2024 International Scientific Conference on Information Technologies (InfoTech-2024)

Department of Corporate Information Systems
MIREA – Russian Technological University

Architecture of a regular expression translator with optimization of intermediate states

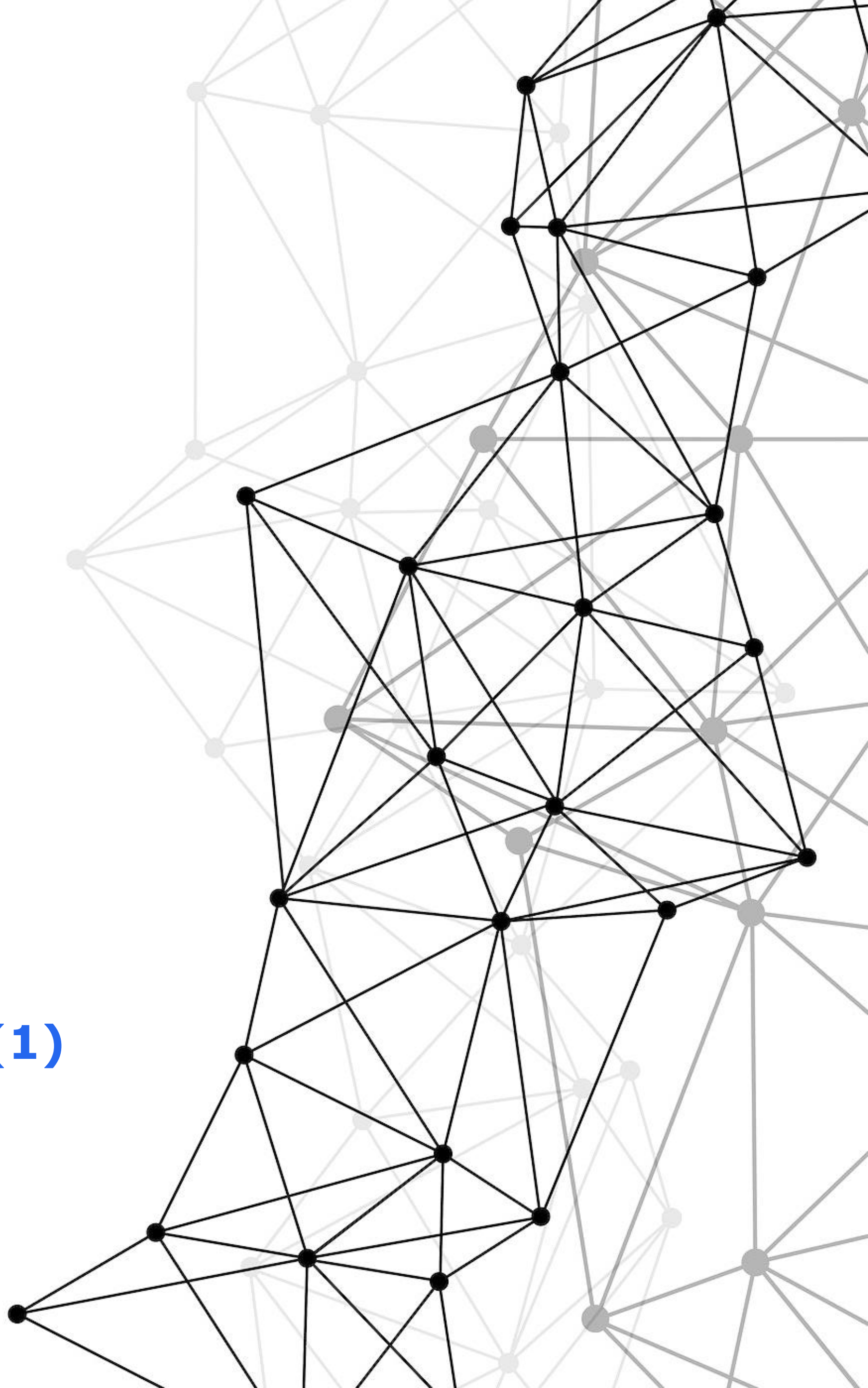
Liliya A. Demidova, Nikita A. Moroshkin

Regular Expressions

Regular expressions are a powerful tool for searching words in a text corpus by a specific pattern. Regular expressions are represented in many programming languages with different implementations.

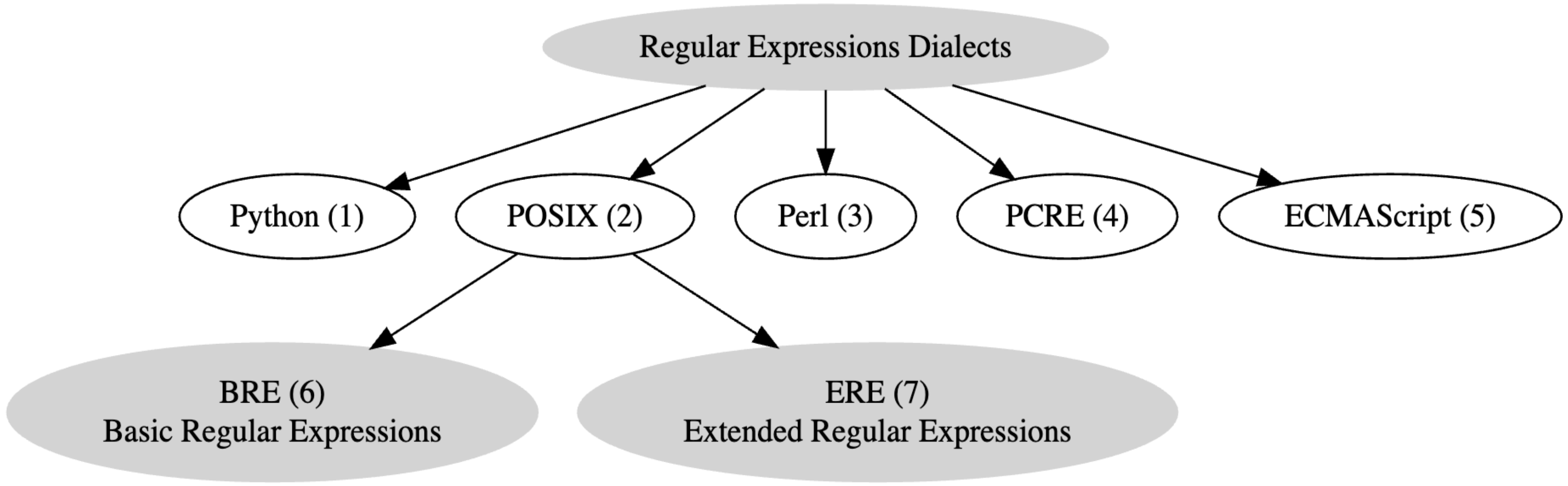
Regular expressions are applied in many fields of text analysis, including social media monitoring, lexical analysis as a stage of code compilation, and many others.

$$\hat{}\left((8|\backslash+7)[\backslash -]?\right)?(\backslash(?[0-9]\{3\}\backslash)?[\backslash -]?)?[0123456789\backslash -]\{7,10\}\$ \quad \textbf{(1)}$$



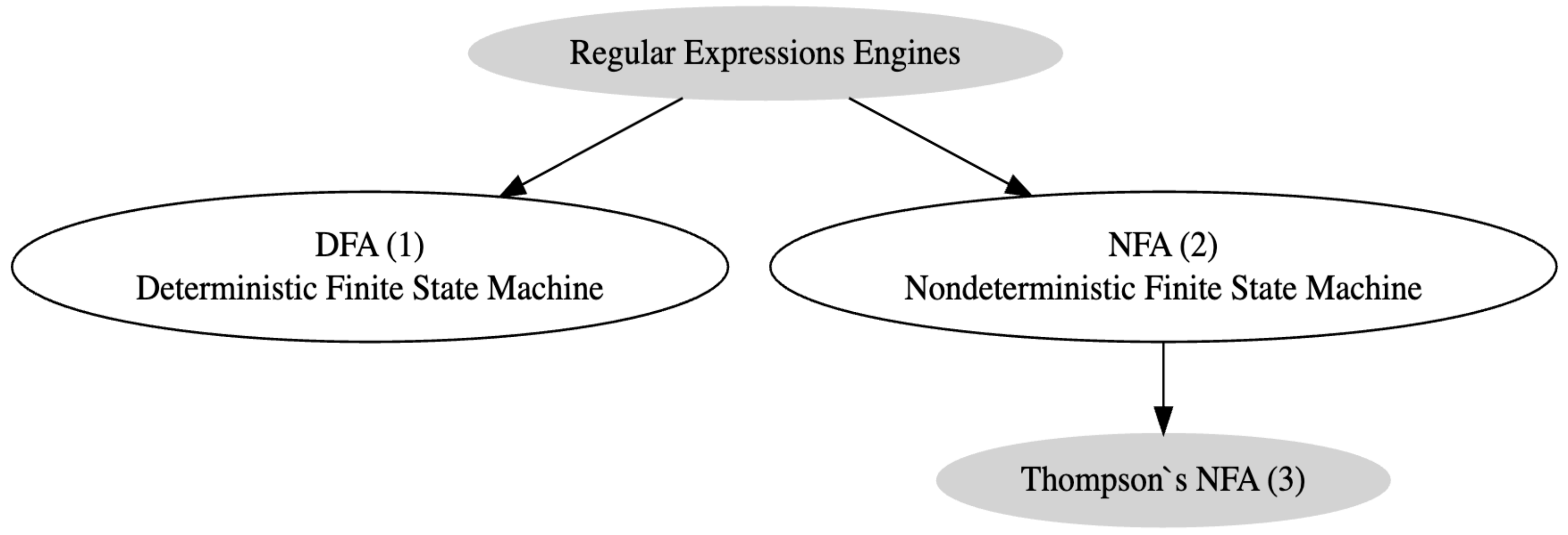
Diversity of regular expressions dialects

Each high-level programming language of the 4th generation (**Python**, **Java**, **PHP**, **Ruby, etc.**) has its versions of regular expressions with its syntax and mathematical base. This makes backward compatibility of various implementations a complex task for researchers, as the syntax and specific features of a particular implementation are often ignored, leading to possible memory leaks, program crashes, and code slowness


Regular Expressions Dialects
Python (1)
POSIX (2)
Perl (3)
PCRE (4)
ECMAScript (5)
BRE (6) Basic Regular Expressions
ERE (7) Extended Regular Expressions

Diversity of regular expressions implementations

Regular expressions can also be classified by the type of software implementation of the finite automaton described by the expressions.
In general, all such software implementations can be divided into two types - those implemented on the basis of a deterministic finite automaton (DFA) and those implemented on the basis of a nondeterministic finite automaton (NFA).

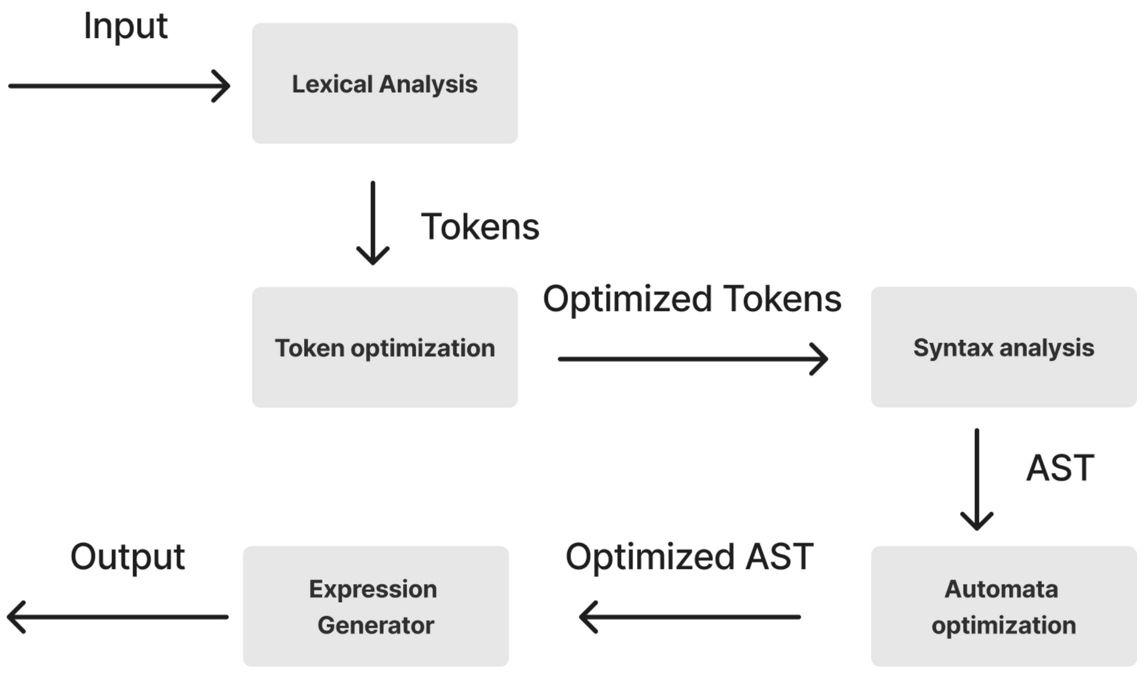

Regular Expressions Engines
DFA (1)
Deterministic Finite State Machine
NFA (2)
Nondeterministic Finite State Machine
Thompson`s NFA (3)

Atchitecture of regular expressions translator

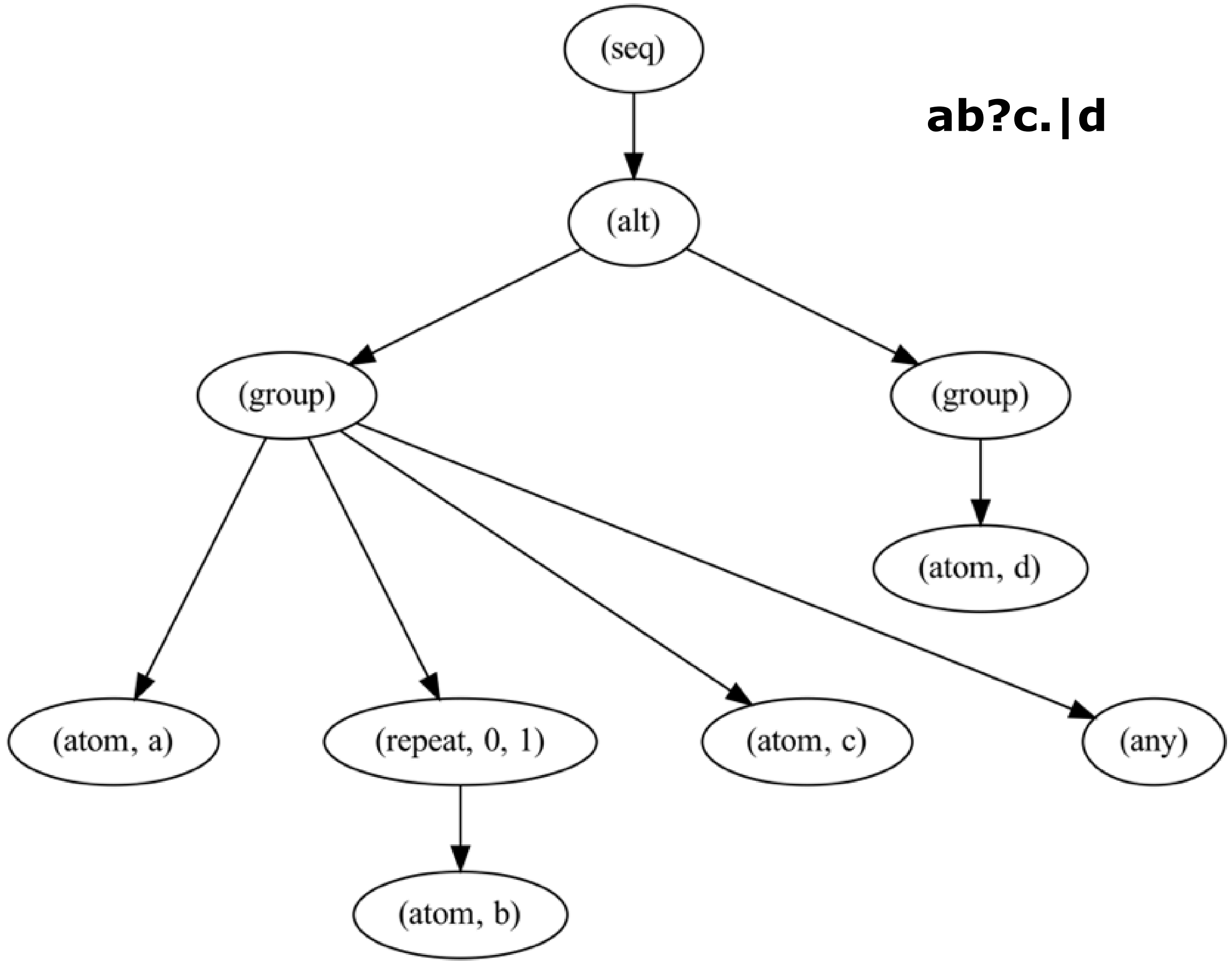
Regular expression translator can solve the backward compatibility problem.
It allows creating a single and independent intermediate representation that will be universal for different regular expression syntaxes

Comparison of regular expression in different implementations

	Average regular expression (1) running time, msec
PCRE library	3,0
PCRE2 library	4,1
re python library	2,8
regexp package in Golang	5,6



Regular expression intermediate state



ab?c.|d

Regular expressions can be represented in the form of an abstract syntax tree. At the stage of lexical analysis, an abstract syntax tree is formed. Each node of such a tree, in essence, represents a construction similar to the Backus-Naur form.

Regular expression optimization (tokens)

The optimization stage is divided into two parts – optimization of tokens (lexemes) of the input expression considering the specifics of the automaton's operation.

For example, most regular expressions have groups of symbol delimiters – quantifiers. These constructs define the number of repetitions of a specific pattern. There are also so-called greedy, lazy, and super-greedy quantifiers, differing in the degree of match strictness.

Comparison of greedy and lazy quantifiers

	re library expression	*regex* library expression
Average number of steps	156	45
Average time, sec	8,4	5,87

Regular expression optimization (AST)

Population algorithms can be used to optimize AST.

Population algorithms are among the most widely used algorithms for solving extreme optimization problems. Three such algorithms are considered – the differential evolution algorithm (DE), the fish school search algorithm (FSS), and the particle swarm optimization algorithm (PSO).

Example of regular expression optimization by population algorithms

AS IS:

```
((i|I)nternational.{0, 1}(c|C)onference.{0, 1}on.{0, 1}(i|I)nformation.{0, 1}(t|T)echnologies
{0.8}(bulgaria|Bulgaria).{0.50}(11-12.september(.2024)?)|(info(t|T)ech\s2024)
```

TO BE:

```
(International.(c|C)onference.{0, 1}on.{0, 1}(i|I)nformation.{0,
1}(t|T)echnologies{0,8}(bulgaria|Bulgaria) .{0,10}(.2024|11-12.september.2024))|(i|I)nfo(t|T)ech\s2024
```

Regular expression optimization (AST)

To achieve these results, the DE algorithm required 35 generations, the particle swarm algorithm required 78, and the fish swarm algorithm required 24. It's worth noting that this experiment was preliminary, aimed primarily at demonstrating the feasibility of working with ASTs of regular expressions through population algorithms.

Comparison of original regexp and optimizated

	Regular expression (AS IS)	*Regular expression (TO BE)*
Average number of steps	33	35
Average time, msec	3,0	2,8

Conclusion

Presented results show a slight speedup over the original expressions. However, as mentioned earlier, regular expressions can be used everywhere, including for processing large amounts of data.

Regular expressions are commonly used as input filters in almost every monitoring system. In such cases, as the amount of input data increases, the advantages and optimizations performed can significantly reduce the time required to process the input stream.