```
%Autómatas de ejemplo. Si agregan otros, mejor.

ejemplo(1, a(s1, [sf], [(s1, a, sf)])).
ejemplo(2, a(si, [si], [(si, a, si)])).
ejemplo(3, a(si, [si], [])).
ejemplo(4, a(s1, [s2, s3], [(s1, a, s1), (s1, a, s2), (s1, b, s3)])).
ejemplo(5, a(s1, [s2, s3], [(s1, a, s1), (s1, b, s2), (s1, c, s3), (s2, c, s3)])).
ejemplo(6, a(s1, [s3], [(s1, b, s2), (s3, n, s2), (s2, a, s3)])).
ejemplo(7, a(s1, [s2], [(s1, a, s3), (s3, a, s3), (s3, b, s2), (s2, b, s2)])).
ejemplo(8, a(s1, [sf], [(s1, a, s2), (s2, a, s3), (s2, b, s3), (s3, a, s1), (s3, b, s2), (s3, b, s4), (s4, f,
sf)])). % No deterministico :)
ejemplo(9, a(s1, [s1], [(s1, a, s2), (s2, b, s1)])).
ejemplo(10, a(s1, [s10, s11],
        [(s2, a, s3), (s4, a, s5), (s9, a, s10), (s5, d, s6), (s7, g, s8), (s15, g, s11), (s6, i, s7), (s13,
l, s14), (s8, m, s9), (s12, o, s13), (s14, o, s15), (s1, p, s2), (s3, r, s4), (s2, r, s12), (s10, s, s11)])).

ejemploMalo(1, a(s1, [s2], [(s1, a, s1), (s1, b, s2), (s2, b, s2), (s2, a, s3)])). %s3 es un estado sin
salida.
ejemploMalo(2, a(s1, [sf], [(s1, a, s1), (sf, b, sf)])). %sf no es alcanzable.
ejemploMalo(3, a(s1, [s2, s3], [(s1, a, s3), (s1, b, s3)])). %s2 no es alcanzable.
ejemploMalo(4, a(s1, [s3], [(s1, a, s3), (s2, b, s3)])). %s2 no es alcanzable.
ejemploMalo(5, a(s1, [s3, s2, s3], [(s1, a, s2), (s2, b, s3)])). %Tiene un estado final repetido.
ejemploMalo(6, a(s1, [s3], [(s1, a, s2), (s2, b, s3), (s1, a, s2)])). %Tiene una transición repetida.
ejemploMalo(7, a(s1, [], [(s1, a, s2), (s2, b, s3)])). %No tiene estados finales.

ejemploPropio(1, a(si, [sf], [(si, a, si), (sf, a, si), (si, b, si)])).
ejemploPropio(2, a(si, [sf], [(si, a, sf), (sf, a, si)])).
ejemploPropio(3, a(s1, [s2, s3], [(s1, a, s2), (s2, a, s2), (s2, b, s3)])).
ejemploPropio(4, a(s1, [s1], [(s1, a, s1), (s1, b, s1)])).
ejemploPropio(5, a(s0, [s1], [(s0, a, s0), (s0, b, s1), (s1, a, s0), (s1, b, s1)])).
ejemploPropio(6,a(s0,[s1,s2], [(s0,a,s1), (s0,a,s2)])).
ejemploPropio(7,a(s0,[s2],[(s0,a,s0),(s0,b,s1),(s1,a,s1),(s1,b,s2),(s2,a,s2),(s2,b,s0)])).
ejemploPropio(8,a(s0,[s3],[(s0,a,s1),(s2,a,s3)])).
ejemploPropio(9,a(s0,[s2],[(s0,a,s1),(s1,b,s0)])).
ejemploPropio(10,a(s0,[s2],[(s0,a,s1),(s0,b,s2)])).

%%Proyectores
inicialDe(a(I, _, _), I).
finalesDe(a(_, F, _), F).
transicionesDe(a(_, _, T), T).

%Auxiliar dada en clase
%desde(+X, -Y).
desde(X, X).
desde(X, Y):-desde(X, Z),  Y is Z + 1.

% Auxiliares propios

sinRepetidos([], []).
sinRepetidos([X|XS], L) :- member(X, XS), sinRepetidos(XS, L).
sinRepetidos([X|XS], [X|L]) :- not(member(X, XS)), sinRepetidos(XS, L).

last(X, [X]).
last(X, [_|XS]) :- last(X, XS).

%%Predicados pedidos.

% 1) %esDeterministico(+Automata)

%% Un automata sin transiciones es deterministico
esDeterministico(a(_, _, [])).

%% Por cada transicion del automata chequeo que no exista otra con el mismo origen y la misma etiqueta
esDeterministico(a(I, F, [T | TS])) :-  noExisteOtraTransicion(T, TS),
                    esDeterministico(a(I, F, TS)).

% O la Transicion tiene distinto Estado Origen, o la etiqueta es distinta
noExisteOtraTransicion(_, []).
noExisteOtraTransicion((TO, TE, TD), [(T2O, _, _) | TS]) :- TO \= T2O,
                              noExisteOtraTransicion((TO, TE, TD), TS), !.

noExisteOtraTransicion((TO, TE, TD), [(_, T2E, _) | TS]) :- TE \= T2E,
                              noExisteOtraTransicion((TO, TE, TD), TS).

% 2) estados(+Automata, ?Estados)
estados(a(S, FS, []), L) :- var(L), append([S], FS, L2), sort(L2, L1), sinRepetidos(L1, L).
estados(a(I, FS, [(TO, _, TD) | TS]), L) :-   var(L),
                                              append([TO, TD], LR, LBruto),
                                              estados(a(I, FS, TS), LR),
                                              sort(LBruto, LConRep),
                                              sinRepetidos(LConRep, L).

estados(A,L ) :- ground(L), estados(A,L2), sort(L,L3), sinRepetidos(L3,L2).
```

```
% 3)esCamino(+Automata, ?EstadoInicial, ?EstadoFinal, +Camino)

%% Cheuqeamos que el primer y ultimo estado del camino se correspondan con los del automata
%% Y luego nos concentramos en el camino intermedio
esCamino(A, EI, EF, [E1 | ES]) :-   EI = E1,
                  last(EF, ES),
                  esCaminoValido(A, [E1 | ES]).

%% Recorremos los estados del camino chequeando que existan transiciones entre cada par de ellos
esCaminoValido(A, [T]) :- estados(A, ES), member(T, ES).
esCaminoValido(A, [E1, E2 | ES]) :- transicionesDe(A, TS),
                  existeTransicion(E1, E2, TS),
                  esCaminoValido(A, [E2 | ES]).

%% Si encuentro una transicion entre los estados dejo de buscar, sino sigo
existeTransicion(EO, ED, [(EO, _, ED)|_]) :- !.
existeTransicion(EO, ED, [(_, _, _)|TS]) :- existeTransicion(EO, ED, TS).


% 4) ¿el predicado anterior es o no reversible con respecto a Camino y por
% qué? No, no es reversible. Esto se debe a que se genera una variable fresca en ES
% por cada transicion que esCaminoValido no puede unificar a traves de
% existeTransicion, por lo tanto si el camino no existe sigue buscando infinitamente.


% 5) caminoDeLongitud(+Automata, +N, -Camino, -Etiquetas, ?S1, ?S2)

%% Si el camino es de longitud 0 la unica posibilidad es que el estado inicial y final sea el mismo
caminoDeLongitud(A, 0, [], [], S1, S1) :- inicialDe(A, S1), finalesDe(A, FS), member(S1, FS).

%% Si el camino es de longitud 1 buscamos una transicion que los una
caminoDeLongitud(A, 1, [S1, S2], ES, S1, S2) :- transicionesDe(A, T),
                                                 member((S1, ET, S2), T), ES = [ET].

%% Iteramos los estados del camino fijandonos que exista una trancion entre el estado actual y el siguiente
caminoDeLongitud(A, N, [S1, C2|CS], [E|ES], S1, S2) :-  N > 1,

transicionesDe(A, T),

member((S1, ET, C2), T),

E = ET,

Nm1 is N - 1,

Nm1 > 0,

caminoDeLongitud(A, Nm1, [C2|CS], ES, C2, S2).

% 6) alcanzable(+Automata, +Estado)
alcanzable(A, S) :- inicialDe(A, EI), estados(A, ES), length(ES, M), between(1, M, N),
            caminoDeLongitud(A, N, _, _, EI, S), !.

% 7) automataValido(+Automata)
automataValido(A) :-  inicialDe(A, I), estados(A, ES), subtract(ES, [I], EsinInicial), forall(member(E,
EsinInicial), alcanzable(A, E)),
                      finalesDe(A, FS), not(not(member(_, FS))),
                      transicionesDe(A, TS), subtract(ES, FS, EsinFinales), forall(member(EsinF,
EsinFinales), member((EsinF, _, _), TS)),
                      sinRepetidos(FS, FS), sinRepetidos(TS, TS), !.

%--- NOTA: De acá en adelante se asume que los autómatas son válidos.

% 8) hayCiclo(+Automata)
hayCiclo(A) :- estados(A, ES), length(ES, N), Nm1 is N + 1, caminoDeLongitud(A, Nm1, _, _, _, _), !.

% 9) reconoce(+Automata, ?Palabra)

%% Si la palabra esta semi-instanciada/instanciada conozco su longitud,
%% entonces puedo llamar a caminoDeLongitud instanciando con esa longitud.
reconoce(A, PS) :-      nonvar(PS), length(PS, Long), inicialDe(A, EI), finalesDe(A, FS), member(F, FS),
                                    caminoDeLongitud(A, Long, _, PS, EI, F).

%% Si hay ciclos la longitud de la palabra puede ser infinita por lo que utilizo el predicado desde para
iterar todos los naturales e ir generando todos los caminos posibles con caminoDeLongitud, que tambien van a
ser infinitos.
reconoce(A, PS) :-      var(PS), hayCiclo(A), desde(0, N), inicialDe(A, EI), finalesDe(A, FS), member(F, FS),
                                    caminoDeLongitud(A, N, _, PS, EI, F).

%% Si no hay ciclos puedo acotar la longitud de una palabra por la cantidad de estados del automata, entonces
utilizo el predicado between para recorrer hasta esa cantidad.
reconoce(A, PS) :-      var(PS), not(hayCiclo(A)), estados(A, S), length(S, M), between(0, M, N),
                                    inicialDe(A, EI), finalesDe(A, FS), member(F, FS),
                                    caminoDeLongitud(A, N, _, PS, EI, F).
```

```
155
      % 10) PalabraMásCorta(+Automata, ?Palabra)
      %% Uso de Generate and Test para verificar que es la palabra mas corta. Se acota el espacio de busqueda
      dado que se busca alguna palabra de longitud menor a la instanciada primero en longitudPalabraMasCorta.
      palabraMasCorta(A, PS) :-       longitudPalabraMasCorta(A, N), length(PS, N), reconoce(A, PS),
160                                                          Nm1 is N -1, not(hayPalabraMasCorta(A, Nm1)).

      longitudPalabraMasCorta(A, N) :- reconoce(A, PS), length(PS, N), Nm1 is N -1,
                                                                  not(hayPalabraMasCorta(A, Nm1)), !.

165   hayPalabraMasCorta(A, N) :- between(1, N, M), length(PS, M), reconoce(A, PS).

      %-----------------
      %----- Tests -----
      %-----------------
170
      % Algunos tests de ejemplo. Deben agregar los suyos.

      test(1) :- forall(ejemplo(_, A),  automataValido(A)).
      test(2) :- not((ejemploMalo(_, A),  automataValido(A))).
175   test(3) :- ejemplo(10, A), reconoce(A, [p, X, r, X, d, i, _, m, X, s]).
      test(4) :- ejemplo(9, A), reconoce(A, [a,  b,  a,  b,  a,  b,  a,  b]).
      test(5) :- ejemplo(7, A), reconoce(A, [a,  a,  a,  b,  b]).
      test(6) :- ejemplo(7, A), not(reconoce(A, [b])).
      test(7) :- ejemplo(2, A),  findall(P, palabraMasCorta(A, P), [[]]).
180   test(8) :- ejemplo(4, A),  findall(P, palabraMasCorta(A, P), Lista), length(Lista, 2), sort(Lista, [[a],
      [b]]).
      test(9) :- ejemplo(5, A),  findall(P, palabraMasCorta(A, P), Lista), length(Lista, 2), sort(Lista, [[b],
      [c]]).
      test(10) :- ejemplo(6, A),  findall(P, palabraMasCorta(A, P), [[b, a]]).
      test(11) :- ejemplo(7, A),  findall(P, palabraMasCorta(A, P), [[a, b]]).
      test(12) :- ejemplo(8, A),  findall(P, palabraMasCorta(A, P), Lista), length(Lista, 2), sort(Lista, [[a,  a,
      b,  f], [a,  b,  b,  f]]).
185   test(13) :- ejemplo(10, A),  findall(P, palabraMasCorta(A, P), [[p, r, o, l, o, g]]).
      test(14) :- forall(member(X, [2, 4, 5, 6, 7, 8, 9]), (ejemplo(X, A), hayCiclo(A))).
      test(15) :- not((member(X, [1, 3, 10]), ejemplo(X, A), hayCiclo(A))).

      %% Tests Propios
190   test(16) :- ejemplo(2, A), reconoce(A, [a, X, a]).
      test(17) :- ejemploPropio(5,A), esDeterministico(A).
      test(18) :- ejemploPropio(6,A), not(esDeterministico(A)).
      test(19) :- ejemploPropio(5,A), estados(A,[s0,s1]).
      test(20) :- ejemploPropio(5,A), esCamino(A,s0,s1,[s0,s0,s0,s0,s0,s1,s1,s1]).
195   test(21) :- ejemploPropio(7,A), esCamino(A,s0,s2,[s0,s0,s1,s1,s2,s2]).
      test(22) :- ejemploPropio(7,A), not(esCamino(A,s0,s2,[s0,s1,s2,s2,s1,s0,s2])).
      test(23) :- ejemploPropio(7,A), alcanzable(A,s2).
      test(24) :- ejemploPropio(8,A), not(alcanzable(A,s2)).
      test(25) :- ejemploPropio(8,A), alcanzable(A,s1).
200   test(26) :- ejemploPropio(5,A), automataValido(A).
      test(27) :- ejemploPropio(6,A), automataValido(A).
      test(28) :- ejemploPropio(7,A), automataValido(A).
      test(29) :- ejemploPropio(8,A), not(automataValido(A)).
      test(30) :- ejemploPropio(9,A), not(automataValido(A)).
205   test(31) :- ejemploPropio(10,A), not(automataValido(A)).
      test(32) :- ejemploPropio(7,A), hayCiclo(A).
      test(33) :- ejemploPropio(6,A), not(hayCiclo(A)).
      test(34) :- ejemploPropio(5,A), reconoce(A,[a,b,a,b,a,a,a,b,b,b]).
      test(35) :- ejemploPropio(5,A), not(reconoce(A,[a,b,a,b,a,a,a,b,b,b,a])).
210   test(36) :- ejemploPropio(7,A), reconoce(A,[a,b,a,b]).
      test(37) :- ejemploPropio(7,A), not(reconoce(A,[a,b,a,b,a,b])).
      test(38) :- ejemploPropio(7,A), palabraMasCorta(A,[b,b]).

      %% Todos los tests
215   tests :- forall(between(1, 38, N), test(N)). %IMPORTANTE: Actualizar la cantidad total de tests para
      contemplar los que agreguen ustedes.
```