

3. Parte 3

3.1.

Al ejecutar código que se nos entrega,

```
amistad(juan,maria).
amistad(maria,elena).
amistad(elena,fernando).
```

```
usuario(X) :- amistad(X,Y).
usuario(Y) :- amistad(X,Y).
```

```
cadena(X,Y,Largo) :- amistad(X,Y), Largo=1. %Caso base
cadena(X,Y,Largo) :- cadena(X,Z,L1), cadena(Z,Y,L2), Largo=L1+L2.
```

Vemos que se ejecuta correctamente, generado un modelo de la siguiente manera:

```
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main)> clingo aux.lp
clingo version 5.7.1
Reading from aux.lp
Solving...
Answer: 1
amistad(juan,maria) amistad(maria,elena) amistad(elena,fernando) usuario(maria) usuario(elena) usuario(fernando) usuario(juan) cadena(juan,maria,1) cadena(maria,elena,1)
cadena(elena,fernando,1) cadena(maria,fernando,2) cadena(juan,elena,2) cadena(juan,fernando,3)
SATISFIABLE

Models      : 1
Calls       : 1
Time        : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 0.002s
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [30]>
```

Sin embargo, si agregamos `amistad(fernando,juan)` al código, vemos que se genera un ciclo, donde tenemos que interrumpir el código ya que no llega nunca a una solución.

```
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [30]> clingo aux.lp
clingo version 5.7.1
Reading from aux.lp
^C*** Info : (clingo): INTERRUPTED by signal!
UNKNOWN

INTERRUPTED : 1
Models      : 0+
Calls       : 1
Time        : 20.124s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 20.106s
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [1]>
```

De esta forma, tenemos que buscar una forma de plantear el código para que evitemos entrar en un ciclo y por resolver el problema.

Con todo esto en marcha, ahora ahora podemos considerar replantear el problema, eliminando las últimas 2 líneas de código, que son las que traen problemas.

Primero agregaremos una línea para asegurar que no importe la posición en cuanto a la amistad.

```
amistad(X,Y) :- amistad(Y,X).
```

Luego utilizando count reformularemos las 2 líneas que traen problemas. De esta forma el código actualizado y funcional queda de la siguiente manera:

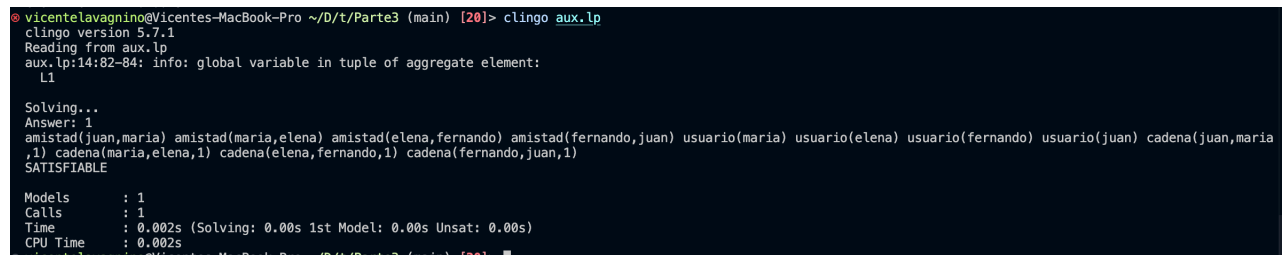
```
amistad(juan,maria).
amistad(maria,elena).
amistad(elena,fernando).
amistad(fernando,juan). % Caso que produce un ciclo

usuario(X) :- amistad(X,Y).
usuario(Y) :- amistad(X,Y).

% cadena(X,Y,Largo) :- amistad(X,Y), Largo=1. % NO USADO
% cadena(X,Y,Largo) :- cadena(X,Z,L1), amistad(Z,Y,L2), Largo=L1+L2. % NO
⇨ USADO

% uso de #count para evitar ciclos en cadena en caso base
cadena(X, Y, Largo) :- amistad(X, Y), Largo = 1.
cadena(X, Y, Largo) :- cadena(X, Z, L1), amistad(Z, Y), Largo = L1 + 1, #
⇨ count { L1 : cadena(X, N, L1) } = Largo, X != Y, Z != X, Z != Y.
```

En particular lo que se hizo fue replicar la estructura existente en la primera línea de `cadena(X,Y,Largo)` y luego modificar la segunda que genera el ciclo para ver, en los casos que no sean iguales las variables, que no se cuantifique recursivamente los casos de unión.



```
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [20]> clingo aux.lp
clingo version 5.7.1
Reading from aux.lp
aux.lp:14:82-84: info: global variable in tuple of aggregate element:
  L1

Solving...
Answer: 1
amistad(juan,maria) amistad(maria,elena) amistad(elena,fernando) amistad(fernando,juan) usuario(maria) usuario(elena) usuario(fernando) usuario(juan) cadena(juan,maria,1)
cadena(maria,elena,1) cadena(elena,fernando,1) cadena(fernando,juan,1)
SATISFIABLE

Models      : 1
Calls       : 1
Time        : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 0.002s
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [20]>
```