```
:- [lab6lmc3].

/* Sa de determine toate functiile de la L2+L2^2 la L3 care pastreaza compunerea relatiei de
succesiune cu ea insasi, si sa se arate ca niciuna nu e injectiva. */

l2plusromb(L,OrdL) :- L=[0,a,b,c,1],
        orddinsucc([(0,a),(a,b),(a,c),(b,1),(c,1)],L,OrdL).

fctposetarb(P,OrdP,Q,OrdQ,LF) :- succdinord(OrdP,SuccP),
        succdinord(OrdQ,SuccQ),
        comprel(SuccP,SuccP,SuccPoSuccP), comprel(SuccQ,SuccQ,SuccQoSuccQ),
        setof(F, (functie(F,P,Q), pastrrel(F,SuccPoSuccP,SuccQoSuccQ)), LF), !.
fctposetarb(_,_,_,_,[]).

fctcerute(LF) :- l2plusromb(L,OrdL), l3(L3,OrdL3),
        fctposetarb(L,OrdL,L3,OrdL3,LF).

nusuntinj(LF) :- not((member(F,LF), write(F), nl, injectiv(F))).

testneinjcerut :- fctcerute(LF), nusuntinj(LF).

niciunainj([]).
niciunainj([F|LF]) :- not(injectiv(F)), write(F), nl, niciunainj(LF).

testulneinjcerut :- fctcerute(LF), niciunainj(LF).

/* Interogati:
```

```
?- fctcerute(LF), afislista(LF).
?- testneinjcerut.
?- testulneinjcerut.
?- nusuntinj([[(a,b),(u,b)],[(x,1),(y,1)],[(x,1),(y,2)],[(a,1)]]).
?- niciunainj([[(a,b),(u,b)],[(x,1),(y,1)],[(x,1),(y,2)],[(a,1)]]).
*/

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/* Reprezint:
        conectorul logic de negatie prin -|,
        faptul ca un enunt e teorema formala (i.e. adevar sintactic), precum si deductia
sintactica, prin |-,
        faptul ca un enunt e tautologie (i.e. adevar semantic, i.e. enunt universal
adevarat, adica adevarat in orice interpretare, adica pentru orice valori de adevar pentru
variabilele din componenta sa), precum si deductia semantica, prin |=.
   Amintesc ca:
        pentru o interpretare (i.e. evaluare, semantica, functie care da variabilelor
propozitionale valori de adevar) h:V->L2, unde V este multimea variabilelor propozitionale
(ale logicii propozitionale clasice), iar L2={0,1} este algebra Boole standard (lantul cu 2
elemente, i.e. algebra Boole a valorilor de adevar pentru logica clasica, unde 0 reprezinta
falsul, iar 1 reprezinta adevarul),
        se noteaza cu h~:E->L2 unica prelungire a lui h la multimea E a enunturilor care
transforma conectorii logici in operatii booleene si astfel calculeaza valorile de adevar
ale tuturor enunturilor pe baza valorilor de adevar pe care h le da variabilelor
propozitionale.
   Posetul ({false,true},<=), cu false<true, se organizeaza ca algebra Boole cu operatiile
```

```
date de aplicarea conectorilor logici constantelor booleene false,true, iar functia
f:L2->{false,true}, definita prin f(0)=false si f(1)=true, este izomorfism boolean. Prin
urmare:
        pentru orice interpretare h:V->L2, intrucat h transforma conectorii logici in
operatii booleene, iar f pastreaza operatiile booleene, rezulta ca foh~:E->{false,true}
transforma conectorii logici in operatii booleene;
        oricare ar fi enuntul beta si interpretarea h:V->L2, are loc:
                h |= beta <=> h~(beta)=1 <=> f(h~(beta))=true.
   Fie enunturile alfa, fi, psi, hi, astfel incat:
        alfa = [fi -> -|(-|psi -> hi)] <-> [(psi -> -|fi) ^ (hi -> -|fi)].
   Sa demonstram ca:
        |= alfa,
i.e. alfa e adevar semantic, i.e. adevarat in orice interpretare, i.e. adevarat indiferent
ce valori de adevar au variabilele din componenta sa, i.e. adevarat indiferent ce valori de
adevar au enunturile fi, psi, hi.
   Consideram variabilele Prolog Fi, Psi, Hi, reprezentand urmatoarele valori booleene:
        Fi = f(h~(fi)), Psi = f(h~(psi)), Hi = f(h~(hi)).
   Atunci valoarea booleana f(h~(alfa)) este egala cu valoarea urmatorului predicat in
tripletul (Fi,Psi,Hi): */

alfa(Fi,Psi,Hi) :- echiv(implica(Fi,not(implica(not(Psi),Hi))),
        (implica(Psi,not(Fi)), implica(Hi,not(Fi)))).

demalfa :- not((listaValBool([Fi,Psi,Hi]), not(alfa(Fi,Psi,Hi)))).

/* Acum sa demonstram ca:
        {fi, fi -> (psi -> hi), -|hi} |= -|psi,
```

```
   i.e., pentru orice h:V->L2:
daca h |= {fi, fi -> (psi -> hi), -|hi}, atunci h |= -|psi,
i.e.: daca h~(fi) = h~(fi -> (psi -> hi)) = h~(-|hi) = 1, atunci h~(-|psi) = 1,
i.e.: daca f(h~(fi)) = f(h~(fi -> (psi -> hi))) = f(h~(-|hi)) = true,
        atunci f(h~(-|psi)) = true: */

demdeductie :- not((listaValBool([Fi,Psi,Hi]),
        Fi, implica(Fi,implica(Psi,Hi)), not(Hi), not(not(Psi))))).

/* Sa demonstram ca, pentru orice multimi de enunturi Sigma1, Sigma2, Sigma3 si orice
enunturi fi, psi, hi, are loc regula de deductie:
        Sigma1 U {fi} |- psi, Sigma2 U {psi^hi} |- fi, Sigma3 U {psi} |- hi
        ____________________________________________________________________
                Sigma1 U Sigma2 U Sigma3 |- fi <-> (psi^hi)
Consideram Sigma1, Sigma2, Sigma3 multimi de enunturi si fi, psi, hi enunturi a.i. Sigma1 U
{fi} |- psi, Sigma2 U {psi^hi} |- fi, Sigma3 U {psi} |- hi,
i.e., conform TD:
        Sigma1 |- fi->psi, Sigma2 |- (psi^hi)->fi, Sigma3 |- psi->hi,
asadar, conform TCT:
        Sigma1 |= fi->psi, Sigma2 |= (psi^hi)->fi, Sigma3 |= psi->hi.
Avem de demonstrat ca Sigma1 U Sigma2 U Sigma3 |- fi <-> (psi^hi). Conform TCT,
e suficient sa demonstram ca: Sigma1 U Sigma2 U Sigma3 |= fi <-> (psi^hi).
Fie h:V->L2 a.i. h |= Sigma1 U Sigma2 U Sigma3, i.e.:
        h |= Sigma1, h |= Sigma2 si h |= Sigma3,
asadar h |= fi->psi, h |= (psi^hi)->fi si h |= psi->hi, i.e.:
        h~(fi->psi) = h~((psi^hi)->fi) = h~(psi->hi) = 1, adica:
        f(h~(fi->psi)) = f(h~((psi^hi)->fi)) = f(h~(psi->hi)) = f(1) = true.
```

```
Avem de demonstrat ca h |= fi <-> (psi^hi), i.e. h~(fi <-> (psi^hi)) = 1,
adica: f(h~(fi <-> (psi^hi))) = true.
Notam cu: Fi = f(h~(fi)), Psi = f(h~(psi)), Hi = f(h~(hi)).
Atunci valorile lui f(h~(fi->psi)), f(h~((psi^hi)->fi)), f(h~(psi->hi)), respectiv f(h~(fi
<-> (psi^hi))) sunt:
*/

ipoteza1(Fi,Psi) :- implica(Fi,Psi).

ipoteza2(Fi,Psi,Hi) :- implica((Psi,Hi),Fi).

ipoteza3(Psi,Hi) :- implica(Psi,Hi).

concluzia(Fi,Psi,Hi) :- echiv(Fi,(Psi,Hi)).

demregded :- not((listaValBool([Fi,Psi,Hi]), ipoteza1(Fi,Psi),
        ipoteza2(Fi,Psi,Hi), ipoteza3(Psi,Hi), not(concluzia(Fi,Psi,Hi)))).

/* Problema cu triburile:
Consideram trei variabile propozitionale a,b,c (i.e. elemente ale lui V) doua cate doua
distincte, pe care le instantiem cu enunturile:
   a: A spune adevarul,
   b: B spune adevarul,
   c: C spune adevarul.
Avem enunturile:
   alfa = (b^c) <-> c,
   beta = (a^c) -> -|[(b^c) -> a],
```

```
   gama = -|b <-> (a v b).
Fie h : V->L2, astfel ca foh : V->{false,true}, iar foh~ : E->{false,true}.
Notam cu variabilele Prolog: A = f(h(a)), B = f(h(b)), C = f(h(c)).
Calculam valorile lui foh~ in enunturile alfa, beta, gama, apoi in enunturile a<->alfa,
b<->beta, c<->gamma si impunem conditia ca acestea din urma sa fie adevarate in
interpretarea h, ale carei valori in a,b,c dau apartenenta bastinasilor A,B, respectiv C la
unul sau altul dintre triburile Tu si Fa: */

alfa(B,C) :- echiv((B,C), C).

beta(A,B,C) :- implica((A,C), not(implica((B,C), A))).

gama(A,B) :- echiv(not(B), A;B).

detTriburi(A,B,C) :- listaBool([A,B,C]), echiv(A,alfa(B,C)),
        echiv(B,beta(A,B,C)), echiv(C,gama(A,B)).

afisTriburi(A,B,C) :- detTriburi(A,B,C), scrieTriburi(['A',A,'B',B,'C',C]).

scrieTriburi([]).
scrieTriburi([Nume,Trib|T]) :- scrieTrib(Nume,Trib), scrieTriburi(T).

scrieTrib(Nume,Trib) :- write('bastinasul '), write(Nume),
        write(' face parte din tribul '), detTrib(Trib), nl.

detTrib(true) :- write('Tu').
detTrib(false) :- write('Fa').
```

```
/* Fie p,q in V, iar fi = (p^q) v -|(p->q) v -|(q->p) v (-|p ^ -|q).
Sa demonstram ca |-fi. Conform TC, e suficient sa demonstram ca |=fi.
Fie variabilele Prolog: P = f(h(p)), Q = f(h(q)).
Calculam f(h~(fi)), apoi aratam ca e intotdeauna adevarata: */

fi(P,Q) :- P,Q ; not(implica(P,Q)) ; not(implica(Q,P)) ; not(P),not(Q).

demfi :- not((listaValBool([P,Q]), not(fi(P,Q)))).

/* Fie alfa, beta, gama, delta, epsilon in V.
Sa demonstram ca multimea:
M = {alfa -> beta, beta -> (gama^delta), gama -> alfa, delta -> -|alfa}
e satisfiabila.
Fie h:V->L2. Consideram variabilele Prolog: A = f(h(alfa)), B = f(h(beta)), C = f(h(gama)),
D = f(h(delta)), E = f(h(epsilon)).
Atunci foh~ are aceste valori in enunturile din multimea M: */

enuntul1(A,B) :- implica(A,B).

enuntul2(B,C,D) :- implica(B,(C,D)).

enuntul3(A,C) :- implica(C,A).

enuntul4(A,D) :- implica(D,not(A)).

/* Daca alfa, beta, gama, delta, epsilon sunt doua cate doua distincte, atunci putem
```

```
demonstra astfel faptul ca M e satisfiabila: */

satisfiabila :- listaBool([A,B,C,D,E]), enuntul1(A,B), enuntul2(B,C,D),
        enuntul3(A,C), enuntul4(A,D), write((A,B,C,D,E)), nl.

% Altfel, o demonstratie corecta de satisfiabilitate pentru M este:

satisfiabilitate :- member(X,[false,true]), enuntul1(X,X), enuntul2(X,X,X),
        enuntul3(X,X), enuntul4(X,X), write(X), nl.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reprezint cuantificatorul universal prin \-/.

/* Fie signatura tau = (1;2;0) si algebra (A,f,R,k) de aceasta signatura, cu multimea suport
A = {a,b,c}, avand |A|=3 (i.e. a=/=b=/=c=/=a), R relatie de ordine pe A a.i. c = min(A,R),
iar a,b sunt elementele maximale ale posetului (A,R), f : A->A strict crescatoare de la
posetul (A,R) la dualul sau (A,R^-1) cu f(c)=a, iar k=max(A,R^-1). Sa se determine daca:
        A |= \-/x \-/y [f(x)=y -> (R(x,y) v y=k)]. */

multA([a,b,c]).

ordA(R) :- multA(A), relord(R,A), min(A,R,c), elemmax(a,A,R), elemmax(b,A,R).

fctF(F) :- multA(A), ordA(R), invrel(R,Q),
        ordstrdinord(R,S), ordstrdinord(Q,T),
        functie(F,A,A), pastrrel(F,S,T), member((c,a),F).
```

```
ctK(K) :- multA(A), ordA(R), invrel(R,Q), max(A,Q,K).

verifAsatenunt :- multA(A), ordA(R), fctF(F), ctK(K),
        not((member(X,A), member(Y,A), write((X,Y)), nl,
        not(implica(member((X,Y),F), member((X,Y),R) ; Y=K)))).
```