Dictionary-Based Tree Compression

Sebastian Maneth[*1]

1 **NICTA and University of New South Wales**
Sydney, Australia
sebastian.maneth@nicta.com.au

Abstract

Trees are a ubiquitous data structure in computer science. LISP, for instance, was designed to manipulate nested lists, that is, ordered unranked trees. Already at that time, DAGs were used to detect common subexpression, a process known as "hash consing." In a DAG every distinct subtree is represented only once (but can be referenced many times) and hence it constitutes a dictionary-based compression method for ordered trees. In our compression scenario we distinguish two kinds of ordered trees: binary and unranked. The latter appear naturally as representation of XML document structures. We survey these dictionary-based compression methods for ordered trees:

1. DAGs
2. hybrid DAGs
3. straight-line context-free tree grammars ("SLT grammars").

We compare the minimal DAG of an unranked tree with the minimal DAG of its binary tree encoding. The latter is obtained by identifying first children of the unranked tree with left children of the binary tree, and next-siblings with the right children. For XML document trees, unranked DAGs are usually smaller than encoded binary DAGs. We show that this holds for arbitrary unranked trees, *on average*. We also present the "hybrid DAG"; its size lower-bounds those of the binary and unranked DAGs.

Finding a smallest SLT grammar for a given tree is NP-complete. We discuss two linear-time approximation algorithms: BPLEX and TreeRePair. For typical XML document trees, TreeRePair produces SLT grammars that are only one fourth of the size of the minimal DAG, and which contain approximately 3% of the edges of the original tree. As far as we know, this gives rise to the smallest existing pointer-based tree representation.

We show that some basic algorithms can be computed directly on the compressed trees, without prior decompression. Examples include the execution of different kinds of tree automata, and the real-time traversal of the original tree. It is even possible to evaluate simple XPath queries directly on the SLT grammars, using deterministic node-selecting tree automata. In this way, impressive speed-ups are achieved over existing XPath evaluators, while at the same time the memory requirement is slashed to only a few percent. For more complex XPath queries that require nondeterministic node-selecting tree automata, efficient evaluation over SLT grammars remains a difficult challenge.

1998 ACM Subject Classification E.4 Data Compaction and Compression

Keywords and phrases Tree grammars, tree automata, straight-line programs

Digital Object Identifier 10.4230/LIPIcs.RTA.2012.5

Category Invited Talk

* The author's present address is: Institut für Informatik, Universität Leipzig, Johannisgasse 26, D-04103 Leipzig, Germany

© Sebastian Maneth;
licensed under Creative Commons License NC-ND
23rd International Conference on Rewriting Techniques and Applications (RTA'12).
Editor: A. Tiwari; pp. 5–5
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany