

# The Effect of Rewriting Regular Expressions on Their Accepting Automata

Michiel Frishert, Loek G. Cleophas, and Bruce W. Watson

Department of Mathematics and Computer Science, Technische Universiteit  
Eindhoven, P.O.Box 513, NL-5600 MB Eindhoven, The Netherlands

Department of Computer Science, University of Pretoria,  
Pretoria 0002, South Africa

m.frishert@stud.tue.nl, loek@loekcleophas.com, bruce@bruce-watson.com

Traditionally, regular expressions are transformed into (deterministic) finite automata, which are then used in, for example, searching and parsing. For an introduction to automata and regular expressions, refer to [1]. For every automaton, there is an infinite number of other automata that recognizes exactly the same language, although memory usage and performance can differ. This has led to the wide-spread use of automata minimization techniques. In the same way, for every regular expression, there exists an infinite number of equivalent regular expressions, some of which lead to smaller/faster automata. This paper compares automaton sizes constructed from 10 different regular expressions, both before and after the rewriting transformation. Rewriting is done based on a rule set directly derived from the regular expression language (i.e. Kleene Algebra) axioms; these are listed in Table 1.

The first three automata tested are the Thompson NFA [2], the Position NFA [3], and the Follow  $\epsilon$ -NFA [4]. The remaining three automata are the determinized counterparts of each NFA. For these DFA, the size of the minimized DFA is provided for comparison. For a list of the regular expressions used in this test, see [5]. In Fig. 1 the results are presented visually. Clearly, rewriting has an impact on the automaton size. However, the impact varies between regular expressions and between automata.

**Table 1.** Rewrite Rules. Note that  $a, E, F$  are regular expressions.

$\epsilon \cdot E \rightarrow E$	$\emptyset^*$	$\rightarrow$	$\epsilon$	$(E^* F)^* \rightarrow (E F)^*$	$E^{+?} \rightarrow E^*$
$E \cdot \epsilon \rightarrow E$	$\emptyset^?$	$\rightarrow$	$\epsilon$	$(\epsilon E) \rightarrow E^?$	$E^{??} \rightarrow E^?$
$E \cdot \emptyset \rightarrow \emptyset$	$a \cdot E a \cdot F \rightarrow a \cdot (E F)$			$E^{**} \rightarrow E^*$	$E^{++} \rightarrow E^+$
$\emptyset \cdot E \rightarrow \emptyset$	$E \cdot a F \cdot a \rightarrow (E F) \cdot a$			$E^{?*} \rightarrow E^*$	$E^{*+} \rightarrow E^*$
$E \emptyset \rightarrow E$	$E E \rightarrow E$			$E^{+*} \rightarrow E^*$	$E^{2+} \rightarrow E^+$
$\epsilon^* \rightarrow \epsilon$	$E E^* \rightarrow E^*$			$E^{*?} \rightarrow E^*$	

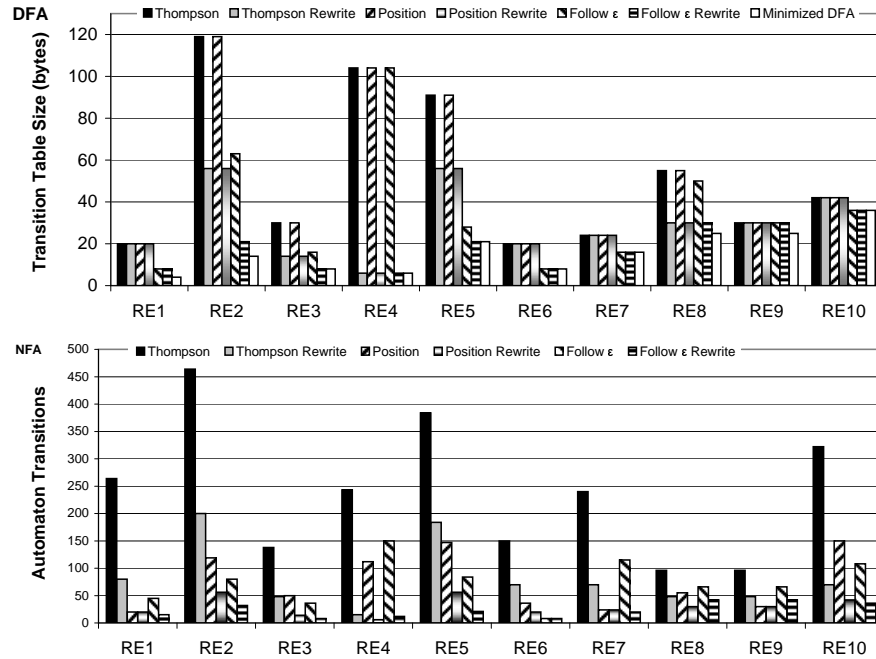


Fig. 1. Effect of rewriting on NFA and DFA sizes

Our first implementation reads its rules from a text file and is therefore quite slow: rewriting takes up to 10 times as long as automaton construction. However, once a set of good rules is selected, they can be hardcoded, thus leading to much better performance.

## References

1. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (1979)
2. Thompson, K.: Regular Expression Search Algorithm. Communications of the ACM **11** (6) (1968) 419–422
3. Glushkov, V.M.: The Abstract Theory of Automata. Russian Math. Surveys **16** (1961) 1–53
4. Ilie, L., Yu, S.: Constructing NFAs by Optimal Use of Positions in Regular Expressions. In: Lecture Notes in Computer Science: Proceedings of the 13th CPM, Springer-Verlag (2002) 279–288
5. Frishert, M., Watson, B.W.: The Effect of Rewriting Regular Expressions on Their Accepting Automata. Technical report, Technische Universiteit Eindhoven (2003)