Sammlung möglicher Klausuraufgaben

24. Januar 2017

1. Gegeben eine Wissensbasis zu einem Familienstammbaum:

```
% et/2
% et(X,Y): X ist ein Elternteil von
et(jutta, mia).
et(johann, mia).
et(jutta, otto).
et(mia, hans).
et(mia, jim).
et(mia, ana).
et(otto, tom).

% ehemann/2
% ehemann(X,Y): X ist Ehemann von Y.
ehemann(johann, jutta).
ehemann(otto, ria).
```

```
% fem/1
% fem(X). X ist weiblich.
fem(mia).
fem(ria).
fem(ana).
fem(jutta).

% mas/1
% mas(X). X ist maennlich.
mas(otto).
mas(johann).
mas(hans).
mas(jim).
mas(tom).
```

   Erweitern Sie die Wissensbasis um Prädikate für

```
mutter/2
sohn/2
schwester/2
grossvater/2
cousine/2
```

```
onkel/2
enkelin/2
schwager/2
ehepartner/2
```

   Hierbei gelten zwei Personen als Geschwister, sobald sie einen gemeinsamen Elternteil haben. Denken Sie daran, dass Onkel und Tante auch angeheiratet sein können.

2. Gegeben die folgende Wissensbasis:

```
directTrain(duesseldorf,koeln).
directTrain(koeln,paris).
directTrain(paris,berlin).
directTrain(duesseldorf,wuppertal).
directTrain(wuppertal,berlin).
directTrain(berlin,hannover).
directTrain(wuppertal,hannover).
```

Schreibe ein Prädikat `travelFromTo/2`, das zwei Orte als Argumente nimmt und wahr ist, wenn der zweite Ort vom ersten per Zug erreicht werden kann. Dabei ist es egal, ob man umsteigen muss oder nicht.

Erweiterung: Schreibe ein Prädikat `travelFromTo/3`, das zwei Orte und eine Zahl als Argumente nimmt und wahr ist, wenn der zweite Ort vom ersten per Zug erreicht werden kann. Die Zahl gibt an, wie oft bei der gewählten Strecke umgestiegen werden muss.

Bonusaufgabe: Schreibe ein Prädikat `travelFromToMin/3`, das zwei Orte und eine Zahl als Argumente nimmt und wahr ist, wenn der zweite Ort vom ersten per Zug erreicht werden kann. Die Zahl gibt dabei an, wie oft man mindestens umsteigen muss, wenn man die optimale Verbindung wählt.

3. Schreiben sie ein Prädikat `tausch12/2`, das zwei Listen als Argumente nimmt und gelingt, wenn sich die beiden Listen nur in der Reihenfolge der ersten beiden Elemente unterscheiden.

```
?- tausch([a,b,c,d],[b,a,c,d]).
true.
```

4. Schreibe ein Prädikat `all_members/2`, das zwei Listen L1 und L2 nimmt und gelingt, wenn alle Elemente von L1 auch Element von L2 sind.

```
?- all_members([a,c],[a,b,c,d]).
true.
?- all_members([a,e],[a,b,c,d]).
false.
?- all_members(a,[a,b,c,d]).
false.
```

5. Was antwortet Prolog auf die folgenden Anfragen:

```
?- food(bread,X) = food(Y,sausage)
?- food(bread,X,beer) = food(Y,kahuna_burger)
?- meal(food(bread),drink(beer)) = meal(X,Y)
?- meal(food(bread),X) = meal(X,drink(beer))
?- g(a,B,c) \= g(A,b,C).
?- g(a,b,c) \= g(A,C).
?- [a,b,c,d] = [a,b,X].
?- [a,b,c,d] = [a,b|X].
?- [[die,Y]|Z]=[[X,katze],[ist,weg]].
?- [anna,X]=[Y|[maria]].
?- [_,X,_,Y|_] = [dead(zed), [2, [b, chopper]], [], []].
?- [_,_,[_|X]|_] = [[], dead(zed), [2, [b, chopper]], [], Z, [2, [b, chopper]]].
?- 3 is 1+2.
?- 3 is +(1,2).
?- 3 is X+2.
?- X is 1+2.
?- 1+2 is 1+2.
```

6. Gegeben die folgende Wissensbasis:

```
verdaut(X,Y) :- hatgegessen(X,Y).
verdaut(X,Y) :- hatgegessen(X,Z),
                verdaut(Z,Y).

hatgegessen(moskito,blut(john)).
hatgegessen(frosch,moskito).
hatgegessen(storch,frosch).
```

   Wie lauten die Antworten auf die folgenden Anfragen? In welcher Reihenfolge werden die Ergebnisse für die letzte Anfrage ausgegeben?

```
1 ?- verdaut(storch,frosch).
2 ?- verdaut(storch,moskito).
3 ?- verdaut(frosch,X).
```

7. Gegeben die folgende Wissensbasis:

```
vorfahr(X,Y):- et(X,Y).
vorfahr(X,Z):-
     vorfahr(Y,Z),
     et(X,Y).

et(albert,kevin).
et(lena,albert).
et(marie,lena).
```

   Was antwortet Prolog auf die folgenden Anfragen? In welcher Reihenfolge werden die Antworten gegeben?

```
?- vorfahr(lena,kevin).
?- vorfahr(kevin,albert).
?- vorfahr(X,kevin).
```

8. Welche der beiden folgenden Definitionen des Prädikats `vorfahr/2` ist besser? Begründen Sie Ihre Antwort in einem Satz.

```
vorfahr(X,Y):- et(X,Y).
vorfahr(X,Z):-
     vorfahr(Y,Z),
     et(X,Y).
```

```
vorfahr(X,Y):- et(X,Y).
vorfahr(X,Z):-
     et(X,Y),
     vorfahr(Y,Z).
```

9. Schreibe ein 2-stelliges Prädikat `haelfte/2`, das zwei Zahlen als Argumente nimmt und wahr ist, wenn das zweite Argument halb so groß ist wie das erste.

```
?- haelfte(2,1).
true.
```

```
?- haelfte(5,2).
false.
?- haelfte(3,X).
X=1.5.
```

10. Gegeben das folgende Prädikat:

```
double([],[]).
double([H1|T1],[H2|T2]):- H2 is H1*2.
```

    Welche Argumente müssen bei dem Aufruf des Prädikats instantiiert sein?

11. Schreiben Sie ein 2-stelliges Prädikat `reiss/3`, das drei Listen als Argumente nimmt. Das Prädikat gelingt, wenn die ersten beiden Listen gleich lang sind und die dritte Liste aus den ersten beiden Listen im Reißverschlußverfahren entsteht: Die Elemente der ersten Liste sollen dabei auf den ungeraden Positionen und die der zweiten Liste auf den geraden Positionen der dritten Liste stehen.

```
?- reiss([a,b,c],[1,2,3],X).
X=[a,1,b,2,c,3].
?- reiss([a,a],[b,b,b],X).
false.
?- reiss([a,a],[b,b],[a,b,a,b]).
true.
```

12. Das folgende Prädikat dreht eine Liste um. Was muss an der Stelle des Fragezeichens in der Prädikatsdefinition stehen?

```
reverse(L,RL):- reverse(L,?,RL).

reverse([],L,L).
reverse([H|T],RT,RL):-
   reverse(T,[H|RT],RL).
```

13. Das folgende Prädikat konkateniert zwei Listen zu einer dritten. Was muss an der Stelle des Fragezeichens in der Prädikatsdefinition stehen?

```
append([],?,L).
append([H|T1],L2,[H|T3]) :- append(T1,L2,T3).
```

14. Übertragen Sie die folgende kontextfreie Sprache in eine Prolog DCG. Welche Sprache wird von der Grammatik generiert?
    S $\rightarrow$ A, S, B
    S $\rightarrow \epsilon$
    A $\rightarrow$ a
    B $\rightarrow$ b

15. Geben Sie eine kontextfreie Grammatik an, die die Sprache generiert, die aus allen Palindromen über dem Alphabet $\{c, d\}$ besteht.

16. Geben Sie die interne Prolognotation zu den folgenden DCG-Klauseln an:

```
det --> [].
adj --> [klein].
simple_adj --> adj, n.
a --> b(p), c(p).
a(p) --> [hu] {mag(popeye,spinat)}.
```

17. Die folgende Grammatik erzeugt NPs. Welche Anfrage müssen Sie stellen, um nacheinander alle grammatisch wohlgeformten NPs angezeigt zu bekommen?

```
np --> det, adjs, n.
adjs --> [].
adjs --> adj, adjs.
adj --> [kleine].
det --> [eine].
n --> [maus].
```

18. Bringen Sie die DCG-Klauseln in der zweiten Zeile in eine möglichst günstige Reihenfolge

```
np --> det, adjs, n.
adjs --> adj, adjs. adjs --> [].
adj --> [kleine].
det --> [eine].
n --> [maus].
```

19. Schreiben Sie eine DCG, die die folgenden Ausdrücke generiert:

```
eins
minus eins
minus minus eins
minus minus minus eins
...
```

    Verwenden Sie dazu folgende Rumpf-DCG:

```
m --> [minus].
e --> [eins].
```

20. Schreiben Sie eine DCG, die die Sprache $a^n b^m c^n d^m$ akzeptiert.