Logik für Informatiker

Vorlesung 13, Teil 1: Resolution

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Prädikatenlogische Resolution

Wiederholung

Grundidee

- Vor Resolutionsschritt durch geeignete Substitution komplementäres Paar von Literalen erzeugen

Prädikatenlogische Resolution

Wiederholung

Möglichkeit für Resolutionsregel

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

wobei

- die Elternklauseln keine Variablen gemeinsam haben (bereinigt)
  $\mapsto$ ggf. umbenennen
- $\sigma(L) = \sigma(L')$

Prädikatenlogische Resolution

Wiederholung

Grundidee

Vor Resolutionsschritt durch geeignete Substitution komplementäres Paar von Literalen erzeugen

Möglichkeit für Resolutionsregel

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

wobei

- die Elternklauseln keine Variablen gemeinsam haben (bereinigt)
  $\mapsto$ ggf. umbenennen
- $\sigma(L) = \sigma(L')$

Nachteil: Viel zu viele Substitutionen σ mit $\sigma(L) = \sigma(L')$
Idee: Wähle die "allgemeinste" Substitution, mit $\sigma(L) = \sigma(L')$

Unifikationsalgorithmus von Martelli/Montanari

$$
\begin{array}{lrcl}
(1) & t \stackrel{?}{=} t, E & \Rightarrow_{MM} & E \\
(2) & f(s_1, \ldots, s_n) \stackrel{?}{=} f(t_1, \ldots, t_n), E & \Rightarrow_{MM} & s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n, E \\
(3) & f(\ldots) \stackrel{?}{=} g(\ldots), E & \Rightarrow_{MM} & \bot \\
(4) & x \stackrel{?}{=} t, E & \Rightarrow_{MM} & x \stackrel{?}{=} t, E[t/x] \\
 & & & \text{falls } x \in var(E), x \notin var(t) \\
(5) & x \stackrel{?}{=} t, E & \Rightarrow_{MM} & \bot \\
 & & & \text{falls } x \neq t, x \in var(t) \\
(6) & t \stackrel{?}{=} x, E & \Rightarrow_{MM} & x \stackrel{?}{=} t, E \\
 & & & \text{falls } t \notin X
\end{array}
$$

Unifikationsalgorithmus von Martelli/Montanari

1. *Sei $\sigma = \varnothing$ die leere (oder "identische") Substitution.*

2. *Falls $|\sigma(K)| = 1$ ist, dann brich ab und gib σ als mgu von K aus.*

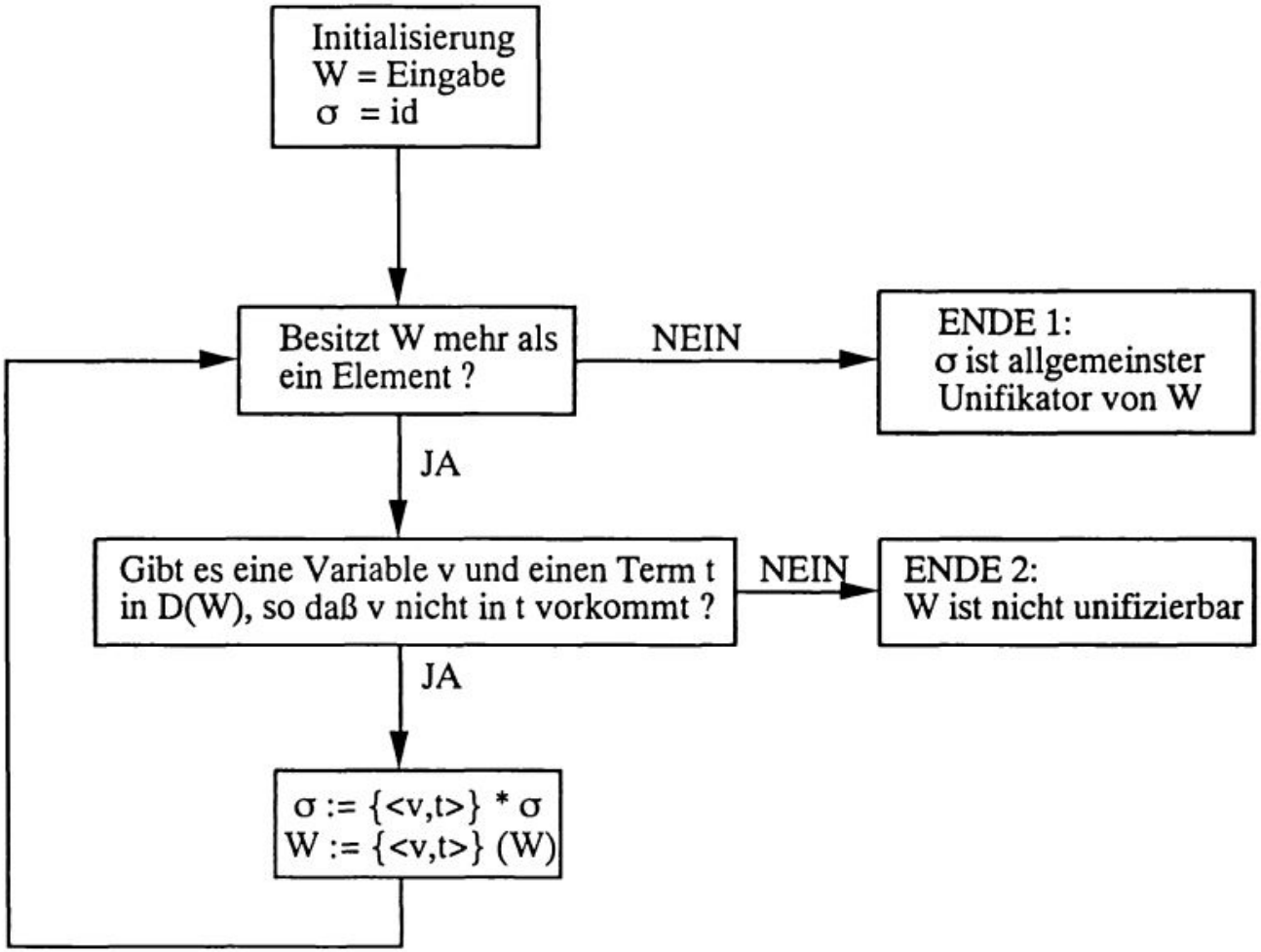
3. *Sonst durchsuche alle $\sigma(L_i)$ parallel von links nach rechts, bis in zwei Literalen die gelesenen Zeichen verschieden sind.*

4. *Falls keines der beiden Zeichen eine Variable ist, dann brich mit* Clash Failure *ab.*

5. *Sonst sei X die Variable und t der Teilterm im anderen Literal (hierbei kann t auch eine Variable sein). Falls X in t vorkommt, dann brich mit* Occur Failure *ab. (Diese Überprüfung bezeichnet man als* Occur Check.*)*

6. *Sonst setze $\sigma = \{X/t\} \circ \sigma$ und gehe zurück zu Schritt 2.*

Unifikationsalgorithmus von Martelli/Montanari



Prädikatenlogische Resolution

Grundidee

Vor Resolutionsschritt durch geeignete Substitution komplementäres Paar von Literalen erzeugen

Möglichkeit für Resolutionsregel

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

wobei

- die Elternklauseln keine Variablen gemeinsam haben (bereinigt)
  $\mapsto$ ggf. umbenennen
- $\sigma = \mathrm{mgu}(L, L')$

Prädikatenlogische Resolution

Resolutionsregel in dieser Form alleine unvollständig für Prädikatenlogik

Beispiel:

$\{\{p(x), p(y)\}, \{\neg p(u), \neg p(v)\}\}$

- unerfüllbar
- aber nur Resolventen der Länge 2

Prädikatenlogische Resolution

Faktorisierung

$$\frac{\{L_1, \ldots, L_n\} \cup C}{(\{L_1, \ldots, L_n\} \cup C)\sigma}$$

wobei

- σ allgemeinster Unifikator (MGU) von $\{L_1, \ldots, L_n\}$ ist

$(\{L_1, \ldots, L_n\} \cup C)\sigma$ heißt Faktor von $\{L_1, \ldots, L_n\} \cup C$

Beispiel für Faktorisierung

$$\frac{\{p(x),p(y),r(y,z)\}}{\{p(x),r(x,z)\}} \qquad \mathrm{mgu}(p(x), p(y)) = [x/y]$$

$$\frac{\{p(x),p(y),p(a),r(y,z)\}}{\{p(a),r(a,z)\}} \qquad \mathrm{mgu}(p(x), p(y), p(a)) = [a/x, a/y]$$

$$\frac{\{p(b),p(y),p(a),r(y,z)\}}{\{p(b),p(a),r(b,z)\}} \qquad \mathrm{mgu}(p(b), p(y)) = [b/y]$$

$$\frac{\{p(b),p(y),p(a),r(y,z)\}}{\{p(b),p(a),r(a,z)\}} \qquad \mathrm{mgu}(p(y), p(a)) = [a/y]$$

Resolutionskalkül RES für allgemeine Klauseln

$$\frac{C \cup \{A_1\} \qquad D \cup \{\neg A_2\}}{(C \cup D)\sigma} \quad \text{falls } \sigma = \text{mgu}(A_1, A_2) \qquad \text{[Resolution]}$$

$$\frac{C \cup \{L_1, L_2\}}{(C \cup \{L_1\})\sigma} \quad \text{falls } \sigma = \text{mgu}(L_1, L_2) \qquad \text{[Faktorisierung]}$$

Es wird immer implizit angenommen, dass die Variablen in einer der beiden Prämissen der Resolutionsregel ggfs. (bijektiv) umbenannt werden, so dass sie disjunkt mit denen der anderen Prämisse sind.

Dieses implizite Umbenennen werden wir nicht formalisieren.

Welche Variablennamen man verwendet ist egal.

Beispielsweise könnte man sich vorstellen, dass am Anfang alle Klauseln paarweise variablendisjunkt sind und das Unifikatoren so gewählt werden, dass in ihrem Wertebereich nur neue Variablen vorkommen.

BEISPIEL

1. $\{P(x), P(f(x)), \neg Q(x)\}$ [Gegeben]
2. $\{\neg P(y)\}$ [Gegeben]
3. $\{P(g(x', x)), Q(x)\}$ [Gegeben]

Beispiel

1.	$\{P(x), P(f(x)), \neg Q(x)\}$	[Gegeben]
2.	$\{\neg P(y)\}$	[Gegeben]
3.	$\{P(g(x', x'')), Q(x'')\}$	[Gegeben; Bereinigt]
4.	$\{P(f(x)), \neg Q(x)\}$	[Res. 1, 2], $\text{mgu}(P(x), P(y)) = [x/y]$
5.	$\{\neg Q(x)\}$	[Res. 4, 2], $\text{mgu}(P(y), P(f(x))) = [f(x)/y]$
6.	$\{Q(x'')\}$	[Res. 3, 2], $\text{mgu}(P(y), P(g(x', x''))) = [g(x', x'')/y$
7.	$\bot$	[Res. 5, 6], $\text{mgu}(Q(x), Q(x'')) = [x/x'']$

Notation

Sei N eine Klauselmenge und

$$\mathrm{Res}(N) = N \cup \{R \mid R \text{ ist eine Resolvente zweier Klauseln aus } N \text{ oder Resultat der Faktorisierung einer Klausel aus } N\}$$

$$\mathrm{Res}^0(N) = N$$

$$\mathrm{Res}^{n+1}(N) = \mathrm{Res}(\mathrm{Res}^n(N))$$

$$\mathrm{Res}^\star(N) = \bigcup_{n \in \mathbb{N}} \mathrm{Res}^n(N)$$

(bezeichnet die Vereinigung der Ergebnisse aus aller möglichen Resolutions- und Faktorisierungsschritte auf N)

Anwendungen

- Logische Programmierung (Prolog, Answer Set Programming): Beweis als Programmablauf
- Industrielle Anwendungen: KI, virtuelle Agenten, Strategien, maschinelles Lernen, etc.
- SAT Solver; Theorem Prover
- Semantic Web: Logik pur
- Software Verifikation: Korrektheitsbeweise von Programmen, Schaltkreisen, etc; diese benötigen Rechnerunterstützung; automatisches Beweisen

Anwendungen

Abstrakte Datentypen

- Definiere Datentypen und finde effiziente Implementationen dazu!
- Dafür werden abstrakte Eigenschaften dieser Datentypen definiert.
- Beispiel: Datentyp *Stack (Stapel)*: Sei Σ ein Alphabet. Um einen Stapel S über Σ zu definieren, sei *clear* eine konstante Funktion.

$$clear \in S$$
$$\forall s \in S \, \forall x \in \Sigma \, (push(s, x) \in S)$$
$$\forall s \in S \, (s \neq clear \Rightarrow pop(s) \in S)$$
$$\forall s \in S \, (s \neq clear \Rightarrow top(s) \in \Sigma)$$
$$\forall s \in S \, (empty(s) \in \{true, false\})$$

Datentyp Stack

- *empty* ist ein Prädikat
- *clear, push, pop* sind Funktionen
- Eigenschaften der *push* Funktion:

$$\forall s \in S\ \forall x \in \Sigma\ (push(s,x) \neq clear)$$
$$\forall s \in S\ \forall x,y \in \Sigma\ (push(s,x) = push(s,y) \Rightarrow x = y)$$
$$\forall s,t \in S\ \forall x \in \Sigma\ (push(s,x) = push(t,x) \Rightarrow s = t)$$

- Dazu kommen noch folgende Eigenschaften:

$$\forall s \in S\ \forall x \in \Sigma \quad (top(push(s,x)) = x \wedge pop(push(s,x)) = s)$$
$$\forall s \in S\ \forall x \in \Sigma \quad (empty(push(s,x)) = false)$$
$$empty(clear) = true$$

Fragestellungen

- Ist diese Spezifikation korrekt? D.h. existiert ein S, so dass die obigen Eigenschaften erfüllt sind?
- Ist diese Spezifikation vollständig? Entsprechen diese Axiomen unseren Erwartungen, was ein Stapel sein sollte?
- Diese sind zentrale Fragen in dem Entwurf formaler Systeme.

Weitere Anwendungen

- Software Entwicklung und Verifikation
- Microsoft Windows bluescreen
- Künstliche Intelligenz
- Knowledge Representation, u.v.m.

Last but not least

Nicht vergessen Spass zu haben!