Einführung in die KI

Prof. Dr. sc. Hans-Dieter Burkhard
Vorlesung Winter-Semester 2003/04

Wissensrepräsentation:
Resolution (im PK1)

2. Resolution

Vorbild für „Formalismus“:

– exakt, präzise, (theoretisch) beherrscht

Aufbau:

–Zeichen

–Ausdrücke (rekursive Definition)

–Sätze („Theorie“) **Th**

•syntaktisch bestimmt:

Th = Abl(**Ax**)

Nach speziellen formalen Regeln erzeugbare Formeln

•semantisch bestimmt:

Th = allgemeingültige Sätze einer Struktur

Beweise

„Irreflexive, transitive Relationen sind asymmetrisch“

Inhaltlicher Beweis:

– Argumentation

Formaler (syntaktischer) Beweis:

– Umformung von Ausdrücken („Kalkül“)

Theorembeweiser:

– (Syntaktisches) Verfahren zur Entscheidung, ob ein Ausdruck zu einer Satzmenge (Theorie) gehört:

$$\boxed{H \in \mathbf{Th}\ ?}$$

Axiomatische Behandlung der Logik

Syntax

Ausdrucksmenge **X**

Ableiten

Ableitbare Sätze
Abl(**X**∪**ag**)

Semantik

Ausdrucksmenge **X**

Folgern

Folgerungen
Fl(**X**)

Korrektheit von Abl : $Abl(\mathbf{X} \cup \mathbf{ag}) \subseteq Fl(\mathbf{X})$

Vollständigkeit von Abl : $Abl(\mathbf{X} \cup \mathbf{ag}) \supseteq Fl(\mathbf{X})$

Äquivalenz von Abl : $Abl(\mathbf{X} \cup \mathbf{ag}) = Fl(\mathbf{X})$

Formales Ableiten (Resolutionsregel)

Für Klauseln („Disjunktion von Literalen“)

Voraussetzung:

K1 und K2 unifizierbar mittels Unifikator σ

K =Res(K1,K2,σ) entsteht durch

- „Vereinigung“ von σ(K1) und σ(K2)
- Streichen komplementärer Literale

Formales Ableiten (Resolution)

KA1:	$\neg R(x,x)$
KA2:	$\neg R(u,y) \vee \neg R(y,z) \vee R(u,z))$
K1:	$R(c,f(c))$
K2:	$R(f(c),c)$

K3 =Res(KA1,KA2,σ): $\neg R(w,y) \vee \neg R(y,w)$

mit $\sigma(u)= \sigma(z)= \sigma(x)= w$,$\sigma(y)= y$

K4 = Res(K1,K3,σ): $\neg R(f(c),c)$

mit $\sigma(w)=c$ $\sigma(y)=f(c)$

K5 = Res(K2,K4,σ): $\square$

Enscheidbarkeit in der Logik

(rein logisch) allgemeingültige Sätze:

ag = Abl(**axp**) = Fl(∅)

axp : Axiomensystem des PK1

H ∈ **ag** ?

- Entscheidbar im AK
- Unentscheidbar im PK1

  (aber aufzählbar, da axiomatisierbar)

Formalisierung einer Domäne

beabsichtigte Bedeutung

Struktur S=(U,I)

Gültigkeit

Gültige Sätze
Th(S)

Formalismus

Axiome: **Ax**∪**axp**

Inferenz-
Maschine

Folgern

Ableitbare Sätze
Abl(**Ax**∪**axp**)

=

Folgerungen
Fl(**Ax**)

Korrektheit der Formalisierung : Abl(**Ax**∪**axp**) ⊆ **Th**(S)

Vollständigkeit der Formalisierung: Abl(**Ax**∪**axp**) ⊇ **Th**(S)

Äquivalenz der Formalisierung : Abl(**Ax**∪**axp**) = **Th**(S)

Beispiel: Formalisierung der Arithmetik

beabsichtigte Bedeutung

Struktur $S=(U,I)$

Gültigkeit

Gültige Sätze
Th(S)

Formalismus

Elementare Arithmetik

Gültigkeit

Theoreme der Elementaren Arithmetik

$\mathbf{Th}(S) = \{\ H \mid \mathrm{Wert}_S(H,\beta) = 1$ für alle β über $U\ \}$

β : Belegung der Variablen
mit Individuen aus dem Universum U

Beispiel: Formalisierung der Arithmetik

Elementare Arithmetik
ElArith

Gültigkeit

Theoreme in
$\mathbf{Th}_{\mathbf{ElArith}}$

Peano-Axiome:
$\mathbf{ax}_{\mathbf{Peano}} \cup \mathbf{axp}$

Ableiten
(PK1)

Folgern

Ableitbare Sätze
$\mathrm{Abl}(\mathbf{ax}_{\mathbf{Peano}} \cup \mathbf{axp})$

=

Folgerungen
$\mathrm{Fl}(\mathbf{ax}_{\mathbf{Peano}})$

Korrektheit der Formalisierung :

$$\mathrm{Abl}(\mathbf{Ax}_{\mathbf{Peano}} \cup \mathbf{axp}) \subseteq \mathbf{Th}_{\mathbf{ElArith}}$$

Unvollständigkeit der Formalisierung:

$$\mathrm{Abl}(\mathbf{Ax}_{\mathbf{Peano}} \cup \mathbf{axp}) \neq \mathbf{Th}_{\mathbf{ElArith}}$$

Adäquatheit der Formalisierung

Adäquatheit:

Die wesentlichen Aspekte werden in der Struktur S bzw. den Axiomen **Ax** korrekt erfaßt.

–Parallelen-Axiom der Geometrie

–Stetigkeitsdefinition in der Analysis

–Modellierung eines Staubsaugers

Modellierung eines Staubsaugers

Alternativen:

$$(S1) \quad \forall staubi : Staubsauger(staubi) \Rightarrow \exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi, l) \wedge Motor(m) \wedge HatAlsTeil(staubi, m) \wedge Stecker(s) \wedge HatAlsTeil(staubi, s)$$

$$(S2) \quad \forall staubi : Staubsauger(staubi) \Leftrightarrow \exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi, l) \wedge Motor(m) \wedge HatAlsTeil(staubi, m) \wedge Stecker(s) \wedge HatAlsTeil(staubi, s)$$

Modellierung eines Staubsaugers

Was folgt für *staubi₁* aus (S1) bzw. (S2) zusammen mit einer der folgenden Aussagen:

$$\exists l,m,s : Lampe(l) \wedge HatAlsTeil(staubi_1,l) \wedge Motor(m) \wedge HatAlsTeil(staubi_1,m) \wedge Stecker(s) \wedge HatAlsTeil(staubi_1,s)$$

$$\neg\exists l,m,s : Lampe(l) \wedge HatAlsTeil(staubi_1,l) \wedge Motor(m) \wedge HatAlsTeil(staubi_1,m) \wedge Stecker(s) \wedge HatAlsTeil(staubi_1,s)$$

Modellierung - Adäquatheit

Mehr Axiome – weniger Modelle - mehr Folgerungen

Mehr Axiome – komplexere Verarbeitung

Adäquatheit hinsichtlich
- Anwendungsproblematik
- Komplexität

Vereinfachung, wenn klar ist, dass es (nur) um Staubsauger geht

Modellierung eines Staubsaugers

Funktionsbeschreibung: Verhalten

$$(M1)\quad \forall m : Brummt(m) \Leftarrow Motor(m) \wedge [\exists staubi, s, sd : Staubsauger(staubi) \wedge HatAlsTeil(staubi, m) \wedge Stecker(s) \wedge HatAlsTeil(staubi, s) \wedge Steckdose(sd) \wedge SteckIn(s, sd)]$$

$$\boldsymbol{Angeschlossen(m)}$$

Diagnose: Fehlverhalten

$$(M2)\quad \forall m : Brummt(m) \Leftarrow Motor(m) \wedge Angeschlossen(m)$$

$$(M3)\quad \forall m : Brummt(m) \Leftarrow Motor(m) \wedge Angeschlossen(m) \wedge Ok(m)$$

Syntax des PK1

Terme: Individuen (Konstante, Variable, Funktionen)

Prädikate (atomare Formeln):

Relationen $R(x_1,\ldots,x_n)$ (wahr/falsch)

Ausdrücke:

logische Beziehungen zwischen Prädikaten mittels

- Aussagenlogischen Operatoren $\neg \wedge \vee \leftrightarrow \rightarrow$
- Quantifikation von Variablen $\forall \exists$

z.B. $\forall x\, \exists y\, R(x, x_1,\ldots,x_n) \wedge \neg R(y, x_1,\ldots,x_n)$

Positives Literal: nicht negierte atomare Formel

Negatives Literal: negierte atomare Formel

Syntax des PK1: Ableiten

Ableiten: Ausdrücke umformen mit Ableitungsregeln

z.B. Abtrennungsregel (modus ponens)

$$\frac{H_1 \; , \; H_1 \rightarrow H_2}{H_2}$$

H *ableitbar aus* **X**,

falls Ableitungsfolge für H aus **X** existiert

X |- H oder: H ∈ **X**|- oder: H ∈ Abl(**X**)

Allgemeingültigkeit, Erfüllbarkeit

Belegung β erfüllt den Ausdruck H in der Struktur S = [U, I] ,

falls $Wert_S(H, \beta) = W$.

(Rein logische) Erfüllbarkeit eines Ausdrucks H:

H heißt erfüllbar, falls β, U,I existieren mit $Wert_{[U,I]}(H, \beta) = W$.

ef : Menge aller erfüllbaren Ausdrücke

(Rein logische) Allgemeingültigkeit eines Ausdrucks H:

H heißt allgemeingültig, falls $Wert_{[U,I]}(H, \beta) = W$ für alle β, U,I .

ag : Menge aller allgemeingültigen Ausdrücke

Freie Variable werden bei Allgemeingültigkeit wie generalisierte Variable behandelt.

Allgemeingültigkeit, Erfüllbarkeit

H ist allgemeingültig gdw. ¬H nicht erfüllbar ist.

H ist erfüllbar gdw. ¬H nicht allgemeingültig ist.

ag axiomatisierbar:

Es gibt aufzählbares Axiomensystem **axp** mit **ag** = Abl(**axp**)

Axiomatisierbar = aufzählbar = partiell entscheidbar
M entscheidbar **gdw**. M und U - M aufzählbar

Allgemeingültigkeit, Erfüllbarkeit

Satz von Church:

Erfüllbarkeit/Allgemeingültigkeit im PK1 sind unentscheidbar.

Menge der allgemeingültigen Ausdrücke: axiomatisierbar.

Menge der nicht erfüllbaren Ausdrücke: axiomatisierbar.

Menge der nicht allgemeingültigen Ausdrücke: nicht axiomat.

Menge der erfüllbaren Ausdrücke: nicht axiomatisierbar.

H ist allgemeingültig gdw. ¬H nicht erfüllbar ist.

H ist erfüllbar gdw. ¬H nicht allgemeingültig ist.

Folgern im PK1

Eine Struktur S = [U, I] und eine Belegung β sind ein *Modell* für eine Menge **X** von Ausdrücken, wenn für alle H $\in$ **X** gilt:
β erfüllt H in der Struktur S = [U, I] , d.h. $\text{Wert}_S(H, \beta) = W$.

Es sei **X** eine Menge von Ausdrücken, H ein Ausdruck.

H *folgt aus* **X** , falls gilt:

Jedes Modell von **X** ist ein Modell von H .

X |= H oder: H $\in$ **X**|= oder: H $\in$ Fl(**X**)

Folgern im PK1

Staubi1 als Modell für

(S1) $\forall staubi : Staubsauger(staubi) \Rightarrow \exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi, l) \wedge Motor(m) \wedge HatAlsTeil(staubi, m) \wedge Stecker(s) \wedge HatAlsTeil(staubi, s)$

(*) $\neg\exists l, m, s : Lampe(l) \wedge HatAlsTeil(staubi_1, l) \wedge Motor(m) \wedge HatAlsTeil(staubi_1, m) \wedge Stecker(s) \wedge HatAlsTeil(staubi_1, s)$

Staubi1 ist auch Modell von : *¬Staubsauger(staubi1)*

Gilt bei allen Modellen,
d.h. *¬Staubsauger(staubi1)* ist Folgerung von (S1) und (*)

Folgern und Ableiten im PK1

Fl ist syntaktisch beschreibbar mittels Abl

Fl = Abl „modulo **ag**“

ag = Fl(∅) = Abl(**axp**)

axp : Axiomensystem des PK1

Fl und Abl sind monoton:

$$\mathbf{X} \subseteq \mathbf{Y} \Rightarrow \mathrm{Fl}(\mathbf{X}) \subseteq \mathrm{Fl}(\mathbf{X})$$

$$\mathbf{X} \subseteq \mathbf{Y} \Rightarrow \mathrm{Abl}(\mathbf{X}) \subseteq \mathrm{Abl}(\mathbf{X})$$

$$(H \rightarrow G) \in \mathrm{Fl}(\mathbf{X}) \Leftrightarrow G \in \mathrm{Fl}(\mathbf{X} \cup \{H\})$$

$$H \in \mathrm{Fl}(\mathbf{X}) \Leftrightarrow (\wedge\mathbf{X} \rightarrow H) \in \mathbf{ag}$$

Formale Theorien im PK1

Als *Theorie*

wird eine bezüglich Folgern (Ableiten)

abgeschlossene Menge **Th**

von Ausdrücken bezeichnet:

Th = Fl(**Th**) = Abl(**Th**) „modulo **ag**“

Formale Theorien im PK1

Theorie mit *semantisch bestimmter Satzmenge*:

Gegeben ist eine Struktur $S = [U, I]$ mit

$$\mathbf{Th} = \{ F \mid F \text{ allgemeingültig in } S \}$$
$$= \{ F \mid \mathrm{Wert}_S (F, \beta) = W \text{ für alle } \beta \text{ über } U \}$$

Arithmetik
Staubsauger

Theorie mit *syntaktisch bestimmter Satzmenge*:

Gegeben ist ein Ausdruckmenge **Ax** („Axiome“) mit

$$\mathbf{Th} = \mathrm{Abl}(\mathbf{Ax} \cup \mathbf{axp}) \quad (= \mathrm{Fl}(\mathbf{Ax}) \quad)$$

Peano-Arithmetik
Staubsauger-Axiomatik

Formalisierung mittels PK1

Ziel: Axiome zur Beschreibung von Sachverhalten finden

Analoges Ziel: **Computerverarbeitung ermöglichen**

1. semantisch definierte Theorie **Th** bestimmen
   (Universum, Relationen/Funktionen, Interpretation)

2. axiomatische Beschreibung von **Th** :
   Axiome **Ax** mit $\mathbf{Th} = \mathrm{Abl}(\mathbf{Ax} \cup \mathbf{axp})$

   (z.B. PROLOG-Programm)

Anwendung PK1

Formalisierung: Axiome **X**

Problem durch Ausdruck H beschreiben

Entscheiden, ob

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

d.h. ob $H \in Abl(\mathbf{X} \cup \mathbf{axp})$

Programme dafür: Theorembeweiser

Beweise

Positiver Kalkül: Allgemeingültigkeit entscheiden $H \in \mathbf{Th}$?

Negativer Kalkül: Unerfüllbarkeit untersuchen

$\{\neg H\} \cup \mathbf{Th}$ widersprüchlich ?

Deduktiver Kalkül:

Erweitern der Axiome um H zu finden (forward chaining)

Testkalkül:

Reduktion von H auf Axiome (backward chaining)

Beispiel:

Resolution (PROLOG) ist *Negativer Testkalkül*

Beweise

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

gilt gdw. $H \in Abl(\mathbf{X} \cup \mathbf{axp})$ → Positiver Kalkül

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

gilt gdw. $(\wedge\mathbf{X} \rightarrow H) \in \mathbf{ag}$

gilt gdw. $\neg(\wedge\mathbf{X} \rightarrow H) \notin \mathbf{ef}$

gilt gdw. Skolemform von $(\wedge\mathbf{X} \wedge \neg H)$ nicht erfüllbar

gilt gdw. Klauselform von $(\wedge\mathbf{X} \wedge \neg H)$ nicht erfüllbar

→ Negativer Kalkül

Beweise

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

gilt gdw. $H \in Abl(\mathbf{X} \cup \mathbf{axp})$ → Positiver Kalkül

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

gilt gdw. $(\wedge\mathbf{X} \rightarrow H) \in \mathbf{ag}$

gilt gdw. $\neg(\wedge\mathbf{X} \rightarrow H) \notin \mathbf{ef}$

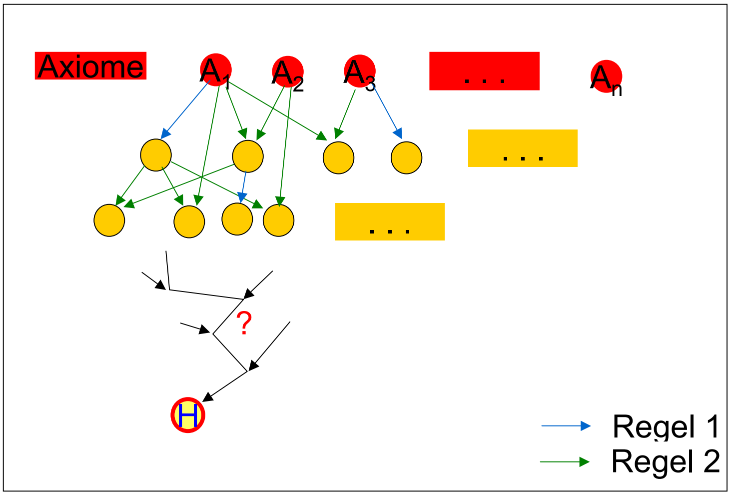
gilt gdw. Skolemform von $(\wedge\mathbf{X} \wedge \neg H)$ nicht erfüllbar

gilt gdw. Klauselform von $(\wedge\mathbf{X} \wedge \neg H)$ nicht erfüllbar

gilt gdw. Leere Klausel mittels Resolution ableitbar

→ Negativer Kalkül

Deduktiver Kalkül



Suchraum bei deduktivem Kalkül

Klassischer AK : 15 Axiome für **ag**, 2 Regeln

sogar entscheidbar, allerdings NP

Klassischer PK1:

– abzählbar viele Axiome für **ag** + weitere für **Th** , 7 Regeln

– Alle allgemeingültigen Ausdrücke im Suchraum **Th** :

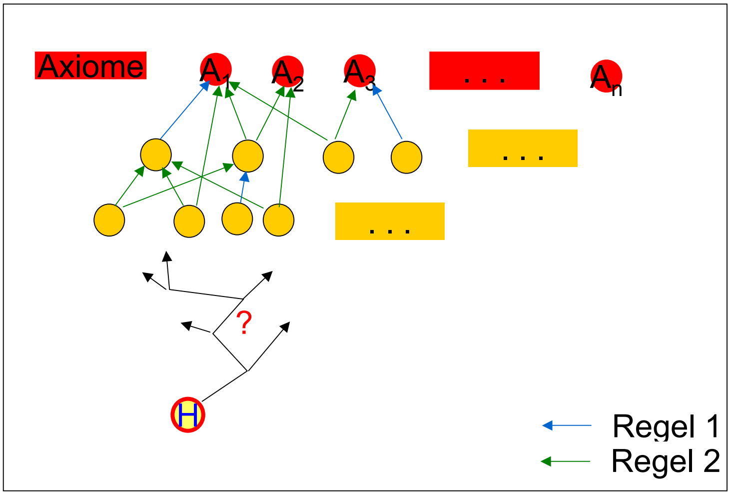
$\mathbf{ag} = Fl(\emptyset) \subseteq Fl(\mathbf{Th}) = \mathbf{Th}$

– Mit H viele weitere Ausdrücke im Suchraum **Th** :

z.B. $H \vee G$ für beliebiges G

Vollständigkeit als Nachteil

Test-Kalkül



Normalformen (Definitionen)

- ein Ausdruck H heißt *bereinigt*, falls gilt:
  - es gibt keine Variable x, die in H sowohl frei als auch gebunden vorkommt,
  - die hinter den Quantoren in H vorkommenden Variablen sind alle verschieden.

- ein Ausdruck H heißt *pränex*, falls gilt:
  - H hat die Form $Q_1 x_1 \ldots Q_n x_n H'$,
    wobei $Q_1, \ldots, Q_n$ Quantoren sind
    und H' keine Quantoren enthält.

Normalformen (Definitionen)

- ein Ausdruck H heißt *pränexe Normalform*, falls gilt:
  - H hat pränexe Form $Q_1 x_1 \ldots Q_n x_n H´$,
  - $H´$ ist eine konjunktive Normalform (KNF)

    $H´ = \wedge \{ \{ \vee L_{ij} \mid i=1\ldots n \} \mid j=1\ldots m_n \}$

    wobei die L_{ij} Literale sind.

> Literal:
> atomare Formel (Prädikat, Relation) $r(X1,\ldots,Xn)$
> oder
> negierte atomare Formel $\neg r(X1,\ldots,Xn)$

Normalformen (Sätze)

1. Zu jedem Ausdruck H_1 existiert ein Ausdruck H_2 in bereinigter Form, so dass H_1 und H_2 semantisch äquivalent sind.
2. Zu jedem Ausdruck H_2 in bereinigter Form existiert ein Ausdruck H_3 in bereinigter pränexer Form, so dass H_2 und H_3 semantisch äquivalent sind.
3. Zu jedem Ausdruck H_3 in bereinigter pränexer Form existiert ein Ausdruck H_4 in bereinigter pränexer Normalform, so dass H_3 und H_4 semantisch äquivalent sind.

> Zu jedem Ausdruck H
> existiert ein semantisch äquivalenter Ausdruck G
> in bereinigter pränexer Normalform.

Skolem-Normalform

Eine Skolem-Normalform ist eine pränexe Normalform, deren Variable sämtlich generalisiert sind.

Umformung einer bereinigten pränexen Normalform G in eine Skolem-Normalform F:

1. $\exists$-Quantoren einführen für alle freien Variablen in G. (erfüllbarkeits-äquivalente Umformung !!)
2. Elimination aller $\exists$-Quantoren mit Hilfe von Skolem-Funktionen.

Die entstehende Formel F ist erfüllbarkeits-äquivalent zu G.

Klauselform

Eine Klausel ist eine Menge von Literalen $K = \{L_1, ..., L_n\}$.

Ein Literal ist eine negierte oder nicht-negierte Atomformel.

Vorteil der Klauselnotation:

Mengenschreibweise beseitigt rein formale Unterschiede äquivalenter Ausdrücke bzgl. Kommutativität/Idempotenz.

Klauselform

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.)

$$\mathbf{F} = \wedge_i \vee_j L_{ij}$$

3. Darstellung der Alternativen als Klauseln:
   d.h. Menge von Literalen $\{ L_{ij} \mid j = 1, ...m_i \}$.
   Darstellung von F als Menge von Klauseln **Kl** :

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

Substitution (Definition)

Eine *Substitution* σ ist eine Abbildung der Individuenvariablen (eines Ausdrucks) in die Menge der Terme.

– Grundsubstitution: Abbildung in die Grundterme.

– Variablenumbenennung: Ersetzung von Variablen durch Variable.

$\sigma(L)$: Ergebnis der Substitution σ für ein Literal L
(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 (Definition)

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

$\sigma_0 = \sigma * \sigma_{00}$ (σ_0 ist spezieller als σ).

Der allgemeinste Unifikator ist eindeutig bis auf Umbenennungen.

Substitution, Unifikation

- Der allgemeinste Unifikator σ einer Menge $\{ L_1,...,L_n \}$ beschreibt die Menge der gemeinsamen Spezialisierungen (Beispiele).
- Werden die Variablen der Literale zuvor separiert, so ergibt sich ein allgemeinerer Unifikator.

Satz

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

Resolutionsregel

$$\begin{array}{ccc} K1 & & K2 \\ & \searrow \quad \swarrow & \\ & K & \end{array}$$

Voraussetzungen:

K_1 und K_2 Klauseln ohne gemeinsame Variablen
(sonst: Separation durch Variablenumbenennungen)

Literale $L'_1, \ldots, L'_{n'} \in K_1$ und $M'_1, \ldots, M'_{m'} \in K_2$, $n', m' > 0$,
die mittels eines Unifikators σ unifizierbar sind.

Durch Resolution entsteht die *Resolvente*

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

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

Resolutionsregel

Zerlegung in zwei Regeln.

Einfache Resolutionsregel:

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

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:

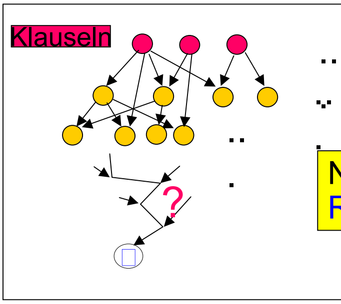
H $\not\in$ **ef** gdw. $\square \in$ Res(**Klauseln**(H))

Beweis: Herbrand-Modelle

Unerfüllbarkeitsnachweis

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

Suche in Res(**Klauseln**(H)) nach □



Nachweis erfolgreich, wenn □ in Res(**Klauseln**(H)) gefunden.

Res(**Klauseln**(H)) i.a. unendlicher Suchraum.

Vollständigkeit des Suchverfahrens?

Resolutionsverfahren

Nachweis von H$\in$Fl (**X**)

Konjunktive Normalform von ($\wedge$**X** $\wedge$ $\neg$H) bilden.

Darstellung als (erfüllbarkeitsäquivalente) Klauselmenge

$$\mathbf{Kl} = \{ \ \{ L_{ij} \ | \ j = 1, \ldots m_i \} \ | \ i = 1, \ldots, n \}$$

Suchverfahren:

- Wiederholtes Erweitern der Klauselmenge durch neue Resolventen.
- Abbruch, wenn leere Klausel □ abgeleitet wurde: EXIT(H $\in$ Fl (**X**)).
- Wenn keine neuen Klauseln ableitbar: EXIT(H $\notin$ Fl (**X**)). (i.a. aber unendlicher Suchraum)

Nachweis von $H \in FI$ (**X**)

Resolutionsverfahren

Resolutionsverfahren liefert Lösung im Fall der Unerfüllbarkeit in endlich vielen Schritten z.B. bei Breite-Zuerst-Verfahren.

Für erfüllbare Formeln dagegen evtl. kein Resultat (unendliches Resolvieren ohne Erreichen von □).

Resolutionsverfahren

- ist Negativer Testkalkül. Verwendet
  - spezielle Normalformen (Klauseln)
  - Resolutionsregel (Schnittregel, Unifikation)

- ist nicht vollständig
- aber widerlegungsvollständig
  - wenn H unerfüllbar, so □ mit Resolution ableitbar
- und widerlegungskorrekt
  - wenn □ mit Resolution ableitbar, so H unerfüllbarist

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 sichern.

Spezielle Resolutionsstrategien

Definition:

- o Eine Atomformel ist ein *positives Literal*, eine negierte Atomformel ist ein *negatives Literal*.
- o Eine *Klausel heißt negativ*, wenn sie nur negative Literale enthält.
- o Eine *Klausel heißt definit*, falls sie genau ein positives Literal (und evtl. noch negative Literale) enthält.
- o Eine 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 jeweils die im vorigen Schritt erzeugte Resolvente.

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

Einheitsresolution:
Resolution benutzt jeweils mindestens eine ein-elementige Klausel.

Spezielle Resolutionsstrategien

Stützmengenresolution:
Resolution benutzt jeweils eine Klausel aus der Stützmenge **T**.
Als Stützmenge wird eine Teilmenge **T** der Ausgangsmenge **KI** verwendet mit **KI - T** erfüllbar.

Satz
P-Resolution, N-Resolution, lineare Resolution und Stützmengen-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" (s.u.)
L = lineare Resolution
D = definite Klauseln

Definition

- o Ein definites Programm **P** ist eine Menge definiter Klauseln.
- o Eine Zielklausel ist eine negative Klausel.
- o Ein SLD-Widerlegungsbeweis für eine (positive) Klausel G aus einem Programm **P** ist SLD-Ableitung von $\square$ aus **P** $\cup \{\neg G\}$.

SLD-Resolution für HORN-Klauseln:

Start mit negativer Klausel $\neg G$.

In jedem Schritt wird eine negative mit einer definiten Klausel aus **P** resolviert. Alle Resolventen sind negativ.

Spezialfall der linearen Resolution, Input-Resolution, N-Resolution.

SLD-Resolution für HORN-Klauseln

Sei σ_i die im i-ten Schritt der SLD-Resolution verwendete Substitution (Unifikator).

Dann ist $\sigma = \sigma_1 * \ldots * \sigma_n$ die beim Erfolg nach n Schritten erzeugte Antwortsubstitution.

Satz

Für jede Antwortsubstitution σ

eines SLD-Widerlegungsbeweises für G aus **P** gilt

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

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

SLD-Resolution für HORN-Klauseln

Jeder Resolutionsschritt der SLD-Resolution benutzt eine negative und eine definite Klausel.

Es sind zwei Entscheidungen zu treffen:

1. Auswahl eines Literals $\neg L$ 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 ggf. eine Lösung.)
- oder Tiefe-Zuerst.