Eigenschaften der SLD-Resolution

Vollständigkeit der SLD-Resolution für Hornklauseln

Sei $\mathcal{F}$ eine inkonsistente Hornklauselmenge.
Dann gibt es eine SLD-Widerlegung von $\mathcal{F}$.

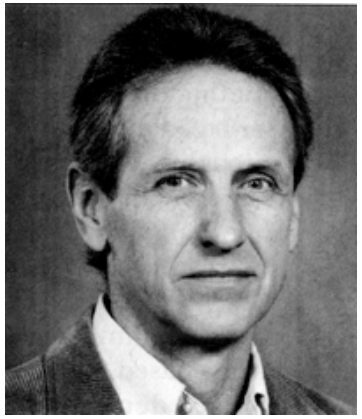
Beweisskizze: Für inkonsistentes $\mathcal{F}$ muss es eine minimale inkonsistente Teilmenge $\mathcal{F}'$ geben, die eine negative Klausel N enthält.

Für $\mathcal{F}'$ gibt es eine Widerlegung durch Input-Resolution (s. Vollst.-Satz). Diese Widerlegung lässt sich zu einer SLD-Widerlegung ausgehend von der negativen Klausel N umsortieren.

Wurzeln von PROLOG

SLD-Resolution ist ein Verfahren mit *Ordnung auf Literalen:* Auf welchem soll nächstes resolviert werden?
Hier nicht im Detail, sondern nur in der Variante für PROLOG.

Alain Colmerauer *1941



Robert Kowalski *1941

Interpreter für
Definite Clause Grammars
(DCGs) zur Verarbeitung
natürlicher Sprache

Algorithm = Logic + Control

PROLOG (aussagenlogische Variante)

Programme bestehen aus:

- **Prozedurklauseln** der Form
  `P :- Q1,...,Qk.` (Prozedur**kopf** `P`, -**rumpf** `Q1,...,Qk`)
  (logische Interpretation: $Q_1 \wedge \ldots \wedge Q_k \Rightarrow P$)
- **Faktenklauseln** der Form
  `P.`
  (logische Interpretation: P)
- Einer **Zielklausel** (Frageklausel) der Form
  `?- Q1,..., Qk.`
  (logische Interpretation: $\neg(Q_1 \wedge \ldots \wedge Q_k)$ oder $\neg Q_1 \vee \ldots \vee \neg Q_k$)

Mit Prozedur- und Faktenklauseln programmiert man;
mit der Zielklausel ruft man das Programm auf.

Programmaufruf entspricht Widerlegungsbeweis von $Q_1 \wedge \ldots \wedge Q_k$!

Programmaufruf als Deduktion

Der Aufruf eines PROLOG-Programms entspricht einer SLD-Widerlegung von Programm-, Fakten- und Zielklausel, ausgehend von der Zielklausel!

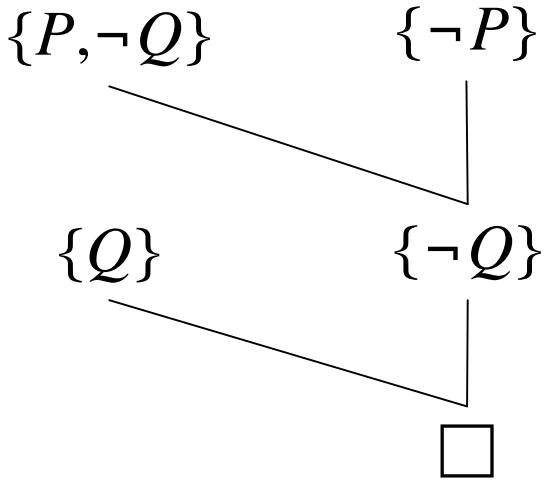
Beispiel

Programm
```
q.
p :- q.
q :- p.
```

Zielklausel
```
?- p.
```

Ergebnis
```
Yes
```

entspricht



Achtung: in SWI-Prolog schreibe Aussagevariablen *klein*!

Literalordnung in PROLOG

Strikte Reihenfolge auf Zielklausel-Literale und Programmklauseln:

- Literale von links nach rechts! (Unterziele links an Zielklausel anfügen)
- Klauseln von oben nach unten!

➥ SLD-Resolution mit Tiefensuch-Strategie

Beispiel

Programm	`q.`
	`p :- q.`
	`q :- p.`
		≠	Programm	`q :- p.`
				`p :- q.`
				`q.`

Zielklausel	`?- p.`		Zielklausel	`?- p.`
Ergebnis	`Yes`		Ergebnis	`ERROR: Out of local stack`

Unvollständigkeit der PROLOG-SLD-Resolution

Sei $\mathcal{P}$ Prolog-Programm aus Fakten- und Programmklauseln, das einer Hornklauselmenge $\mathcal{F}$ entspricht; sei Z Zielklausel, die der negativen Klausel N entspricht, sodass $\mathcal{F} \cup N$ inkonsistent ist.

Dann besteht die Möglichkeit, dass PROLOG nicht mit einer Antwort für Z aus $\mathcal{P}$ terminiert.

Beweis: Siehe Beispiel auf der vorigen Folie.

Bemerkung

Die Literal-Reihenfolge in einer Klausel ist irrelevant für die Vollständigkeit (lediglich Änderung der Resolutions-Reihenfolge).

Es gibt immer eine Klausel-Reihenfolge, mit der die Herleitung der Zielklausel gelingt.

Negation in PROLOG

PROLOG hat ein Negationsprädikat: `not(p)`

Achtung: *Nicht* identisch mit dem Junktor ¬!

negation as failure, genauer: *negation by finite failure*

`not(p)` ist wahr, wenn `p` nicht bewiesen werden kann.

Beispiel

Programm
```
r.
s.
p:-not(r).
```

Zielklausel `?- p.`

Ergebnis `No`

Programm
```
r.
s.
p:-not(r).
p :- s.
```

Zielklausel `?- p.`

Ergebnis `Yes`

Widerspruch zu Monotonie! ➥ kein klassisch-logischer Operator

„Direkte“ Implementierung von Ableitung

Aussagenlogik (endliche Symbolmenge!) erlaubt, Ableitung in Form von Propagierung direkt zu implementieren.
Hornklauseln vereinfachen das weiter.

Dazu gibt es spezielle Datenstrukturen
(z.B. Und/Oder-Graphen, *binary decision diagrams* (BDDs))
und spezielle Algorithmen.
Diese sind nicht ohne Weiteres verallgemeinerbar!

Vorwärtsverkettung für AL-Hornklauseln

- Verwende positive Einsklauseln, um über Implikationen weitere positive Einsklauseln abzuleiten
- Packe dabei jede Aussagevariable maximal einmal an
  - ➥ Laufzeit $O(n)$ in Variablenzahl!
    (bei effizienter Indizierung der Klauseln)
- „Verpacke“ Zielklausel $(Q_1 \wedge \ldots \wedge Q_k)$ als Einsklausel: $(Q_1 \wedge \ldots \wedge Q_k) \Rightarrow Q$ für neue Variable Q.
- Zielklausel ableitbar gdw. Q in abgeleitete Einsklauseln

PL-FC-ENTAILS

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true

  while agenda is not empty do
      p ← POP(agenda)
      unless inferred[p] do
          inferred[p] ← true
          for each Horn clause c in whose premise p appears do
              decrement count[c]
              if count[c] = 0 then do
                  if HEAD[c] = q then return true
                  PUSH(HEAD[c], agenda)
  return false
```

Ein Hornklauselvorwärtsverkettungsbeispiel

Klauselmenge

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

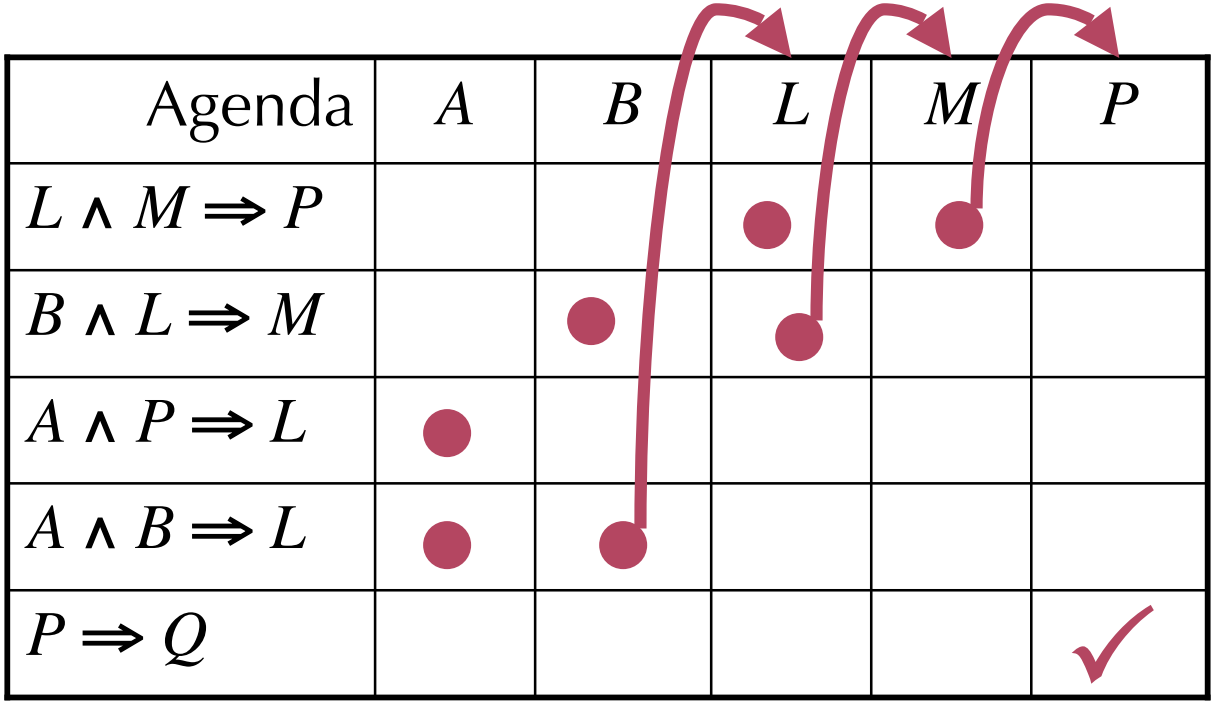
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

$P \Rightarrow Q$ (Ziel)



Verbindung zu DPLL-Algorithmus!

Fazit Aussagenlogik und Resolution

- Aussagenlogik ist ein Formalismus zur Wissensrepräsentation, aber mit schwacher Ausdrucksfähigkeit
- AL hat formal definierte Syntax und Semantik – wie jede Wissensrepräsentationssprache
- In AL sind (semantische) Folgerung und (syntaktische) Ableitbarkeit formal definiert; es gibt korrekte und vollständige Kalküle
- AL-Ableitbarkeit ist entscheidbar. Model-Checking und DP funktionieren. Resolution desgleichen.
- Hornklauseln kommen in der Wissensrepräsentation auf natürliche Weise vor. Es gibt effiziente Kalküle für Ableitbarkeit aus Hornklauselmengen. PROLOG nutzt das aus.

… und was hat das mit KI und Agenten zu tun?

Zur Erinnerung:

„Die KI ist der Teil der Informatik, der mittels algorithmischer Modelle Leistungen des Denkens, Tuns und Wahrnehmens untersucht.“

Zur Erinnerung:

„Ein Agent ist rational, wenn er stets die Aktion ausführt, die seine Performanz unter der aktuellen Perzept-Folge und seinem gegebenen Wissen maximiert.“

z.B. Ziele verfolgender Agent

Von AL zu PL

- Aussagen in der Aussagenlogik haben keine innere Struktur:
  Verknüpfung nur über Wahrheitswert möglich,
  nicht über Inhalt von Aussagen
- Prädikatenlogik kennt darüber hinaus
  - **Objekte/Individuen**: z.B. Menschen, Tiere, Sensationen, Ereignisse, Vorlesungen, Zahlen, Agenten, …
  - **Relationen**: z.B. Ist_Kanzler(.), Geschah_vor(.,.), >(.,.), Haben_unterschiedliche_Farbe(.,.,.), …
  - **Funktionen**: z.B. Kanzler_von(.), Sinus(.), Spielte_im_WM-Finale(.,.), …

Sprachelemente der PL

- **Variable**: x_i (i=1, …)
- **Funktionssymbole**: f_i^k (i=1, …) (k=0, …: Stelligkeit)
  0-stellige Funktionen heißen Konstante
- **Prädikatensymbole**: P_i^k (i=1, …) (k=0, …: Stelligkeit)
- **Junktoren** (wie Aussagenlogik)
- **Quantoren**: ∃ (Existenzquantor), ∀ (Allquantor)

Für Variable, Funktionen und Prädikate verwende im Folgenden auch andere, Sinn tragende Namen.

Syntax der PL

1. Terme

- Eine Variable ist ein Term
- Sind $t_1,\ldots,t_k$ Terme, so ist $f_i^k(t_1,\ldots,t_k)$ ein Term
  Ein variablenfreier Term heißt Grundterm

2. Formeln:

- Ist P_i^k ein Prädikatensymbol und sind $t_1,\ldots,t_k$ Terme, so ist $P_i^k(t_1,\ldots,t_k)$ eine Formel (atomare Formel)
  Formel ohne Variable: Grundformel bzw. Grundatomformel
- Sind α und β Formeln, so auch $\neg\alpha$, $\alpha\vee\beta$, $\alpha\wedge\beta$.
  [Wie in AL: $\alpha\Rightarrow\beta$ steht für $\neg\alpha\vee\beta$, $\alpha\Leftrightarrow\beta$ für $(\alpha\Rightarrow\beta)\wedge(\beta\Rightarrow\alpha)$]
- Ist x eine Variable, α Formel, so sind $\exists x.\alpha$ und $\forall x.\alpha$ Formeln
  Kommt x in α vor, so heißt es gebunden. (Gegenteil: frei)

Beispiel: Formulierungen in PL

Everybody loves somebody sometime. *Irving Taylor/Dean Martin*

$$\forall x.\ \exists y.\ \exists t.\ \text{loves}(x,y,t)$$

oder: $\forall x.\ \exists t.\ \exists y.\ \text{loves}(x,y,t)$

aber nicht: $\exists t.\ \forall x.\ \exists y.\ \text{loves}(x,y,t)$

Wer an der UOS studiert, ist schlau.

$$\forall x.\ \text{studUOS}(x) \Rightarrow \text{schlau}(x)$$

Es gibt eine/n schlaue/n Student/in an der UOS.

$$\exists x.\ \text{studUOS}(x) \wedge \text{schlau}(x)$$