- Grundresolutionsalgorithmus
- Unifikation

Wiederholung

Unifikation

Definition

Eine Klausel $K = \{L_1, \ldots, L_n\}$ ist unifizierbar gdw. es eine Substitution σ mit $\sigma(L_1) = \cdots = \sigma(L_n)$ gibt (d.h., $\mid \sigma(K) \mid = 1$). Solch eine Substitution heißt ein Unifikator von K. Ein Unifikator σ heißt allgemeinster Unifikator (most general unifier, mgu), falls es für jeden Unifikator Σ_0 eine Substitution δ gibt mit $\Sigma_0(X) = \delta(\sigma(X))$ für alle $X \in \mathcal{V}$.

- Falls eine Klausel unifizierbar ist, so existiert auch ein allgemeinster Unifikator, der bis auf Variablenumbenennungen eindeutig ist.

Wiederholung

Unifikationsalgorithmus

1. *Sei $\sigma = \varnothing$ die leere (oder "identische") Substitution.*

2. *Falls $|\sigma(K)| = 1$ ist, dann brich ab und gib σ als mgu von K aus.*

3. *Sonst durchsuche alle $\sigma(L_i)$ parallel von links nach rechts, bis in zwei Literalen die gelesenen Zeichen verschieden sind.*

4. *Falls keines der beiden Zeichen eine Variable ist, dann brich mit* Clash Failure *ab.*

5. *Sonst sei X die Variable und t der Teilterm im anderen Literal (hierbei kann t auch eine Variable sein). Falls X in t vorkommt, dann brich mit* Occur Failure *ab. (Diese Überprüfung bezeichnet man als* Occur Check.*)*

6. *Sonst setze $\sigma = \{X/t\} \circ \sigma$ und gehe zurück zu Schritt 2.*

Terminierung und Korrektheit des Unifikationsalgorithmus

Satz

Der Unifikationsalgorithmus terminiert für jede Klausel K und er ist korrekt, d.h., er liefert einen mgu für die Klausel K gdw. K unifizierbar ist.

Beweis

- Die Terminierung des Algorithmus folgt, da in jedem Durchlauf der Schleife von Schritt 2 - 6 die Zahl der Variablen in $\sigma(K)$ um 1 abnimmt.
- Falls der Algorithmus mit Erfolg terminiert und eine Substitution σ ausgibt, so ist σ offensichtlich ein Unifikator von K, da $\mid \sigma(K) \mid = 1$ gilt.
- Sofern die Klausel K also nicht unifizierbar ist, muss der Algorithmus daher mit einem Clash oder Occur Failure abbrechen.

Beweis

- Es bleibt zu zeigen, dass bei jeder unifizierbaren Klausel auch tatsächlich ein Unifikator gefunden wird und dass dieser Unifikator dann auch ein allgemeinster Unifikator ist.
- Sei $m \geq 0$ die Anzahl der Schleifendurchläufe, die bei Eingabe der Klausel K stattfinden.
- Für alle $0 \leq i \leq m$ sei σ_i der Wert von σ nach dem i-ten Schleifendurchlauf.
- Wir zeigen die folgende Behauptung für alle $0 \leq i \leq m$:
  (*) Für jeden Unifikator σ' von K gilt $\sigma' = \sigma' \circ \sigma_i$.

Beweis

- Aus der Behauptung (*) folgt, dass zum Schluss nicht mit Clash oder Occur Failure abgebrochen werden kann.
- Denn wurde im (m + 1)-ten Schleifendurchlauf abgebrochen, so wäre $\sigma_m(K)$ nicht unifizierbar.
- Da aber K unifizierbar ist, hat K einen Unifikator $\sigma' = \sigma' \circ \sigma_m$. Somit ist σ' auch Unifikator von $\sigma_m(K)$.
- Da die Schleife nur m mal durchlaufen wird, muss dann also $\mid \sigma_m(K) \mid = 1$ gelten, d.h., σ_m ist Unifikator von K.
- Da außerdem für jeden Unifikator σ_0 von K eine Substitution $\delta = \sigma_0$ mit $\sigma_0 = \delta \circ \sigma_m$ existiert, ist σ_m dann auch mgu von K.

Beweis

- Wir beweisen nun die Behauptung (*) durch Induktion über i.
- Im Induktionsanfang $i = 0$ ist $\sigma_0 = \varnothing$ die Identität. Daher gilt dann auch $\sigma' = \sigma' \circ \sigma_0$ für alle σ'.
- Im Induktionsschritt $i > 0$ wird im i-ten Schleifendurchlauf eine Variable X im einen Literal und ein Term t im anderen Literal gefunden und es ergibt sich $\sigma_i = \{X/t\} \circ \sigma_{i-1}$.
- Für jeden Unifikator σ' von K gilt nach der Induktionshypothese $\sigma' = \sigma' \circ \sigma_{i-1}$.
- Damit folgt:

Beweis

- $\sigma' \circ \sigma_i$
  $= \sigma' \circ \{X/t\} \circ \sigma_{i-1}$ (Def. von σ_i)
  $= \sigma' \circ \sigma_{i-1}$ (da $\sigma' \circ \{X/t\} = \sigma'$)

Beweis

- Um $\sigma' \circ \{X/t\} = \sigma'$ zu zeigen, erkennt man zunächst, dass die beiden Substitutionen auf allen Variablen $Y \neq X$ offensichtlich identisch sind.
- Bei der Variablen X ergibt sich

$$(\sigma' \circ \{X/t\})(X) = \sigma'(t) = \sigma'(X),$$

da σ' Unifikator von $\sigma_{i-1}(K)$ ist (denn $|\ \sigma'(K)\ | = |\ \sigma'(\sigma_{i-1}(K))\ | = 1$) und jeder Unifikator von $\sigma_{i-1}(K)$ auch die Terme X und t unifizieren muss.

Prädikatenlogische Resolution

Definition

Seien K_1 und K_2 Klauseln. Dann ist die Klausel R Resolvent von K_1 und K_2 gdw. die folgenden drei Bedingungen gelten:

- *Es gibt Variablenumbenennungen v_1 und v_2, so dass $v_1(K_1)$ und $v_2(K_2)$ keine gemeinsamen Variablen enthalten.*
- *Es gibt Literale $L_1, \ldots, L_m \in v_1(K_1)$ und Literale $L'_1, \ldots, L'_n \in v_2(K_2)$ mit $n, m \geq 1$, so dass $\{L_1, \ldots, L_m, L'_1, \ldots, L'_n\}$ mit einem mgu σ unifizierbar ist.*
- $R = \sigma((v_1(K_1) \setminus \{L_1, \ldots, L_m\} \cup (v_2(K_2) \setminus \{L'_1, \ldots, L'_n\}))$

Prädikatenlogische Resolution

- Für eine Klauselmenge $\mathcal{K}$ definieren wir wie bei der aussagenlogischen Resolution:
- $Res(\mathcal{K}) = \mathcal{K} \cup \{R \mid R \text{ ist Resolvent zweier Klauseln aus } K\}$
- $Res^0(\mathcal{K}) = \mathcal{K}$
- $Res^{n+1}(\mathcal{K}) = Res(Res^n(\mathcal{K}))$ für alle $n \geq 0$
- $Res^*(\mathcal{K}) = \bigcup_{n \geq 0} Res^n(\mathcal{K})$

Prädikatenlogische Resolution

- Offensichtlich ist die aussagenlogische Resolution ein Spezialfall der prädikatenlogischen Resolution, denn bei variablenfreien Klauseln fällt die Definition der prädikatenlogischen Resolution mit der Definition der aussagenlogischen Resolution zusammen.
- Analog zur aussagenlogischen Resolution gilt $\square \in Res(\mathcal{K})$ gdw. es eine Folge von Klauseln $K_1, \ldots, K_m$ gibt, so dass $K_m = \square$ ist und so dass für alle $1 \leq i \leq m$ gilt:
  - $K_i \in \mathcal{K}$ oder
  - K_i ist eine Resolvente von K_j und K_t für $j, t < i$.

Wiederholung: Aussagenlogische Resolution

Resolution: Idee

Beobachtung

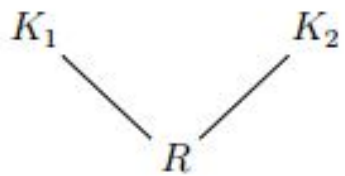
- Prüfen auf Allgemeingültigkeit kann auf Prüfen von Unerfüllbarkeit zurückgeführt werden.
- Formel ϕ allgemeingültig gdw. $\neg\phi$ unerfüllbar.

Resolution: Idee

- Methode, um Unerfüllbarkeit einer Formel ϕ zu prüfen.
- Idee: Leite aus ϕ neue Formeln ab, die aus ϕ logisch folgen.
- Wenn leere Klausel $\square$ abgeleitet werden kann, dann ist ϕ unerfüllbar.

Prädikatenlogische Resolution

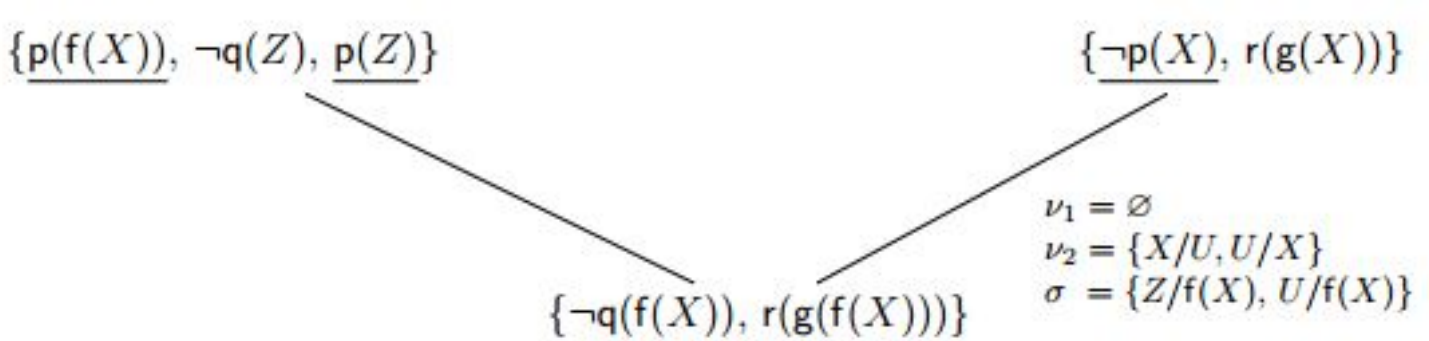
Zur Darstellung von Resolutionsbeweisen schreiben wir wieder das folgende Diagramm, um deutlich zu machen, dass R durch Resolution aus K_1 und K_2 entsteht.



Eine genauere Darstellung ist möglich, indem man die eliminierten Literale unterstreicht und die Umbenennungen und den Unifikator explizit angibt.

BEISPIEL

$p, q, r \in \Delta_1, f, g \in \Sigma_1$



Prädikatenlogisches Resolutionslemma

Lemma

Sei $\mathcal{K}$ eine Menge von Klauseln. Falls $K_1, K_2 \in \mathcal{K}$ und R Resolvente von K_1 und K_2 ist, dann sind $\mathcal{K}$ und $\mathcal{K} \cup \{R\}$ äquivalent.

Beweis

- Wie in der Aussagenlogik folgt aus $S \models \mathcal{K} \cup \{R\}$ trivialerweise $S \models \mathcal{K}$ für alle Strukturen S.
- Es genügt, hierbei Strukturen statt Interpretationen zu betrachten, da jede Klausel die allquantifizierte Disjunktion ihrer Literale repräsentiert und jede Klauselmenge die Konjunktion ihrer Klauseln.
- Somit gilt $\mathcal{K} \cup \{R\} \models \mathcal{K}$.

Beweis

- Umgekehrt sei nun S eine Struktur, die $\mathcal{K}$ erfüllt.
- Es gilt
  $R = \sigma((v_1(K_1) \setminus \{L_1, \ldots, L_m\} \cup (v_2(K_2) \setminus \{L'_1, \ldots, L'_n\}))$.
  Hierbei sind v_1 und v_2 Variablenumbenennungen, so dass $v_1(K_1)$ und $v_2(K_2)$ keine gemeinsamen Variablen enthalten.
- Außerdem sind $L_1, \ldots, L_m \in v_1(K_1)$ und $'_1, \ldots, L'_n \in v_2(K_2)$ mit $n, m \geq 1$, so dass $\{L_1, \ldots, L_m, L'_1, \ldots, L'_n\}$ mit dem mgu σ unifizierbar sind.
- Es gilt also $\sigma(L_1) = \cdots = \sigma(L_m) = L$ und $\sigma(L'_1) = \cdots = \sigma(L'_n) = L$ für ein Literal L.

Beweis

Wir nehmen an, dass $S \not\models \mathcal{K} \cup \{R\}$ gilt. Aus $S \models \mathcal{K}$ folgt dann $S \not\models R$. Sei

$$\nu_1(K_1) = \{L_1, \ldots, L_m, L_{m+1}, \ldots, L_p\} \qquad \text{und} \qquad \nu_2(K_2) = \{L'_1, \ldots, L'_n, L'_{n+1}, \ldots, L'_q\}$$

mit $p \geq m$ und $q \geq n$. Dann entsteht R durch Allquantifizierung der Formel

$$\sigma(L_{m+1} \vee \ldots \vee L_p \vee L'_{n+1} \vee \ldots \vee L'_q).$$

Sei $S = (\mathcal{A}, \alpha)$. Es existiert also eine Interpretation $I = (\mathcal{A}, \alpha, \beta)$ mit

$$I \not\models \sigma(L_{m+1} \vee \ldots \vee L_p \vee L'_{n+1} \vee \ldots \vee L'_q).$$

Beweis

- Sei $\sigma = \{X_1/t_1, \ldots, X_k/t_k\}$ und sei I' die Interpretation $I[\![X_1/I(t_1), \ldots, X_k/I(t_k)]\!]$.
- Nach dem Substitutionslemma gilt dann

$$(**) \qquad I' \not\models L_{m+1} \vee \cdots \vee L_p \vee L'_{n+1} \vee \cdots \vee L'_q.$$

- Da aber $S \models K_1$ und $S \models K_2$ gilt, folgt auch $S \models v_1(K_1)$ und $S \models v_2(K_2)$ und somit $I' \models L_1 \vee \cdots \vee L_m \vee L_{m+1} \vee \cdots \vee L_p$ und $I' \models L'_1 \vee \cdots \vee L'_n \vee L'_{n+1} \vee \cdots \vee L'_q$.

Beweis

- Mit (**) folgt $I' \vDash L_1 \vee \cdots \vee L_m$ und $I' \vDash L'_1 \vee \cdots \vee L'_n$
- Mit dem Substitutionslemma ergibt sich $I \vDash \sigma(L_1 \vee \cdots \vee L_m)$ und $I \vDash \sigma(L'_1 \vee \cdots \vee L'_n)$.
- Wir erhalten also den folgenden Widerspruch: $I \vDash L$ und $I \vDash \overline{L}$.

Resolutionsalgorithmus

- Überführe $\mathcal{K}$ in KNF, bzw. in die entsprechende Klauselmenge $\mathcal{K}(\psi)$.
- Berechne $Res^*(\mathcal{K}(\psi))$. Sofern die leere Klausel gefunden wurde, brich ab und gib true zurück. Sofern $Res(\mathcal{K}(\psi))$ komplett berechnet wurde, brich ab und gib false zurück.

- Dieser SemiEntscheidungsalgorithmus kann auch mit false abbrechen, denn falls $Res^*(\{K(\psi)\})$ endlich ist und die leere Klausel nicht enthält, folgt die Erfüllbarkeit der Klauselmenge $K(\psi)$ aus der Vollständigkeit des Resolutionskalküls.
- Im Allgemeinen ist $Res^*(\{K(\psi)\})$ allerdings unendlich und dann terminiert der obige Algorithmus bei erfüllbaren Klauselmengen $K(\psi)$ nicht.

Die Effizienz dieses Algorithmus läßt sich allerdings noch weiter verbessern, denn bislang müssen noch alle möglichen Resolutionsschritte zwischen allen Klauseln betrachtet werden. Hierbei müssen natürlich auch die Klauseln betrachtet werden, die im Laufe der Resolution entstanden sind.

Einschränkungen der Resolution

- Das Problem beim Unerfüllbarkeitsnachweis durch Resolution ist, dass es sehr viele Resolutionsmöglichkeiten gibt, was zu einer kombinatorischen Explosion führt.
- Unser Ziel ist daher, den Suchraum durch Verwendung spezieller Resolutionsstrategien zu verkleinern, ohne jedoch dabei die Vollständigkeit zu verlieren.
- Lineare Resolution
- Input Resolution: vollständig nur für Horn Klauselmengen.
- SLD Resolution: wird bei der Logikprogrammierung verwendet.

Lineare Resolution

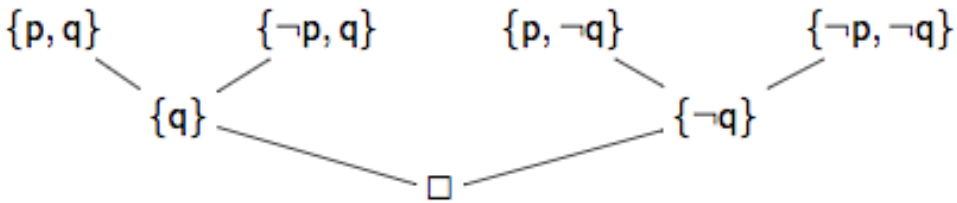
- Die Idee der linearen Resolution besteht darin, dass bei jedem Resolutionsschritt eine der beiden Elternklauseln der zuletzt erzeugte Resolvent sein muss.

Lineare Resolution

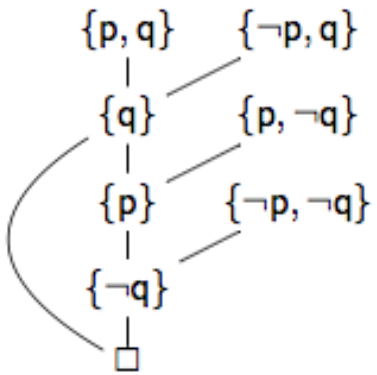
Definition

Sei $\mathcal{K}$ eine Klauselmenge. Die leere Klausel $\square$ ist aus der Klausel K in $\mathcal{K}$ linear resolvierbar gdw. es eine Folge von Klauseln $K_1, \ldots, K_m$ gibt, so dass $K_1 = K \in \mathcal{K}$ und $K_m = \square$ ist und so dass für alle $2 \leq i \leq m$ gilt: K_i ist ein Resolvent von K_{i-1} und einer Klausel aus $\{K_1, \ldots, K_{i-1}\} \cup \mathcal{K}$.

Beispiel



Diese Herleitung entspricht aber keinem linearen *Resolutionsbeweis, denn wenn man im ersten Schritt den Resolvent* {q} *bildet, so müsste dieser Resolvent dann eine der Elternklauseln im nächsten Schritt sein. Eine Herleitung der leeren Klausel mit linearer Resolution wäre in diesem Beispiel wie folgt möglich:*



Korrektheit und Vollständigkeit der linearen Resolution

Sei $\mathcal{K}$ eine Menge von Klauseln. Dann ist $\mathcal{K}$ unerfüllbar gdw. $\square$ aus einer Klausel K in $\mathcal{K}$ linear resolvierbar ist. Falls $\mathcal{K}$ eine minimale unerfüllbare Menge ist (d.h., falls für jedes $K \in \mathcal{K}$ die Menge $\mathcal{K} \setminus \{\mathcal{K}\}$ erfüllbar ist), dann ist $\square$ sogar aus jeder Klausel K in $\mathcal{K}$ linear resolvierbar.

Input Resolution

- Um die Möglichkeiten der Resolution noch weiter zu reduzieren, schränken wir die lineare Resolution nun noch weiter ein.
- Bei der linearen Resolution muss eine der Elternklauseln in jedem Resolutionsschritt der letzte Resolvent sein. Die andere Elternklausel konnte jedoch noch frei gewählt werden (d.h., es konnte eine Klausel aus der ursprünglichen Klauselmenge oder aber ein bereits früher gebildeter Resolvent sein).
- Wir verbieten nun die letzte Möglichkeit: nun muss in jedem Schritt zwischen dem zuletzt gebildeten Resolvent und einer der ursprünglichen Eingabe - Klauseln resolviert werden.
- Aus diesem Grund bezeichnet man diese Einschränkung als Input - Resolution.

Definition

Sei $\mathcal{K}$ eine Klauselmenge. Die leere Klausel $\square$ ist aus der Klausel K in $\mathcal{K}$ durch Input - Resolution herleitbar gdw. es eine Folge von Klauseln $K_1, \ldots, K_m$ gibt, so dass $K_1 = K \in \mathcal{K}$ und $K_m = \square$ ist und so dass für alle $2 \leq i \leq m$ gilt: K_i ist ein Resolvent von K_{i-1} und einer Klausel aus $\mathcal{K}$.

Aus der Definition folgt sofort, dass die Input - Resolution ein Spezialfall der linearen Resolution ist, d.h., jeder Input - Resolutionsbeweis ist auch ein linearer Resolutionsbeweis. Der Umkehrschluss gilt natürlich nicht. In der Tat ist die Input - Resolution im Unterschied zur linearen Resolution nicht mehr vollständig, wie das folgende Beispiel zeigt.

Beispiel

- Wir betrachten das Beispiel auf S. 54:
- Während wir dort die Unerfüllbarkeit mit linearer Resolution nachweisen konnten, gelingt dies mit Input - Resolution nicht. Durch Resolution zweier Klauseln aus der ursprünglichen Klauselmenge lassen sich nur die folgenden Klauseln herleiten:

$$\{q\}, \{\neg q\}, \{p\}, \{\neg p\}, \{q, \neg q\}, \{p, \neg p\}.$$

BEISPIEL

- Falls man eine der ersten vier entstandenen Klauseln wieder mit einer Klausel der ursprünglichen Menge resolviert, so entsteht wieder eine der ersten vier Klauseln. Wenn man eine der letzten beiden (allgemeingültigen) Klauseln mit einer Klausel der ursprünglichen Menge resolviert, so erhält man die Klausel der ursprünglichen Menge.
- Eine Herleitung der leeren Klausel ist also nur dann möglich, wenn man zwei der durch Resolution entstandenen Klauseln (wie z.B. $\{q\}$ und $\{\neg q\}$) miteinander resolviert. Dies ist jedoch bei Input - Resolution nicht zulässig.

- Die Input-Resolution ist auf Horn-Klauseln vollständig!

Horn-Klauseln

Definition

Eine Klausel K ist eine Hornklausel gdw. sie höchstens ein positives Literal enthält (d.h., höchstens eines ihrer Literale ist eine atomare Formel und die anderen Literale sind negierte atomare Formeln). Eine Hornklausel heißt negativ, falls sie nur negative Literale enthält (d.h., falls sie die Gestalt $\{\neg A_1, \ldots, \neg A_k\}$ für atomare Formeln $A_1, \ldots, A_k$ hat). Eine Hornklausel heißt definit, falls sie ein positives Literal enthält (d.h., falls sie die Gestalt $\{B, \neg C_1, \ldots, \neg C_n\}$ für atomare Formeln $B, C_1, \ldots, C_n$ hat).

- Eine Menge definiter Hornklauseln entspricht einer Konjunktion von Implikationen
- $\{\{p, \neg q\}, \{\neg r, \neg p, s\}, \{s\}\}$ ist äquivalent zur Formel

$$((p \vee \neg q) \wedge (\neg r \vee \neg p \vee s) \wedge s)$$

und damit zur folgenden Formel

$$(q \to p) \wedge (r \wedge p \to s) \wedge s).$$

Zusammenhang zur Logikprogrammierung

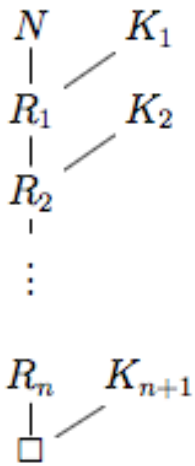
- Fakten sind definite Hornklauseln ohne negative Literale (d.h., sie enthalten genau ein positives Literal). Ein Beispiel ist die Klausel $\{s\}$. In der Logikprogrammierung schreibt man: `s.`
- Regeln sind definite Hornklauseln mit negativen Literalen. Ein Beispiel ist die Klausel $\{\neg r, \neg p, s\}$. In der Logikprogrammierung schreibt man: `s :- r, p.`
- Anfragen sind negative Hornklauseln. Ein Beispiel ist die Klausel $\{\neg p, \neg q\}$. In der Logikprogrammierung schreibt man: ?- p,q.

SLD-Resolution

Definition

Sei $\mathcal{K}$ eine Hornklauselmenge mit $\mathcal{K} = \mathcal{K}^d \uplus \mathcal{K}^n$, wobei $\mathcal{K}^d$ die definiten Klauseln und $\mathcal{K}^n$ die negativen Klauseln von $\mathcal{K}$ enthält. Die leere Klausel $\square$ ist aus der Klausel K in $\mathcal{K}^n$ durch SLD-Resolution herleitbar gdw. es eine Folge von Klauseln $K_1, \ldots, K_m$ gibt, so dass $K_1 = K \in \mathcal{K}^n$ und $K_m = \square$ ist und so dass für alle $2 \leq i \leq m$ gilt: K_i ist ein Resolvent von K_{i-1} und einer Klausel aus K_d.

Man erkennt sofort, dass in einer SLD–Resolution alle Klauseln $K_1, \ldots, K_m$ negativ sind. SLD–Resolutionen haben also die folgende Gestalt:



Hierbei sind $K_1, \ldots, K_{n+1} \in \mathcal{K}^d$ definite Hornklauseln aus der Eingabemenge, $N \in \mathcal{K}^n$ ist eine negative Hornklausel aus der Eingabemenge und die Resolventen $R_1, \ldots, R_n$ sind ebenfalls negative Hornklauseln.

Korrektheit und Vollständigkeit der SLD-Resolution

Sei $\mathcal{K}$ eine Menge von Hornklauseln. Dann ist $\mathcal{K}$ unerfüllbar gdw. $\square$ aus einer negativen Klausel N in $\mathcal{K}$ durch SLD - Resolution herleitbar ist.