BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2023

Matching Regexes with Back-References Using Register Set Automata

Jan Vašák, 2023
Supervisor: Ing. Ondřej Lengál Ph.D

Preliminaries

Register Automata

Register automata (RAs) extend finite automata by adding a finite set of *registers*. Each register in an RA can hold one value from the input string or it can be empty. Registers are updated on each transition by either (i) the current input symbol, denoted as $r \leftarrow in$, (ii) a value stored in a register, denoted as $r_1 \leftarrow r_2$, or (iii) they can be emptied, denoted as $r \leftarrow \bot$. Updates in the form of $r \leftarrow r$ are treated as implicit. Furthermore, registers can *guard* a transition by testing whether the current input symbol is (or is not) stored in them, denoted as $in = r$ or $in \neq r$. We also allow sets of characters on transitions instead of single characters. Σ denotes the entire alphabet (i.e., any character can be used to take the transition).

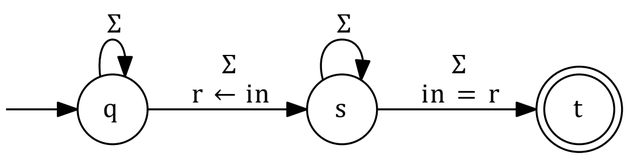


Figure 1: A non-deterministic RA accepting the language of strings whose last symbol appears in the string more than once

Register Set Automata

Register set automata (RsAs), presented in [2], extend finite automata by adding a finite set of *set-registers*. Each set-register holds a set of characters. They are updated on each transition with a set that can contain set-registers of the automaton and in. The new value is union of the set-registers in the update set, with in added if it is a member of the update set. Updates in the form of $r \leftarrow \{r\}$ are treated as implicit. Like in RAs, the set-registers can guard a transition, but instead of an equality test, it is a (non-)membership test, denoted as $in \in r$ and $in \notin r$.

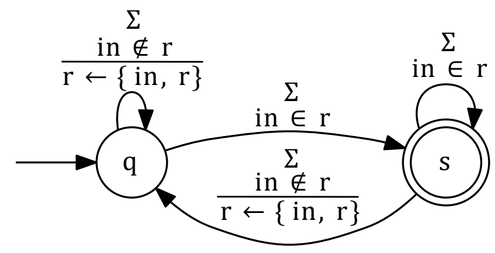


Figure 2: A deterministic RsA, accepting the same language as the RA in Figure 1

Matching regexes with back-references

Backtracking vs. Automata Algorithms

Regexes with back-references are not expressible with finite automata and their common extensions (such as RAs) that can express them are generally not determinisable. For these reasons, matching regexes with back-references is usually done using back-tracking algorithms (which can lead to catastrophic backtracking, causing a large slowdown). However, a large class of RAs can be converted to *deterministic* RsAs using an algorithm presented in [2].

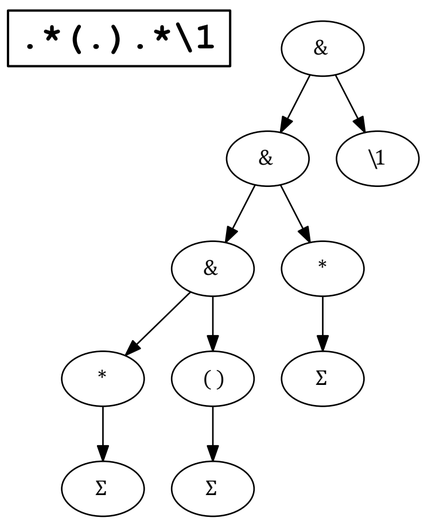


Figure 3: A regex and its syntax tree, & denotes concatenation

Converting regexes to RAs

Regular expressions with back-references can be converted to RAs. This is done by parsing the regex and creating a syntax tree. From this syntax tree, an RA is created by creating RAs for each sub-regex and merging them together.

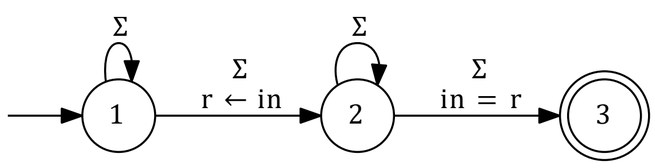


Figure 4: An RA accepting the language described by the regex in Figure 3

In order to determinize an RA, it needs to be converted to an equivalent RA that is *register-local*.

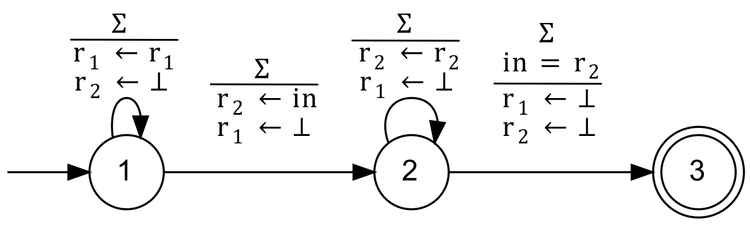


Figure 5: A register-local version of the RA in Figure 4

Output of the Determinisation Algorithm

The algorithm works similarly to the subset construction algorithm [3] with additional handling of registers. The determinised automaton can then be used to match strings to the specified pattern by simply *running* the automaton over the string.

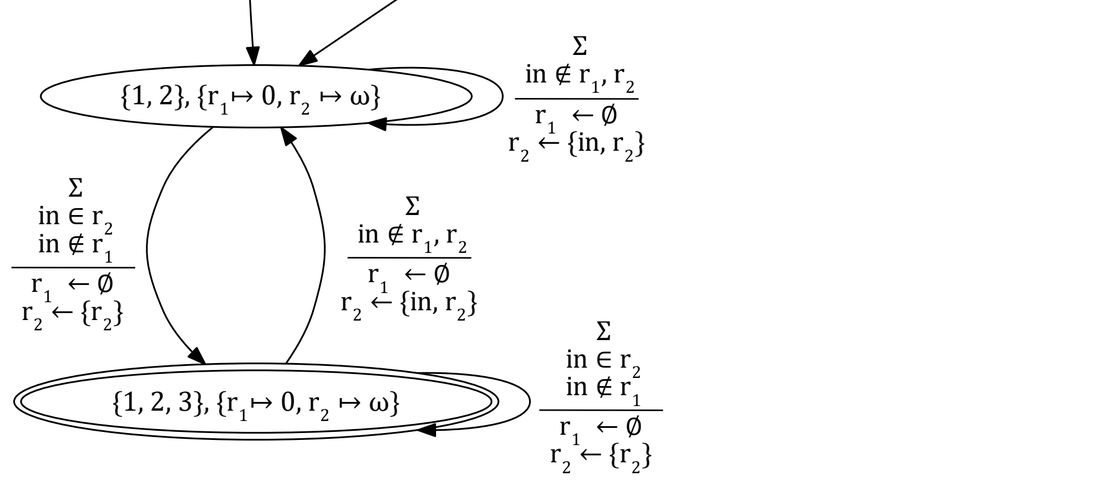
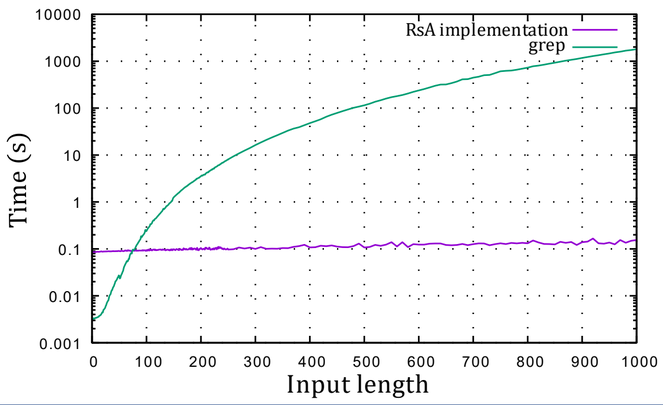


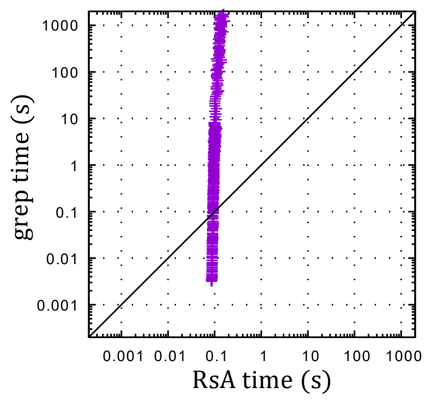
Figure 6: The output of the determinisation algorithm for the RA in Figure 5

Experiments

A prototype of a regex matcher using RsAs was implemented in Python and measured against the pattern matching tool `grep` (version 3.6 [1]). The chosen regex was `/^.*(.).*\1.*;.*;.*\1$/` with the input words being `a;a;a;a`, `a;;a;a;a`, ..., `a;`^{994}`a;a;a`. Results show that the RsA implementation has a near constant matching time, while `grep`'s time increases exponentially.

Figure 7 (left): A graph showing the time to match depending on the length of input
Figure 8 (right): A scatter plot of times to match





[1] Free Software Foundation, Inc. GNU grep 3.6 [online]. [cit. 2022-02-02]. Available at: https://git.savannah.gnu.org/cgit/grep.git.
[2] Gulčíková, S. and Lengál, O. Register Set Automata (Technical Report). arXiv, 2022. [cit. 2022-02-02]. DOI: 10.48550/ARXIV.2205.12114.]. Available at: https://arxiv.org/abs/2205.12114.
[3] Rabin, M. O. and Scott, D. Finite Automata and Their Decision Problems. IBM Journal of Research and Development. 1959, vol. 3, no. 2, p. 114–125. DOI: 10.1147/rd.32.0114.