```
:- op(1, fy, -). % negación

:- op(2, xfy, &). % conjunción

:- op(2, xfy, v). % disyunción

:- op(3, xfy, =>). % implicación

:- op(3, xfy, <=>). % Doble Implicación


%si al evaluar la formula en M el resultado es 1, se dice que M es modelo de la formula

es_modelo(ESTRUCTURA,FORMULA) :-
valor(FORMULA,ESTRUCTURA,V),nl,write(ESTRUCTURA),write("  "),write(V),V = 1.


% descompone la formula y evalua. p.e (p => q) v (q => p) se descompone en A = (p =>q) v B = (q =>
p).

% %Luego descompone A = (p) => B= (q). Con memberchk obtiene el valor de p y de q, finalmente

%evalua de adentro hacia afuera (por orden de operadores) las definiciones de las funciones, en
este caso v y => (definidas en eval)

valor(F, ESTRUCTURA, V) :-memberchk((F,V), ESTRUCTURA).

valor(-A, ESTRUCTURA, V) :-valor(A, ESTRUCTURA, VA),eval(-, VA, V).

valor(F, ESTRUCTURA, V) :-F =..[Op,A,B], valor(A, ESTRUCTURA, VA),valor(B, ESTRUCTURA, VB),

eval(Op, VA, VB, V).


%eval contiene las definiciones de

eval(v, 0, 0, 0) :- !.%Definición de disyuncion (or)

eval(v, _, _, 1).

eval(&, 1, 1, 1) :- !.%Definición de conjuncion (and)

eval(&, _, _, 0).

eval(=>, 1, 0, 0) :- !.%definición de implicación (=>)

eval(=>, _, _, 1).

eval(<=>, X, X, 1) :- !.%definición de doble implicación (<=>)

eval(<=>, _, _, 0).

eval(-, 1, 0). %definición de negación (-)
```

```
eval(-, 0, 1).


%obtención de las estructuras M. p.e para el conjunto de variables proposicionales
% ingresas un conjungo de variables p.e [p,q], devuelve las posibles estructuras:
% [(p,0),(q,0)] , [(p,0),(q,1)], [(p,1),(q,0)], [(p,1),(q,1)]
getEstructuras([],[]).
getEstructuras([VARIABLE|VARIABLES_COLA],[(VARIABLE,VALOR)|ML]) :-
valor_verdad(VALOR),
getEstructuras(VARIABLES_COLA,ML).


%Los valores que pueden tomar las variables son 0 y 1, si ponemos valor_verdad(X), regresa 0 y 1
valor_verdad(0).
valor_verdad(1).
%La condición de satisfacción de una formula es que tenga al menos 1 modelo. Un modelo es una
%estructura que al aplicar a la formula hace que de verdadero. Para saber si una formula
%es satisfacible es conveniente revisar estructura por estructura (00,01,10,11; para dos variables por ejemplo)
% hasta que una de ellas sea un modelo es decir
es_satisfacible(FORMULA,VARIABLES) :-
getEstructuras(VARIABLES,ESTRUCTURA), %obtiene las posibles estructuras
es_modelo(ESTRUCTURA,FORMULA).% evalua las estrcuturas para comprobar si alguna es modelo, si lo es entonces la formula es satisfacible
```