

Resolution

wichtiges Werkzeug zur Automatisierung von Beweisen und logischer Inferenz. Resolution spielt zentrale Rolle in AI (v.a. im Automatischen Beweisen) und in der Logikprogrammierung.

Indirekter Beweis: Um A zu beweisen, wird $\neg A$ (in NF) auf Widerspruch geführt.

Transformation in Normalform:

- Normalform 1:
Elimination von \rightarrow , \neg nur noch vor Atomen,
Variablenstandardisierung
- Normalform 2:
Elimination der \exists Quantoren (Skolemform)
- Weglassen der \forall Quantoren (NNF)
- Transformation in KNF

Normalformenbildung: SCHRITT 1

Transformation von A in eine logisch äquivalente Form B, wobei B \rightarrow nicht mehr enthält und \neg nur vor Atomformeln vorkommt.

Transformationsregeln:

$$(T1) \quad (A \rightarrow B) \Rightarrow \neg A \vee B$$

$$(T2) \quad \neg(A \wedge B) \Rightarrow \neg A \vee \neg B$$

$$(T3) \quad \neg(A \vee B) \Rightarrow \neg A \wedge \neg B$$

$$(T4) \quad \neg\neg A \Rightarrow A$$

$$(T5) \quad \neg(\forall x A) \Rightarrow (\exists x) \neg A$$

$$(T6) \quad \neg(\exists x A) \Rightarrow (\forall x) \neg A$$

Beispiel:

$$\neg(\forall x)(\exists y)((P(x,y) \wedge Q(y)) \rightarrow R(y)) \Rightarrow^{(T5)}$$

$$(\exists x) \neg(\exists y)((P(x,y) \wedge Q(y)) \rightarrow R(y)) \Rightarrow^{(T6)}$$

$$(\exists x)(\forall y) \neg((P(x,y) \wedge Q(y)) \rightarrow R(y)) \Rightarrow^{(T1)}$$

$$(\exists x)(\forall y) \neg(\neg(P(x,y) \wedge Q(y)) \vee R(y)) \Rightarrow^{(T3)}$$

$$(\exists x)(\forall y) (\neg\neg(P(x,y) \wedge Q(y)) \wedge \neg R(y)) \Rightarrow^{(T4)}$$

$$(\exists x)(\forall y) ((P(x,y) \wedge Q(y)) \wedge \neg R(y))$$

Schritt1: Variablen-Normalisierung

Zur Vereinfachung der weiteren NF-bildung schränken wir die Variablenstruktur von PL-Formeln ein:

(Q1) Verschiedene Vorkommen von Quantoren binden verschiedene Variablen (d.h., es ist unmöglich, dass $(\exists x)$ oder $(\forall x)$ mehr als einmal in einer Formel vorkommen.

(Q2) Variablen kommen nicht gleichzeitig frei und gebunden in einer Formel vor.

Diese Syntaxeigenschaften lassen sich immer durch Umbenennung gebundener Variablen herbeiführen, wobei die logische Äquivalenz erhalten bleibt.

Beispiel:

$$\begin{aligned} (\forall x) A(x) \wedge (\exists x) A(x) &\Rightarrow (\forall x) A(x) \wedge (\exists y) A(y) \\ A(x) \wedge (\forall x) A(x) &\Rightarrow A(x) \wedge (\forall y) A(y) \end{aligned}$$

Normalform1

Eine Formel, die (Q_1) und (Q_2) erfüllt, und irreduzibel unter $(T1) - (T6)$ ist, befindet sich in **Normalform 1**.

Im nächsten Schritt ersetzen wir nun jede Variable, die an einen Existenzquantor gebunden ist, durch eine Skolem Konstante oder Funktion.

Seien $Q_1, Q_2 \in \{\forall, \exists\}$ und $F \in PL$. Wir sagen **(Q_2x) ist im Bereich von (Q_1y)** (in F) wenn es Teilformeln G, H von F gibt mit $G = (Q_1y) H$ und (Q_2x) kommt in H vor.

Sei A eine geschlossene PL-Formel in Normalform 1. Sei A' gleich A nach Auslassung des ersten \exists -Quantors von links (gibt es keine \exists - Quantoren in A , so ist $A' = A$).

Schritt 2: Elimination der \exists Quantoren



Transformation $\alpha: PL \rightarrow PL$

Annahme: $(\exists x)$ ist der erste \exists -Quantor von links.

- a) A ist \exists -frei: $\alpha(A)=A$
- b) $(\exists x)$ nicht im Bereich von $(\forall -)$ Quantoren: $\alpha(A)=(A')(x/a)$
wobei a Konstantensymbol, das nicht in A vorkommt.
- c) $(\exists x)$ liegt im Bereich der Allquantoren $(\forall y_1)\dots(\forall y_k)$. Sei $f \in FS_k - FS(A)$ (d.h., f kommt nicht in A vor), dann setzen wir: $\alpha(A)=(A')(x/f(y_1,\dots,y_k))$.

Beispiel:

$$A = (\forall x) (\forall y) (\exists z) P(x,y,z) \vee (\exists u) Q(u,a)$$

$$\alpha(A) = (\forall x) (\forall y) P(x,y,f(x,y)) \vee (\exists u) Q(u,a)$$

$$\alpha(\alpha(A)) = (\forall x) (\forall y) P(x,y,f(x,y)) \vee Q(b,a)$$

Normalform 2



Thoralf Skolem
1887 - 1963

Sei $\alpha^0(A) = A$, $\alpha^{n+1}(A) = \alpha(\alpha^n(A))$

Eine Formel ist in **Normalform 2**, wenn sie in Normalform 1 und \exists -frei ist. Ist B eine Normalform 1 von A und ist $\alpha^n(B)$ in Normalform 2, so heisst $\alpha^n(B)$ **Skolemform** von A.

Transformation in Skolemform erhält NICHT die logische Äquivalenz! Jedoch:

Ist A eine geschlossene Formel in Normalform 1, und ist B Skolemform von A, so gilt $A \sim_e B$

Schritt 3: Negationsnormalform

Weglassen der Allquantoren in der Skolemform

Stellung der Quantoren unerheblich, Äquivalenz zu Pränexform.

beruht auf Verschiebungsregeln:

$$(\forall x) A \vee B = (\forall x) (A \vee B)$$

$$B \vee (\forall x) A = (\forall x) (B \vee A)$$

$$(\forall x) A \wedge B = (\forall x) (A \wedge B)$$

$$B \wedge (\forall x) A = (\forall x) (B \wedge A)$$

Sei A eine Formel in Normalform 2 und A^0 sei A nach Weglassung aller Quantorenausdrücke aus A . A^0 ist dann in **Negationsnormalform** (NNF).

NNFs sind offene Formeln, müssen aber als geschlossene interpretiert werden!

Schritt 4: Transformation in KNF

Literale sind (negierte) Atomformeln.

Sei $F = C_1 \wedge \dots \wedge C_n$ wobei alle C_i von der Form $(L_{i,1} \vee \dots \vee L_{i,k_i})$ für Literale $L_{i,j}$ sind. Wir sagen dann, F ist in **konjunktiver Normalform** (KNF).

Transformation in KNF durch Distributivregeln:

$$A \vee (B \wedge C) \Rightarrow (A \vee B) \wedge (A \vee C)$$

$$(B \wedge C) \vee A \Rightarrow (A \vee B) \wedge (A \vee C)$$

Danach: Übersetzung der KNF in eine Menge von Klauseln.

Resolution

Resolution auf aussagenlogischen Formeln ist nichts anderes als die Schnittregel. Bei Klauseln in der Prädikatenlogik ist die Lage etwas komplizierter, da eine Klausel eine geschlossene All-Formel beschreibt und dadurch (i.a. unendlich viele) Substitutionsinstanzen besitzt.

Sind nämlich x_1, \dots, x_n die Variablen in einer Klausel C , so ist für beliebige Terme t_1, \dots, t_n die Klausel $C(x_1/t_1) (x_2/t_2) \dots (x_n/t_n)$ eine logische Konsequenz von C . Wir definieren erst den formalen Begriff der Substitution.

Substitution

$\sigma: V \rightarrow T$ mit $\sigma(v) \neq v$ nur für endlich viele Variablen $v \in V$.

Notation für $\sigma(x_1)=t_1, \dots, \sigma(x_n)=t_n$ für $x_i \neq t_i$ und $\sigma(v)=v$ für $v \notin \{x_1, \dots, x_n\}$:

$$\sigma = \{x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n\}$$

Homomorphe Erweiterung:

- $\sigma(c)=c$ für $c \in KS$
- $\sigma(f(t_1, \dots, t_m)) = f(\sigma(t_1), \dots, \sigma(t_m))$ für $f \in FS_m$
- $\sigma(P(t_1, \dots, t_k)) = P(\sigma(t_1), \dots, \sigma(t_k))$ für $P \in PS_k$

Erweiterung auf Mengen:

$$\sigma\{X_1, \dots, X_n\} = \{\sigma(X_1), \dots, \sigma(X_n)\}$$

Substitution: Zusammensetzung, Allgemeinheit

Zusammensetzung von Substitutionen:

$$\tau \circ \sigma(v) = \tau(\sigma(v))$$

Allgemeinheit:

A_1, A_2 Atome

A_1 heisst allgemeiner als A_2 wenn ein σ existiert mit

$$\sigma(A_1) = A_2$$

Schreibweise: $A_1 \leq_s A_2$

σ heisst allgemeiner als τ ($\sigma \leq_s \tau$) wenn ein θ existiert mit

$$\theta \circ \sigma = \tau$$

Beispiel

$$A = P(x, f(y), f(y))$$

$$B = P(x', y', f(x'))$$

Sei t ein Term welcher von x und x' verschieden ist.

Dann gilt für

$$\sigma_t = \{ x \leftarrow t, y \leftarrow t, x' \leftarrow t, y' \leftarrow f(t) \}$$

$$\sigma_t(P(x, f(y), f(y))) = \sigma_t(P(x', y', f(x'))) = P(t, f(t), f(t))$$

Ebenso gilt für die Substitution $\sigma = \{ y \leftarrow x, x' \leftarrow x, y' \leftarrow f(x) \}$

$$\sigma(P(x, f(y), f(y))) = \sigma(P(x', y', f(x'))) = P(x, f(x), f(x))$$

Offensichtlich gilt $\sigma \leq_s \sigma_t$ wegen $\{ x \leftarrow t \} \circ \sigma = \sigma_t$

Unifikation: (allgemeinster) Unifikator

Sei A nicht-leere Menge von Atomen oder von Termen.

σ heisst **Unifikator** von A wenn für alle $A', A'' \in A$ gilt:

$$\sigma(A') = \sigma(A'')$$

σ heisst **allgemeinster Unifikator** (A.U.) von A wenn für jeden Unifikator θ von A gilt: $\sigma \leq_s \theta$.

Bemerkung:

Ist $A = \{A'\}$ dann ist jede Substitution Unifikator von A und die identische Substitution $\varepsilon = \{\}$ ist der A.U. von A .

Unifikation

Gibt es immer einen A.U. einer unifizierbaren Menge?
Sind A.U. algorithmisch berechenbar?

JA!

Die Unifikation von Atomen kann stets auf die Unifikation von Termen reduziert werden.

Jeder Algorithmus, der die Unifizierbarkeit von Termmengen entscheidet und im Falle der Unifizierbarkeit einen A.U. berechnet, wird als Unifikationsalgorithmus bezeichnet.

Unifikationsalgorithmus

Regel-basierte Methode zur Lösung von Termgleichungssystemen

Ein **System von Termgleichungen** ist eine endliche Menge der Form $E = \{ s_1 \doteq t_1, \dots, s_n \doteq t_n \}$ wobei s_i und t_i Terme sind.

Eine Substitution θ heisst

Lösung von E wenn für alle $i = 1, \dots, n$ gilt

$$\theta(s_i) = \theta(t_i)$$

Eine Lösung σ heisst **allgemeinste Lösung** von E , wenn für alle Lösungen θ von E $\sigma \leq_s \theta$ gilt.

Äquivalenz und gelöste Form

Zwei Systeme E und E' heissen **äquivalent**, wenn sie dieselbe Menge von Lösungen besitzen.

Sei $E = \{ s_1 \doteq t_1, \dots, s_n \doteq t_n \}$.

E ist in **gelöster Form** wenn $\{ s_1, \dots, s_n \} \subseteq V$ und jedes s_i nur einmal in E vorkommt.

Regeln

Redundanzelimination:

(delete) Wenn E eine Gleichung $s \doteq s$ enthält, dann ersetze E durch $E - \{s \doteq s\}$

Gleichungen orientieren:

(orient) Wenn E eine Gleichung der Form $t \doteq v$ enthält, wobei v eine Variable ist, t aber nicht, dann ersetze E durch $(E - \{t \doteq v\}) \cup \{v \doteq t\}$

Unlösbarkeit: Problemreduktion auf \perp

(clash) Wenn E eine Gleichung der Form $s \doteq t$ enthält, und s, t Funktionsterme mit verschiedenen Führungssymbolen sind, dann ersetze E durch \perp .

(occurs check) Ist s eine Variable mit $s \neq t$ und kommt s in t vor, dann ersetze E durch \perp .

Regeln

Dekomposition und Ersetzung:

(decompose) Sei $s \doteq t$ eine Gleichung in E sodass ein Funktionssymbol $f \in FS_n$ und Terme $s_1, \dots, s_n, t_1, \dots, t_n$ existieren mit

$$s = f(s_1, \dots, s_n) \text{ und } t = f(t_1, \dots, t_n)$$

dann ersetze E durch $(E - \{s \doteq t\}) \cup \{s_1 \doteq t_1, \dots, s_n \doteq t_n\}$

(eliminate) Sei $s \doteq t$ eine Gleichung in E mit den Eigenschaften

- (1) s ist eine Variable
- (2) s kommt nicht in t vor und
- (3) s kommt in $E - \{s \doteq t\}$ vor

dann ersetze E durch $\{s \leftarrow t\}(E - \{s \doteq t\}) \cup \{s \doteq t\}$

(d.h., s wird im Rest von E durch Substitution eliminiert.)

Irreduzible Form

Sei R die Menge der Regeln (delete), (orient), (clash), (occurs check), (decompose), (eliminate).

Wenn E mittels einer Regel in R in E' transformiert wird, so schreiben wir $E \succ E'$. Für die reflexive und transitive Hülle von \succ schreiben wir \succ^* .

Ein System E heisst **irreduzibel**, wenn keine Regel in R auf E anwendbar ist. \perp ist per definitionem irreduzibel.

Unifikation bedeutet also nichts anderes als die Transformation eines Systems in irreduzible Form, wobei eine beliebige Reihenfolge der Regelanwendung gestattet ist.

Unifikationstheorem

Jedes System in gelöster Form ist irreduzibel. Umgekehrt ist jedes irreduzible System entweder in gelöster Form oder \perp .

Unifikationstheorem:

R ist ein Unifikationssystem, das bedeutet

- R terminiert immer
- Wenn E lösbar ist, dann existiert ein System E' mit
$$E \succ^* E',$$

wobei E' in gelöster Form ist und die durch E' definierte Substitution die allgemeinste Lösung von E ist.

Der Resolutionskalkül

Resolution: Unifikation und Schnitt

notwendig:

- Umbenennen von Variablen
- Faktorisierung

Beispiel: Umbenennung

Klauseln $C = \{Q(x), P(x)\}$, $D = \{\neg P(f(x)), Q(x)\}$

$\{P(x), \neg P(f(x))\}$ nicht unifizierbar!
 $\{x \doteq f(x)\}$ unlösbar

Daher **Umbenennung**:

$C = \{Q(x), P(x)\}$, $D' = \{\neg P(f(y)), Q(y)\}$

$\{P(x), \neg P(f(y))\}$ unifizierbar
A.U. $\sigma = \{x \leftarrow f(y)\}$

Beispiel: Faktorisierung

$$C = \{ C_1: \{P(x), P(y)\}, C_2: \{\neg P(u), \neg P(v)\} \}$$

Hier gibt es 4 Möglichkeiten, je zwei Atome zu unifizieren, und dann die Schnittregel anzuwenden.

In allen Fällen ergibt sich die Klausel $C_3: \{P(x), \neg P(u)\}$

Nach entsprechender Variablenumbenennung liefern die Klauseln $C_1: \{P(x), P(y)\}$ und $C_3: \{P(z), \neg P(u)\}$ nach Unifikation und anschließendem Schnitt die Klausel $\{P(y), P(z)\}$.

Lösung:

Unifikation innerhalb von Klauseln:

Wende auf C_1 $\sigma = \{ x \leftarrow y \}$ und auf C_2 $\tau = \{ u \leftarrow v \}$ an, dann gilt $\sigma(C_1) = \{P(y)\}$ und $\tau(C_2) = \{\neg P(v)\}$.

Mit $\lambda = \{ y \leftarrow v \}$ gilt dann

$$\lambda \circ \sigma(C_1) = \{P(v)\}$$

$$\lambda \circ \tau(C_2) = \{\neg P(v)\}$$

was, wendet man nun den Schnitt an, die leere Klausel ergibt.

Resolutionsregel

Seien $A_1 \in C$ und $\neg A_2 \in D$ für variablenfremde Klauseln C, D und A_1, A_2 unifizierbare Atomformeln.

Resolutionsregel:

$$\frac{C \qquad D}{(\sigma(C) - \sigma(A_1)) \cup (\sigma(D) - \sigma(A_2))} \quad \text{Resolvent von } C \text{ und } D$$

wobei σ der A.U. von A_1 und A_2 ist.

Resolutionskalkül

Der **Resolutionskalkül** ist jener Kalkül auf Klauseln, dessen einzige Regel die Resolution ist.

Eine Klausel, die nicht Konklusion einer Regelanwendung ist, heisst **Axiomklausel**.

Eine **Resolutionsableitung** heisst Resolutionsableitung aus C , wenn die Axiomklauseln Klauseln aus C oder Varianten von Klauseln aus C sind.

Eine Resolutionsableitung von \square aus einer Menge von Klauseln C heisst **Resolutionswiderlegung** von C .

Beispiel

$$A = P(a) \wedge (\forall x) (P(x) \rightarrow P(f(x))) \rightarrow P(f(f(a)))$$

Klauselform von $\neg A$:

$$C = \{ \{P(a)\}, \{\neg P(x), P(f(x))\}, \{\neg P(f(f(a)))\} \}$$

Resolutionswiderlegung von C:

$$\begin{array}{c} \frac{\{P(a)\} \quad \{\neg P(x), P(f(x))\}}{\{P(f(a))\} \quad \{\neg P(y), P(f(y))\}} \\ \frac{\{P(f(f(a)))\} \quad \{\neg P(f(f(a)))\}}{\square} \end{array}$$

A.U. Nr. 1: $\{x \leftarrow a\}$ A.U. Nr. 2: $\{y \leftarrow f(a)\}$

Korrektheit und Vollständigkeit

Resolution ist **korrekt**:

Wenn es eine Resolutionswiderlegung einer Klauselmeng C gibt, dann ist C unerfüllbar.

Resolution ist **vollständig**:

Ist C eine unerfüllbare Klauselmeng dann existiert eine Resolutionswiderlegung von C .