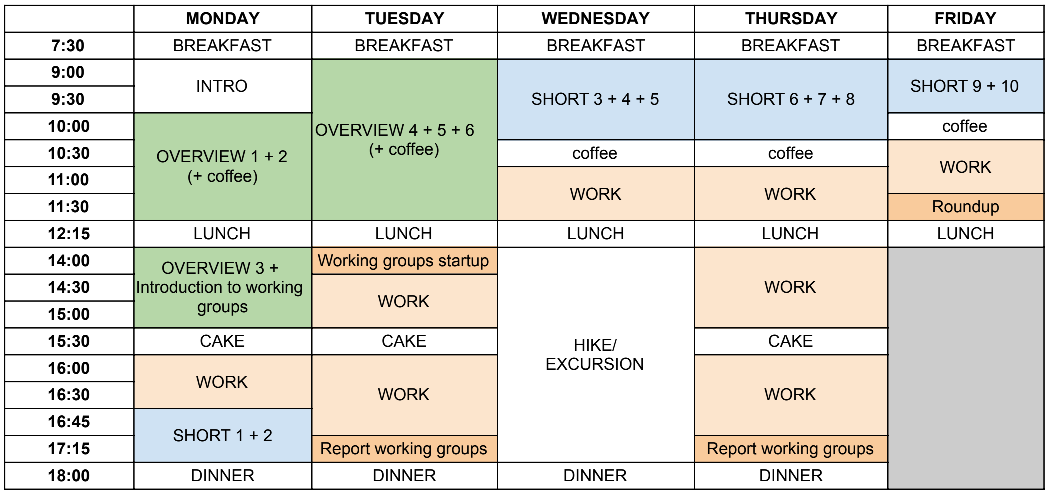
Dagstuhl Seminar 24472
Regular Expressions: Matching and Indexing
Program

Regular expressions and finite automata lie at the foundations of Computer Science and have been used since the sixties in basic problems like compiler design. The key algorithmic challenge is regular expression matching, that is, efficiently identifying words of a regular language within a sequence.

Over the years, there have been numerous algorithmic advances around the topic, while at the same time, their applications have spread over too many different areas like information retrieval, databases, bioinformatics, security, and others, which not only make use of standard results but also pose new and challenging variants of the regular expression matching problem. The use of regular expressions has made its way even into current standards like SQL:2016 and SPARQL. Bringing together researchers from core stringology and relevant application areas will benefit both sides, giving the opportunity to exchange novel problems and solutions of theoretical and practical nature.

The seminar aims to bring together researchers from various research directions within algorithmic aspects of regular expressions and finite automata. Furthermore, the seminar will inspire the exchange of theoretical and practical results. Our aims are to identify practically relevant restrictions and extensions of regular expression matching, as well as variants that work on graphs rather than sequences, and propose matching and indexing algorithms to handle those, together with related impossibility results.

1 Program



	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
7:30	BREAKFAST	BREAKFAST	BREAKFAST	BREAKFAST	BREAKFAST
9:00	INTRO	OVERVIEW 4 + 5 + 6 (+ coffee)	SHORT 3 + 4 + 5	SHORT 6 + 7 + 8	SHORT 9 + 10
9:30					
10:00	OVERVIEW 1 + 2 (+ coffee)				coffee
10:30			coffee	coffee	WORK
11:00			WORK	WORK	
11:30					Roundup
12:15	LUNCH	LUNCH	LUNCH	LUNCH	LUNCH
14:00	OVERVIEW 3 + Introduction to working groups	Working groups startup	HIKE/ EXCURSION	WORK	
14:30		WORK			
15:00					
15:30	CAKE	CAKE		CAKE	
16:00	WORK	WORK		WORK	
16:30					
16:45	SHORT 1 + 2				
17:15		Report working groups		Report working groups	
18:00	DINNER	DINNER	DINNER	DINNER	

2 Titles and abstracts

Overview talks: 40 minutes + questions. Short talks: 20 minutes + questions.

OVERVIEW 1. Philip Bille, Algoritmic techniques for regexp matching

OVERVIEW 2. Gonzalo Navarro, Bit-parallel sequential search for regular expressions

OVERVIEW 3. Wim Martens, Regular Expressions in Information Extraction and Graph Databases I will give an overview to the use of regular expressions in two areas in databases: information extraction and graph databases. In information extraction, regular expressions play a central role in the document spanner framework. This framework aims at transforming text into a relation of spans, i.e., intervals of start and end positions in the text. Within the spanner framework, the class of regular spanners is particularly interesting, since it is based on regular expressions with capture variables. I will also briefly touch upon frequent sequence mining. In graph databases, regular expressions have been interesting since the beginning, since they form the foundation of regular path queries, which are the quintessential feature of graph database query languages. Recent developments in graph query languages, among which the development of Cypher and the standardization of GQL and SQL/PGQ, motivate new ways of evaluating regular path queries in practice, which raises many new research questions.

OVERVIEW 4. Paweł Gawrychowski, Regular language membership in small space

OVERVIEW 5. Moshe Lewenstein, Text Indexing in the context of Regular Expressions

OVERVIEW 6. Nicola Prezza, From text indexing to regular language indexing Since the invention of suffix sorting (in particular, of suffix trees) in the 70s, the problem of indexed pattern matching has been heavily studied in the literature. This problem has a natural language-theoretic interpretation: given a string S, build a (linear-space) data structure answering membership queries in the substring closure of S. This interpretation was recently made more interesting by several works showing that suffix sorting can be naturally extended to some nonlinear structures, notably labeled trees and de Bruijn graphs. This line of work culminated in the invention of Wheeler automata, a class of NFAs admitting efficient and elegant solutions to a large number of hard problems on automata (including membership). In this talk, I will first give an introduction to the rich theory of Wheeler automata and Wheeler languages. I will then show how these ideas can be generalized to arbitrary NFAs, comparing this solution to other existing parameterized approaches for matching strings on regular languages.

SHORT 1. Cristian Riveros, REmatch: theory and practice for evaluating regex and finding all matches

SHORT 2. Dominik D. Freydenberger, Treating Regex as Database Queries Most modern implementations of regular expressions have back-references, which express repetitions of submatches. These allow the definition of non-regular languages, like the copy language ww. The price for this gain in expressive power is that matching becomes NP-hard.

In this talk, I present a relational algebra on strings that allows us to adapt techniques from database theory to matching of regular expressions with back-references. It also provides us with a framework to combine these with parsing techniques.

SHORT 3. Yasuhiko Minamide, Complexity Analysis of Regular Expression Matching Based on Backtracking Until recently, regular expression matching has mostly been implemented using backtracking, which led to a denial-of-service vulnerability known as ReDoS. In ReDoS, matching does not complete in linear time and can take an excessive amount of time. In this talk, we will review previous works that detect such problematic regular expressions through ambiguity analysis of nondeterministic automata and growth rate analysis of string-to-tree transducers. For a given regular expression, it is possible to precisely determine the time complexity (order) of its matching with respect to the length of the input string.

SHORT 4. Konstantinos Mamouras, Efficient Matching of Regular Expressions with Lookaround Assertions Regular expressions can be extended with lookaround assertions, which are subdivided into lookahead and lookbehind assertions. These constructs are used to refine when a match for a pattern occurs in the input text based on the surrounding context. Current implementation techniques for lookaround involve backtracking search, which can give rise to running time that is super-linear in the length of input text. In this talk, we first present a formal mathematical semantics for lookaround, which complements the commonly used operational understanding of lookaround in terms of a backtracking implementation. This formal semantics allows us to establish several equational properties for simplifying lookaround assertions. Additionally, we propose a new algorithm for matching regular expressions with lookaround that has time complexity $O(m \cdot n)$, where m is the size of the regular expression and n is the length of the input text. The algorithm works by evaluating lookaround assertions in a bottom-up manner. It makes use of a new notion of nondeterministic finite automata (NFAs), which we call oracle-NFAs. These automata are augmented with epsilon-transitions that are guarded by oracle queries that provide the truth values of lookaround assertions at every position in the text. We provide an implementation of our algorithm that incorporates three performance optimizations for reducing the work performed and memory used. We present an experimental comparison against PCRE and Java's regex library, which are state-of-the-art regex engines that support lookaround assertions. Our experimental results show that, in contrast to PCRE and Java, our implementation does not suffer from super-linear running time and is several times faster.

SHORT 5. Markus Schmid, Subsequence Expressions with Gap Constraints A subsequence expression is a regular expression of the form $p = A^*x_1A^*x_2A^* \ldots A^*x_mA^*$, where A is an alphabet and $x_i \in A$. Matching p to a string w means to search w for the subsequence $x_1x_2 \ldots x_m$. A match can be formalised as an embedding $e : |p| \to |w|$, which then, for every $1 \leq i < j \leq |p|$, induces an (i, j)-gap, i.e., the substring of w that occurrs strictly between $w[e(i)]$ and $w[e(j)]$. Gap constraints are constraints for the gaps induced by an embedding, e.g., "the (i, j)-gap should be no longer than 7", "the (i, j)-gap should not contain the symbol c", etc.

We investigate the problem of matching a subsequence expression to a string under the presence of gap constraints, where the gap constraints are given as regular languages. Our results cover hardness results as well as polynomial upper bounds and conditional lower bounds.

SHORT 6. Gonzalo Navarro, Evaluating Regular Path Queries on Compressed Adjacency Matrices

SHORT 7. Adrián Gómez-Brandón, Optimizing RPQs over a Compact Graph Representation We propose techniques to evaluate regular path queries (RPQs) over labeled graphs (e.g., RDF). We apply a bit-parallel simulation of a Glushkov automaton representing the query over a Ring: a compact wavelet-tree-based index of the graph. To the best of our knowledge, our approach is the first to evaluate RPQs over a compact representation of such graphs, where we show the key advantages of using Glushkov automata in this setting. We introduce various optimizations, such as the ability to process several automaton states and graph nodes/labels simultaneously, and to accurately estimate relevant selectivities. Experiments show that our approach uses 3-5x less space, and is over 5x faster, on average, than the next best state-of-the-art system for evaluating RPQs.

SHORT 8. Roberto Grossi, McDag: An Index for Maximal common subsequences Maximal Common Subsequences (MCSs)—the inclusion-maximal sequences of non-contiguous symbols shared by strings—are a natural extension of known methods like Longest Common Substrings but have only recently gained attention. This talk presents McDag, an efficient graph-based tool to index MCSs on real genomic data, showing it can handle sequence pairs over 10,000 base pairs in minutes with minimal extra storage.

Joint work with Giovanni Buzzega, Alessio Conte and Giulia Punzi

SHORT 9. Manuel Ariel Caceres Reyes, Parameterized linear-time algorithms for String Matching to DAGs

SHORT 10. James C Davis, Regular Expression Denial of Service: Past, Present, and Future