Master MLDM/DSC/CPS2 - First year
Introduction to Artificial Intelligence - Exam on Prolog

Maximum time allocated: 3h00 - No documents allowed. Scoring will depend on the cleanliness of your examination paper and the clarity of the explanations. TAKE CARE: any cheating will be severely punished and will lead to a formal complaint to the disciplinary council of the university.

1 Vocabulary (2 points)

Answer the following questions, related to the Prolog language, in one or two sentences.

1. What is a Horn clause?
2. What is a fact?
3. What is a rule?
4. What is a recursive clause?
5. What is a compound term?
6. What is the Closed-World Assumption?
7. What is the Occurs Check?
8. What does DCG mean?

2 Proof tree (3 points)

Consider the Prolog program below:

```
s(a).       s(b).       s(c).
q(a).       q(c).
r(f,a,g).   r(s,b,d).
p(X,Y) :- s(X), q(Y).
p(X,Y) :- q(Z), r(X,Z,Y).
```

Draw the proof tree of the resolution of the goal: `?- p(A,B).`

3 Unification (2 points)

Suppose the occurs check is activated. For each of the goals below, say whether it is true or false. In cases where it makes sense, give the value(s) of the variables that make(s) a goal true.

1. `?- [ a, X, b, c] = [a, b, c].`
2. `?- p(a, f(b)) = p(f(X), Y).`
3. `?- p(a, g(X), Y) = p(Y, g(f(X)), a).`
4. `?- [a,b,c,d,e,f] = [a, X, c, Y | Z].`
5. `?- '2' = 2.`
6. `?- Harry_Potter = harry_potter.`
7. `?- X = 2 + 1.`
8. `?- sentence([det(a),adj(little),adj(big),noun(dog)]) = sentence([X,adj(Y)|Z]).`

4 Knowledge base modeling and querying (3 points)

Convert the following information into a Prolog program (to choose the appropriate Prolog representation, you should look at the goals we then want to prove): The job of Chopin, Liszt and Rachmaninov is classical music composer. The job of Jagger and Presley is rock singer. The job of Sartre is philosopher and novelist. The job of Levy is novelist. If someone's job is classical music composer or rock singer, then the working area of this person is music. If someone's job is philosopher of novelist, then the working area of this person is writing. If the domain area of somebody is music or writing, then this person is an artist.

Considering the Prolog program you just wrote, convert the following english queries into Prolog goals:

1. What are the names of the classical music composers?
2. What are the names of all the artists?
3. Are there some persons that are both writer and musician?
4. What are the jobs of Sartre?
5. What is the working area of Sartre?
6. Is Sartre an artist?

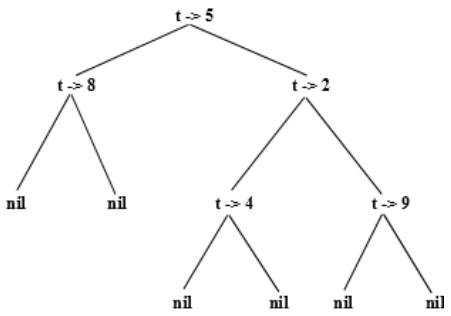
5 Lists (4 points)

Define the following predicates that specify some relationships between lists.

1. `isin/2` where `isin(X,L)` is true if X is an element of the list L.
2. `mylast/2` where `mylast(X,L)` is true if X is the last element of the list L.
3. `element_k/2` where `element_k(X,L,K)` is true if the value X is in position K in the list L.
4. `duplicate/2` where `duplicate(L1,L2)` is true if the elements of L1 are duplicated twice in the list L2.
5. `myreverse/2` where `myreverse(L1,L2)` is true if L2 is the list L1 reversed. To write this predicate you are **not** allowed to use the built-in predicate `append` or any equivalent predicate.
6. `compress/2` where `compress(L1,L2)` is true if L2 is equal to L1 without any consecutive duplicated value.

6 Trees (3 points)

A tree can be represented by a compound term t(V,L,R) where V is the value of the root of the tree, L is the left subtree of the tree, and R is the right subtree of the tree. The empty tree is denoted `nil`.
For example the compound term t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil)) is a Prolog representation of the following tree:



Define the following predicates :

1. `is_a_tree/1` where `is_a_tree(T)` is true if T is an empty tree denoted `nil` or if T is a tree of the form t(V,L,R) where L and T are trees.
2. `count_leaves/2` where `count_leaves(T,N)` is true if N is the number of leaves of the tree T.
3. `collect_leaves/2` where `collect_leaves(T,L)` is true if L is the list containing all the leaves of the tree T.

(in the tree above, there are three leaves: 8, 4 and 9)

7 DCG (3 points)

Consider the formal grammar of arithmetic expressions :

exp → exp '+' exp1
exp → exp '-' exp1
exp → exp1
exp1 → exp1 '*' exp2
exp1 → exp1 '/' exp2
exp1 → exp2
exp2 → '(' exp ')'
exp2 → '0'
exp2 → '1'
exp2 → '2'
exp2 → '3'

1. Convert this grammar to a DCG that can be used to prove if an arithmetic expression is syntactically correct and to evaluate it.
2. Write the Prolog goal you have to run to prove that the arithmetic expression 2 * (2 + 1) - 1 is syntactically correct and to evaluate it.

① **Horn Clause:** A horn clause is a clause with at most one positive literal.

Fact: A fact is a predicate expression that makes a declarative statement about the problem domain.

Rule:- A rule is a predicate expression that uses logical implication (:-) to describe a relationship among facts.

Recursive clause: Predicate (expression) which uses recursion or a rule.

Compound term:- It's made up of
↳ an atom
↳ some terms between parenthesis

Closed word Assumption:- It's the assumption that which's not known to be false, so that absence of information is interpreted as a negative information.

Occurs Check:- It causes unification of a variable V & a structure S if S contains V.

DCG:- A prolog DCG contains a list.
DCG rule Head --> Body.

② ① False
② False
③ False
④ X = b
Y = d
Z = [e, f]

⑤ False
⑥ True
⑦ X = 2+1 (True)
⑧ X = det(a)
Y = little
Z = [adj(big), noun(dog)].

② P(A,B)
A=-1 B=-2
A=-1 B=-2 Z=-3
S(-1), 9(-2)
9(-3), r(-1,-3,-2)
-1=a -1=b
S(a), 9(-2) S(b), 9(-2)
-2=e -2=a -2=a -2=b
S(a), 9(e) S(a), 9(a) S(a), 9(e) S(b), 9(a) S(b), 9(e)
True True True True
(A=a B=e) (A=a, A=a) (A=a B=b)

① job('chopin', musicComposer).
job('liszt', musicComposer).
job('Rachmaninov', musicComposer).
job('Jagger', rockSinger).
job('Presely', rockSinger).
job('Satre', philosopher).
job('Satre', novelist).
job('Levy', novelist).

work(X, music) :- job(X, musicComposer).
work(X, music) :- job(X, rockSinger).
work(X, writing) :- job(X, philosopher).
work(X, writing) :- job(X, novelist).
artist(X) :- work(X, writing); work(X, music).

① job(X, musicComposer).
② artist(X).
③ work(X, writing), work(X, musicComposer).
④ job('Satre', Y).
⑤ work('Satre', X).
⑤ artist('Satre').

⑤
```
isin([H|T], H):-!.
isin([H|T], L) :- isin(T, L).
```

②
```
mylast([H], H):-!.
mylast([H|T], L):- mylast(T, L).
```

③
```
element_K([H|T], H, 1):-!.
element-K([H|T], L, K):-
        K1 is K-1,
        element_K(T, L, K1).
```

④
```
duplicate([],[]).
duplicate([H1|T1],
          [H1,H1|T2]):-
        duplicate(T1, T2).
```
[a,b,c] → [a,a,b,b,c,c]

⑤
```
myreverse(L1,L2) :- myreverse2(L1,L2,L).

myreverse2([], L2, L2).
myreverse2([H1|T1], L2, L) :- myreverse2
                              (T1, L2, [H1|L]).
```

⑥
```
Comprem([],[]).
Comprem([X],[X]).
Comprem([X,X|L1], L2):- Comprem([X|L1], L2).
Comprem([X,Y|L1], [X|L2]) :-
        X \= Y,
        Comprem([Y|L1], L2).
```

⑥
```
is_a_tree(t(value,[],[])).
is_a_tree(t(V,L,R)):-
        is_a_tree(L),
        is_a_tree(R).
```

⑦
```
count_leaves(t(V,[],[]), 1).
count_leaves(t(V,L,R), N) :-
        count_leaves(L, N1),
        count_leaves(R,N2),
        N is N1+N2.
```

②
```
collect_leaves(t(V,nil,nil), L) :- is_in(V, L).

collect_leaves(t(V,Left,Right), L) :-
        collect_leaves(Left),
        collect_leaves(Right).
```

⑦
```
exp --> exp, ['+'], exp1.
exp --> exp, ['-'], exp1.
exp --> exp1.
exp1 --> exp1, ['*'], exp2.
exp1 --> exp1, ['/'], exp2.
exp1 --> exp2.
exp2 --> ['('], exp, [')'].
exp2 --> ['0'].
exp2 --> ['1'].
exp2 --> ['2'].
exp2 --> ['3'].
exp2 --> ['4'].
exp2 --> ['5'].
exp2 --> ['6'].
exp2 --> ['7'].
exp2 --> ['8'].
exp2 --> ['9'].
exp2 -->
```

①
```
phrase(exp, ['2','*','(','2','+','1',')','-','1']).
```