Was bisher geschah

Wissensrepräsentation und -verarbeitung in Logiken

klassische Aussagenlogik:

- Syntax (Wiederholung)
- Semantik (Wiederholung)
- Resolution

klassische Prädikatenlogik:

- Syntax (Wiederholung)
- Semantik (Wiederholung)
- Normalformen:
  - bereinigt
  - Pränex
  - Skolem ($\exists$-Eliminierung)
  - Klausel (Menge von Klauseln, Notation ohne Quantoren)
- Substitutionen,Unifikatoren
- allgemeinster Unifikator
- Unifikationsalgorithmus

Modellierungsbeispiel

Wissensbasis:

- Otto kann Salsa tanzen.
- Paul kann Tango und Walzer tanzen.
- Maria kann Salsa tanzen.
- Salsatänzer (beliebigen Geschlechts) mögen jeden, der Salsa tanzen kann.
- Anna mag alle, die (irgendeinen Tanz) tanzen können.
- Anna tanzt jeden Tanz, den jemand tanzt, den sie mag.

Fragen:

- Tanzt Anna Tango?
- Wer mag Otto?
- Welche Tänze tanzt Anna?
- Wer tanzt welche Tänze?

Lösungsidee (Wiederholung)

1. Formulierung der Wissensbasis als Formelmenge (Regelmenge)
2. Formulierung der Frage als Formel (Behauptung, evtl. mit Variablen)
3. Transformation in Klauselmenge
4. Grundinstanziierung (sofern möglich)
5. Lösung durch aussagenlogische Resolution

Nachteil:
Grundinstanziierung ergibt i.A. sehr große (evtl. unendliche) Formelmengen

Prädikatenlogische Resolution

Wiederholung: Resolutionsregel (Aussagenlogik)

$$\frac{l_1 \vee \cdots \vee l_n \vee l \qquad l'_1 \vee \cdots \vee l'_m \vee \neg l}{l_1 \vee \cdots \vee l_n \vee l'_1 \vee \cdots \vee l'_m}$$

Ziel: Erweiterung auf Prädikatenlogik
Beispiel: Für jeden Indianer eines Stammes gilt:
Sein Vater trägt eine rote Feder und er selbst trägt keine rote Feder.
$\varphi = \forall x(R(f(x)) \wedge \neg R(x))$ ist unerfüllbar.
Interpretiert in einer Struktur $\mathcal{A} = (A, [\![\cdot]\!]_{\mathcal{A}})$, repräsentiert φ die Formelmenge (Grundinstanziierung)

$$\bigcup_{a \in A} \{R(f(a)) \wedge \neg R(a), R(f(f(a))) \wedge \neg R(f(a)), \ldots\}$$

φ repräsentiert die Klauselmenge

$$\bigcup_{a \in A} \{\underline{R(f(a))}, \neg R(a), R(f(f(a))), \underline{\neg R(f(a))}, \ldots\}$$

unerfüllbar nach (aussagenlogischer) Resolution
Ziel: Resolution geeigneter Instanzen der zu resolvierenden Klauseln.

Prädikatenlogische Resolution

(Einfache) Resolutionsregel:
falls $\theta = \text{mgu}(l, l')$ existiert:

$$\frac{l_1 \vee \cdots \vee l_n \vee l \qquad l'_1 \vee \cdots \vee l'_m \vee \neg l'}{\theta(l_1 \vee \cdots \vee l_n \vee l'_1 \vee \cdots \vee l'_m)}$$

Beispiel:

$$\{P(f(x)) \vee \underline{R(g(x), x)}, \underline{\neg R(u, v)} \vee \neg S(u, v)\} \models P(f(x)) \vee \neg S(g(x), x)$$

mit Unifikator $\theta = [u \mapsto g(x), v \mapsto x]$

Volle Resolutionsregel erlaubt Eliminierung aller unifizierbaren Instanzen eines Literals

Beispiel:

$$\{\underline{P(x, b)} \vee \underline{P(a, y)} \vee Q(x, f(y)), \underline{\neg P(z, w)} \vee \neg Q(w, z)\} \models Q(a, f(b)) \vee \neg Q(b, a)$$

mit Unifikator $\theta = [x \mapsto a, y \mapsto b, z \mapsto a, w \mapsto b]$

Prädikatenlogische Resolution

Wiederholung:
In einer Logik L (definiert durch Syntax, Semantik)
ist der Kalkül (Schlussverfahren) $\vdash \subseteq 2^L \times L$
gegenüber der Folgerungsrelation $\models \subseteq 2^L \times L$ in

korrekt gdw.
für jede Formelmenge $\Phi \subseteq L$ und jede Formel $\psi \in L$
aus $\Phi \vdash \psi$ folgt $\Phi \models \psi$

vollständig gdw.
für jede Formelmenge $\Phi \subseteq L$ und jede Formel $\psi \in L$
aus $\Phi \models \psi$ folgt $\Phi \vdash \psi$

Satz
In AL(P) *und* FOL($\Sigma, \mathbb{X}$) *ist das prädikatenlogische Resolutionsverfahren korrekt und vollständig.*

Prädikatenlogische Resolution – Beispiele

- Gilt

$$\left\{ \begin{array}{l} E(a,b), E(a,c), E(b,c), E(c,d), \\ E(x,y) \rightarrow P(x,y), \\ (E(u,v) \wedge P(v,w)) \rightarrow P(u,w) \end{array} \right\} \models P(a,d)?$$

- $\{\neg b(x) \vee r(y,y)) \vee r(x,y), \neg b(z) \vee \neg r(u,u)) \vee \neg r(z,u), b(a)\}$ ist unerfüllbar

Beispiel: Barbier (Russellsches Paradox)

informal: Der Barbier im Dorf rasiert (genau) diejenigen, die sich nicht selbt rasieren.

Problem: Frage: Rasiert der Barbier (b) sich selbst? 2 Fälle:

1. Ann.: b rasiert sich selbst $\rightarrow$ b rasiert sich nicht
2. Ann.: b rasiert sich nicht selbst $\rightarrow$ b rasiert sich

Widerspruch in beiden Fällen

Folgerung: Also existiert kein solcher Barbier.

(analog existiert keine Menge aller Mengen)

formal: Kontext: $\forall x(r(b,x) \leftrightarrow \neg r(x,x))$

in Klauselform: $\Phi =$

$\{\neg r(b,x) \vee \neg r(x,x), r(x,x) \vee r(b,x)\}$

Resolution: 2 Fälle: Frage (als Behauptung)

$\psi \in \{r(b,b), \neg r(b,b)\}$

1. $\{\neg r(b,x) \vee \neg r(x,x), r(x,x) \vee r(b,x), \neg r(b,b)\}$
2. $\{\neg r(b,x) \vee \neg r(x,x), r(x,x) \vee r(b,x), r(b,b)\}$

in beiden Fällen Resolution zu $\mathbf{f}$ mit $\sigma = \{x \mapsto b\}$

Horn-Logik (definite Regeln)

Fragment der klassischen Prädikatenlogik (der ersten Stufe):

Horn-Klausel (endliche) Disjunktion von Literalen, die höchstens ein positives Literal enthalten.

Horn-Formel (endliche) Konjunktion von Horn-Klauseln

(definites) logisches Programm (endliche) Menge von Hornklauseln mit je einem positiven Literal

Darstellung von Horn-Klauseln:

- Hornklausel mit einem positiven Literal:
  $\neg b_1 \vee \ldots \vee \neg b_n \vee h \equiv (b_1 \wedge \ldots \wedge b_n) \rightarrow h$
  Regeln
  Spezialfall Fakt h (Regel ohne Rumpf)
- Hornklausel ohne positive Literale:
  $\neg b_1 \vee \ldots \vee \neg b_n \vee \equiv \neg b_1 \vee \ldots \vee \neg b_n \vee \mathbf{f} \equiv (b_1 \wedge \ldots \wedge b_n) \rightarrow \mathbf{f}$
  Anfrage (Query, Zielklausel, herzuleitende Formel)

Achtung:
Nicht zu jeder Formel existiert eine äquivalente Formel in Horn-Logik (z.B. $p \vee q$, Formalisierung Barbier).

Horn-Logik ist also ein echtes Fragment der klassischen Prädikatenlogik

Logische Programmierung – Prolog

Regel: Hornklausel, d.h. Klausel $\neg b_1 \vee \ldots \vee \neg b_n \vee h$ mit genau einem positiven Literal
äquivalent: $(b_1 \wedge \ldots \wedge b_n) \rightarrow h$
Rumpf $b_1 \wedge \ldots \wedge b_n$, Kopf h
Prolog-Syntax: `h :- b1, ..., bn.`

Fakt: Atom h (positives Literal, Regel ohne Rumpf)
Prolog-Syntax: `h.`

Zielklausel (Query): Klausel $\neg b_1 \vee \ldots \vee \neg b_n$ ohne positives Literal (Regel ohne Kopf)
äquivalent: $\neg(b_1 \wedge \ldots \wedge b_n)$
Prolog-Syntax: `?- b1, ..., bn.`

logisches Programm (Wissensbasis):
endliche Menge von (Fakten und) Regeln.

übliche Semantik der klassischen Prädikatenlogik

Logische Programmierung – Beispiel

Wissensbasis: Prolog-Programm P

```
liest(paul,krimi).
liest(bob,zeitung).
liest(tina,arztroman).
mag(tina,X) :- liest(X,krimi).
mag(X,Y) :- liest(X,krimi), liest(Y,zeitung).
```

repräsentiert die Formelmenge

$$\Phi = \left\{ \begin{array}{l} l(p,k), l(b,z), l(t,a), \forall x\,(l(x,k) \rightarrow m(t,x))\,, \\ \forall x\ \forall y\ \ ((l(x,k) \wedge l(y,z)) \rightarrow m(x,y)) \end{array} \right\}$$

Aufgabenbeschreibung: Zielklausel (Anfrage, Query)

```
?- mag (tina,paul).
```

repräsentiert die Formel $\psi = m(t,p)$

Paar (Programm, Zielklausel) repräsentiert die Frage:
(Für welche Belegungen) Gilt $\Phi \models \psi$?

Lösung ja und Belegung β mit $\Phi \models \psi$
oder nein

Lösungsverfahren Resolution auf $\Phi \cup \{\neg\psi\}$

Logische Programmierung – Antworten

Variablen in Zielklauseln:

```
?- mag (tina,X).
```

repräsentiert die Formel $\psi = \exists x\ m(t, x)$
Anfrage: Für welche x gilt $\Phi \models \psi$?
Antwort: erfüllende Belegung für x
Lösung durch Resolution auf

$$\Phi \cup \{\neg\exists x\ \psi\} = \Phi \cup \{\neg\exists x\ m(t, x)\} = \Phi \cup \{\neg\exists x\ m(t, y)\} = \Phi \cup \{\forall y\ \neg m(t, y)\}$$

Beispiel als Klauselmenge:

$$\{l(p, k), l(b, z), l(t, a), l(x, k) \rightarrow m(t, x), \neg m(t, y)\}$$

mögliche Antworten:
während der Resolution berechnete Substitutionen für x
Was bedeutet `?- mag (X,Y).`
Welche Antworten?

Prolog: Semantik

Wissensbasis: Prolog-Programm (Programmklauseln)

$$P = \{b_{i1} \wedge \cdots \wedge b_{in_i} \rightarrow h_i \mid i \in \{1, \ldots n\}\}$$

Zielklausel: $b_1 \wedge \cdots \wedge b_m$

Antwort: no

yes und evtl. Variablenbelegung β

Was ist die Bedeutung der drei Komponenten Wissensbasis, Zielklausel und Antwort? Wie hängen sie zusammen?

deklarative Semantik: logische Bedeutung der Komponenten und der Antwort

operationale Semantik: Verfahren zum maschinellen Finden der Antwort (durch den Prolog-Interpreter) aus den Komponenten

Prolog – deklarative Semantik

1. Programm P repräsentiert die Formelmenge $\Phi \subseteq \mathrm{FOL}(\Sigma, \mathbb{X})$

   $$\Phi = \{\forall\,(b_{i1} \wedge \cdots \wedge b_{in_i} \rightarrow h_i) \mid i \in \{1, \ldots n\}\}$$

   ($\forall$-Abschlüsse aller Regeln)
   Σ enthält genau alle in P vorkommenen Funktions- und Relationssymbole

2. Zielklausel $\psi = (b_1 \wedge \cdots \wedge b_m) \in \mathrm{FOL}(\Sigma, \mathbb{X})$

3. Antwort:

   yes und Belegung $\beta : \mathbb{X} \rightarrow \mathrm{Term}(\Sigma_F, \emptyset)$,
   falls $\Phi \cup \{\beta(\psi)\}$ unerfüllbar
   no , falls $\Phi \cup \{\psi\}$ erfüllbar

Beispiele:

- Programm `a :- b,c. a :- d. b. b :- d. c. d :- e. d.`
  Anfrage `?- a.`
- Programm
  `p(a). p(b). q(a). q(b). r(b). s(X):-p(X),q(X),r(X).`
  Anfrage `?- s(X).`

Prolog – operationale Semantik

SLD-Resolution:
linear resolution with selection function for definite clauses

- definite clauses:
  Horn-Programme (höchstens ein positives Literal je Klausel)
- linear resolution:
  Folge von Resolutionsschritten, wobei in jedem Schritt mit einer Programmklausel resloviert wird
- Input-Resolution
  Folge von Resolutionsschritten, wobei in jedem Schritt nach einem Literal in der im vorigen Schritt erzeugten Klausel (zu Beginn Zielklausel) resolviert wird
- selection function: Auswahl
  - einer Klausel (unter mehreren passenden):
    von oben nach unten (von links nach rechts)
  - des Literals in der Klausel: von links nach rechts

  Tiefensuche mit Backtracking

Beispiel: $\{a \vee b, \neg a \vee b, a \vee \neg b, \neg a \vee \neg b\}$ unerfüllbar