9-7-2008

1. Carga de las topologías

- He escrito una primera versión del proceso de carga de los ficheros GML, basándome en la información de la web `http://www.infosun.fim.uni-passau.de/Graphlet/GML/`. El proceso se divide en dos etapas: conversión de una cadena de texto a un árbol sintáctico, y extracción del grafo del árbol sintáctico. He llamado a estas funciones *string2gml* y *gml2graph*. También he creado dos funciones, llamadas *gml2string* y *graph2gml*, que realizan los pasos contrarios de cara a guardar topologías en disco.

- Para guardar los datos de las etapas anteriores, se definen dos tipos de datos, *GML* y *Grafo*:

```
data GML = VI Int
         | VR Double
         | VS String
         | VM (Map Clave [GML])

data Grafo = Grafo {
        nodos :: Map Id Nodo,
        arcos :: [Arco],
        atributos :: Atributos
        }
```

  Un árbol sintáctico puede ser un número entero, un real, una cadena de texto, o una lista de pares clave-valor. Por motivos de eficiencia se usa el tipo de datos *Map*. Los grafos son tuplas de tres elementos: el conjunto de nodos, indexados por su identificador; la lista de arcos, que son a su vez tuplas con el origen, el destino y los atributos del arco; y el resto de atributos del grafo. Estas estructuras no son definitivas, es bastante probable que tenga que cambiarlas.

- He procurado contemplar el mayor número de condiciones de error posibles:
  - Que el fichero de entrada no contenga grafos.
  - Que un grafo no contenga nodos o arcos.
  - Que un nodo no tenga identificador, o sea incorrecto.
  - Que un arco no tenga origen o destino.
  - Que el tipo de datos de algún campo básico sea incorrecto (por ejemplo, que el identificador no sea un número entero).

- Aún así, hay varios casos que no se comprueban, entre ellos los siguientes:
  - Que el identificador de cada objeto sea único.
  - Que los arcos unan nodos existentes.

- Para probar que las funciones de conversión son correctas, he decidido implementar una serie de pruebas con QuickCheck. Las pruebas consisten esencialmente en generar un grafo aleatorio, pasarlo a texto, volver a convertirlo a un grafo, y compararlo con el original. Si no son idénticos, es que algo ha ido mal.

- También he estado analizando la manera de obtener la lista de escenarios lineales a partir del grafo. Creo que el algoritmo más adecuado sería una búsqueda en profundidad, ya que tiene menor complejidad espacial que la búsqueda en anchura. Se me ocurren dos maneras de hacer la búsqueda: escribir una función que recorra el grafo directamente, o bien desenrollar el grafo en un árbol y recorrer el árbol. Creo que la segunda opción es más elegante, ya que facilitaría la modificación del código si en el futuro se quiere añadir alguna heurística (por ejemplo), pero tengo que estudiarlo más a fondo.

2. Estadísticas

- Como ya se habló, para representar los arcos de un grafo hay dos opciones: usar una matriz de adyacencia o una lista de enlaces. Para decidir cual de las dos opciones es la más adecuada, he encontrado una medida llamada *densidad* del grafo, que para grafos no dirigidos se define como:

$$D = \frac{2\,|A|}{|V|\,(|V|-1)}$$

Donde A es el número de arcos y V el número de vértices. La densidad mínima es 0, y la máxima es 1 para un grafo completamente conexo. He modificado el programa para que midiera la densidad de cada topología, obteniendo los siguientes resultados:

Fichero	Nodos	Enlaces	Enlaces por nodo (mínimo, máximo, media)			Densidad
abilene.gml	N/A	N/A	N/A	N/A	N/A	N/A
abovenet.6461.r0.cch.gml	368	966	1	39	5,25	0,0008
att.7018.r0.cch.gml	731	2253	1	55	6,16	0,0003
att.gml	154	367	2	58	4,77	0,0023
belnet.gml	N/A	N/A	N/A	N/A	N/A	N/A
colt.gml	43	107	2	15	4,98	0,0076
cw.gml	33	107	2	13	6,48	0,0058
dfn.gml	30	97	1	20	6,47	0,0064
ebone.1755.r0.cch.gml	161	307	1	17	3,81	0,0034
exodus.3967.r0.cch.gml	246	540	1	18	4,39	0,0017
geant-20041125.gml	23	37	2	6	3,22	0,0345
geant.gml	23	37	2	6	3,22	0,0345
level3.3356.r0.cch.gml	625	5298	1	169	16,95	0,0000
simple2.gml	4	4	2	2	2	0,6667
simple.gml	3	3	2	2	2	1,0000
sprint.1239.r0.cch.gml	549	1593	1	51	5,8	0,0004
switch.gml	31	80	2	19	5,16	0,0098
telekom.gml	10	34	4	14	6,8	0,0178
tiscali.3257.r0.cch.gml	248	405	1	31	3,27	0,0030
verrio.2914.r0.cch.gml	911	2217	1	46	4,87	0,0004
vsnl.4755.r0.cch.gml	12	12	1	4	2	0,1818

- Como se puede ver, excepto en casos muy excepcionales el grado de conectividad de los nodos es bastante bajo. Por tanto, se deduce que la mejor opción es usar listas de nodos adyacentes.