Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Succinctness of regular expressions with interleaving, intersection and counting[☆]

Wouter Gelade [*,1]

Hasselt University and transnational University of Limburg, School for Information Technology, Department WNI, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium

ARTICLE INFO

Article history:
Received 1 October 2008
Received in revised form 3 February 2010
Accepted 20 April 2010
Communicated by M. Ito

Keywords:
Regular Expressions
Succinctness

ABSTRACT

In this paper, we study the succinctness of regular expressions (REs) extended with interleaving, intersection and counting operators. We show that in a translation from REs with interleaving to standard regular expressions a double exponential size increase cannot be avoided. We also consider the complexity of translations to finite automata. We give a tight exponential lower bound on the translation of REs with intersection to NFAs, and, for each of the three classes of REs, we show that in a translation to a DFA a double exponential size increase cannot be avoided. Together with known results, this gives a complete picture of the complexity of translating REs extended with interleaving, intersection or counting into (standard) regular expressions, NFAs, and DFAs.

1. Introduction

Regular expressions are used in many applications such as text processors, programming languages [34], and XML schema languages [5,32]. These applications, however, usually do not restrict themselves to the standard regular expression using disjunction (+), concatenation (·) and star (*), but also allow the use of additional operators. Although these operators mostly do not increase the expressive power of the regular expressions, they can have a drastic impact on succinctness, thus making them harder to handle. For instance, it is well known that expressions extended with the complement operator can describe certain languages non-elementary more succinct than standard regular expressions or finite automata [33].

In this paper, we study the succinctness of regular expressions extended with counting (RE(#)), intersection (RE(∩)), and interleaving (RE(&)) operators. The counting operator allows for expressions such as $a^{[2,5]}$, specifying that there must occur at least two and at most five a's. These RE(#)*s* are used in egrep [20] and Perl [34] patterns and in the XML schema language XML Schema [32]. The class RE(∩) is a well studied extension of the regular expressions, and is often referred to as the semi-extended regular expressions. The interleaving operator allows for expressions such as $a \,\&\, b \,\&\, c$, specifying that a, b, and c may occur in any order, and is used, for instance, in the XML schema language Relax NG [5].

A problem we consider, is the translation of extended regular expressions into (standard) regular expressions. For RE(#) and RE(∩) the complexity of this translation has already been settled and is exponential [22] and double exponential [13], respectively. We show that also in constructing an expression for the interleaving of a set of expressions (an hence also for an RE(&)) a double exponential size increase cannot be avoided. This is the main technical result of the paper. Apart from a pure mathematical interest, the latter result has two important consequences. First, it prohibits an efficient translation

[☆] An extended abstract of this paper appeared in the proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2008).

[*] Tel.: +32 11 268222.
E-mail addresses: wouter.gelade@uhasselt.be, wouter.gelade@gmail.com.

[1] Research Assistant of the Fund for Scientific Research - Flanders (Belgium).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.04.036

Table 1
An overview of the results in this paper concerning the complexity of translating extended regular expressions into NFAs, DFAs, and regular expressions. Proposition and theorem numbers are given in brackets. The result concerning the translation from RE(&) to RE has been improved in [18] to $2^{2^{\Omega(n)}}$.

	NFA	DFA	RE
RE(#)	$2^{\Omega(n)}$ [22]	$2^{2^{\Omega(n)}}$ (Proposition 7)	$2^{\theta(n)}$ [22]
RE(∩)	$2^{\Omega(n)}$ (Proposition 5)	$2^{2^{\Omega(n)}}$ (Theorem 8)	$2^{2^{\Omega(n)}}$ [18]
RE(&)	$2^{\Omega(n)}$ [24]	$2^{2^{\Omega(\sqrt{n})}}$ (Theorem 10)	$2^{2^{\Omega(\sqrt{n})}}$ (Theorem 20)
RE(&, ∩, #)	$2^{\mathcal{O}(n)}$ (Proposition 4)	$2^{2^{\mathcal{O}(n)}}$ (Proposition 6)	$2^{2^{\mathcal{O}(n)}}$ (Proposition 11)

from Relax NG (which allows interleaving) to XML Schema Definitions (which does not). However, as XML Schema is the widespread W3C standard, and Relax NG is a more flexible alternative, such a translation would be more than desirable. A second consequence concerns the automatic discovery of regular expression describing a set of given strings. The latter problem occurs in the learning of XML schema languages [1–3]. At present these algorithms do not take into account the interleaving operator, but for Relax NG this would be wise as this would allow to learn significantly smaller expressions.

It should be noted here that Gruber and Holzer independently obtained a similar result [18]. They show that any regular expression defining the language $(a_1b_1)^* \& (a_2b_2)^* \& \cdots \& (a_nb_n)^*$ must be of size at least double exponential in n. Compared to the result in this paper, this gives a tighter bound ($2^{2^{\Omega(n)}}$ instead of $2^{2^{\Omega(\sqrt{n})}}$), and shows that the double exponential size increase already occurs for very simple expressions. On the other hand, the alphabet of the counterexamples grows linear with n, whereas the alphabet size is constant for the languages in this paper. Later, they also adapted this result to obtain a (tighter) $2^{2^{\Omega(n/\log n)}}$ bound over a binary alphabet [18].

We also consider the translation of extended regular expressions to NFAs. For the standard regular expressions, it is well known that such a translation can be done efficiently [4]. Therefore, when considering problems such as membership, equivalence, and inclusion testing for regular expressions the first step is almost invariantly a translation to a finite automaton. For extended regular expressions, such an approach is less fruitful. We show that an RE(&, ∩, #) can be translated in exponential time into an NFA. However, it has already been shown by Kilpeläinen and Tuhkanen [22] and Mayer and Stockmeyer [24] that such an exponential size increase cannot be avoided for RE(#) and RE(&), respectively. For the translation from RE(∩) to NFAs, a $2^{\Omega(\sqrt{n})}$ lower bound is reported in [29], which we here improve to $2^{\Omega(n)}$.

As the translation of extended regular expressions to NFAs already involves an exponential size increase, it is natural to ask what the size increase for DFAs is. Of course, we can translate any NFA into a DFA in exponential time, thus giving a double exponential translation, but can we do better? For instance, from the results in [13] we can conclude that given a set of regular expressions, constructing an NFA for their intersection cannot avoid an exponential size increase. However, it is not too hard to see that also a DFA of exponential size accepting their intersection can be constructed. In the present paper, we show that this is not possible for the classes RE(#), RE(∩), and RE(&). For each class we show that in a translation to a DFA, a double exponential size increase cannot be avoided. An overview of all results is given in Table 1. Note that as we give upper bounds for the translation from RE(&, ∩, #) and lower bounds for the smaller classes, most of these results imply matching upper and lower bounds.

Related work. The different classes of regular expressions considered here have been well studied; in particular, RE(∩) and its membership [21,23,29] and equivalence and emptiness [10,28,30] problems. Also the classes RE(#) [22,27] and RE(&) [12,24] have received interest. Succinctness of regular expressions has been studied by Ehrenfeucht and Zeiger [8] and, more recently, by Ellul et al. [9], Gelade and Neven [13], Gruber and Holzer [15–18], and Gruber and Johannsen [19]. See the Ph.D. theses of Gruber [14] and Gelade [11] for an overview of many of the recently obtained results. Schott and Spehner give lower bounds for the translation of the interleaving of words to DFAs [31]. Also related, but different in nature, are the results on state complexity [36], in which the impact of the application of different operations on finite automata is studied.

Outline. In Section 2 we give the necessary definitions and present some basic results. In Sections 3–5 we study the translation of extended regular expressions to NFAs, DFAs, and regular expressions, respectively.

2. Definitions and basic results

2.1. Regular expressions

By $\mathbb{N}$ we denote the natural numbers without zero. For the rest of the paper, Σ always denotes a finite alphabet. A *Σ-string* (or simply string) is a finite sequence $w = a_1a_2 \cdots a_n$ of Σ-symbols. We define the length of w, denoted by $|w|$, to be n. We denote the empty string by ε. The set of *positions of w* is $\{1, \ldots, n\}$ and the *symbol of w at position i* is a_i. By $w_1 \cdot w_2$ we denote the *concatenation* of two strings w_1 and w_2. As usual, for readability, we denote the concatenation of w_1 and w_2 by w_1w_2. The set of all strings is denoted by Σ^*. A *string language* is a subset of Σ^*. For two string languages $L, L' \subseteq \Sigma^*$, we define their concatenation $L \cdot L'$ to be the set $\{ww' \mid w \in L, w' \in L'\}$. We abbreviate $L \cdot L \cdots L$ (i times) by