2 Parte 2

2.1 Respuesta 1

En pocas palabras, lo que pasa al agregar un ciclo en el grafo de nuestro programa, este se queda encerrado en un *loop* infinito. Esto ocurre porque, según la definición algorítmica de un modelo, la ejecución sólo terminará de evaluar el set de reglas, cuando ocurra una iteración en la cual no cambie el modelo actual M. Si hay un ciclo en el programa, en cada iteración se encontrará un camino de mayor longitud. Por lo tanto, crecerá el conjunto M, y se volverá a iterar con el set actual, ocurriendo lo mismo que en la iteración anterior.

Ejemplificando con un grafo básico, supongamos que existen los arcos `arco(a,b)`, y `arco(b,a)`, con `a` y `b` nodos. En la primera iteración, se descubren `camino(a,b,1)` y `camino(b,a,1)`. Sin embargo, ahora nos podemos dar cuenta que estos átomos cumplen la regla `camino(X,Y,Largo) :- camino(X,Z,L1), camino(Z,Y,L2), Largo=L1+L2.`, por lo que se añadirían los nuevos átomos `camino(a,a,2)` y `camino(b,b,2)`. Sin embargo, vemos que estos también cumplen la regla anterior, y posteriormente se generarían caminos como `camino(a,a,4)`, `camino(b,b,4)`, `camino(a,b,3)` y `camino(b,a,3)`. Ahí nos podemos dar cuenta que la ejecución no se detendrá, debido a que M sigue creciendo con cada iteración.

Para solucionar el problema anterior para grafos con ciclos, y valiéndose del operador `count`, sólo bastaría con agregar una cota superior al largo máximo de camino. Si hay un largo de camino mayor a la cantidad de arcos totales del grafo, significa que este contiene un ciclo. Por lo tanto, si modificamos la última regla agregando la condición `Largo <= #count{A,B : arco(A,B)}` (notar el menor o igual), el programa dejará de ejecutarse infinitamente, y al mismo tiempo se encontrarán todos los caminos sin ciclos.

2.2 Respuesta 4