

Correspondencia de grafos RDF

Técnicas de bases de datos en la WEB - CC71K

Profs.: Claudio Gutiérrez - Carlos Hurtado

Alumno: Marcelo Mendoza

Departamento de Ciencias de la Computación

Universidad de Chile

mmendoza@dcc.uchile.cl

October 16, 2002

Abstract En este reporte se presenta un resumen de las principales ideas expuestas en el paper *Matching RDF Graphs* de Jeremy Carroll [1]. El standard RDF tiene como modelo de datos el grafo generado a partir de las triplas declaradas. Entonces es necesario poder comparar dos grafos RDF para determinar si describen lo mismo. Para ello se aplican algoritmos clásicos de isomorfismos de grafos mostrándose que son suficientes para comparar efectivamente grafos RDF.

1 Introducción

Uno de los objetivos relevantes del RDF-concepts [4] fue definir con precisión la equivalencia entre grafos RDF. Debido a que la función de etiquetación de nodos es parcial, esto es existen nodos sin etiquetas llamados nodos blancos, la equivalencia no es trivial debido a que es necesario establecer si ambos grafos son isomórficos.

En [1] se aborda este problema mostrando que es posible determinar que dos grafos son isomórficos en el sentido RDF a partir de los algoritmos clásicos de isomorfismos en grafos ([2] y [3]). Además se presentan mejoras de eficiencia de los algoritmos usando las versiones de clasificación de vértices iterativas [3]. Finalmente se muestran otras variaciones basadas en el hashCode de las etiquetas de los nodos.

2 Descripción del problema

La estructura asociada a una sentencia RDF es un grafo dirigido etiquetado. Cada expresión corresponde a dos nodos y un arco. Se define un nodo sujeto, un arco que declara una propiedad y un nodo objeto referenciado por el sujeto con la propiedad declarada en el arco. Si se consideran todas las expresiones RDF de un archivo, la estructura asociada es el grafo resultante a partir de la unión de los subgrafos asociados a cada declaración.

Los arcos de un grafo RDF son etiquetados con URIs. Los nodos de un grafo RDF pueden ser etiquetados con URIs, literales o nada (estos últimos se conocen como nodos blancos). Para comparar dos grafos RDF G_1, G_2 es necesario determinar si describen las mismas declaraciones. Entonces basta con determinar una biyección B entre los vértices V_1, V_2 de ambos grafos de manera que $v_{2,j} = B(v_{1,i})$ si y solo si las URIs con que han sido etiquetados los nodos coinciden caracter a caracter.

Debido a que la función de etiquetamiento de nodos es parcial, la comparación de dos grafos RDF debe ser determinada asociando una biyección entre todos los nodos etiquetados y buscando una correspondencia entre los nodos blancos. El problema está en determinar la asociación entre nodos blancos debido a que no podemos comparar etiquetas. Es necesario establecer, entonces, si los grafos son isomórficos en el sentido RDF, para lo que basta chequear que los arcos que unen los nodos bajo la biyección son los mismos [4].

3 Algoritmos de clasificación de vértices iterativos

En [1] se aborda el problema mostrando cuatro algoritmos. El primero de ellos es el de fuerza bruta. Escencialmente verifica todas las asignaciones posibles entre los nodos verificando que los arcos correspondan. En el peor caso (no son isomórficos) las verifica todas, lo que tiene costo $O(n!)$.

El segundo algoritmo presentado usa clasificación de nodos. La idea es particionar el grafo en clases considerando un invariante (por ejemplo el grado de cada nodo). A partir de esta clasificación se asocian clases de acuerdo al invariante y luego se verifican las asignaciones dentro de cada clase. Esta mejora reduce la cantidad de asignaciones a verificar a $n_c!$ por clase, donde n_c es la cantidad de nodos en la clase c . En este algoritmo la clasificación se hace sólo una vez.

El tercer algoritmo está basado en la clasificación de nodos iterativos (o refinamiento de la partición) de [3]. La idea es usar la información que entrega la clasificación actual para reclasificar los nodos en clases de acuerdo a los invariantes entre nodos de la misma clase y entre las clases. Esto permite fijar una clase

(la que tenga menos elementos) y verificar las asignaciones en ella hasta satisfacer el chequeo de arcos. Luego se fija la biyección de esta clase y se refina la partición en los nodos restantes. El algoritmo decide si es isomórfico cuando las clases resultantes tienen cardinalidad uno. Al igual que el algoritmo anterior, usa como invariante la adyacencia de nodos.

Finalmente en [1] se propone un cuarto algoritmo basado en refinamiento de particiones (al igual que el algoritmo 3), pero con invariante basado en el hashCode (el de Java) ponderado de cada tripla en la que participa un nodo blanco. Este invariante es más simple de calcular que la adyacencia del nodo debido a que basta usar el hashCode de Java (asociado a string).

4 Presentación oral

En la presentación oral del paper se mostrará el problema descrito en este reporte. Luego se mostrarán los cuatro algoritmos descritos. Además se ilustrará su uso con un ejemplo basado en RDF. Finalmente se expondrán conclusiones.

References

- [1] Carroll, J., "*Matching RDF Graphs*", Tech. Report HPL-2001-293 Hewlett-Packard Laboratories Bristol, UK, <http://www.hpl.hp.com/techreports/2001/HPL-2001-293.html>, (2001).
- [2] McKay, B., "*Practical Graph Isomorphism*", *Congressus Numerantium* 30, pp.45 - 87, <http://cs.anu.edu.au/bdm/papers/pgi.pdf>, (1981).
- [3] Read, R. & Corneil, D., "*Graph Isomorphism Disease*", *Journal Graph Theory* 1, pp. 339 - 363, (1977).
- [4] W3C Working Draft, "*RDF: Concepts and Abstract Data Model*", <http://www.w3.org/TR/rdf-concepts/>, (Aug 2002).