Expertensysteme

Wissensbasis ⟷ [Inferenz-maschine ⟷ Benutzer-schnittstelle] ⟷ Benutzer

Shell

Probleme lösen, Lösungen erklären, Unsicherheiten behandeln

Wenn-Dann-Regeln zur Wissensrepräsentation

Bedingte Anweisungen unterschiedlich interpretiert:

Wenn Vorbedingung V,	dann Folgerung F
Wenn Situation S,	dann Aktion A
Wenn Bedingung B1 und B2 gelten,	dann gilt B nicht.

Eigenschaften:
modular, erweiterbar, modifzierbar, transparent

Knowledge-Engineering:
Extrahieren eines Verständnisses von Experten und transferieren in einen gewählten Formalismus zur Wissensrepräsentation.

a) kategorisches Wissen	Wenn ... dann ...
b) probabilistisches Wissen	Wenn ... dann ... mit Sicherheit F

Wissensbasis zur Identifikation von Tieren

(Auszug)

```
:- op(900, xfx, #).
:- op(100, xfx, [hat, gibt, 'tut nicht', frisst, legt, ist_ein]).
:- op(880, fx, wenn).
:- op(870, yfx, dann).
:- op(550, xfy, oder).
:- op(540, xfy, und).

regel1# wenn Tier hat haare oder Tier gibt milch
        dann Tier ist_ein saeugetier.
regel2# wenn Tier hat federn oder Tier fliegt und
                              Tier legt eier
        dann Tier ist_ein vogel.
...
fakt# X ist_ein tier :- member(X, [gepard, tiger, pinguin, albatros]).

fragbar(_ gibt _, 'Tier' gibt 'Was').
fragbar(_ fliegt, 'Tier' fliegt).
...
```

Entwicklung der Shell

Bsp.-Dialog:

Ihre Frage bitte:
tweety fliegt und tweety ist_ein X.

Ist folgendes wahr: tweety hat federn?
warum.

Um zu untersuchen mit Regel2, tweety ist_ein vogel

Ist folgendes wahr: tweety hat federn?
ja.
...
X = vogel ??
(tweety ist_ein vogel) ist wahr

Moechten Sie sehen, wie dies ermittelt wurde?

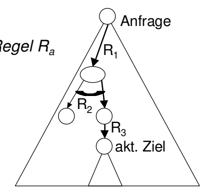
Skizze des Folgerungsprozess

Die Suche nach einer Antwort Antw auf eine Frage F:

- Ist F ein Fakt der Wissensbasis, dann ist Antw „F ist wahr".
- Gibt es in der Wissensbasis eine Regel der Form
  „Wenn Bedingung dann F"
  so erkunde Bedingung, um eine Antwort Anw zu finden.
- Ist F eine „fragbare" Frage, dann frage den Benutzer nach F.
- Ist F von der Gestalt „F1 und F2", dann untersuche F1. Ist F1 falsch, dann ist Antw „F ist falsch", ansonsten erkunde F2 und kombiniere die Antworten von beiden F1 und F2, geeignet zu Antw.
- Ist F von der Gestalt „F1 oder F2", dann untersuche F1. Wenn F1 wahr ist, dann ist Antw „F ist wahr", oder alternativ, untersuche F2 und kombiniere die Antworten von beiden F1 und F2 zu Antw.

Warum und Wie-Fragen

Ist a wahr?
warum
ich kann a benutzen zur Erforschung von b mit Regel R_a
ich kann b benutzen ...
z war ihre ursprüngliche Frage.



(z) ist wahr.

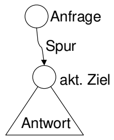
Moechten Sie sehen, wie dies ermittelt wurde?
ja
z
wurde abgeleitet mit Regel R_y von
y ...
...

Implementierung

erforsche(Ziel, Spur, Antwort)

benutzerantwort(Ziel, Spur, Antwort)

vorhanden(Antwort)



Ziel: Bsp. X hat federn oder X fliegt und X legt eier
Spur ist eine Liste von Elemente der Form: Ziel durch Regel
Bsp. [(tweety ist_ein vogel) durch Regel2, (tweety hat federn) durch ...]
Antwort ist ein Lösungsbaumvon der Form: Folgerung wurde Gefunden
Bsp. (tweety hat federn) ist wahr wurde mitgeteilt
oder (tweety ist_ein vogel) ist wahr wurde 'abgeleitet‘ durch Regel von (...)

Benutzerschnittstelle

benutzerantwort(Ziel, Spur, Antwort)

* Ziel fragbar, dann
  * zeige Ziel an und frage den Benutzer
    * Antwort: warum => zeige_spur und frage erneut
    * Antwort: ja => frage ggf. die Variablen in Ziel vom Benutzer ab

in der Wissensbasis:
fragbar(X frisst Y, ‘Tier‘ frisst ‘Irgendetwas‘).

Verbesserung von benutzerantwort

a) Frage nie, was bereits gefragt wurde
assert(wurde_gesagt(tweety legt eier, wahr, Index)).

b) Zeige dem Benutzer keine „kryptischen“ Variablen an
Bsp. fragbar(X frisst Y, ‘Tier‘ frisst ‘Irgendetwas‘).

?- benutzerfrage(X frisst Y, [], Antwort).

Ist folgendes wahr: Tier frisst Irgendetwas?

Haupttreiber experte

```
experte :-                                                        (Auszug)
        hole_frage(Frage),          %Eingabe der Benutzerfrage
        (antworte_ja(Frage)         %Suche positive Antwort
         ;antworte_nein(Frage)      %gibt es keine,suche negative A.
        ).

antworte_ja(Frage) :-                        %suche pos. Antwort auf Frage
        markiere_status(negativ),            %noch keine pos. Antwort erhalten
        erforsche(Frage, [], Antwort),
        positiv(Antwort),                    %suche positive Antworten
        markiere_status(positiv),            %positive Antwort gefunden
        vorhanden(Antwort), nl,
        write('Weitere Loesungen? '),
        hole_antwort(Erwiderung),
        Erwiderung = nein.                   %ja, dann backtracking nach
                                             %"erforsche"
```

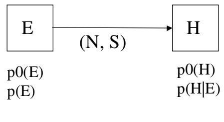

Anmerkung

* dem Shell-Programm mangelt es teilweise an der sonst typischen deklarativen Klarheit
  Ursache:
  prozedurale Ansteuerung der Benutzerschnittstelle

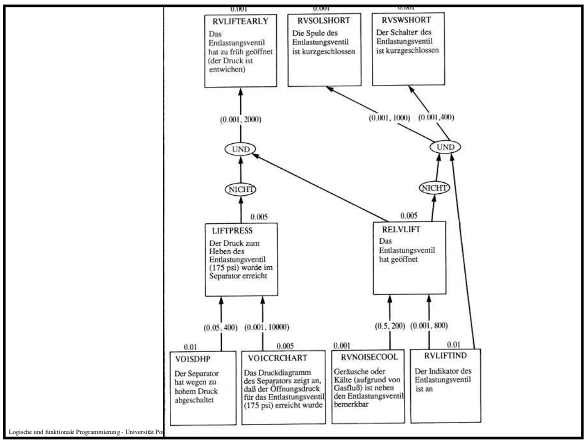
* negierte Ziele nicht(tiger frisst fleisch)
  Instantiierung des Arguments zwingend notwendig

* Wissensbasis kann Zyklen enthalten
  Bsp. regel1# wenn flasche_leer dann johannes_betrunken
  regel2# wenn johannes_betrunken dann flasche_leer

Exkurs: Umgang mit Unsicherheit



wenn E dann H mit Wahrscheinlichkeit W


0.001
RVLIFTEARLY
Das Entlastungsventil hat zu früh geöffnet (der Druck ist entwichen)
RVSOLSHORT
Die Spule des Entlastungsventil ist kurzgeschlossen
RVSWSHORT
Der Schalter des Entlastungsventil ist kurzgeschlossen
(0.001, 2000)
(0.001, 1000)
(0.001,400)
UND
UND
NICHT
NICHT
0.005
LIFTPRESS
Der Druck zum Heben des Entlastungsventil (175 psi) wurde im Separator erreicht
0.005
RELVLIFT
Das Entlastungsventil hat geöffnet
(0.05, 400)
(0.001, 10000)
(0.5, 200)
(0.001, 800)
0.01
VO1SDHP
Der Separator hat wegen zu hohem Druck abgeschaltet
0.005
VO1CCRCHART
Das Druckdiagramm des Separators zeigt an, daß der Öffnungsdruck für das Entlastungsventil (175 psi) erreicht wurde
0.001
RVNOISECOOL
Geräusche oder Kälte (aufgrund von Gasfluß) ist neben den Entlastungsventil bemerkbar
0.01
RVLIFTIND
Der Indikator des Entlastungsventil ist an
Logische und funktionale Programmierung - Universität Po