Information and Computation 245 (2015) 181–196

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Deciding determinism of unary languages ☆

Ping Lu [a,b,c], Feifei Peng [a,b,c,1], Haiming Chen [a,*], Lixiao Zheng [d,*]

[a] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[b] Graduate University of Chinese Academy of Sciences, China
[c] University of Chinese Academy of Sciences, China
[d] College of Computer Science and Technology, Huaqiao University, Xiamen Fujian 361021, China

ARTICLE INFO

Article history:
Received 28 October 2013
Received in revised form 28 May 2015
Available online 1 October 2015

Keywords:
Regular expressions
Regular expressions with counting
Deterministic languages
Minimal DFA
coNP-complete
$\mathbf{\Pi_2^p}$

ABSTRACT

In this paper, we investigate the complexity of deciding determinism of unary languages. First, we give a method to derive a set of arithmetic progressions from a regular expression E over a unary alphabet, and establish relations between numbers represented by these arithmetic progressions and words in $\mathcal{L}(E)$. Next, we define a problem relating to arithmetic progressions and investigate the complexity of this problem. Then by a reduction from this problem we show that deciding determinism of unary languages is **coNP**-complete. Finally, we extend our derivation method to expressions with counting, and prove that deciding whether an expression over a unary alphabet with counting defines a deterministic language is in $\mathbf{\Pi_2^p}$. We also establish a tight upper bound for the size of the minimal DFA for expressions with counting.

1. Introduction

XML (Extensible Markup Language) has important applications in data exchange [1], database [2], etc. XML schema languages, e.g., DTD and XML Schema, are used to specify the constraints which XML documents should obey [3]. However, designing a correct schema is not an easy job [4,5]. One difficulty is the Unique Particle Attribution (UPA) constraint [6], which requires that content models should be deterministic [7,8]. Intuitively, determinism means that a symbol in the input word should be matched to a unique position in the regular expression without looking ahead in the word [6,9]. For example, $A \to a^*a$ is a simple example of a DTD. This is not a correct DTD, because the content model a^*a is not deterministic. Consider the word a. Without knowing the length of the word, we do not know that the only symbol a in the word should match the first a or the second one in a^*a.

Deterministic expression is defined in a semantic way, without a known simple syntax definition [8]. It is not easy for users to understand such kind of expressions. Studying properties of deterministic expressions can help reduce this difficulty. Lots of work [8–15] studied properties of deterministic expressions and gave methods to help users write deterministic expressions. Meanwhile, studying properties of deterministic languages is equally important. For example, when the user writes a nondeterministic expression E and we know that $L(E)$ is deterministic, we can automatically generate a

☆ Parts of this work have been presented as conference abstracts appeared in DLT 2013 under the title "Deciding Determinism of Unary Languages Is coNP-Complete". Work supported by the National Natural Science Foundation of China under Grants 61472405 and 61070038.

* Corresponding authors.
E-mail addresses: luping@ios.ac.cn (P. Lu), chm@ios.ac.cn (H. Chen).

[1] Part of this work was done while Feifei Peng was a student in China Agricultural University.

http://dx.doi.org/10.1016/j.ic.2015.08.005
0890-5401/

deterministic expression describing $L(E)$ for the user. However only little progress has been made about determinism of languages.

For standard regular expressions, Brüggemann-Klein and Wood [9] showed that the problem, whether a regular language defined by a standard regular expression can be described by a standard deterministic expression, is decidable. Bex et al. [8], Czerwiński et al. [16], and P. Lu et al. [17] proved that this problem is **PSPACE**-complete. The problem becomes much harder when we consider expressions with counting. Czerwiński et al. [16] also proved that deciding whether a regular language defined by a regular expression with counting can be described by a standard deterministic expression is **EXPSPACE**-complete [16]. Recently Latte et al. [18] showed that whether a regular language defined by a standard regular expression can be described by a deterministic expression with counting is in 2-**EXPSPACE**. And an **NL** lower bound was given there [18,17]. In this paper, we try to show that this problem is **coNP**-hard, and improve the existing lower bound.

In [19], Gelade et al. showed that for unary languages, deterministic expressions with counting are expressively equivalent to standard deterministic expressions. Hence considering determinism of regular languages described by standard expressions over a unary alphabet can give a lower bound for the problem, whether a regular language can be described by a deterministic expression with counting. Moreover, in the lower bound proofs of [8] and [16], the alphabet size of constructed expressions is at least 4. So it is possible that the complexity of the problem, whether a regular language defined by a standard regular expression over a unary alphabet can be described by a standard deterministic expression, is lower than **PSPACE**. This is our starting point. In the following, unless explicitly stated otherwise, all regular expressions are expressions over the alphabet $\{a\}$.

Our contributions are listed as follows:

(1) We show that deciding whether a standard expression denotes a deterministic language is **coNP**-complete. Then we conclude that deciding whether a language can be defined by a deterministic expression with counting is **coNP**-hard.
(2) For any expression E with counting, we show that there is a DFA with less than $2^{O(|E|)}$ states accepting $\mathcal{L}(E)$. It has been shown that there exists an expression E with counting such that every DFA accepting this language has at least exponential number of states [20,21]. So our upper bound is tight. For the case $|\Sigma| = 2$, there is an expression E such that the minimal DFA accepting $\mathcal{L}(E)$ has $\Omega(2^{2^{|E|}})$ states [21].
(3) Using the result in (2), we devise a non-deterministic algorithm to check determinism of languages defined by expressions with counting, and show that the problem, whether an expression with counting denotes a deterministic language, is in $\mathbf{\Pi_2^p}$.

The rest of the paper is organized as follows. Section 2 gives some basic definitions and some facts from the number theory, which we will use later. We associate a set of arithmetic progressions with a given regular expression in Section 3. Section 4 shows the complexity of deciding determinism of unary languages. Section 5 deals with expressions with counting. Section 6 gives the conclusion and the future work.

2. Preliminaries

Let $\Sigma = \{a\}$ be an alphabet of symbols. A standard regular expression over Σ is recursively defined as follows: $\emptyset$, ε and a are regular expressions; for any two regular expressions E_1 and E_2, the union $E_1 + E_2$, the concatenation E_1E_2 and the star E_1^* are regular expressions. For a regular expression E, we denote $\mathcal{L}(E)$ as the language specified by E and $|E|$ as the size of E, which is the sum of the number of symbol occurrences in E and the number of used operators.

Expressions with counting, denoted by R(#), extend standard expressions by using counting operator: $E^{[m,n]}$ or $E^{[m,\infty]}$, where ∞ stands for infinity. Since $E^* = E^{[0,\infty]}$, we do not consider the star operator in regular expressions in R(#). The size of an expression E in R(#), denoted by $|E|$, is the sum of the number of symbol occurrences, the number of used operators, and the lengths of the binary encodings of all counting numbers [19].

To define deterministic regular expressions, we need the following notations. We mark each symbol a in E with a different integer i such that each marked symbol a_i occurs only once in the marked expression. For example, $a_1^*a_2$ is a marking of a^*a. The marking of E is denoted by $\overline{E}$. We use $E^\natural$ to denote the result of dropping subscripts from the marked symbols. These notations are extended for words and sets of symbols in an obvious way.

Deterministic regular expressions are defined as follows.

Definition 1. (See [9].) An expression E is deterministic, if and only if, for all words $uxv, uyw \in \mathcal{L}(\overline{E})$ where $|x| = |y| = 1$, if $x \neq y$ then $x^\natural \neq y^\natural$. A regular language is deterministic if it is denoted by some deterministic expression.

For example, a^*a is not deterministic, since $a_2, a_1a_2 \in \mathcal{L}(a_1^*a_2)$. Deterministic regular expressions denote a proper subclass of regular languages [9].

A nondeterministic finite automaton (NFA) [22] is a 5-tuple $\mathcal{N} = (Q, \{a\}, \delta, q_0, F)$, where Q is a finite set of states, a is the input symbol, $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of final states, $\delta : Q \times \{a\} \to 2^Q$ is the transition function. The size of an NFA $\mathcal{N}$, denoted as $|\mathcal{N}|$, is defined as the number of states of $\mathcal{N}$. An NFA $\mathcal{N} = (Q, \{a\}, \delta, q_0, F)$ is in Chrobak normal form [23] if the following conditions hold: (1) $Q = \{q_0, q_1, \ldots, q_m\} \cup \{q_{1,0}, q_{1,1}, \ldots, q_{1,i_1}\} \cup \{q_{2,0}, q_{2,1}, \ldots, q_{2,i_2}\} \cup$