Logische Sprachen

Grundidee:

statt (N. Wirth):

Algorithmen + Datenstrukturen = Programm

jetzt (Kowalski):

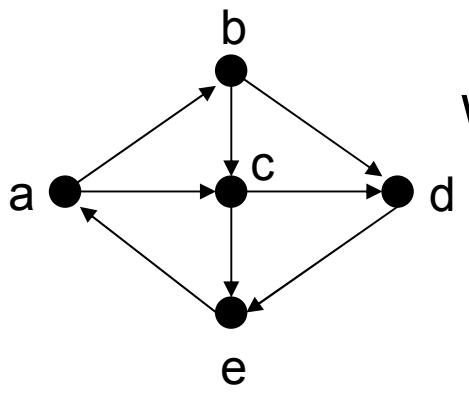
Algorithmus = Logik + Steuerung

- Wissensbasis aus Informationen
- beschrieben als Formeln eines logischen Kalküls
- Programmlauf: Anfrage des Benutzers (im Prädikatenkalkül)
- Antwort: ob oder welche Variablenbelegung logische Folgerung aus Wissensbasis ist
- zur Antwortermittlung: Resolutionsverfahren

Logische Sprachen

Beispiel:

kante(a,b).
kante(a,c).
kante(b,c).
kante(b,d).
kante(c,d).
kante(c,e).
kante(d,e).
kante(e,a).



Weggraph

„Programm“ zur <u>Wegsuche</u>:

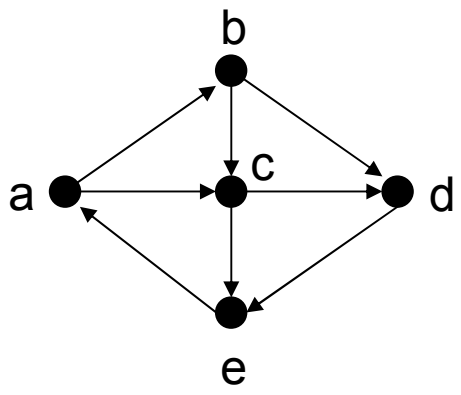
Für alle X führt ein Weg von X nach X.
Für alle X,Y und Z gilt: wenn eine Kante von X nach Z führt und ein Weg von Z nach Y, dann führt auch ein Weg von X nach Y.

In Symbolen:

weg(X,X).
weg(X,Y) $\leftarrow$ kante(X,Z) $\wedge$ weg(Z,Y).

Logische Sprachen

Anfragen:



Gibt es einen Weg von a nach e?

?- weg(a,e)

Gibt es einen Knoten X, der durch einen Weg von a aus erreichbar ist?

?- weg(a,X)

Gibt es einen Knoten Z, von dem aus Kanten nach d und nach e führen?

?- kante(Z,d) $\wedge$ kante(Z,e).

Logische Sprachen

Wissensbasis:

Grundlegende Elemente: Klauseln

- Fakten
- Regeln

Fakten:

- Aussagen über Objekte
- besteht aus Funktor und (1-n) Argumenten
- Prolog-Syntax:
  - Funktor und Argumente beginnen mit Kleinbuchstaben,
  - bestehen aus Buchstaben, Ziffern und _
  - Argumente in ( ), mehrere: durch Komma getrennt
  - jede Faktendefinition endet mit Punkt

Logische Sprachen

Fakten:

```
kartoffelgericht(roesti).
kartoffelgericht(pommes_frites).
fleisch(schnitzel).
fleisch(frikadelle).


vater(zeus,ares).        /*Zeus ist Vater von Ares*/
vater(adam,abel).
mutter(hera,ares).
mutter(eva,kain).
weiblich(eva).
eltern(adam,eva,kain).  /*Elternteil, Elternteil, Kind*/
```

Logische Sprachen

Regeln:

- logische Aussage
- folgert aus bekannten Fakten ein neues Faktum
- besteht aus Regelkopf und Regelrumpf
- Rumpf: ein oder mehrere bekannte Klauseln, durch Komma/Semikolon getrennt
- Rumpf endet mit .
- Prolog-Syntax: :- verbindet Kopf und Rumpf
- Beispiel:

```
obst(apfel).
gemuese(tomate).
gesund(X):- obst(X); gemuese(X).
```

X ist Variable

Logische Sprachen

Variablen:

- Stellvertreter für bel. Objekte
- Prolog-Syntax: Bezeichner, beginnend mit Großbuchstaben

```
eltern(V,M,K) :-vater(V,K), mutter(M,K).
schwester(X,Y) :- weiblich(X), eltern(V,M,X),
                  eltern(V,M,Y).
grossvater(X,Z) :- vater(X,Y), vater(Y,Z).
grossvater(X,Z) :- vater(X,Y), mutter(Y,Z).
```

- Gültigkeitsbereich einer Variablen: Klausel, in der sie auftritt

Logische Sprachen

Anfragen an die Wissensbasis

- System soll Gültigkeit von ein oder mehreren Ausdrücken nachweisen
- Prolog-Prompt im Anfragemodus: ?-
- Anfrage: Ausdruck, mit Punkt abgeschlossen

- Bsp: ?- kartoffelgericht(pommes_frites).
  yes

- ist Ziel beweisbar: yes, sonst: no

Logische Sprachen

- Anfrage mit Variablen:

```
?- kartoffelgericht(X).
X=roesti
yes
```

- X logische Variable
- Ziel: kann X mit Wert instanziiert werden, so dass kartoffelgericht(X) beweisbar ist?
- Wenn ja: Ausgabe von erstem möglichen Wert
- Eingabe von **;** lässt System nach weiteren Lösungen suchen:

```
;
X=pommes_frites
;
no
```

Logische Sprachen

- Anfrage aus mehreren Ausdrücken:

```
?- kartoffelgericht(K), fleisch(F).
K=roesti, F=schnitzel

?- kartoffelgericht(K), fleisch(K).
no
```

- Variablen mit gleichem Namen: mit gleichem Wert instanziieren
- besondere Variable _ : anonyme Variable
- Bedeutung Anfrage mit _: „Gibt es überhaupt einen Wert, der Anfrage beantwortet?“
- → einzige Antworten: yes, no

```
?- kartoffelgericht(_).
yes
?- kartoffelgericht(_), fleisch(_).
yes
```

Logische Sprachen

- Arbeitsweise zur Lösungsfindung:
  - Unifikation
  - Backtracking

```
vater(zeus,ares).
vater(adam,abel).
mutter(hera,ares).
mutter(eva,kain).
```

- Unifikation:
  - suche mit Anfrage unifizierendes Faktum oder Regelkopf
  - unifiziert: Variablen lassen sich so ersetzen, dass beide Ausdrücke gleich sind
  - Bsp.: ?- vater (V,abel).
    - durchsuche Wissensbasis von vorne nach hinten
    - suche Klausel mit Funktor „vater“ und 2. Argument „abel“ oder 2. mit „abel“ unifizierbaren Argument
    - unifiziere Variable V mit 1. Argument
    - V=adam

Logische Sprachen

- Anfrage mit Regelkopf unifizierbar: versuche Subgoals im Rumpf zu beantworten
- Bsp.:

```
vorfahr(X,Y) :- mutter(X,Y).          %X ist Vorfahr von Y
vorfahr(X,Y) :- vater(X,Y).
vorfahr(X,Y) :- mutter(X,Z), vorfahr(Z,Y).
vorfahr(X,Y) :- vater(X,Z), vorfahr(Z,Y).
mutter(helga, peter).                 % (Mutter, Kind)
mutter(eva, sabine).
mutter(sabine, andrea).
mutter(andrea, oliver).
vater(karl, peter).                   % (Vater, Kind)
vater(karl, sabine).
vater(peter, doris).

?- vorfahr(eva, oliver)
```

Unifikation mit 1. Regel: mutter(eva,oliver) schlägt fehl, Backtracking
Unifikation mit 2. Regel: vater(eva,oliver) schlägt fehl, Backtracking
Unifikation mit 3. Regel: mutter(eva,Z),vorfahr(Z,oliver)

Logische Sprachen

weise Teilziele nach:

mutter(eva,Z) unifiziert mit mutter(eva, sabine),
weise vorfahr(sabine, oliver) nach
gelingt mit 3. Regel : mutter(sabine, Z), vorfahr(Z, oliver)
unifiziere mutter(sabine,Z) mit mutter(sabine, andrea)
weise vorfahr(andrea, oliver) nach
ist in Wissensbasis

kann für Teilziel keine Klausel gefunden werden (kein Kopf unifiziert mit Teilanfrage) : Suche nach Alternativen zu vorheriger Unifikation

→ Backtracking

Logische Sprachen

- Backtracking:
  - Choicepoint: letzte Stelle in Lösungsermittlung, wo Variablenbindung durchgeführt wurde
  - bei Fehlschlägen:
    - Rückkehr zum letzten Choicepoint
    - Aufheben der Variablenbindung ab diesem Point
    - Suche nach alternativer Klausel mit passendem Kopf
  - Bsp.:

    ?- vorfahr(karl, N)

    Alle Regeln passen: 1. Choicepoint, Wahl der 1. Regel
    → X=karl, Y=N
    Neues Teilziel: mutter(karl,N) schlägt fehl
    ⇒ Rückkehr zum Choicepoint, Freigabe der Variablen, Unifikation mit 2. Regel
    → X=karl, Y=N,
    neues Teilziel: vater(karl,N)
    Teilziel Faktum im Wissensbasis: 1. Lösung erreicht mit N=peter, 2. Choicepoint wird gesetzt

Logische Sprachen

- Backtracking:

  - Ist eine Lösung erreicht:
    Eingabe von **;** stößt Backtracking an,
    Suche nach weiteren Lösungen
  - Bsp.: Eingabe ;
    2. Choicepoint wird aufgehoben,
    Unifikation mit Alternative für Teilziel
    => vater(karl,N) unifizieren mit vater(karl,sabine)
    neuer Choicepoint wird gesetzt, Ausgabe N=sabine

    erneute Eingabe von ;
    letzter Choicepoint wird aufgehoben,
    keine weitere Alternative für vater(karl,N)
    => Zurücksetzen des vorletzten Choicepoints,
    neue Unifikation vorfahr(karl,N) mit
    mutter(karl,Z), vorfahr(Z,N)
    u.s.w.
    Welche Nachfahren erhält man insgesamt?