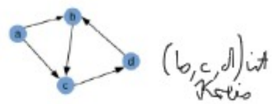
8.4 Gerichtete Graphen*

Schreiben Sie ein Prolog-Programm, das in einem gerichtetem Graphen Wege zwischen Knoten ausgibt. Bearbeiten Sie dafür die folgenden Teilaufgaben:

a) Ist der Graph in Abbildung 2 azyklisch (ohne Kreise)?

b) Definieren Sie in Prolog ein Prädikat `kante/2`, das alle Kanten des Graphs in Abbildung 3 als Fakten auflistet.

c) Definieren Sie rekursiv ein Prädikat `verbunden/2`, das gilt, wenn es einen Pfad zwischen den Eingabeknoten gibt. Welche Probleme stellen Sie fest, wenn Sie diese Wissensbasis auf 3 und auf 2 testen?



(b,c,d) ist Kreis

Abbildung 2: Ein gerichteter Graph.

azyklisch

Abbildung 3: Ein azyklischer, gerichteter Graph.

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

c) Zwei Knoten x und y sind verbunden, wenn es einen Weg (evtl. über mehrere Kanten) von x nach y gibt

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

?- verbunden(a,d)
- ?- kante(a,d) → †
- ?- kante(a,Z), verbunden(Z,d)
  - Z=b → verbunden(b,d)
    - ?- kante(b,d) → †
    - ?- kante(b,Q), verbunden(Q,d)
      - Q=c → verbunden(c,d)
        - kante(c,d) → □

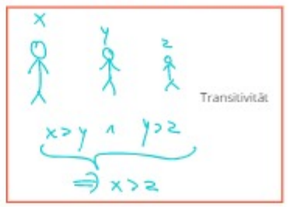
?- verbunden(d,a)
- ?- kante(d,a) → †
- ?- kante(d,Z), verbunden(Z,a)
  - Z=b → ?- verbunden(b,a)
    - ?- kante(b,a) → †
    - ?- kante(b,Q), verbunden(Q,a)
      - Q=c → ?- verbunden(c,a)
        - ?- kante(c,a) → †
        - ?- kante(c,R), verbunden(R,a)
          - R=d → verbunden(d,a)

Endlosschleife

Aufgabe 4.8 (Reicher_als). Seien Fakten über Personen gegeben, die ausdrücken, wer reicher ist:

```
reicher(adam,berta). % Adam ist reicher als Berta.
reicher(berta,clemens).     reicher(adam,erwin).
reicher(eva,adam).
```

Aus den Fakten folgt, dass Eva reicher als Clemens ist. PROLOG würde uns aber FALSE antworten. Entwickeln Sie Regeln für `reicher_als`, die sowohl die reicher-Fakten als auch die transitiven Beziehungen enthält. `reicher_als` soll also die transitive Hülle von `reicher` sein.



reicher_als(X,Y) :- reicher(X,Y).
reicher_als(X,Y) :- reicher(X,Z), ~~reicher(Z,Y)~~ reicher_als(Z,Y)

?- reicher_als(eva, clemens)

?- reicher(eva,clemens)
+

?- reicher(eva,Z), reicher_als(Z,clemens)
Z = adam

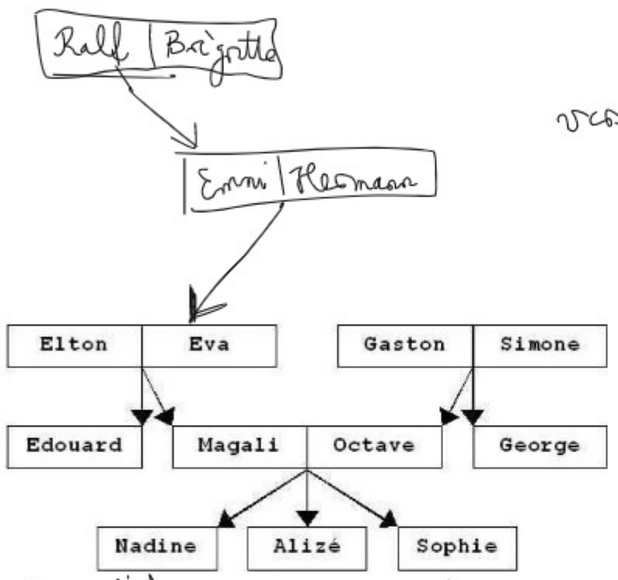
?- reicher_als(adam,clemens)

?- reicher(adam,clemens)
+

?- reicher(adam,Q), reicher_als(Q,clemens)
Q = berta

?- reicher_als(berta,clemens)

reicher(berta,clemens)



vorfahre(ernst, sophie)

X ist vorfahre von Y

vorfahre(X,Y) :- parent(X,Y).
vorfahre(X,Y) :- parent(X,Z), ~~parent(Z,Q), parent(Q,R), parent(R,Y)~~
vorfahre(Z,Y).

2 „kind"

1a)

Aufgabe) Benutze die Prädikate mann(X) und frau(X) und parent(X,Y), um die Wissensbasis zu beschreiben

a) enfant(X,Y)
b) fils(X,Y)
c) fille(X,Y)
d) pere(X,Y)
e) mere(X,Y)
f) grand_parent(X,Y)
g) grand_pere(X,Y)
h) grand_mere(X,Y)
i) petit_enfant(X,Y)
j) petit_fils(X,Y)
k) petite_fille(X,Y)

frau(eva).
mann(elton).
parent(elton, edouard).
parent(eva, edouard).

2a) kind(X,Y) :- parent(Y,X).
sohn(X,Y) :- kind(X,Y), mann(X).
tochter(X,Y) :- parent(Y,X), frau(X).
vater(X,Y) :- parent(X,Y), mann(X).
mutter(X,Y) :- parent(X,Y), frau(X).
grand_parent(X,Y) :- parent(X,Z), parent(Z,Y).
grand_father(X,Y) :- ~~parent(X,Z), parent(Z,Y), mann(X).~~
grand_parent(X,Y)
grand_mother(X,Y) :- parent(X,Z), parent(Z,Y), frau(X).
enkel_kind(X,Y) :- grand_parent(Y,X).
enkel_sohn(X,Y) :- enkel_kind(X,Y), mann(X).
enkel_tochter(X,Y) :- " , frau(X).

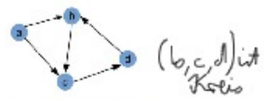
8.4 Gerichtete Graphen*

Schreiben Sie ein Prolog-Programm, das in einem gerichtetem Graphen Wege zwischen Knoten ausgibt. Bearbeiten Sie dafür die folgenden Teilaufgaben:

a) Ist der Graph in Abbildung 2 azyklisch (ohne Kreise)?

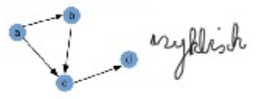
b) Definieren Sie in Prolog ein Prädikat `kante/2`, das alle Kanten des Graphs in Abbildung 3 als Fakten auflistet.

c) Definieren Sie rekursiv ein Prädikat `verbunden/2`, das gilt, wenn es einen Pfad zwischen den Eingabeknoten gibt. Welche Probleme stellen Sie fest, wenn Sie diese Wissensbasis auf 3 und auf 2 testen?



(b,c,d) ist Kreis

Abbildung 2: Ein gerichteter Graph.



azyklisch

Abbildung 3: Ein azyklischer, gerichteter Graph.

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

d) Aufgabe:) Definiere ein Prädikat "verbunden", sodass der Aufruf verbunden(d,a) zu keiner Endlosschleife mehr führt!

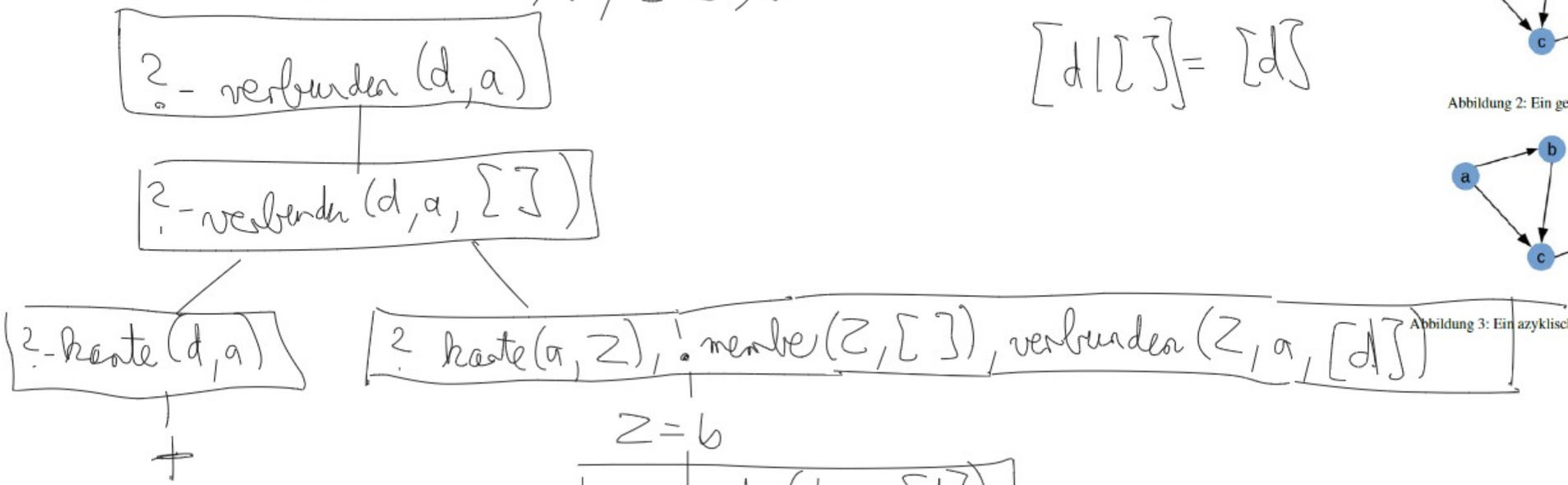
Liste von Zwischenknoten, die man nicht verwenden darf

verbunden(X,Y,L) :- kante(X,Y).

verbunden(X,Y,L) :- kante(X,Z), nicht member(Z,L), verbunden(Z,Y,[X|L]).

verbunden(X,Y) :- verbunden(X,Y,[]).

[d|[]] = [d]



verbunden(Z, Y, [X|L]).

verbunden(X,Y) :- verbunden(X, Y, [ ]).

[d|[ ]] = [d]

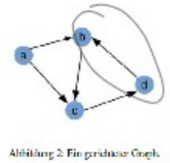


Abbildung 2: Ein gerichteter Graph.

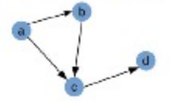


Abbildung 3: Ein zyklischer, gerichteter Graph.

?- verbunden(d,a)

?- verbunden(d,a,[ ])

?- kante(d,a) | ?- kante(a,Z), !, member(Z,[ ]), verbunden(Z,a,[d])

+

Z=b

verbunden(b,a,[d])

?- kante(b,a) | ?- kante(b,Q), !, member(Q,[d]), verbunden(Q,a,[b,d])

+

Q=c

verbunden(c,a,[b,d])

?- kante(c,a) | ?- kante(c,R), !, member(R,[b,d]), verbunden(R,a,[b,d,c])

+

R=d

verbunden(d,a,[b,d,c])

?- kante(d,a) | ?- kante(d,S), !, member(S,[b,d,c]), verbunden(S,a,[b,d,c,d])

+

S=b

!- member(b,[b,d,c]), verbunden(...)

+