Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 59 (2006) 652–680
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

A graph grammars based framework for querying graph-like data ☆

Sergio Flesca [a], Filippo Furfaro [a,*], Sergio Greco [a,b]

[a] *Dipto Elettronica Informatica e ISI-CNR, Universita della Calabria, Sistemistica, Via P Rucci 41 C, 87030 Rende-Cosenza, Italy*
[b] *ICAR-CNR, 87030 Rende, Italy*

Received 1 November 2005; accepted 1 November 2005
Available online 1 December 2005

Abstract

The widespread use of graph-based models for representing data collections (e.g. object-oriented data, XML data, etc.) has stimulated the database research community to investigate the problem of defining declarative languages for querying graph-like databases. In this paper, a new framework for querying graph-like data based on graph grammars is proposed. The new paradigm allows us to verify structural properties of graphs and to extract sub-graphs. More specifically, a new form of query (namely *graph query*) is proposed, consisting in a particular graph grammar which defines a class of graphs to be matched on the graph representing the database. Thus, differently from path queries, the answer of a graph query is not just a set of nodes, but a subgraph, extracted from the input graph, which satisfies the structural properties defined by the graph grammar. Expressiveness and complexity of different forms of graph queries are discussed, and some practical applications are shown.

Keywords: Query language; Semistructured data; Graph grammars

1. Introduction

The widespread use of graph based models for representing data collections (e.g. object-oriented data, XML data, etc.) has stimulated the database research community to investigate the problem of defining declarative languages for querying graph-like databases [1,3,17,18,22]. Recently, several languages and prototypes have been proposed for searching both generic graph-like data and specific types of graph data such as XML. The most widely used mechanism for extracting information from graphs is that of *path queries*, due to its simplicity and declarative nature. Basically path queries are navigational queries expressed by means of regular expressions denoting paths in the graph [2,4,6,12,19,23]. A path query of the form $\langle \Gamma, r \rangle$, where Γ is a set of node labels and r a regular expression, defines the query "*find all the nodes reachable from a node whose label*

☆ Work partially supported by a MURST grant under the project "D2I".
* Corresponding author.
E-mail addresses: flesca@deis.unical.it (S. Flesca), furfaro@si.deis.unical.it (F. Furfaro), greco@deis.unical.it (S. Greco).

0169-023X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2005.11.001

belongs to Γ through paths spelling a string of the language defined by the regular expression r''. However, this kind of navigational query is not completely satisfactory since in many cases we would like to express queries verifying whether the graph has a given structure (e.g. a tree or a chain), or we need to extract from the input graph not simply a set of nodes, but a complex subgraph satisfying a given property [7,20,24].

Example 1. Consider the labeled rooted graph shown in Fig. 1, where some pieces of information about a collection of books are represented.

In the above graph, book details (name and surname of authors, title, publisher) are represented by labels associated with leaf nodes, whereas edge labels describe the type of information contained in the descending ''subtree''. For instance, any sub-tree identified by an edge with label ''*written_by*'' contains information about the book authors.

Assume now that we want to extract the sub-trees corresponding to the books written by *Ullman*. We could use path queries to extract separately all the pieces of the available information about the desired book, but we cannot use path queries to extract the desired information preserving its structure. That is, we could extract the titles of all Ullman's books (''*A First course in Database Systems*'', ''*Principles of Databases and Knowledge Systems*'') by means of the path query $\langle\{$''*Ullman*''$\}$, *surname.author.written_by.title*$\rangle$, and extract their publishers by means of the path query $\langle\{$''*Ullman*''$\}$, *surname.author.written_by.pub*$\rangle$. But the information returned by the former query is disjoint from the information returned by the latter one. As path queries return sets of nodes, we are not able to reconstruct the correspondence between titles and publishers. It is worth noting that if the rooted graph is replaced by an acyclic rooted digraph (tree), even the single pieces of information cannot be extracted, as arcs cannot be navigated from the destination node to the source node.

In order to overcome the limited expressive power of path queries, without completely renouncing to their simplicity and declarative nature, some languages (such as XQuery [27]) embed the path query mechanism into a more general and more expressive query paradigm. However this result in procedural query languages making query specification rather complex. Our proposal consists in a framework for defining graph patterns which can be used to extract information from the input graph. A pattern is a graph which defines the shape (or, more generally, structural properties) and the content of the subgraph to be extracted from the input graph. An example of pattern (called *query graph*) is shown on the left-hand side of Fig. 2. Such a query graph defines a class of graphs to be matched on the input data graph, where labels associated with nodes may be specific (e.g. ''`Ullman`'') or generic (e.g. `$n`). The matching between any graph in the class defined by the query graph with the input data graph permits us to extract portions of the graph of Fig. 1 corresponding to Ullman's books. Node labels whose first symbol is $ are used to define variables: the node with label `$u` ''extracts'' the name of Ullman, nodes with label `$n`, `$s` are associated with the name and the surname of possible Ullman's co-authors, whereas nodes with label `$t`, `$p` are associated, respectively, to the title and the publisher of each Ullman's book. The two sub-graphs extracted by means of this pattern, containing the available information about the two books written by Ullman, are shown on the right-hand side of Fig. 2.

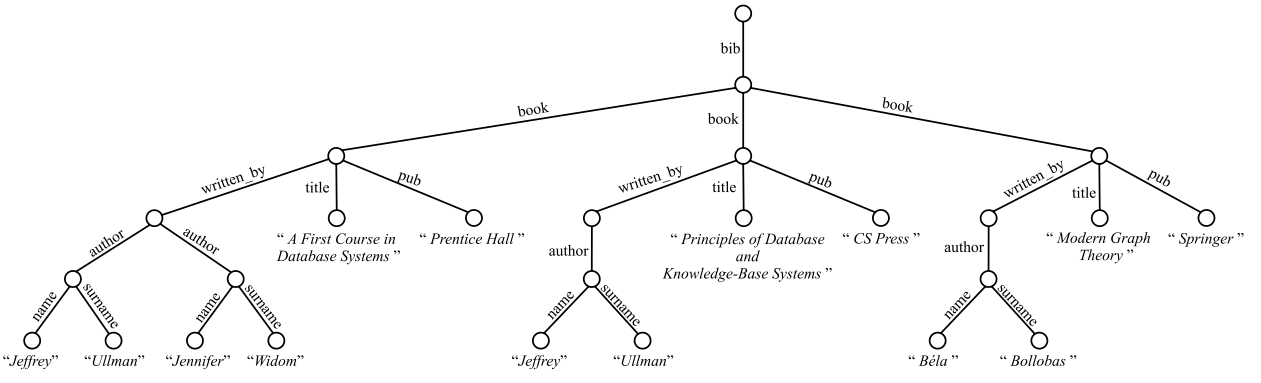

Fig. 1. A tree containing some information about books.