Was bisher geschah

Wissensrepräsentation und -verarbeitung in Logiken

klassische Aussagenlogik: Syntax, Semantik, Resolution

klassische Prädikatenlogik:

- Wiederholung Syntax, Semantik
- Normalformen: bereinigt, Pränex-, Skolem-, Klausel-
- Substitutionen, Unifikatoren
- allgemeinster Unifikator ($\mathrm{mgu}(s, t)$), Unifikationsalgorithmus
- Prädikatenlogisches Resolutionsverfahren

logische Programmierung in Prolog:

- Syntax: Horn-Klauseln in Regel-Form
- Wissensbasis: logisches Programm aus Fakten und (definiten) Regeln
- Anfrage: Konjunktion von Literalen
- Semantik:

  deklarativ: analog FOL, jedoch nur kleinstes Termmodell

  operational: SLD-Resolution

Vergleich LP (Prolog) mit FP (Haskell)

Deklarative Programmierung enthält z.B.

- logische Programmierung
- funktionale Programmierung
- Constraint-Programmierung

und Kombinationen dieser Paradigmen

Paradigma	logisch	funktional
Beispielsprache	Prolog	Haskell
Berechnungsmodell	FOL	λ-Kalkül
Programmbaustein	Horn-Klausel	Termgleichung
Ausdrücke	Atome	Terme
Auszuwerten	Konjunktion von Atomen	Term
Finden passender Programmteile	Unifikation	Matching
Auswertung	SLD-Resolution	Reduktion von λ-Ausdrücken, Termersetzung

Beispiel – Peano-Zahlen

Prolog:
```
plus(zero,Y,Y).
plus(succ(X),Y,succ(Z)) :- plus(X,Y,Z).
```

Anfragen, z.B.:

```
?- plus(succ(zero),succ(succ(zero)),X).
?- plus(succ(zero),X,succ(succ(zero))).
?- plus(X,Y,succ(succ(zero))).
```

Haskell:
```
plus Zero y        = y
plus ( Succ x ) y = Succ (plus x y)
```

Anfrage, z.B.:

```
plus (Succ Zero) (Succ (Succ Zero))
```

Prolog:
```
times(zero,Y,zero).
times(succ(X),Y,Z) :- times(X,Y,H), plus(H,Y,Z).
```

Haskell:
```
times Zero y        = Zero
times ( Succ x ) y = plus (times x y) y
```

Logische Programme zur Lösung von Planungs-Aufgaben

generisches Programm:

```
plan(P) :- start(Start),
           action(Start,[Start],Q),
           reverse(Q,P).

action(S,P,P) :- goal(S).
action(S,Visited,P) :- next(S,N),
                       safe(N),
                       no_loop(N,Visited),
                       action(N,[N|Visited],P).

no_loop(S,Visited) :- \+member(S,Visited).
```

für jeweilige Aufgabe zu implementieren:

```
start(...).
goal(...).
next(S,N) :- ...
safe(S) :- ...
```

Beispiele: Münzspiel, Kannibalen, Ziege

Datalog

Datalog: Anfragesprache für relationale Datenbanken
(Tabellen repräsentieren Relationen, definieren Signatur)

FOL($\Sigma, \mathbb{X}$)-Fragment mit den folgenden Eigenschaften:

Syntax Σ_F enthält nur Konstantensymbole (nullstellige Funktionssymbole)

Semantik Interpretation über einer festen Struktur (definiert durch Datenbank) mit endlicher Trägermenge (Menge aller in der DB vorkommenden Konstanten)

extensionale Prädikate:
in Datenbank definiert
(Tabellen-, Relationenschemata)

intensionale Prädikate:
im Datalog-Programm (Regelköpfen) definiert

Einschränkung (Syntax): Kein Prädikat darf sowohl extensional als auch intensional vorkommen.

Datalog: Syntax und Semantik

Syntax: wie Prolog ohne Funktionssymbole mit Stelligkeit > 1

Datalog-Term: Konstantensymbol oder Variable

Datalog-Atom: `p( t1 , ... , tn )` mit n-stelligem
Relationssymbol $p \in \Sigma_R$ und Termen
`t1 , ... , tn` (Variablen oder Konstanten)

Datalog-Klausel: Regel `h :- b1 , ... , bn.` mit
Datalog-Atomen `b1 , ... , bn , h`
intensionales Prädikat in `h`,
intensionale oder extensionale Prädikate in `bi`

Datalog-Fakt: Datalog-Klausel mit $n = 0$

Datalog-Wissensbasis: endliche Menge von Datalog-Klauseln

Datalog-Anfrage: Formel `?- b1, ... , bn.` mit
Datalog-Atomen `b1, ... , bn`

deklarative Semantik wie in Prolog

Fixpunkt-Semantik über Konsequenzoperator

Beispiel

Aus der Wissensbasis

F1	Tom ist ein Baby. `b(t).`	(extensional)
F2	Tom ist männlich. `m(t).`	(extensional)
R1	Babies sind Kinder. `k(X) :- b(X).`	(intensional)
R2	Männliche Kinder sind Jungen. `j(X) :- k(X),m(X).`	(intensional)
R3	Weibliche Kinder sind Mädchen. `g(X) :- k(X),w(X).`	(intensional)

folgt (ohne gezielte Anfrage): Tom ist ein Kind. Tom ist ein Junge.

Instanz $b(t) \rightarrow k(t)$ der Regel R1 mit Substitution $x \mapsto t$
„feuert“ in der Faktenmenge $F = \{b(t), m(t)\}$,
weil $b(t) \in F$, also Voraussetzung (Rumpf der Regel R1) in F erfüllt.

schrittweise Erweiterung der Faktenmenge um gültige Fakten:

$$
\begin{aligned}
F_0 &= \{b(t), m(t)\} \\
F_1 &= \{b(t), m(t), k(t)\} \quad \text{wegen R1} \\
F_2 &= \{b(t), m(t), k(t), j(t)\} (= F_3) \quad \text{wegen R2}
\end{aligned}
$$

Erweiterung der Faktenmenge

gegeben: logisches Programm $P = F \cup R$ mit

- Faktenmenge $F \subseteq \text{Atom}(P)$ (interpretiert als Zustand)
  repräsentiert Menge aller Instanzen der Fakten,
  Menge von Grundatomen (Herbrand-Interpretation)
  (aus Datenbank oder Grundinstanziierung)
- Regelmenge R

Folge $F_0, F_1, \ldots$ von Faktenmengen (Zuständen) $F_i \in 2^{\text{Atom}(P)}$

$$
\begin{aligned}
F_0 &= F \\
\forall i \in \mathbb{N} : F_{i+1} &= \{h \mid ((B \rightarrow h) \in (F \cup R)) \wedge (F_i \models B)\} \\
F^* &= \bigcup_{n \in \mathbb{N}} F_n
\end{aligned}
$$

Sind alle Regeln Hornklauseln, dann ist F^* kleinstes Modell für P.

(datenorientierte Suche)

Konsequenzoperator

gegeben: Wissensbasis (logisches Programm) $P = F \cup R$
P definiert den Konsequenzoperator $T_P : 2^{\text{Atom}(P)} \to 2^{\text{Atom}(P)}$
auf Faktenmegen $M \subseteq \text{Atom}(P)$:

$$\begin{aligned} T_P(M) &= \{h \mid b \to h \in F \cup R \text{ und } M \models b\} \\ &= \{h \mid (b_1 \wedge \cdots \wedge b_n) \to h \in F \cup R \text{ und } \{b_1, \ldots, b_n\} \subseteq M\} \end{aligned}$$

definiert die Mengen

$$\begin{aligned} T_P^0(\emptyset) &= F \\ T_P^{n+1}(\emptyset) &= T_P\left(T_P^n(\emptyset)\right) \\ &\vdots \\ T_P^*(\emptyset) &= \bigcup_{i \in \mathbb{N}} T_P^i(\emptyset) \end{aligned}$$

Falls $T_P^{n+1} = T_P^n$ gilt, dann ist $T_P^* = T_P^n$.
Für endliche Wissensbasen $P = F \cup R$ wird der Fixpunkt $T_P^*(\emptyset)$ nach endlich vielen Anwendungen von T_P erreicht.
Fixpunkt-Semantik für Datalog:
Ein Atom a ist genau dann aus P ableitbar, wenn $a \in T_P^*(\emptyset)$.

Hülleneigenschaften

Ein Hüllenoperator ist eine Funktion $f : 2^M \to 2^M$ mit den folgenden Eigenschaften (Hülleneigenschaften)

- Für alle Mengen $m, n \in 2^M$ folgt aus $m \subseteq n$, dass $f(m) \subseteq f(n)$ gilt.
  f ist monoton
- Für jede Menge $m \in 2^M$ gilt $m \subseteq f(m)$
  f ist extensiv
- Für jede Menge $m \in 2^M$ gilt $f(f(m)) = f(m)$
  f ist idempotent

Für Wissensbasen $P = (R, F)$ aus definiten Regeln (ohne negative Literale in Rümpfen) ist die Funktion T_P^* ein Hüllenoperator.

Regelbasierte Systeme

Regel : Implikation $r = (\varphi \rightarrow \psi)$

definite Regel (Hornklausel, ohne negative Voraussetzungen):
Implikation $r = (\varphi \rightarrow \psi)$ mit
$\varphi = (b_1 \wedge \ldots \wedge b_n)$ und $\psi = h$
mit (aussagen- oder prädikatenlogischen) Atomen
$b_1, \ldots, b_n, h$

Bestandteile der Regel $r = (\varphi \rightarrow \psi)$:

Kopf ψ (Folgerung)

Rumpf φ (Voraussetzung)
oft $\varphi = (b_1 \wedge \ldots \wedge b_n \wedge \neg c_1 \wedge \ldots \wedge \neg c_m)$ mit
positiven Voraussetzungen $b_1, \ldots, b_n$
und negativen Voraussetzungen $c_1, \ldots, c_m$

mögliche Interpretation des Regelkopfes ψ als

Aussage , z.B. Kälte $\wedge$ Niederschlag $\rightarrow$ Schnee
Deduktionsregeln

Aktion , z.B. heiße Stirn $\rightarrow$ Fieber messen
Aktionsregeln, Produktionsregeln

Konflike bei Aktionsregeln

gegeben: Faktenmenge F, Regelmenge R (z.B. definit)

Regel $b_1 \wedge \cdots \wedge b_n \rightarrow h$ in F genau dann anwendbar, wenn ihr Rumpf $b_1 \wedge \cdots \wedge b_n$ in F erfüllt ist (also wenn $\{b_1, \ldots, b_i\} \in F$)

Problem:

- Faktenmenge (Zustand) F' mit mehreren anwendbaren Regeln aus R, die gegenteilige Aktionen auslösen
- Aktionsteil welcher Regel soll ausgeführt werden?

Konfliktmenge zu einer Faktenmenge F':
Menge aller in F' anwendbaren Regeln aus R

Beispiel

Wissensbasis:

R1 Wenn Besuch zum Essen kommt, gibt es Weiß- oder Rotwein.

R2 Zum Fisch gibt es Weißwein.

R3 Wenn Bob da ist, gibt es Rotwein.

F Bob kommt zum Fischessen.

Frage: Welches Getränk soll serviert werden?

Konfliktlösestrategien

Konfliktlösung durch Auswahl einer Teilmenge aller anwendbaren Regeln, durch deren Anwendung kein Konflikt entsteht.

oft durch (schrittweises) Entfernen ausgewählter Regeln

(einige) Konfliktlösestrategien:

Refraktionsprinzip: Keine Regelinstanz darf zweimal nacheinander feuern

Präferenzen: Bei der Wissensrepräsentation definierte Ordnung auf Regeln
Regeln niederer Präferenzen feuern nicht, wenn eine Konflikt-Regel höherer Präferenz angewendet wurde.

Spezifität: spezifischere Regelinstanzen werden bevorzugt

Aktualität: erstmals feuernde Regeln werden bevorzugt

zufällige Auswahl einer Regel

Wissensrepräsentation durch aussagenlogische Regeln

Wissensbasis: Menge aussagenlogischer Regeln
verschiedene Repräsentationsformen möglich, z.B.

- Menge von Regeln
- Formelmenge $\subseteq \mathrm{AL}(P)$
- Entscheidungstabelle
- Entscheidungsbaum
- Entscheidungsdiagramm

Anfrage (Fall): Menge von Fakten als

- Formel
- Formelmenge

Lösung: ja / nein , evtl. mit Modell

Problemlöseverfahren: abhängig von der Art der Regeldarstellung

Beispiel

Einlösen eines Schecks unter den folgenden Bedingungen:

- R1 Wenn die vereinbarte Kreditgrenze des Ausstellers eines Schecks überschritten wird, das bisherige Zahlungsverhalten einwandfrei war und der Überschreitungsbetrag kleiner oder gleich 250 € ist, dann wird der Scheck eingelöst.
- R2 Wenn die Kreditgrenze überschritten wird, das bisherige Zahlungsverhalten einwandfrei war, aber der Überschreitungsbetrag über 250 € liegt, dann wird der Scheck eingelöst und dem Kunden werden neue Konditionen vorgelegt.
- R3 Der Scheck wird nicht eingelöst, wenn die Kreditgrenze überschritten wird und das Zahlungsverhalten nicht einwandfrei war.
- R4 Der Scheck wird eingelöst, wenn die Kreditgrenze nicht überschritten wird.

Entscheidungstabellen

kompakte Darstellung einer endlichen Menge von Regeln

Aufbau einer Entscheidungstabelle:

- Bedingungsteil
  je Bedingung eine Zeile, je Regel eine Spalte
- Folgerungs- bzw. Aktionsteil
  je Folgerung bzw. Aktion eine Zeile
  in jeder Spalte alle Folgerungen / Aktionen der entsprechenden Regel markiert.

Beispiel Scheckeinlösen (Tafel)

Wissensrepräsentation und -verarbeitung durch Entscheidungstabellen

Wissensbasis (Kontextwissen):
Entscheidungstabelle (repräsentiert Menge von Regeln)

Problem (konkreter Fall):
Menge von Fakten (Aussagen, Merkmalswerten)
Menge möglicher Aussagen (oder Aktionen)

Lösung: zum konkreten Fall passende Aussagen (oder Aktionen)

Wissensverarbeitung (Problemlösung):
Suche der gegebenen Aussagenkombination im Bedingungsteil der Entscheidungstabelle,
Ablesen der Aussagen (oder Aktionen) im Folgerungsteil

Eigenschaften von Entscheidungstabellen

Entscheidung zwischen mehreren Alternativen
im Beispiel: einlösen, nicht einlösen

- vollständig:
  je Kombination von Bedingungen eine Spalte
  (Wahrheitswerttabelle)
- konsolidiert (kompakte Darstellung):
  Zusammenfassung von Spalten mit denselben Aktionen
  (Wildcards in Bedingungen)

Übersetzung: Regelmenge $\leftrightarrow$ Entscheidungstabelle

- Regelmenge $\rightarrow$ Entscheidungstabelle
  (i.A. keine vollständige ET)
- Entscheidungstabelle $\rightarrow$ Regelmenge (einfach):
  jede Spalte repräsentiert eine oder mehrere Regeln
  (nach Anzahl der markierten Folgerungen, Aktionen)

Verknüpfung mehrerer Entscheidungstabellen

Menge $\{T_1, \ldots, T_n\}$ von Entscheidungstabellen

spezielle Aktion: Übergang zu einer Tabelle T_i

Ablaufsteuerung durch mehrere Entscheidungstabellen:

- Sequenz: Übergang zur Folge-ET unter allen Bedingungen
- Verzweigung: von Bedingungen abhängiger Übergang zu verschiedenen Folge-ET
- Rekursion: Übergang zur selben ET

Regeln mit Alternative

Regeln der Form: Falls A, dann B sonst C

als logische Formel:
$(A \to B) \wedge (\neg A \to C) \equiv (A \wedge B) \vee (\neg A \wedge C) = \text{ite}(A, B, C)$
mit dreistelligem Junktor ite

$\{\text{ite}, \mathbf{t}, \mathbf{f}\}$ ist eine Junktorbasis.

Zu jeder aussagenlogischen Formel $\varphi \in \text{AL}(P)$ existiert sogar eine äquivalente Formel ψ, die nur die Junktoren ite, $\mathbf{t}, \mathbf{f}$ enthält und jede Teilformeln mit Wurzel ite die Form $\text{ite}(p, \varphi_1, \varphi_2)$ mit $p \in P$ hat (ite-Formel).

Regeln mit Alternative

(Induktive) Definition der Menge aller ite-Formeln

IA: $\mathbf{t}$ und $\mathbf{f}$ sind ite-Formeln.

IS: Sind $p \in P$ eine Aussagenvariable und φ und ψ ite-Formeln, dann ist auch $\text{ite}(p, \varphi, \psi)$ eine ite-Formel

Beispiele (Tafel):

- $\neg p \equiv \text{ite}(p, \mathbf{f}, \mathbf{t})$
- $p \wedge q \equiv \text{ite}(p, \text{ite}(q, \mathbf{t}, \mathbf{f}), \mathbf{f})$
- $b \wedge (a \vee c) \equiv \text{ite}(a, \text{ite}(b, \mathbf{t}, \mathbf{f}), \text{ite}(c, \text{ite}(b, \mathbf{t}, \mathbf{f}), \mathbf{f}))$

Binäre Entscheidungsbäume

Binärer Entscheidungsbaum:
Graph $T = (V, E)$ mit den folgenden Eigenschaften:

- Baum T hat eine Wurzel $r \in V$,
- jeder innere Knoten hat Grad 2,
  je einen mit 0 und einen mit 1 markierten Nachfolger
- jedes Blatt hat Grad 0
- jeder Knoten hat genau einen Vorgänger

und Knotenmarkierungen

- innere Knoten: Variablen (Entscheidungskriterien, Merkmale)
- Blätter: Werte oder Aktionen

Beispiel: $b \wedge (a \vee c)$

Jeder binäre Entscheidungsbaum repräsentiert eine Boolesche Funktion.

Wissensrepräsentation und -verarbeitung durch Entscheidungsbäume

Wissensbasis (Kontextwissen):
Entscheidungsbaum (repräsentiert Menge von Regeln)

Problem (konkreter Fall):
Menge von Fakten (Aussagen, Merkmalswerten)
Menge möglicher Aussagen (oder Aktionen)

Lösung: zum konkreten Fall passende Aussagen (oder Aktionen)

Wissensverarbeitung (Problemlösung):
Beginn in der Wurzel,
schrittweises Verfolgen der Kanten im Entscheidungsbaum entsprechend der Merkmalswerte in jedem Knoten
Ablesen der Aussage (oder Aktion) im erreichten Blatt

Übersetzung: Regelmenge $\leftrightarrow$ Entscheidungsbaum

- Entscheidungsbaum $\rightarrow$ Regelmenge (einfach):
  Jeder Pfad von der Wurzel zu einem Blatt repräsentiert eine Regel.
  Innere Knoten und die dazu ausgewählten Kanten bestimmen die Voraussetzungen, das Blatt die Folgerung.
- Regelmenge $\rightarrow$ Entscheidungsbaum (weniger einfach), z.B. über ite-Formel oder Entscheidungstabelle

Allgemeine Entscheidungsbäume

Verallgemeinerung binärer Entscheidungsbäume:

- Knotenmarkierung: Variable mit mehreren möglichen Werten,
- innere Knoten mit beliebig vielen Nachfolgern
  (mehrere Alternativen möglich)

Allgemeiner Entscheidungsbaum:
Baum (gerichteter azyklischer Graph) mit den Eigenschaften

- eine Wurzel,
- jeder Knoten hat genau einen Vorgänger,
- Blätter, Knotenmarkierung: Klassen
- innere Knoten mit Knotenmarkierung: Merkmal,
  (Frage nach Merkmalswert)
  ausgehende Kanten markiert mit möglichen Werten oder
  Wertebereichen dieses Merkmals (Antworten)
  Jeder innere Knoten hat so viele Nachfolger wie mögliche
  Alternativen (Werte oder Wertebereiche)