Was bisher geschah

Wissensrepräsentation und -verarbeitung in

- Zustandsübergangssystemen
- klassischer Aussagenlogik
- Prädikatenlogik und Fragmente,
  z.B. Hornlogik (Prolog)
- Logische Programmen
  Beispiele zum Planen
- Regelsystemen:
  - Wissensrepräsentation: Mengen von Regeln
    z.B. Prolog, Datalog
  - Wissensverarbeitung:
    - Rückwärtsverkettung (Ziel-orientiert)
      z.B. prädikatenlogische Resolution (Prolog)
    - Vorwärtsverkettung (Daten-orientiert)
      z.B. Erweiterungen von Faktenmengen (Datalog)
- Entscheidungstabellen
- Entscheidungsbäume

Beispiel Erreichbarkeit (ÜA)

Wissensbasis:

- Regelmenge:
  - R1 Feldwege sind befahrbar.
  - R2 Landstraßen sind befahrbar.
  - R3 Flüsse sind in Flussrichtung befahrbar.
- Faktenmenge:
  - F1 Feldwege gibt es zwischen A und C und zwischen B und D.
  - F2 Landstraßen gibt es zwischen C und D und zwischen B und E.
  - F3 Flüsse fließen von A nach B und von E nach D.

Frage: Ist D von A erreichbar?

neue Informationen:

- R1' (statt R1): Feldwege sind nur befahrbar, wenn es nicht regnet.
- F4 Es regnet.

Regeln (Wiederholung)

Formen von Regeln:

allgemeine Implikation: $\varphi \rightarrow \psi$
mit beliebigen (aussagen- oder prädikatenlogischen) Formeln

definite Regel: $l_1 \wedge \ldots \wedge l_n \rightarrow h$
mit Atomen l_i, h (Horn-Klausel)

normale Regel: $l_1 \wedge \ldots \wedge l_n \rightarrow h$
mit Literalen l_i und Atom h

allgemeine Regel: $l_1 \wedge \ldots \wedge l_n \rightarrow h$
mit Literalen l_i, h

Allgemeine (normale, definite) logische Programme (Regelmengen) sind endliche Mengen von allgemeinen (normalen, definiten) Regeln.

Wiederholung Konsequenzoperator

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

$$T_P(F) = \{h \mid b \to h \in R \text{ und } F \models b\}$$

für definite Programme (nur positive Bedingungen):

$$T_P(F) = \{h \mid b_1 \wedge \cdots \wedge b_n \to h \in R \text{ und } \{b_1, \ldots, b_n\} \subseteq F\}$$

definiert die Mengen

$$\begin{aligned} T_P^0(\emptyset) &= F_0 = F \\ T_P^{n+1}(\emptyset) &= F_{n+1} = T_P\left(T_P^n(\emptyset)\right) \\ &\vdots \\ T_P^*(\emptyset) &= F_* = \bigcup_{i \in \mathbb{N}} T_P^i(\emptyset) \end{aligned}$$

Fixpunkt-Semantik (definiter) logischer Programme

$T_P^*(\emptyset)$ ist der kleinste Fixpunkt des Operators T_P.

Für definite Programme P:

- gilt $T_P^*(\emptyset) = \bigcap \mathrm{Mod}(P)$
- ist $T_P^*(\emptyset)$ das eindeutige kleinste Modell für P.
- Falls $T_P^n(\emptyset) = T_P(T_P^n(\emptyset))$ gilt, dann ist $T_P^n(\emptyset) = T_P^*(\emptyset)$.
- Für endliche (grundinstanziierte) Programme P wird $T_P^*(\emptyset)$ nach endlich vielen Anwendungen von T_P erreicht.

Folgern / Schließen aus

- Ein Atom a folgt genau dann aus P ($P \models a$), wenn $a \in T_P^*(\emptyset)$.
- Ein negiertes Atom $\neg a$ folgt genau dann aus P ($P \models \neg a$), wenn $a \notin T_P^*(\emptyset)$.
- Vorsicht bei Fortsetzung auf Formeln, erfüllt übliche Semantik der Junktoren nicht immer

Nichtmonotones Schließen

Syntax der Wissensbasis:
endliche Menge von Regeln mit negierten Atomen im Rumpf (und Kopf)

Problem beim Schließen mit Regeln mit negativen Bedingungen:

- Als falsch angenommene Voraussetzungen können sich später im Laufe der Fixpunkt-Berechnung als wahr herausstellen.
- Voraussetzungen früher angewendeter Regeln gelten damit evtl. nicht mehr.

Konsequenzoperator T_P für nicht definite Programme ist i.A. nicht monoton (d.h. $F \subseteq F' \not\Rightarrow T_P(F) \subseteq T_P(F')$).

Beispiele:

- $P = \{\neg p \to q, \neg q \to p\}$
- $P' = P \cup \{p\}$

Beispiel

Wissensbasis:

F1 mann(Paul).

F2 mann(Otto).

F3 verheiratet(Paul).

R junggeselle(X) : – mann(X), ¬ verheiratet(X)

- Otto-Instanz der Regel R
  mann(Otto) ∧¬ verheiratet(Otto)
  feuert in der Faktenbasis $\{F1, F2, F3\}$
- Paul-Instanz der Regel R
  mann(Paul) ∧¬ verheiratet(Paul)
  feuert in der Faktenbasis $\{F1, F2, F3\}$ wegen F3 nicht

Negative Voraussetzungen

Wann gilt $\neg p$ in einer Faktenbasis F?

verschiedene Ansätze:

1. starke Negation: $\neg p$ gilt genau dann, wenn $(\neg p) \in F$
   Vorteil: positive Antwort immer korrekt
   Probleme:
   - erfordert Verwaltung negativer Fakten in Faktenbasis F
   - Was gilt, falls weder p noch $\neg p$ in F? (Unbestimmtheit)
   - Was gilt, falls sowohl p als auch $\neg p$ in F? (Inkonsistenz)
2. schwache Negation:
   Nicht aus der Wissensbasis ableitbare Aussagen werden als falsch angenommen. (Freispruch aus Mangel an Beweisen)
   Vorteil: ergibt immer eine Antwort (zweiwertig)
   Problem: nach Erweiterung der Wissensbasis evtl. nicht mehr gültig
3. Nutzer fragen (falls möglich)
   Vorteil: Antwort führt zu Erweiterung des Wissens
   keine Antwort vom Nutzer $\rightarrow$ Fall 1 oder 2

Closed World Assumption

CWA: Der Anwendungsbereich ist durch die Wissensbasis vollständig beschrieben.

Damit gilt insbesondere

- Jede im Anwendungsbereich gültige Aussage ist aus der Wissensbasis ableitbar.
- Jede nicht aus der Wissensbasis ableitbare Aussage gilt im Anwendungsbereich nicht.
  (also gilt ihre Negation)

entspricht der Idee der schwachen Negation

Fixpunkt-Semantik logischer Programme mit Negation

WH: Für definite Programme P folgt
die Formel φ ($P \models \varphi$) genau dann aus P, wenn $T_P^*(\emptyset) \models \varphi$.
Problem: Für normal logische Programme P hat T_P i.A.
keinen kleinsten Fixpunkt, sondern eine (evtl. leere) Menge
minimaler Fixpunkte (intuitive Modelle).
Beispiel: $P = \{\neg p \to q, \neg q \to p\}$

Zwei prominente intuitive Folgerungsbegriffe aus Menge $M(P)$
mehrerer intuitiver Modelle von P: Formel φ folgt aus P

leichtgläubig (credulous, $P \models_c \varphi$)
gdw. $\exists M \in M(P) : M \models \varphi$
mit P aus Beispiel oben:
$P \models_c p$, $P \models_c q$, $P \models_c p \vee q$, aber nicht $P \models_c p \wedge q$

skeptisch ($P \models_s \varphi$)
gdw. $\forall M \in M(P) : M \models \varphi$
mit P aus Beispiel oben:
$P \not\models_s p$, $P \not\models_s q$, aber $P \models_s p \vee q$

Regeln mit Negation im Rumpf

Regeln der Form

$$p_1 \wedge \cdots \wedge p_{n_i} \wedge \neg q_1 \wedge \cdots \wedge \neg q_{m_i} \rightarrow h$$

mit

- positiven Bedingungen $p_1, \ldots, p_{n_i}$
- negativen Bedingungen $q_1, \ldots, q_{m_i}$

ist in der Faktenmenge F genau dann anwendbar, wenn

$$F \models (p_1 \wedge \cdots \wedge p_{n_i} \wedge \neg q_1 \wedge \cdots \wedge \neg q_{m_i})$$

also

- $\{p_1, \ldots, p_{n_i}\} \subseteq F$ und
- $\{q_1, \ldots, q_{m_i}\} \cap F = \emptyset$

Vorwärtsverkettung auch möglich für Wissensbasen mit Regeln mit (schwacher) Negation

Normal logische Programme

(negative Voraussetzungen erlaubt)

normal logisches Programm P (Wissensbasis) enthält:

- Menge R von Regeln der Form

$$p_1 \wedge \cdots \wedge p_{n_i} \wedge \neg q_1 \wedge \cdots \wedge \neg q_{m_i} \rightarrow h$$

  mit Atomen p_i, q_i, h

spezielle Regeln:

- Regeln mit leerem Rumpf: h (Fakten)
- Regeln mit leerem Kopf: $p_1 \wedge \cdots \wedge p_{n_i} \wedge \neg q_1 \wedge \cdots \wedge \neg q_{m_i}$ (Constraints)

Beispiel: $\{p \rightarrow q, \neg q \rightarrow r, \neg r \rightarrow q, p\}$

Modelle normal logischer Programme

für normal logisch Programme P:

Herbrand-Universum U_P von P:
Menge aller Grundatome über der Signatur von P, betrachtet als Aussagevariablen (Grundinstanziierung)

Herbrand-Interpretation I für P:
Menge von Grundatomen $I \subseteq U_P$ aus dem Herbrand-Universum von P
Belegung der Aussagevariablen ist charakteristische Funktion von I

Herbrand-Modell für P:
Herbrand-Interpretation $I \subseteq U_P$ mit $I \in \text{Mod}(P)$
(Belegung = charakteristische Funktion)

Beispiel: $P = \{p \to q, \neg q \to r, \neg r \to q, p\}$

- Herbrand-Universum $U_P = \{p, q, r\}$
- (eine mögliche) Herbrand-Interpretation: $I = \{p, r\}$
- $\{p, q, r\}, \{p, q\}$ sind Herbrand-Modelle für P
- $\{q, r\}, \emptyset$ sind keine Herbrand-Modelle für P

Auswahl intuitiver Modelle

für definite logische Programme:

- existiert ein (eindeutiges) kleinstes Modell $\min \mathrm{Mod}(P)$ für P
- gilt: $\min \mathrm{Mod}(P) = \bigcap \mathrm{Mod}(P)$
  $= T_P^*(\emptyset) =$ kleinster Fixpunkt von T_P

Beispiele: $P = \{p \to q\}$, $P' = \{p \to q, p\}$

$\min \mathrm{Mod}(P)$ ist das eindeutige intuitive Modell

intuitiver Folgerungsbegriff:
Aussage a folgt aus P gdw. $a \in \min \mathrm{Mod}(P)$

für normal logische Programme:

- existiert i.A. kein kleinstes Modell für P
- hat T_P i.A. keinen kleinsten Fixpunkt
- sind die Fixpunkte von T_P gute Kandidaten für intuitive Modelle

Beispiel: $P = \{\neg p \to q\}$
intuitiver Folgerungsbegriff $P \models \varphi$ bei mehreren Modellen ?

Eigenschaften intuitiver Modelle

Eigenschaften von Interpretationen I des logischen Programmes P:

abgeschlossen unter P:
für jede Regelinstanz $B \rightarrow h$ aus P gilt:
falls $I \models B$, dann $h \in I$

begründet: für jedes $p \in I$ existiert eine Ableitung (Begründung) für p in I

Eigenschaften von Modellen I eines logischen Programmes P:

minimal: falls $J \subseteq I$ und $J \in \mathrm{Mod}(P)$, dann gilt $J = I$

Intuitive Modelle: (minimale) Modelle für P, die begründet und unter P abgeschlossen sind.

Gelfond-Lifschitz-Transformation

gegeben:

- normal logisches Programm P
- Herbrand-Interpretation (Modell) I für P

I-Redukt von P:

$$P^I = \left\{ p_1 \wedge \ldots \wedge p_m \rightarrow h \mid \begin{array}{l} p_1 \wedge \ldots \wedge p_m \wedge \neg q_1 \wedge \ldots \wedge \neg q_n \rightarrow h \in R \\ \text{und} \{q_1, \ldots, q_n\} \cap I = \emptyset \end{array} \right\}$$

Beispiel: $P = \{p \rightarrow q, \neg q \rightarrow r, \neg r \rightarrow q, p\}$, $I = \{p, r\}$
Programmtransformation von P abhängig von I:

1. Alle Regeln mit negativen Bedingungen $\neg q_i$ mit $q_i \in I$ aus P entfernen.
2. Alle negativen Bedingungen aus allen in P verbliebenen Regeln entfernen.

Für jedes normale logische Programm P und jede Interpretation I ist das I-Redukt P^I ein definites Programm.

Der Konsequenzoperator T_{P_I} von P_I ist also monoton.

Stabile Modelle normaler logischer Programme

Bestimmung der intuitiven Modelle für ein normal logisches Programm P: Auswahl einer Teilmenge von $\mathrm{Mod}(P)$
Modell I für P heißt stabiles Modell für P, falls

$$T^*_{(P^I)}(\emptyset) = I$$

Beispiele:

- $P_1 = \{p \to q, \neg q \to r, \neg r \to q, p\}$
  - $I_1 = \{p, q\}$ ist stabiles Modell für P_1, weil
    ${P_1}^{\{p,q\}} = \{p \to q, q, p\}$ und $T^*_{\left(P_1^{\{p,q\}}\right)}(\emptyset) = \{p, q\} = I_1$
  - $I_2 = \{p, q, r\}$ ist kein stabiles Modell für P_1, weil
    ${P_1}^{\{p,q,r\}} = \{p \to q, p\}$ und $T^*_{\left(P_1^{\{p,q,r\}}\right)}(\emptyset) = \{p, q\} \neq I_2$
- $P_2 = \{\neg p \to q, \neg q \to p\}$
- $P_3 = \{\neg p \to q, p \to q, \neg q \to p\}$
- $P_4 = \{\neg p \to p\}$

Beispiel: gefärbte Graphen

Wissensbasis (Beschreibung des Problems):

- Knotenmenge $V = \{v_1, \ldots, v_n\}$
  `ecke(v1), ..., ecke(vn)`
- Kantenmenge $E = \{(v_i, v_j), \ldots\}$
  `kante(vi, vj), ...`
- Menge $C = \{r, g, b\}$ von Farben

Erzeugung der Kandidaten (jede Ecke genau eine Farbe):

```
farbe(X,r) :- ecke(X), not farbe(X,b), not farbe(X, g).
farbe(X,b) :- ecke(X), not farbe(X,r), not farbe(X, g).
farbe(X,g) :- ecke(X), not farbe(X,r), not farbe(X, b).
```

Bedingung für korrekte Färbung (Ausschlusskriterium):

```
:-  kante(X,Y), farbe(X,Z), farbe(Y,Z).
```

Stabile Modelle repräsentieren Lösungen (korrekte Färbungen)

Unvollständiges Wissen

kommt praktisch überall vor

einige mögliche Quellen der Unvollständigkeit:

- Aussagen mit unbekanntem Wahrheitswert
- Unvollständige Beschreibung der Situation
- Abstraktion von unwichtig erscheinenden Details
- Falsche Wahrnehmung
- Kein sicheres Wissen über zukünftige Aussagen
- natürlichsprachliche ungenaue Formulierungen

Schließen und Treffen sinnvoller Entscheidungen oft trotzdem möglich.

Beispiel

Wissensbasis (Problembereich): Baustein-Welt:

- Turm aus drei Bausteinen
  (von oben nach unten: A,B,C)
- A ist grün.
- C ist nicht grün.

Formeln ...

Problem Steht ein grüner Baustein direkt auf einem nicht-grünen?

Lösung ...

Inkonsistentes Wissen

widersprüchliche Aussagen bzw. Aktionen

fast immer unvermeidbar, (einige) mögliche Ursachen:

- ungenaue natürlichsprachliche Formulierungen
- Ausnahmen, Spezialfälle
- ungenaue Modellierung
- Nichtdeterminismus
- Wissen aus mehreren verschiedenen Quellen

Beispiel

Wissensbasis:

- Regelmenge:
  - Vögel können fliegen.
  - Pinguine sind Vögel.
  - Pinguine können nicht fliegen.
- Faktenmenge:
  Tweety ist ein Pinguin.

Problem: Kann Tweety fliegen?

Logische Programme mit negierten Folgerungen

Ziel:

- Darstellung von Regeln mit negierten Folgerungen
- Kombination starker und schwacher Negation

Syntax: erweiterte logische Programme:
im Regelrumpf sowohl starke $^-$ (`~ x`) als auch schwache Negation $\neg$ (`not x`)
im Regelkopf nur starke $^-$ Negation

Semantik: Faktenmengen
(Mengen von Atomen und stark negierten Atomen)

Problem:

- falls ein Atom a mit $\{a, \overline{a}\} \subseteq F$ existiert, ist die Faktenmenge F inkonsistent
(Semantik nicht definiert)
- bei jeder Erweiterung der Faktenmenge ist deren Konsistenz zu garantieren

Erweiterte logische Programme

Idee: p und $\overline{p}$ als unabhängige Atome betrachten
Konsistenz durch Constraints garantieren (z.B. $p \wedge \overline{p} \rightarrow \mathbf{f}$)

(erweitertes) logisches Programm P (Wissensbasis) enthält:

- Menge R von Regeln der Form

$$p_1 \wedge \cdots \wedge p_{n_i} \wedge \neg q_1 \wedge \cdots \wedge \neg q_{m_i} \rightarrow h$$

  mit „Atomen“ p_i, q_i, h

spezielle Regeln:

- Regeln mit leerem Rumpf: h (Fakten)
- Regeln mit leerem Kopf: $p_1 \wedge \cdots \wedge p_{n_i} \wedge \neg q_1 \wedge \cdots \wedge \neg q_{m_i}$ (Constraints)

Beispiel: $\{p \rightarrow \overline{q}, \neg\overline{q} \rightarrow r, \neg r \rightarrow \overline{q}, \overline{p}\}$

Semantik: Answer-Sets

Answer-Sets:
ausgewählte Modelle erweiterter logischer Programme

Interpretation eines erweiterten logischen Programmes P:
Menge F von „Grundatomen“
(Grundatom oder stark negiertes Grundatom mit derselben Signatur wie P)

Interpretation F ist Answer-Set für P gdw.

abgeschlossen unter P:
für jede Regelinstanz $B \rightarrow h$ aus P gilt:
falls $F \models B$, dann $h \in F$
und konsistent, d.h.
für kein Atom $p \in F$ gilt $\{p, \overline{p}\} \subseteq F$

begründet: für jedes „Atom“ $\in F$ existiert eine Ableitung (Begründung) für in F

Wissensverarbeitung mit Answer-Sets

Bestimmung von Answer-Sets:
Interpretation F ist Answer-Set für P gdw.
F Modell des F-Reduktes P^F (analog stabilen Modellen)
Beispiel: $P = \{\neg c \to a, \neg b \to c, \neg b \to \overline{d}, a \wedge \neg \overline{b} \to b\}$
$F = \{a, b\}$, $F' = \{c, \overline{d}\}$
ASP-Solver-Notation:

```
a :- not c.
c :- not b.
~d :- not b.
b :- a, not ~b.
```

Wahrheitswert eines Atoms a in Answer-Set F (dreiwertige Logik):

wahr	gdw. $a \in F$
falsch	gdw. $\overline{a} \in F$
undefiniert	gdw. weder $a \in F$ noch $\overline{a} \in F$

Beispiel: in $F' = \{c, \overline{d}\}$ sind a und b undefiniert, c wahr und d falsch

Beispiel: Modellierung von Ausnahmen

Wissensbasis:

- Regelmenge:
  - Vögel können fliegen.
  - Pinguine sind Vögel.
  - Pinguine können nicht fliegen.
- Faktenmenge:
  Tweety ist ein Pinguin.

```
f :- v, not ~f.
v :- p.
~f :- p.
p.
```

hat Answer-Set $\{p, b, \overline{f}\}$

Anwendungsbeispiel: Terminplanung

Faktenbasis (Beschreibung des speziellen Problemes):

```
termin(m1), . . . , termin(mn)
zeit(t1), . . . , zeit(ts), raum(r1), . . . , raum(rm)
person(p1), . . . , person(pk)
mit(p1,m1), . . . , mit(p2,m3), . . .
```

Zuordnung von Zeiten und Räumen zu Terminen:

```
um(M, T)  :- termin(M), zeit(T), not ~um(M, T).
~um(M, T) :-  termin(M), zeit(T), not um(M, T).
in(M,R)   :- termin(M), raum(R), not ~in(M,R).
~in(M,R)  :- termin(M), raum(R), not in(M,R).
zeitvergeben(M) :-  um(M, T).
raumvergeben(M) :-  in(M,R).
```

Bedingungen:

```
:-  termin(M), not zeitvergeben(M).
:-  termin(M), not raumvergeben(M).
:-  termin(M), um(M, T), um(M, T'), T <> T'.
:-  termin(M), in(M,R), in(M,R'), R <> R'.
:-  in(M,X), in(M',X), um(M, T), um(M', T),M <> M'.
:-  mit(P,M), mit(P,M'), M <> M', um(M, T), um(M', T).
```