



## On Regular Expression Matching and Deterministic Finite Automata

**Bille, Philip**

*Published in:*  
Tiny ToCS

*Publication date:*  
2015

*Document Version*  
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

*Citation (APA):*  
Bille, P. (2015). On Regular Expression Matching and Deterministic Finite Automata. *Tiny ToCS*, 3, 1.  
<http://tinytocs.org/>

---

### General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

# On Regular Expression Matching and Deterministic Finite Automata

Philip Bille\*  
Technical University of Denmark, DTU Compute

## ABSTRACT

Given a regular expression  $R$  and a string  $T$  the regular expression matching problem is to determine if  $T$  matches any string in the language generated by  $R$ . The best known solution to the problem uses linear space and  $O\left(\frac{nm \log \log n}{\log^{3/2} n} + n + m\right)$  time in the worst-case [2], where  $m$  and  $n$  are the lengths of  $R$  and  $T$ , respectively. A common misconception is that we can solve the problem efficiently by building a deterministic finite automaton (DFA) for  $R$  using  $2^{O(m)}$  space and then run it on  $T$  in  $O(n)$  time [1]. However, this analysis completely ignores issues of addressing into exponential sized data structures. An address in a DFA of size  $2^{\Omega(m)}$  requires  $\Omega(m)$  bits. Hence, on a standard unit-cost word RAM with word length  $\Theta(\log n)$  [3], we need at least  $\Omega(m/\log n)$  time to simply write an address in the DFA. It follows that traversing the DFA for  $R$  uses at least  $\Omega(nm/\log n + n + m)$  worst-case time (note that we do not even include DFA construction time). This bound can only be  $O(n)$  when  $m = O(\log n)$  and is never better than the above best known bound.

## BODY

*Even ignoring construction time, deterministic finite automata do not solve regular expression matching in worst-case linear time.*

## REFERENCES

- [1] Regular expression, Wikipedia. [http://en.wikipedia.org/wiki/Regular\\_expression](http://en.wikipedia.org/wiki/Regular_expression). Accessed Jan. 2015.
- [2] P. Bille and M. Thorup. Faster regular expression matching. In *Proc. 36th ICALP*, pages 171–182, 2009.
- [3] T. Hagerup. Sorting and searching on the word RAM. In *Proc. 15th STACS*, pages 366–398, 1998.

---

\*Supported by the Danish Research Council (DFR 4005-00267, DFR 1323-00178) and the Advanced Technology Foundation.