



deerwalk
DWIT College

Lab Report 4

Submitted by:

Abhishek Kadariya

0501

2019

Submitted to:

Birodh Rijal

(Artificial Intelligence Lecturer)

1. Operators on list

a. Write on list

writelist([]) :- nl.

writelist([H|T]) :- write(H),nl,writelist(T).

?- writelist([1,2,3,4]).

1

2

3

4

b. Membership

member(X,[X|List]).

member(X,[Element|List]) :- member(X,List).

?- member(1, [1,2,3,4]).

true.

?- member(5, [1,2,3,4]).

false.

c. Concatenation

conc([],L,L).

conc([X|L1],L2,[X|L3]) :- conc(L1,L2,L3).

?- conc(['a','b'], ['c','d'], L).

L = [a, b, c, d].

d. Take the n-th element

take(1,[H|_],H).

$\text{take}(N,[_|T],X) : -N1 \text{ is } N - 1, \text{take}(N1,T,X).$

?- take(3,[1,2,3], 3).

true.

?- take(3,[1,2,3], 4).

false.

e. Length of a list

$\text{lengths}([],0).$

$\text{lengths}([H|T],N) :- \text{lengths}(T,M), N \text{ is } M + 1.$

?- lengths([1,2,3,4,5], 7).

false.

?- lengths([1,2,3,4,5], 5).

true.

f. Sum of elements

$\text{sum}([],0).$

$\text{sum}([X|L],\text{Sum}) :- \text{sum}(L,\text{SL}), \text{Sum} \text{ is } X + \text{SL}.$

?- sum([1,2,3],6).

true.

g. Reverse of a list

$\text{reverse}([],X,X).$

$\text{reverse}([X|Y],Z,W) :- \text{reverse}(Y,[X|Z],W).$

?- reverse([1,2,3],L).

$L = [3, 2, 1].$

h. Append

`append([],L,L).`

`append([H|T],L,[H|TL]) :- append(T,L,TL).`

`?- append([1,2,3,4],[5],[1,2,3,4,5]).`

true.

`?- append([1,2,3,4],[5, 6],[1,2,3,4,5,6]).`

true.

`?- append([1,2,3,4],[5, 6, 5],[1,2,3,4,5,6]).`

false.

2. DFA with input as a list

`% Code snippet begin`

```
t(0,a,1).      t(0,b,2).  
t(1,a,1).      t(1,b,1).  
t(2,a,2).      t(2,b,2).
```

```
startstate(0).  % 0 is a starting state  
finalstate(1).  % 1 is a final state
```

`% Code snippet end`

Implement a predicate `checkinput(Start,Input)` that checks if a word (here, input) given as a list (e.g. `[a,b,b,a,b]`) is accepted by the DFA starting from a start state (here `State`).

Answer:

`t(0,a,1).`

`t(0,b,2).`

`t(1,a,1).`

`t(1,b,1).`

t(2,a,2).

t(2,b,2).

checkinput(Start, []) :- Start is 1.

checkinput(Start, [H|T]) :- t(Start,H,Next), checkinput(Next, T).

?- checkinput(0,[a]).

true .

?- checkinput(1,[a]).

true .

?- checkinput(1,[b]).

true .

?- checkinput(0,[a,b]).

true .

3. Using Structures;

family(

person(homer,simpson,date(7,may,1960),works(Inspector,6000)),

person(marge,simpson,date(7,may,1965),housewife), [

person(bart,simpson,date(7,may,1967),student),

person(lisa,simpson,date(7,may,1965),student)].

Using the family predicate, implement the following relation as rules:

a. husband(X) : true if X is someone's husband

Answer:

husband(H) :- family(person(H,_,_,_),_,_).

```
?- consult("F:\\DWIT\\5. Fifth semester\\4. Artificial Intelligence\\lab\\lab 5\\struct.pl").  
true.
```

```
?- husband(homer).  
true.
```

b. wife(X) : true if X is someone's wife

Answer:

```
wife(W) :- family(_,person(W,_,_),_).
```

```
?- wife(marge).  
true.
```

c. child(X) : true if X is someone's child

Answer:

```
child(X) :- family(_,_,Children), member(person(X,_,_),  
Children).
```

```
?- child(bart).  
true .
```

d. exists(Person) : true if the person is in the database

Answer:

```
exists(Person) :-  
husband(Person);wife(Person);child(Person).
```

?- exists(homer).

true .

?- exists(bart).

true .

?- exists(ram).

false.