

Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Paradigmas de Lenguajes de Programación

Trabajo Práctico 2

Segundo Cuatrimestre de 2014

Grupo: Perdimos La Pelota

Apellido y Nombre	LU	E-mail
Heredia, Martin	146/11	martin.herediaf@gmail.com
Izcovich, Sabrina	550/11	sizcovich@gmail.com
Vita, Sebastián	149/11	sebastian_vita@yahoo.com.ar

%Aut matas 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)])) .
ejemplo(11, a(s1, [s2, s3], [(s1, a, s2), (s2, b, s3)])) .
ejemploAgregado(12, a(s1, [s2, s3], [])) .
ejemploAgregado(13, a(s1, [], [])) .
ejemploAgregado(14, a(s1, [], [(s1, e, s2), (s2, l, s3), (s3, /, s4), (s4, t, s5), (s5, p, s6), (s6, /, s7),
(s7, a, s8), (s8, n, s9), (s9, d, s10), (s10, a, s11), (s11, /, s12), (s12, b, s13), (s13, i, s14),
(s14, e, s15), (s15, n, s16)])) .

ejemploMalo(1, a(s1, [s2], [(s1, a, s1), (s1, b, s2), (s2, b, s2), (s2, a, s3)])) . %3 es un
%estado sin salida.
ejemploMalo(2, a(s1, [sf], [(s1, a, s1), (sf, b, sf)])) . %f no es alcanzable.
ejemploMalo(3, a(s1, [s2, s3], [(s1, a, s3), (s1, b, s3)])) . %2 no es alcanzable.
ejemploMalo(4, a(s1, [s3], [(s1, a, s3), (s2, b, s3)])) . %2 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 transici n
%repetida.
ejemploMalo(7, a(s1, [], [(s1, a, s2), (s2, b, s3)])) . %No tiene estados finales.

%%Proyectores
inicialDe(a(I, -, -), I).

finalesDe(a(-, F, -), F).

transicionesDe(a(-, -, T), T).

% transicionDesde(+L,-D)
transicionDesde((D,-,-),D).

% transicionPor(+L,-P)
transicionPor((- ,P,-),P).

% transicionHacia(+L,-H)
transicionHacia((- ,-,H),H).

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

%%Predicados pedidos.

% 1) %esDeterministico(+Automata)
esDeterministico(a(-,-,[])) .
esDeterministico(a(I,F, [X|L])) :- transicionDesde(X,D), transicionPor(X,P), transicionHacia(X,H),
not((member((D,P,H2),L), H2\=H)), esDeterministico(a(I,F,L)).

% Chequea que no existan 2 transiciones que empiecen en el mismo nodo, tengan el mismo label en la
% transicion, y vayan a nodos diferentes

```

```

% 2) %estados(+Automata, ?Estados)
estados(A, E):- var(E), estadosSinRepetidos(A,E).
estados(A, E):- nonvar(E), estadosSinRepetidos(A,M), setof(X, member(X,E),N), M=N.
%Tiene xito cuando Estados es la lista ordenada y sin repetidos de los estados del aut mata.

%concatenar (?Lista1,?Lista2,?Lista3)
concatenar([X|Xs], L2, [X|Ys]):- concatenar(Xs, L2, Ys).
concatenar([], L2, L2).
%Lista3 es el resultado de concatenar Lista1 y Lista2.

%estadosDeLasTransiciones(+Transiciones, ?Estados)
estadosDeLasTransiciones([], []).
estadosDeLasTransiciones([X|Ls],L):- estadosDeLasTransiciones(Ls,M), transicionDesde(X,D),
transicionHacia(X,H), concatenar([D],[H],L1), concatenar(L1,M,L).
%Estados, es la lista con todos los estados que pertenecen a las transiciones de la lista Transiciones.

%estadosSinRepetidos(+Automata, -Estado)
estadosSinRepetidos(A,E):- inicialDe(A,I), finalesDe(A,F), transicionesDe(A,T),
estadosDeLasTransiciones(T,T1), concatenar([I],F,E1), concatenar(E1,T1,E2),
setof(X, member(X,E2),E).
%Dado un Automata, genera una lista sin repetidos con todos sus estados.

% 3) %hayTransicion(+Automata, +EstadoInicial, +EstadoFinal)
hayTransicion(A,I,F):- transicionesDe(A,T), Transicion = (I,-,F), member(Transicion,T).
%Tiene exito si existe una transicion entre el EstadoInicial y el EstadoFinal

%esCamino(+Automata, ?EstadoInicial, ?EstadoFinal, +Camino)
esCamino(A, X, X, [X]):- estados(A,E), member(X,E).
esCamino(A, X, F, [X|[Y|Ls]]):- hayTransicion(A,X,Y), esCamino(A,Y,F,[Y|Ls]).
%Se fija si hay una transicion entre todos los estados del camino comenzando por el inicial y  
terminando en el final

% 4) el predicado anterior es o no reversible con respecto a Camino y por qu ?
% Respuesta: El predicado esCamino, de la manera en que fue definido, no es reversible con respecto a  
% Camino.
% Cuando el parametro de "camino" queda sin instanciar, y el automata A contiene ciclos, el predicado  
% se cuelga. Supongamos que el ciclo es el siguiente: s1->s2->...->sn->s1. El predicado esCamino  
% siempre va a detectar: hayTransicion(A,s1,s2), luego hayTransicion(A,s2,sn) ... hayTransicion(sn,s1)  
% y luego va a volver a detectar hayTransicion(A,s1,s2). Es por esto que se cuelga.
% Ejemplo de automata que se cuelga usando esCamino: ejemplo(4,A), esCamino(A, s1, s3, C).
% En cambio, cuando el automata A no tiene ciclos, no se cuelga, y el predicado ser a reversible con  
% respecto a Camino en ese caso

% 5) caminoDeLongitud(+Automata, +N, -Camino, -Etiquetas, ?S1, ?S2)
caminoDeLongitud(A, 1, [S2], [], S2, S2):- estados(A,E), member(S2,E).
caminoDeLongitud(A, N, C, Etiquetas, S1, S2):- N>1, estados(A,E), member(S1,E), transicionesDe(A,T),
Transicion = (S1,Etiqueta,S3),member(Transicion,T),
Nmenos1 is N-1, caminoDeLongitud(A,Nmenos1,C1,E1,S3,S2),
append([S1],C1,C), append([Etiqueta],E1,Etiquetas).
%Crea una transicion a partir de un nodo inicial. Luego, la concatena a una lista de longitud N-1  
unificando al primer nodo de dicha lista, con el nodo destino de la transicion.

% 6) alcanzable(+Automata, +Estado)
alcanzable(A,E) :- inicialDe(A,I), estados(A,K), length(K,L), between(1,L,N),
caminoDeLongitud(A,N,-,I,E), !.
%La idea es que a partir del estado inicial se verifique si existe un camino a E de longitud N, con  
% 1<=N<cantidadDeEstados

% 7) automataValido(+Automata)
automataValido(A) :- estadosNoFinalesSonSalientes(A), todosAlcanzables(A), hayEstadoFinal(A),
noHayEstadosFinalesRepetidos(A), noHayTransicionesRepetidas(A).

```

%Eval a todas los predicados presentados con A.

%estadosNoFinalesSonSalientes(+A)

estadosNoFinalesSonSalientes(A) :- estados(A,E), finalesDe(A,F), subtract(E,F,EstadosNoFinales),
forall(member(X,EstadosNoFinales), caminoDeLongitud(A,2,-,-,X,-)), !.

%Verifica que exista al menos un camino desde cada estado, a menos que el mismo sea uno final

%todosAlcanzables(+A)

todosAlcanzables(A) :- estados(A,E), inicialDe(A,I), subtract(E,[I],M), forall(member(X,M),
alcanzable(A,X)).

%Verifica que todos los estados sean alcanzables desde el inicial, menos el inicial.

%hayEstadoFinal(+A)

hayEstadoFinal(A) :- finalesDe(A,F), not(F = []).

%Dados los estados finales de un aut mata, se fija que la lista no sea vac a.

%noHayEstadosFinalesRepetidos(+A)

noHayEstadosFinalesRepetidos(A) :- finalesDe(A,F), borrarDuplicados(F,X), X = F.

%Compara la lista de estados finales con ella misma sin sus repetidos.

%borrarDuplicados(+L, -T):

borrarDuplicados([],[]).

borrarDuplicados([X|Xs], F) :- member(X, Xs), borrarDuplicados(Xs, F).

borrarDuplicados([X|Xs], [X|F]) :- not(member(X, Xs)), borrarDuplicados(Xs, F).

%Elimina los duplicados de cualquier lista.

%noHayTransicionesRepetidas(+A)

noHayTransicionesRepetidas(A) :- transicionesDe(A,T), borrarDuplicados(T,X), X = T.

%Compara la lista de transiciones con ella misma sin repetidos.

%— NOTA: De ac en adelante se asume que los aut matas son v lidos.

% 8) hayCiclo(+Automata)

hayCiclo(A) :- estados(A,E), length(E,L), member(X,E), R is L+1, between(2,R,N),
caminoDeLongitud(A,N,-,-, X, X), !.

*%La idea es que a partir de cada estado de A se fije si existe un camino de un estado a s
% mismo de longitud N, con 1<N<cantidadDeEstados+1*

% 9) reconoce(+Automata, ?Palabra)

reconoce(A, P) :- nonvar(P), length(P,Len), CantEstados is Len+1, inicialDe(A,Init),
finalesDe(A,Finales), caminoDeLongitud(A, CantEstados, -, P, Init, Fin),
member(Fin, Finales).

reconoce(A, P) :- var(P), not(hayCiclo(A)), estados(A,Estados), length(Estados,Len),
between(1,Len,N), inicialDe(A,Init), finalesDe(A,Finales),
caminoDeLongitud(A, N, -, P, Init, Fin), member(Fin, Finales).

reconoce(A, P) :- var(P), hayCiclo(A), desde(1,N), inicialDe(A,Init), finalesDe(A,Finales),
caminoDeLongitud(A, N, -, P, Init, Fin), member(Fin, Finales).

%Aca se utiliza la tecnica de Generate & Test.

%Se separan en 2 casos:

% 1) P esta instanciada o contiene variables libres:

*% En este caso chequeo si P es una de las posibles listas de Etiquetas de longitud |P|+1
% que me genera el automata*

% 2) P no esta instanciada:

% Aca es necesario volver a separar en 2 casos.

% A) El automata tiene ciclos:

*% En este caso genero todas las palabras (listas de Etiquetas) que reconoce el automata.
% Por cada numero natural N, genero las lista de etiquetas de longitud N y chequeo si es
% una palabra valida*

% B) El automata no tiene ciclos:

*% Es analogo al caso A), con la diferencia que se busca con N desde 1 hasta la cantidad de estados
% del automata (Esto es porque la longitud maxima posible de estados de una palabra reconocida por
% un automata sin ciclos, es justamente la cantidad de estados). Si se buscara para todo numero*

```
%      natural N, entonces este predicado se colgaria una vez que ya detecto todas las palabras que se
%      reconocen.
```

```
%10) %PalabraMasCorta(+Automata, ?Palabra)
```

```
palabraMasCorta(A, P) :- minimaLongitudAceptada(A, Len), inicialDe(A, Init), finalesDe(A, Finales),
                           caminoDeLongitud(A, Len, _, P, Init, Fin), member(Fin, Finales).
```

```
% Chequeo cual es la minima longitud de palabra que acepta el automata (con el predicado anterior),
% y chequeo cuales son las listas de etiquetas P que son reconocidas con esa longitud.
```

```
%minimaLongitudAceptada(+A,-L)
```

```
minimaLongitudAceptada(A, N) :- inicialDe(A, Init), finalesDe(A, Finales), desde(1, N),
                                caminoDeLongitud(A, N, _, _, Init, Fin), member(Fin, Finales), !.
```

```
% Como aca ya puedo suponer que el automata es valido, entonces se que al menos reconoce 1 palabra.
```

```
% Es por esto que este predicado no se va a colgar (si se cumple esa precondition)
```

```
% Se hace una busqueda, empezando con N=1 (y aumentando de a 1) hasta encontrar un camino de
```

```
% longitud N, y que la palabra que genera sea reconocida por el automata.
```

```
% Una vez encontrada esta palabra, se corta el arbol de busqueda
```

```
%_____
```

```
% Tests _____
```

```
%_____
```

```
% Algunos tests de ejemplo. Deben agregar los suyos.
```

```
test(1) :- forall(ejemplo(_, A), automataValido(A)).
test(2) :- not((ejemploMalo(_, A), automataValido(A))).
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), []).
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]]).
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))).
test(16) :- ejemplo(5, A), not(alcanzable(A, s5)).
test(17) :- ejemplo(5, A), alcanzable(A, s3).
test(18) :- ejemploMalo(3, A), not(alcanzable(A, s2)).
test(19) :- ejemploMalo(3, A), not(hayCiclo(A)).
test(20) :- ejemploMalo(5, A), not(hayCiclo(A)).
test(21) :- ejemploMalo(2, A), not(alcanzable(A, sf)).
test(22) :- ejemploMalo(5, A), estados(A, [s1, s2, s3]).
test(23) :- ejemploMalo(5, A), estados(A, [s1, s3, s2, s3]).
test(24) :- ejemploMalo(5, A), estados(A, E), E = [s1, s2, s3].
test(25) :- ejemploMalo(7, A), estados(A, [s1, s2, s3]).
test(26) :- ejemploAgregado(12, A), estados(A, [s1, s2, s3]).
test(27) :- ejemploAgregado(13, A), estados(A, [s1]).
test(28) :- ejemplo(5, A), esCamino(A, s1, s3, [s1, s2, s3]).
test(29) :- ejemplo(5, A), not(esCamino(A, s3, s1, [s1, s2, s3])).
test(30) :- ejemplo(5, A), not(esCamino(A, s3, s1, [s1, s2, s3])).
test(31) :- ejemplo(5, A), not(esCamino(A, s1, s5, [s1, s2, s3])).
test(32) :- ejemplo(5, A), esCamino(A, S1, S3, [s1, s2, s3]), S1=s1, S3=s3.
test(33) :- ejemplo(5, A), esCamino(A, s1, S3, [s1, s2, s3]), S3=s3.
test(34) :- ejemplo(7, A), caminoDeLongitud(A, 1, C, E, S1, S2), member(C, [[s1], [s2], [s3]]),
    E=[], S1=S2, member(S1, [s1, s2, s3]).
test(35) :- ejemplo(7, A), caminoDeLongitud(A, 2, C, E, s1, s3), C=[s1, s3], E=[a].
```

```

test(36) :- ejemplo(7, A), not(caminoDeLongitud(A, 2, -, -, -, s1)).
test(37) :- ejemploAgregado(14,A), caminoDeLongitud(A,16,C,E,s1,s16),
           C=[s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16],
           E=[e,l,/,t,p,/,a,n,d,a,/,b,i,e,n].
test(38) :- ejemplo(5,A), esDeterministico(A).
test(39) :- ejemplo(4,A), not(esDeterministico(A)).
test(40) :- ejemplo(7,A), palabraMasCorta(A,[a,b]).
test(41) :- ejemplo(4,A), findall(L, palabraMasCorta(A,L), Lista), length(Lista, 2),
           sort(Lista, [[a], [b]]).
tests :- forall(between(1, 41, N), test(N)). ¡IMPORTANTE: Actualizar la cantidad total
de tests para contemplar los que agreguen ustedes.

```