08261 Abstracts Collection
Structure-Based Compression of Complex Massive Data
— Dagstuhl Seminar —

Stefan Böttcher[1], Markus Lohrey[2], Sebastian Maneth[3] and Wojciech Rytter[4]

[1] Universität Paderborn, D
[2] Universität Stuttgart, D
lohrey@informatik.uni-leipzig.de
[3] Univ. of New South Wales, AUS
sebastian.maneth@nicta.com.au
[4] University of Warsaw, PL

Abstract. From June 22, 2008 to June 27, 2008 the Dagstuhl Seminar 08261 "Structure-Based Compression of Complex Massive Data" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Data compression, algorithms for compressed strings and trees, XML-compression

08261 Executive Summary – Structure-Based Compression of Complex Massive Data

From 22nd June to 27th of June 2008, the Dagstuhl Seminar "08261 Structure-Based Compression of Complex Massive Data" took place at the Conference and Research Center (IBFI) in Dagstuhl. 22 researchers with interests in theory and application of compression and computation on compressed structures met to present their current work and to discuss future directions.

Keywords: Compression, Succinct Data Structure, Pattern Matching, Text Search, XML Query

Joint work of: Böttcher, Stefan ; Lohrey, Markus ; Maneth, Sebastian ; Rytter, Wojciech

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1681

From Rewriting to Compression to Querying to Rewriting

Siva Anantharaman (Université d'Orléans, FR)

We present briefly a few issues that link Rewriting to Compression and Querying.

Compression vs Queryability - A Case Study

Siva Anantharaman (Université d'Orléans, FR)

Some compromise on compression is known to be necessary, if the relative positions of the information stored by semi-structured documents are to remain accessible under queries. With this in view, we compare, on an example, the 'query-friendliness' of XML documents, when compressed into straightline tree grammars which are either regular or context-free. The queries considered are in a limited fragment of XPath, corresponding to a type of patterns; each such query defines naturally a non-deterministic, bottom-up 'query automaton' that runs just as well on a tree as on its compressed dag.

Keywords: XML, XPath, Compression, Tree Grammars, Patterns, Query Automata

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1676

The XQueC Project: Compressing and Querying XML

Angela Bonifati (ICAR - CNR - Rende, FR)

We outline in this paper the main contributions of the XQueC project. XQueC, namely XQuery processor and Compressor, is the first compression tool to seamlessly allow XQuery queries in the compressed domain. It includes a set of data structures, that basically shred the XML document into suitable chunks linked to each other, thus disagreeing with the "homomorphic" principle so far adopted in previous XML compressors. According to this principle, the compressed document is homomorphic to the original document. Moreover, in order to avoid the time consumption due to compressing and decompressing intermediate query results, XQueC applies "lazy" decompression by issuing the queries directly in the compressed domain.

Keywords: XML compression, Data structures, XQuery querying

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1691

Queryable, Updateable, and Cacheable XML Compression

Stefan Böttcher (Universität Paderborn, DE)

We are especially interested in XML compression schemes that support queries and updates on the compressed data format without fully decompressing the whole document. We have developed a query engine that evaluates core XPath queries including comparisons to values.

As a plug-in, we have developed several XML compressors that can be used for compressing or decompressing XML data and for operations on compressed data like running queries. Furthermore, we were looking into updates on compressed data, and into caching in client-server scenarios. We provided some results comparing the query compression strength and the query evaluation performance of succinct XML compression and DTD subtraction each implemented on both, XML trees and XML DAGs. Furthermore, we identified a couple open questions for further directions in XML compression research.

Keywords: Schema-based compression, succinct, DAG, queries on compressed data streams, updates, caching

Clone Detection via Structural Abstraction

William Evans (University of British Columbia - Vancouver, CA)

This paper describes the design, implementation, and application of a new algorithm to detect cloned code. It operates on the abstract syntax trees formed by many compilers as an intermediate representation. It extends prior work by identifying clones even when arbitrary subtrees have been changed. On a 440,000-line code corpus, 20- 50%of the clones it detected were missed by previous methods. The method also identifies cloning in declarations, so it is somewhat more general than conventional procedural abstraction.

Keywords: Clone Detection

Joint work of: Evans, William S. ; Fraser, Christoph W. ; Ma, Fei

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1678

Automata for Positive Core XPath Queries on Compressed XML Documents

Barbara Fila-Kordy (Université d'Orleans, FR)

Given any dag t representing a fully or partially compressed XML document, we present a method for evaluating the positive unary queries of some type, on t, without unfolding t into a tree.

To each Core XPath query of a basic type, we associate a word automaton. These automata run on the graph of dependency between the non-terminals of the straightline regular tree grammar asso- ciated to the given dag, or along complete sibling chains in this grammar.

Any given Core XPath query can be decomposed into queries of the basic type, and the answer to the query, on the dag t, can then be expressed as a sub-dag of t suitably labeled under the runs of such automata.

Keywords: Automata, Tree grammars, Dags, Compressed documents, XML, Core XPath.

Joint work of: Fila-Kordy, Barbara; Anantharaman, Siva

A Rewrite Approach for Pattern Containment – Application to Query Evaluation on Compressed Documents

Barbara Fila-Kordy (Université d'Orleans, FR)

In this paper we introduce an approach that allows to handle the containment problem for the fragment XP(/,//,[ ],*) of XPath. Using rewriting techniques we define a necessary and sufficient condition for pattern containment. This rewrite view is then adapted to query evaluation on XML documents, and remains valid even if the documents are given in a compressed form, as dags.

Keywords: Pattern Containment, Compressed Documents

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1679

First order data encoding by prefix codes

Wojciech Fraczak (Université du Québec en Outaouais, CA)

We investigate the use of prefix codes for symbolic encoding of data in the context of model checking of real-time systems.

Keywords: Model checking, data representation, prefix code

Matching Integer Intervals by Minimal Sets of Binary Words with "don't cares"

Wojciech Fraczak (Université du Québec en Outaouais, CA)

An interval $[p, q]$, where $0 \leq p \leq q < 2^n$, can be considered as the set X of n-bit binary strings corresponding to encodings of all integers in $[p, q]$.

A word w with *don't care* symbols is *matching* the set $L(w)$ of all words of the length $|w|$ which can differ only on positions containing a *don't care*. A set Y of words with *don't cares* is *matching* X iff $X = \bigcup_{w \in Y} L(w)$. For a set X of codes of integers in $[p, q]$ we ask for a minimal size set Y of words with *don't cares* matching X. Such a problem appears in the context of network processing engines using *Ternary Content Addressable Memory* (TCAM) as a lookup table for IP packet header fields. The set Y is called a *template* in this paper, and it corresponds to a TCAM representation of an interval. It has been traditionally calculated by a heuristic called "prefix match", which can produce a result of the size approximately twice larger than the minimal one.

We propose two fast (linear time in the size of the input and the output) algorithms for finding minimal solutions for two natural encodings of integers: the usual binary representation (lexicographic encoding) and the reflected Gray code.

Keywords: TCAM, two-level logic minimization, minimal DNF

See also: Wojciech Fraczak, Wojciech Rytter, Mohammadreza Yazdani: Matching Integer Intervals by Minimal Sets of Binary Words with don't cares. CPM 2008: 217-229

An Efficient Algorithm to Test Square-Freeness of Strings Compressed by Balanced Straight Line Program

Shunsuke Inenaga (Kyushu University, JP)

In this paper we study the problem of deciding whether a given compressed string contains a square. A string x is called a square if $x = zz$ and $z = u^k$ implies $k = 1$ and $u = z$. A string w is said to be square-free if no substrings of w are squares. Many efficient algorithms to test if a given string is square-free, have been developed so far. However, very little is known for testing square-freeness of a given compressed string. In this paper, we give an $O(max(n^2; nlog^2N))$-time $O(n^2)$-space solution to test square-freeness of a given compressed string, where n and N are the size of a given compressed string and the corresponding decompressed string, respectively. Our input strings are compressed by balanced straight line program (BSLP). We remark that BSLP has exponential compression, that is, $N = O(2^n)$. Hence no decompress-then-test approaches can be better than our method in the worst case.

Keywords: Square Freeness, Straight Line Program

Joint work of: Matsubara, Wataru ; Inenaga, Shunsuke ; Shinohara, Ayumi

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1680

XML Compression Techniques

Gregory Leighton (University of Calgary, CA)

This talk summarizes existing research on XML-conscious compression techniques and highlights possible areas for future work.

Keywords: XML, compression, survey

Algorithmics on compressed strings

Markus Lohrey (Universität Leipzig, DE)

In recent years, straight-line programs (SLPs) turned out to be a convenient formalism for investigating algorithms on compressed string data. An SLP is a context-free grammar that generates exactly one string w. Since the length of the generated string w may grow exponentially with the size of the SLP, the latter may be seen as a compressed representation of w. The output of many practical compression algorithms, like for instance those of the Lempel-Ziv family, can be considered as SLPs. The talk will give an overview on recent complexity results for algorithmic string problems, where the input string is given compressed as an SLP. Also applications in other fields like for instance algorithmic groups theory will be presented.

Keywords: Algorithms on compressed strings, computational complexity, word problems

Grammar-Based Tree Compression

Sebastian Maneth (Univ. of New South Wales, AU)

Implementations that load XML documents and give access to them via, e.g., the DOM, suffer from huge memory demands: the space needed to load an XML document is usually many times larger than the size of the document. A considerable amount of memory is needed to store the tree structure of the XML document. In this paper, a technique is presented that allows to represent the tree structure of an XML document in an efficient way. The representation exploits the high regularity in XML documents by compressing their tree structure; the latter means to detect and remove repetitions of tree patterns. Formally, context-free tree grammars that generate only a single tree are used for tree compression. The functionality of basic tree operations, like traversal along edges, is preserved under this compressed representation. This allows to directly execute queries (and in particular, bulk operations) without prior decompression.

The complexity of certain computational problems like validation against XML types or testing equality is investigated for compressed input trees.

Keywords: Tree, Compression, XML, straight-line context-free tree grammar

Full Paper:
http://dx.doi.org/10.1016/j.is.2008.01.004

Storage and Retrieval of Individual Genomes

Veli Mäkinen (University of Helsinki, FI)

A repetitive sequence collection is one where portions of a *base sequence* of length n are repeated many times with small variations, forming a collection of total length N. Examples of such collections are version control data and genome sequences of individuals, where the differences can be expressed by lists of basic edit operations.

Flexible and efficient data analysis on a such typically huge collection is plausible using suffix trees. However, suffix tree occupies $O(N \log N)$ bits, which very soon inhibits in-memory analyses. Recent advances in full-text *self-indexing* reduce the space of suffix tree to $O(N \log \sigma)$ bits, where σ is the alphabet size. In practice, the space reduction is more than 10-fold for example on suffix tree of Human Genome. However, this reduction remains a constant factor when more sequences are added to the collection We develop a new self-index suited for the repetitive sequence collection setting. Its expected space requirement depends only on the length n of the base sequence and the number s of variations in its repeated copies. That is, the space reduction is no longer constant, but depends on N/n.

We believe the structure developed in this work will provide a fundamental basis for storage and retrieval of individual genomes as they become available due to rapid progress in the sequencing technologies.

Keywords: Pattern matching, text indexing, compressed data structures, comparative genomics

Joint work of: Mäkinen, Veli; Navarro, Gonzalo; Sirén, Jouni; Välimäki, Niko

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2008/1674

XSAQCT. XML Schema-Aware Queryable Compression Technique

Tomasz Müldner (Acadia University - Wolfville, CA)

Our work is focused on Schema-based queryable compression, in which the compressor and the decompressor share the schema. Given a document D valid in S, we produce and compress the annotated schema graph for D.

The second part of our work is focused on compression of a schema-free XML document D, for which we build an annotated schema graph G and compress it.

Keywords: XML schema, queryable compression

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1673

Practical Search on Compressed Text

Gonzalo Navarro (Universidad de Chile - Santiago, CL)

In this talk I survey some of the current practical results on compressed indexed text searching, highlighting in particular the intricate relation between indexing and compression.

Keywords: Wavelet tree, burrows-wheeler transform, Ziv-Lempel compression, compact data structures, rank-select

An In-Memory XQuery/XPath Engine over a Compressed Structure Representation

Gonzalo Navarro (Universidad de Chile - Santiago, CL)

We describe the architecture and main algorithmic design decisions for an XQuery/ XPath processing engine over XML collections which will be represented using a self-indexing approach, that is, a compressed representation that will allow for basic searching and navigational operations in compressed form.

The goal is a structure that occupies little space and thus permits manipulating large collections in main memory.

Joint work of: Bonifati, Angela ; Leighton, Gregory ; Mäkinen, Veli ; Maneth, Sebastian ; Navarro, Gonzalo ; Pugliese, Andrea

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1677

Optimizing XML Compression in XQueC

Andrea Pugliese (University of Calabria, IT)

We present our approach to the problem of optimizing compression choices in the context of the XQueC compressed XML database system. In XQueC, data items are aggregated into containers, which are further grouped to be compressed together. This way, XQueC is able to exploit data commonalities and to perform query evaluation in the compressed domain, with the aim of improving both compression and querying performance. However, different compression algorithms have different performance and support different sets of operations in the compressed domain. Therefore, choosing how to group containers and which compression algorithm to apply to each group is a challenging issue. We address this problem through an appropriate cost model and a suitable blend of heuristics which, based on a given query workload, are capable of driving appropriate compression choices.

Keywords: XML compression

Joint work of: Arion, Andrei ; Bonifati, Angela ; Manolescu, Ioana ; Pugliese, Andrea

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1692

A Space-Saving Approximation Algorithm for Grammar-Based Compression

Hiroshi Sakamoto (Kyushu Institute of Technology, JP)

A space-efficient approximation algorithm for the grammar-based compression problem, which requests for a given string to find a smallest context-free grammar deriving the string, is presented. For the input length n and an optimum CFG size g, the algorithm consumes only O(g log g) space and O(n log*n) time to achieve O((log*n) log n) approximation ratio to the optimum compression, where log*n is the maximum number of logarithms satisfying log log ... logn $>$ 1. This ratio is thus regarded to almost O(log n), which is the currently best approximation ratio. While g depends on the string, it is known that $g = \Omega(logn)$ and $g = O(n/log_k n)$ for strings from a k-letter alphabet [12].

Keywords: Grammar based compression, space efficient approximation

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1693