Eigenschaften der Resolution für PL1 Formeln

Widerlegungsvollständigkeit (ohne Beweis):

- Sofern man Resolution auf eine widersprüchliche Klauselmenge anwendet, so existiert eine endliche Folge von Resolutionsschritten, mit denen man den Widerspruch aufdecken kann

  Aber: die Anzahl *n* der Beweisschritte kann sehr groß werden und damit das Verfahren unpraktikabel machen.

Kein Entscheidungsverfahren:

- Terminierung nicht garantiert, falls Klauselmenge nicht widersprüchlich.

Maßnahmen zur Effizienzsteigerung

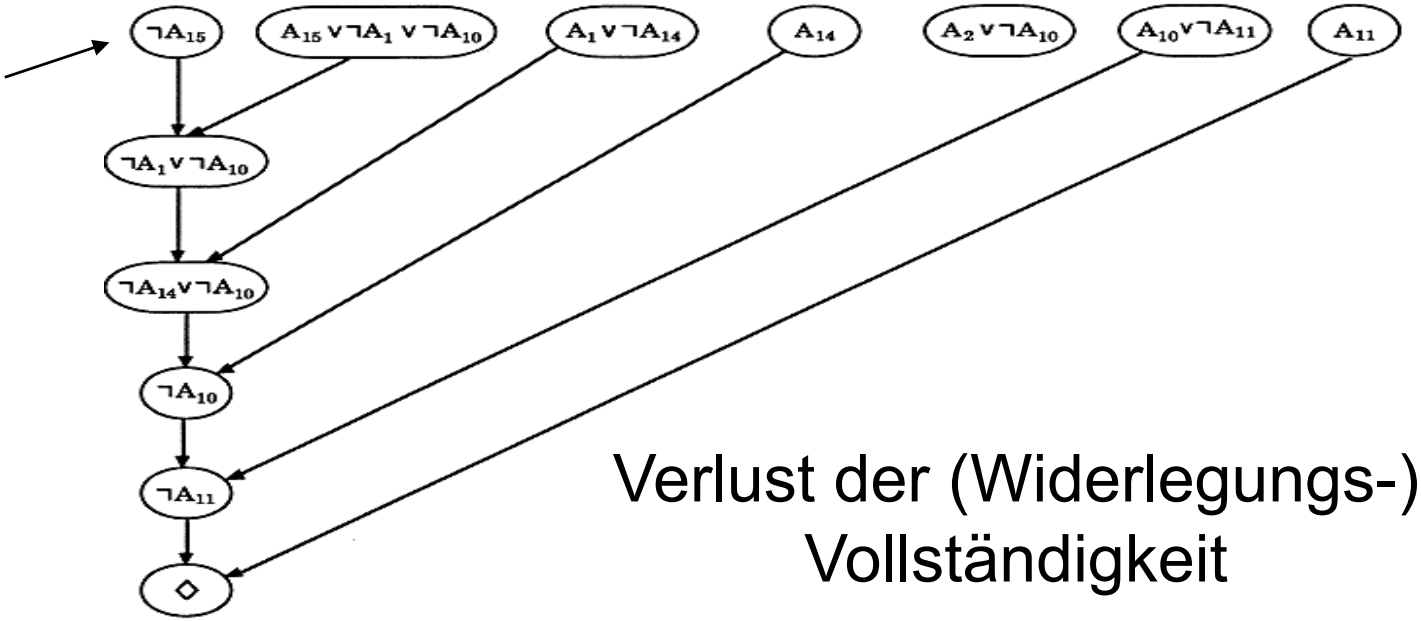
Ansatz 1: Entwicklung von Heuristiken zur Klauselauswahl

- Idee: man betrachtet die Klauselauswahl als Suchproblem und setzt „intelligente“ Suchverfahren ein.

  Beispiele für mögliche Strategien:

  - bevorzuge möglichst kurze Klauseln
  - entferne Klauseln, die von anderen „subsumiert“ werden.
  - **Set-of-Support-Strategie:** Die Resolution wird stets auf eine der Klauseln gerichtet, die von der negierten Behauptung abstammen.

negierte Behauptung



Verlust der (Widerlegungs-) Vollständigkeit

Maßnahmen zur Effizienzsteigerung

Ansatz 2: Einschränkung der Ausdrucksmächtigkeit

- Idee: Für viele Anwendungen benötigt man eine Ausdrucksmächtigkeit, die zwischen Aussagenlogik und PL1 liegt.

- Einschränkung auf eine weniger ausdrucksmächtige Sprache, z.B.:
  - nur allquantifizierte Variablen
  - keine Disjunktion in Termen

- **In der Praxis:**
  - Einschränkung auf praktisch wichtige Spezialform, die sog. **Hornklauseln** (benannt nach dem Logiker Alfred Horn).

Hornklauseln

- **Hornklauseln** sind Klauseln ***mit höchstens einem positiven*** Literal

- Es gibt drei Arten von Hornklauseln:

  i) **Regel:** $\mathbf{P \leftarrow P_1 \wedge P_2 \wedge \ldots \wedge P_n}$

  ist äquivalent zu $\mathbf{\{P, \neg P_1, \neg P_2, \ldots, \neg P_n\}}$

  - in Hornklauseln steht das positive Literal immer vorne
  - das positive Literal P bezeichnet man als ***Konklusion***
  - die negativen Literale $\neg P_1, \neg P_2, \ldots, \neg P_n$ heißen ***Prämissen***
  - das Zeichen $\leftarrow$ ist angelehnt an das Zeichen $\Rightarrow$ (log. Implikation)

  ii) **Fakt:** **P** ist äquivalent zu {P}

  iii) **Zielklausel:** $\mathbf{\leftarrow P_1 \wedge P_2 \wedge \ldots \wedge P_n}$ ist äquivalent zu $\{\neg P_1, \neg P_2, \ldots, \neg P_n\}$

  auch: Integritätsbedingung (integrity constraint)

  - kein positives Literal

Hornklauseln

Eine typische Hornklausel-Wissensbasis:

```
man(X) ← human(X) ∧ male(X)
woman(X) ← human(X) ∧ female(X)
parent(X,Y) ← mother(X,Y)
parent(X,Y) ← father(X,Y)
ancestor(X,Y) ← parent(X,Y)
ancestor(X,Y) ← parent(X,Z)∧ ancestor(Z,Y)

human(john)
human(paul)
human(mary)
male(john)
male(paul)
female(mary)
father(john,mary)
mother(mary,paul)
```

Regeln

(Regelbasis)

Fakten

(Faktenbasis)

Hornklauseln

Beispiele für mögliche Anfragen:

```
human(john)
                        Resultat: true      ;;; Wahrheitswert
female(john)
                        Resultat: unknown
female(peter)
                        Resultat: unknown
human(X)
                        Resultat: X = john    oder    ;;; Belegungen, für
                                  X = paul     oder   ;;; die Anfrage wahr
                                  X = mary            ;;; wird
man(john)
                        Resultat: yes
man(X)
                        Resultat: X = john     oder
                                  X = paul
parent(mary,X)
                        Resultat: X = paul
ancestor(X,Y) ∧ male(X)
                        Resultat: X = john, Y = mary   oder
                                  X = john, Y = paul
```

Beantwortung von Anfragen

Gegeben:

- Wissensbasis mit Regeln, Fakten, Integritätsbedingungen

Anfrage:

- Zu beweisen ist die Gültigkeit einer Aussage: $\mathbf{P_1 \wedge P_2 \wedge \ldots \wedge P_n}$

Ansatz:

- Mit Widerspruchsbeweis (Resolution), wobei man von der negierten Aussage ausgeht:

  $\mathbf{\neg(P_1 \wedge P_2 \wedge \ldots \wedge P_n)}$ ist äquivalent zu Zielklausel

  $\mathbf{\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_n}$ (in Hornklauselschreibweise: $\mathbf{\leftarrow P_1 \wedge P_2 \wedge \ldots \wedge P_n}$)

- Der Widerspruch ist dann aufgedeckt, wenn man die „leere Klausel“ $\_ \leftarrow \_$ ableiten kann.

Beispiel: Beantwortung von Anfragen

Gegeben:

- Wissensbasis mit:
  - i) einer Regel: **sterblich(X) ← mensch(X)**
    (Mengenschreibweise: {**sterblich(X) , ¬mensch(X)** })
  - ii) einem Faktum: **mensch(sokrates) ←**
    (Mengenschreibweise: { **mensch(sokrates)** }

Anfrage:

- Gilt, dass Sokrates sterblich ist? Also: folgt **sterblich(sokrates)** aus der Wissensbasis?
- Also füge hinzu **← sterblich(sokrates)**
  (Mengenschreibweise: { **¬ sterblich(sokrates)** }
- Führe Widerspruchsbeweis., d.h. leite aus der Klauselmenge die leere Klausel ab:

$$\left\{ \begin{array}{l} \{\textbf{sterblich(X) , ¬mensch(X)}\}, \\ \{\textbf{mensch(X)}\}, \\ \{\ \textbf{¬ sterblich(sokrates)}\ \} \end{array} \right\} \models \square$$

(als Hornklausel: _ ← _)

Inferenzregel für Hornklauseln

- **Inferenzregel (OLD-Resolution)**

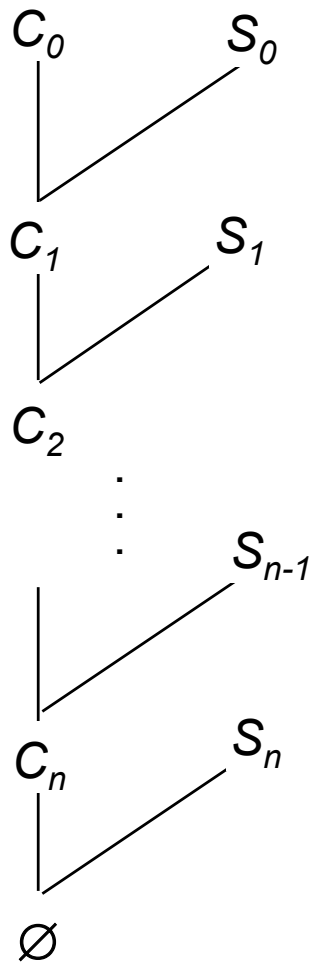
$\mathbf{H_1}$ mit $\mathbf{H}$ unifizierbar: $\mathbf{H_1\sigma = H\sigma}$

$$\frac{\{\neg\mathbf{H_1}, \neg H_2, \ldots, \neg H_m,\} \quad , \quad \{\mathbf{H}, \neg B_1, \neg B_2, \ldots, \neg B_n\}}{\{\neg\mathbf{H_2}, \ldots, \neg\mathbf{H_m}, \neg\mathbf{B_1}, \neg\mathbf{B_2}, \ldots, \neg\mathbf{B_n}\}\ \sigma}$$

- **Ordered-Linear Resolution for Definite Clauses**

  - starte mit negierter Anfrage als einer Klausel C_0
  - Lineare Resolution: nimm im nächsten Resolutionsschritt die Resolvente c_i als eine der beiden Klauseln (vgl. Set-of-Support-Strategie) und resolviere mit geeigneter Seitenklausel S_i
  - Literale einer Klausel sind geordnet, resolviere mit dem ersten Literal.

  => in jedem Schritt wird ein negatives Literal abgearbeitet. Stopp bei leerer Resolvente



C_i: Center Clauses (Zentrumsklauseln)

S_i: Side Clauses (Seitenklauseln)

OLD-Resolution

- Eine Regel ist anwendbar, wenn die *Konklusion* der Regel mit der Anfrage unifiziert werden kann:
  - Variablen werden mit Konstanten belegt
  - eine Variable muss bei jedem Vorkommen den gleichen Wert haben.
- Enthält die Anfrage Variablen, so werden gültige Werte der Variablen berechnet.

- Ein Literal der Anfrage wird beantwortet (bewiesen), wenn ein Fakt in der Datenbasis existiert, der mit der Anfrage unifiziert werden kann.

- Falls für ein Literal der Anfrage keine anwendbare Regel oder Fakt gefunden wird, ist das Literal nicht beweisbar (vgl. Closed-World Assumption)

  => Rücksetzen (Backtracking) und mit alternativer Belegung für die zu unifizierenden Variablen.

- Die OLD-Resolution nennt man auch Rückwärtsverkettung, da die Regeln rückwärts angewendet werden.
- Die Rückwärtsverkettung ist abgeschlossen, wenn die Liste der Anfragen leer ist. Das Ergebnis ist die Menge der Variablen mit ihren Werten (die Substitution).

Zusammenfassung: Hornklauseln

Pro

- Mit Hornklauseln lassen sich sog. Wissensbasen aufbauen, in denen Wissen unterteilt wird in
  - Fakten (zur Repräsentation von Objekten und Sachverhalten)
  - Regeln und Integritätsbedingungen (zur Repräsentation von Beziehungen, Abhängigkeiten, Kausalitäten, Strukturbeziehungen, usw.).
  - Verwendung von Variablen in Regeln (und Fakten) erlaubt die Abstraktion von konkreten Instanzen (Vorteil gegenüber der Aussagenlogik).

- Anfragen an die Wissensbasis werden als spezielle Klauseln formuliert (:- P) und mittels eines linearen Resolutionsbeweises beantwortet. (Günstig für eine Automatisierung).
- Der Hornklauselkalkül ist widerlegungsvollständig

Contra

- Weniger ausdruckstark als Prädikatenlogik 1. Stufe
  - keine alternativen Konklusionen der Form $A \vee B \leftarrow C$
  - keine negierten Prämissen (bzw. Anfragen der Form: $\leftarrow \neg C$)

Anwendung der Hornklausel-Logik

Bedeutung der Hornklausellogik:

- Grundlage für Sprachen zur logischen Programmierung (insbesondere Prolog)
- Grundlage deduktiver Datenbanken

PROLOG: PROgramming in **LOG**ic

- basiert auf Hornklauseln und (linearem) Resolutionsbeweis:

  (Genauer: SLD-Resolution:

  Linear resolution with **S**election function for **D**efinite clauses)
- Existenzquantor entfällt, Allquantor wird implizit vorausgesetzt.
- Terme als Datenstrukturen
  - Variablen
  - Konstante
  - Funktionssymbol mit Argumentliste (wiederum Terme)