Klauselform

Eine Klausel ist eine Menge von Literalen $K = \{L_1, ..., L_n\}$.
Vorteil der Klauselnotation: Mengenschreibweise beseitigt rein formale Unterschiede äquivalenter Ausdrücke bzgl. Kommutativität/Idempotenz.

Umformung einer Skolem-Normalform F in Klauselform:

1. Verteilung der Allquantoren auf die Alternativen.
   (Semantisch äquivalente Umformung)
2. Gebundenene Umbenennungen derart, dass sich insgesamt
   alle Quantoren auf unterschiedliche Variablen beziehen.
   (Semantisch äquivalente Umformung)
3. Darstellung der Alternativen als Klauseln (Menge von Literalen).
   Darstellung von F als Menge von von Klauseln **Kl** :

$$\mathbf{Kl} = \{ \{ L_{ij} \mid j = 1, ...m_i \} \mid i = 1, ..., n \}$$

Klauselform

Insgesamt:
Umformung eines Ausdrucks H in eine Klauselmenge **Kl**, derart dass
H erfüllbar gdw. **Kl** erfüllbar.

Widerlegung (Nicht-Erfüllbarkeit) von H
durch Widerlegung (Nicht-Erfüllbarkeit) von **Kl**

Klauselform

A1: $\forall x \neg R(x,x)$

A2: $\forall x \, \forall y \, \forall z \, ((R(x,y) \wedge R(y,z)) \rightarrow R(x,z))$

Sem.äq.: $\forall x \, \forall y \, \forall z \, (\neg(R(x,y) \wedge R(y,z)) \vee R(x,z))$

Sem.äq.: $\forall x \, \forall y \, \forall z \, (\neg R(x,y) \vee \neg \, R(y,z) \vee R(x,z))$

Klauseln für A1:

KA1: $\neg R(x,x)$

Klauseln für A2:

KA2: $\neg R(u,y) \, , \neg \, R(y,z), \, R(u,z))$

Klauselform

Negation bei Resolutionsbeweis

H: $\forall x \, \forall y \, (R(x,y) \rightarrow \neg R(y,x))$

$\neg$H: $\neg \forall x \, \forall y \, (R(x,y) \rightarrow \neg R(y,x))$

Sem.äq.: $\exists x \, \exists y \, \neg(\neg \, R(x,y) \vee \neg \, R(y,x))$

Sem.äq.: $\exists x \, \exists y \, (R(x,y) \wedge R(y,x))$

Erf.äq.: $R(c,f(c)) \wedge R(f(c),c)$

Klauseln für $\neg$H:

K1: $R(c,f(c))$

K2: $R(f(c),c)$

Substitution

Substitution σ :

Abbildung der Individuenvariablen in die Menge der Terme.

Variablenumbenennung: Ersetzung von Variablen durch Variable.

$\sigma(L)$: Ergebnis der Substitution σ für ein Literal (rekursiv definiert).

Hintereinanderausführung von Substitutionen:

$$\sigma_1 * \sigma_2 (x) = \sigma_2(\sigma_1(x))$$

$\sigma(L)$ ist jeweils „Spezialfall“ von L .

L ist die allgemeinste Form bzgl. aller $\sigma(L)$ für beliebige σ.

Substitution, Unifikation

- Eine Substitution σ heißt *Unifikator* für eine Menge $\{L_1, ..., L_n\}$ von Literalen, falls gilt: $\sigma(L_1) = \sigma(L_2) = ...= \sigma(L_n)$ (syntaktische Gleichheit).

- Ein Unifikator σ heißt *allgemeinster Unifikator* (m.g.u. = most general unifier), falls für jeden Unifikator σ_0 eine Substitution σ_{00} existiert mit

$$\sigma_0 = \sigma * \sigma_{00} \quad (\sigma_0 \text{ ist spezieller als } \sigma).$$

Der allgemeinste Unifikator ist eindeutig bis auf Umbenennungen.

Substitution, Unifikation

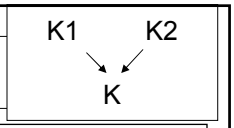
Der allgemeinste Unifikator σ einer Menge $\{ L_1,...,L_n \}$ von Literalen beschreibt die Menge der gemeinsamen Spezialisierungen (Beispiele).

Werden die Variablen der Literale zuvor separiert, so ergibt sich ein allgemeinerer Unifikator.

Satz

- Es ist entscheidbar, ob ein Unifikator existiert.
- Zu jeder unifizierbaren Menge von Literalen existiert ein allgemeinster Unifikator.
- Ein allgemeinster Unifikator kann ggf. konstruiert werden.

Resolutionsregel



- K_1 und K_2 Klauseln ohne gemeinsame Variablen
  (sonst: Separation durch Variablenumbenennungen)
- Negative Literale $\neg L'_1,..., \neg L'_{n'} \in K_1$
  Positive Literale $M'_1,...,M'_{m'} \in K_2$ ($n',m' >0$)
  $\{L'_1,...,L'_{n'}, M'_1,...,M'_{m'}\}$ unifizierbar mittels σ .

Durch Resolution von K_1 und K_2 entsteht die *Resolvente*

$K = \sigma ((K_1 - \{\neg L'_1,..., \neg L'_{n'}\}) \cup (K_2 - \{M'_1,...,M'_{m'}\}))$

$\{L_1,..,L_n, \neg L'_1,.., \neg L'_{n'}\}, \{M_1,.., M_m,M'_1,..,M'_{m'}\}, \sigma(L'_1)=..=\sigma(L'_n)=\sigma(M'_1)=..=\sigma(M'_m)$

$\sigma (\{ L_1,..,L_n , M_1,..,M_m \})$

Resolutionsregel

Zerlegung in zwei Regeln.

Einfache Resolutionsregel:

$$\frac{\{L_1,..,L_n, \neg L'\}, \{M_1,.., M_m, M'\}, \sigma(L')=\sigma(M')}{\sigma(\{L_1,..,L_n, M_1,..,M_m\})}$$

Faktorisierungsregel:

$$\frac{\{L_1,..,L_n, L_1', L_2'\}, \sigma(L_1')=\sigma(L_2')}{\sigma(\{L_1,..,L_n, L_1'\})}$$

Spezialfälle

Schnittregel

$$\frac{A \vee B, \neg B' \vee C, \sigma(B)=\sigma(B')}{\sigma(A \vee C)}$$

Modus ponens

$$\frac{B, B \rightarrow C}{C}$$

Indirekter Beweis

$$\frac{B, \neg B}{\square}$$

Unerfüllbarkeitsnachweis

Res(**K**) = Menge aller mit Resolution in endlich vielen Schritten aus Klauselmenge **K** ableitbarer Klauseln

Klauseln(H) = Klauseldarstellung eines Ausdrucks H (nicht eindeutig)

Satz: H $\notin$ **ef** gdw. □ $\in$ Res(**Klauseln**(H))

□ ist die leere Klausel („Widerspruch“)

Beweis: Herbrand-Modelle („alle“ Beispiele)

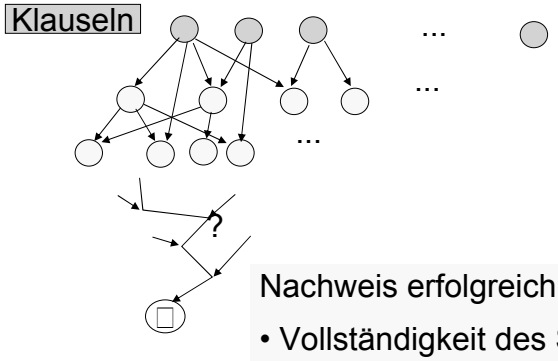
- Bei Resolution werden Gegenbeispiele gesucht
- Allgemeinster Unifikator: „Lazy evaluation“

Unerfüllbarkeitsnachweis

Nachweis

der Unerfüllbarkeit (H $\notin$ **ef**) eines Ausdrucks H :

Erfolgreiche Suche in Res(**Klauseln**(H)) nach □



Nachweis erfolgreich, wenn □ gefunden wird

- Vollständigkeit des Suchverfahrens?
- i.a. unendlicher Suchraum

Resolutionsverfahren

Aufgabe: Nachweis von $H \in Fl(\mathbf{X})$

- Konjunktive Normalform von $(\bigwedge \mathbf{X} \wedge \neg H)$ bilden.
- Darstellung als (erfüllbarkeitsäquivalente) Klauselmenge

$$\mathbf{Kl} = \{ \{ L_{ij} \mid j = 1, ...m_i \} \mid i = 1, ..., n \}$$

Suchverfahren:

–Wiederholtes Erweitern der Klauselmenge durch neue Resolventen.

–Abbruch, wenn leere Klausel $\square$ abgeleitet wurde:

EXIT($H \in Fl(\mathbf{X})$).

– Wenn keine neuen Klauseln ableitbar:

EXIT($H \notin Fl(\mathbf{X})$)

(i.a. unendlicher Suchraum)

Resolution

Zu zeigen:

aus A1: $\forall x \neg R(x,x)$

A2: $\forall x \forall y \forall z (R(x,y) \wedge R(y,z) \rightarrow R(x,z))$

folgt H: $\forall x \forall y (R(x,y) \rightarrow \neg R(y,x))$

Resolutions-Beweis:

aus den Klauseln für A1, A2 und $\neg H$:

KA1: $\neg R(x,x)$

KA2: $\neg R(u,y) \vee \neg R(y,z) \vee R(x,u))$

K1: $R(c,f(c))$

K2: $R(f(c),c)$

ist die leere Klausel mit Resolutionsregel abkeitbar

Resolutionsverfahren

i.a. unendlicher Suchraum:

Resolutionsverfahren liefert Lösung im Fall von $H \in Fl(\mathbf{X})$
(bzw. Unerfüllbarkeit der entsprechenden Klauselmenge)
in endlich vielen Schritten bei geeignetem Suchverfahren
(z.B. Breite-Zuerst).

Im Fall von $H \notin Fl(\mathbf{X})$
(bzw. Erfüllbarkeit der entsprechenden Klauselmenge)
dagegen evtl. kein Resultat
(unendliches Resolvieren ohne Erreichen von □).

Resolutionsverfahren

<u>ist ein Negativer Testkalkül</u>
verwendet
- spezielle Normalformen (Klauseln)
- Resolutionsregel (Unifikation, Resolventen)

- widerlegungsvollständig
  - wenn H unerfüllbar, so □ mit Resolution ableitbar
- und widerlegungskorrekt
  - wenn □ mit Resolution ableitbar, so H unerfüllbar

H unerfüllbar gdw. □ mit Resolution ableitbar

Spezielle Resolutionsstrategien

Resolutionsmethode ist Grundlage für
Deduktionssysteme, Theorembeweiser, ...
Problem: Kombinatorische Explosion des Suchraums

Effizienzverbesserungen durch
1) Streichen überflüssiger Klauseln, z.B. :
- tautologische Klauseln K , d.h. $\{A, \neg A\} \subseteq K$
- subsumierte Klauseln: K_1 wird von K_2 subsumiert,
falls $\sigma(K2) \subseteq K1$ bei einer geeigneten Substitution σ.

2) Heuristiken für Suchstrategie, Klauselgraphen

Problem: Widerspruchsvollständigkeit gewährleisten.

Spezielle Resolutionsstrategien

- Atomformel: *positives Literal*,
negierte Atomformel: *negatives Literal*.
- K*lausel heißt negativ*, wenn sie nur negative Literale enthält.
- *Klausel heißt definit*, falls sie genau ein positives Literal (und evtl. noch negative Literale) enthält.
- Klausel heißt *HORN-Klausel*, wenn sie höchstens ein positives Literal (und evtl. noch negative Literale) enthält.

Hornklausel: definit (Prolog-Klauseln)
oder negativ (Prolog-Anfrage)

Spezielle Resolutionsstrategien

P-Resolution:
Eine der resolvierenden Klauseln enthält nur positive Literale.

N-Resolution:
Eine der resolvierenden Klauseln enthält nur negative Literale.

Lineare Resolution:
Resolution benutzt die im vorigen Schritt erzeugte Resolvente.

Input-Resolution (Spezialfall der linearen Resolution):
Resolution benutzt die im vorigen Schritt erzeugte Resolvente und eine Klausel der Ausgangsmenge.

Einheitsresolution:
Resolution benutzt mindestens eine ein-elementige Klausel.

Spezielle Resolutionsstrategien

- P-Resolution, N-Resolution, lineare Resolution sind widerlegungsvollständig.
- Input-Resolution und Einheitsresolution sind widerlegungsvollständig für HORN-Klauseln (aber nicht im allgemeinen Fall).

SLD-Resolution für HORN-Klauseln

S = „Selection Function“
L = lineare Resolution
D = definite Klauseln

- Programm **P** sei eine Menge definiter Klauseln.
- Ein SLD-Widerlegungsbeweis für eine (positive) Ziel-Klausel G aus dem Programm **P** ist SLD-Ableitung von □ aus **P** ∪ {¬G}

- Start mit negativer Klausel ¬G .
- In jedem Schritt wird eine negative mit einer definiten Klausel aus dem Programm **P** resolviert.
- Alle Resolventen sind negativ (Spezialfall der **linearen** Resolution, Input-Resolution, N-Resolution).

SLD-Resolution für HORN-Klauseln

Jeder Resolutionsschritt der SLD-Resolution benutzt
- eine negative (vorherige Resolvente) und
- eine **definite** Klausel (aus Programm **P**).

Es sind zwei Entscheidungen zu treffen:

1. Auswahl eines Literals ¬L der negativen Klausel mittels „**selection** function“ (im Prinzip beliebig: *Und-Verzweigung*).
2. Auswahl einer Klausel mit einem positiven Literal M, das mit L unifizierbar ist (*Oder-Verzweigung*).

Suche im Und-Oder-Baum, z.B.
- Breite-Zuerst (- Findet eine existierende Lösung. -) oder
- Tiefe-Zuerst.

SLD-Resolution für HORN-Klauseln

σ_i sei die im i-ten Schritt der SLD-Resolution verwendete Substitution (Unifikator).
Dann ist $\sigma = \sigma_1 * ... * \sigma_n$ die beim Erfolg nach n Schritten erzeugte Antwortsubstitution.

Für jede Antwortsubstitution σ
eines SLD-Widerlegungsbeweises für G aus **P** gilt:

$$\mathbf{P} \models \sigma(G) .$$

Falls $\mathbf{P} \models \sigma(G)$,
so existiert ein SLD-Widerlegungsbeweis für G aus **P**
mit einer Antwortsubstitution σ' , die allgemeiner als σ ist.

PROLOG

Logische Programmiersprache.

Algorithmus = Logik + Steuerung

HORN-Klauseln + programmiersprachliche Konstrukte:
- E/A-Funktionen
- meta-logische Funktionen (Programm-Modifikation)
- Eingriff in Suchprozedur (Cut)

Steuerung durch Interpreter mit Auswahlstrategie:
1. links vor rechts
2. oben vor unten

und Suchstrategie:
Tiefe-Zuerst.

Prolog-Interpreter

- Umsetzung des SLD-Verfahrens als Zustandsraumsuche (Tiefe-Zuerst)
- Verwendung des Backtracking-Prinzips
  (Organisiert durch Prozedurkeller)
- Variablenbindungen durch Unifikation
- Optimierungen