Tree Matching Techniques for Textual Tree Databases

Holger Meuss Klaus U. Schulz
Center for Information and Language Processing (CIS)
University of Munich
Oettingenstr. 67 80538 Munich, Germany
e-mail: {meuss,schulz}@cis.uni-muenchen.de

September 1, 1999

Abstract

The use of markup languages like SGML, HTML, or XML for encoding the structure of documents, as well the recent interest in large-scale annotated corpora and similar repositories in computational linguistics has lead to many databases where entries are adequately described as trees where leaves have some form of textual contents. In order to access the information available in such databases, querying formalisms are interesting that offer the possibility to refer both to textual content and logical structure. In the intersection of information retrieval and database theory a number of such formalisms have been developed recently. Among those formalisms that allow for structural information in answers, Kilpeläinen's Tree Matching formalisms is one of the most natural and expressive ones. In this approach, queries are formalized as trees, answers are formalized as suitable homomorphisms to the database. One problem with this approach is the following: a simple enumeration of all answers will often suffer from the effect that many mappings/subtrees have common subparts. From a theoretical point of view this may lead to an exponential time complexity of the presentation of all answers. In this talk we show how to circumvent this problem as well as a number of other deficiencies of the formalism.

Concentrating on the language of so-called tree queries—a variant and extension of Kilpeläinen's Tree Matching formalism—we introduce the notion of a "complete answer aggregate" for a given query. A complete answer aggregate offers a compact view of the set of all answers that seems attractive for practical use in systems. Since complete answer aggregates use an exhaustive structure sharing mechanism their maximal size is of order $\mathcal{O}(d \cdot h \cdot q)$ where d (q) is the size of the database (query) and h is the maximal depth of a path of the database. We give an algorithm that computes a complete answer aggregate for a given tree query in time $\mathcal{O}(d \cdot log(d) \cdot h \cdot q)$. For the sublanguage of so-called rigid tree queries, as well as for so-called "non-recursive" databases, an improved bound of $\mathcal{O}(d \cdot log(d) \cdot q)$ is obtained. The algorithm is based on a specific index structure for textual contents that supports practical efficiency.

Keywords: Tree databases, tree matching, information retrieval, structured documents, query languages, logic.