Deciding Determinism of Unary Languages Is coNP-Complete[⋆]

Ping Lu[1,2,3], Feifei Peng[4], and Haiming Chen[1]

[1] State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
{luping,chm}@ios.ac.cn
[2] Graduate University of Chinese Academy of Sciences
[3] University of Chinese Academy of Sciences
[4] China Agricultural University, Beijing 100083, China

Abstract. In this paper, we give the complexity of deciding determinism of unary languages. First, we derive a set of arithmetic progressions from an expression E over a unary alphabet, and give the relations between numbers in these arithmetic progressions and words in $L(E)$. Next, we define a problem related to arithmetic progressions and investigate the complexity of this problem. Finally, by reduction from this problem we show that deciding determinism of unary languages is **coNP**-complete.

1 Introduction

The XML schema languages, e.g., DTD and XML Schema, require that the content model should be deterministic, which ensures fast parsing documents [20,1]. 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 [21,4].

However, this determinism is defined in a semantic way, without a known simple syntax definition [1]. It is not easy for users to understand such kind of expressions. Lots of work [1,4,3,11,14,5,15] studied properties of deterministic expressions. But most of these work merely handled determinism of expressions and only little progress has been made about determinism of languages.

For standard regular expressions, Brüggemann-Klein and Wood [4] showed that the problem, whether an expression denotes a deterministic language, is decidable. Recently Bex et al. [1] proved that the problem is **PSPACE**-hard, but it is unclear whether the problem is in **PSPACE**. The problem becomes much harder when we consider expressions with counting. It is not known to be decidable whether a language can be defined by a deterministic expression with counting. In [9], Gelade et al. showed that for unary languages, deterministic expressions with counting are expressively equivalent to standard deterministic

⋆ Work supported by the National Natural Science Foundation of China under Grant No. 61070038.

expressions. Hence considering determinism of standard expressions over a unary alphabet can give a lower bound for the problem. This is our starting point.

Covering systems were introduced by Paul Erdős [7,2]. This is an interesting topic in mathematics and there are many unsolved problems about covering systems [12]. Here, we are only concerned with the problem whether a set of arithmetic progressions forms a covering system. This problem has been shown to be **coNP**-complete [19,8].

Unary languages are actually sets of numbers. Then for an expression E, we derive a set of arithmetic progressions and show the relations between these arithmetic progressions and $L(E)$. After that, we give the complexity of deciding determinism of unary languages by reduction from covering systems.

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 gives the conclusion and the future work.

2 Preliminaries

Let Σ be an alphabet of symbols. A regular expression over Σ is recursively defined as follows: $\emptyset$, ε and $a \in \Sigma$ 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 $L(E)$ as the language specified by E and $|E|$ as the size of E, which is the number of symbols occurrence in E.

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 the obvious way.

Deterministic regular expressions are defined as follows.

Definition 1 ([4]). *An expression E is deterministic if and only if, for all words $uxv, uyw \in L(\overline{E})$ 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 $a_2, a_1a_2 \in L(a_1^*a_2)$. A natural characterization of deterministic regular expressions is that: E is deterministic if and only if the Glushkov automaton of E is deterministic [3]. Deterministic regular expressions denote a proper subclass of regular languages [4].

The following notations are basic mathematical operators [10]: $\lfloor x \rfloor = \max\{\, n \mid n \leq x, n \in \mathbb{Z}\}$; $x \bmod y = x - y\lfloor \frac{x}{y} \rfloor$, for $y \neq 0$; $x \equiv y \pmod p \Leftrightarrow x \bmod p = y \bmod p$; $m|n \Leftrightarrow m > 0$ and $n = mx$ for some integer x; $gcd(x_1, x_2, \ldots, x_n) = \max\{k|(k|x_1) \wedge (k|x_2) \wedge \ldots (k|x_n)\}$; $lcm(x_1, x_2, \ldots, x_n) = \min\{k|k > 0 \wedge (x_1|k) \wedge (x_2|k) \wedge \ldots (x_n|k)\}$. Notice that $gcd(0, 0, \ldots, 0)$ is undefined and $lcm(x_1, x_2, \ldots, x_n)$ is also undefined when one of the parameters is not larger than 0. In this paper, we denote $gcd(0, 0, \ldots, 0) = 0$ and $lcm(x_1, x_2, \ldots, x_n) = 0$ when one of the parameters is 0.

Here, we give some facts, which we will use later.

Lemma 1 ([22]). *Given two integers $a, b > 0$, each number of the form $ax+by$, with $x, y \geq 0$, is a multiple of $gcd(a, b)$. Furthermore, the largest multiple of $gcd(a, b)$ that cannot be represented as $ax+by$, with $x, y \geq 0$, is $lcm(a, b)-(a+b)$.*

From Lemma 1, we can obtain more generalized results as follows.

Corollary 1. *Given two integers $a, b > 0$, each number of the form $ax + by$, with $x \geq X \geq 0$ and $y \geq Y \geq 0$, is a multiple of $gcd(a, b)$. Furthermore, the largest multiple of $gcd(a, b)$ that cannot be represented as $ax + by$, with $x \geq X$ and $y \geq Y$, is $aX + bY + lcm(a, b) - (a + b)$.*

Corollary 2. *Given n $(n \geq 2)$ integers $a_1 > 0, a_2 > 0, \ldots, a_n > 0$, each number of the form $a_1x_1 + a_2x_2 \ldots + a_nx_n$, with $x_1 \geq X_1 \geq 0$, $x_2 \geq X_2 \geq 0$, $\ldots$, and $x_n \geq X_n \geq 0$, is a multiple of $gcd(a_1, a_2, \ldots, a_n)$. Furthermore, all multiples of $gcd(a_1, a_2, \ldots, a_n)$ no less than $\sum\limits_{i=1}^{n} a_iX_i + n \prod\limits_{i=1}^{n} a_i$ can be represented as $a_1x_1 + a_2x_2 \ldots + a_nx_n$, with $x_1 \geq X_1 \geq 0$, $x_2 \geq X_2 \geq 0$, $\ldots$, and $x_n \geq X_n \geq 0$.*

In this paper, we primarily discuss unary languages. For a regular language L over the alphabet $\{a\}$, there is a correspondence between words in L and their lengths. For convenience when we say the word n, we mean the word a^n.

3 The Arithmetic Progressions of Unary Languages

In this section, we handle the following problem: Given an expression E, how to construct a set $\mathcal{P} = \{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \ldots, \langle l_n, s_n\rangle\}$ of arithmetic progressions such that any word $w \in L(E)$ satisfies $|w| \equiv s_i \pmod{l_i}$ for some i $(1 \leq i \leq n)$, and $\mathcal{P}$ reflects some structural properties of E.

To this end, we define the following function for an expression E.

Definition 2. *The function $\mathcal{S}(E)$ is defined as*

$$\begin{aligned}
&\mathcal{S}(\varepsilon) = \{\langle 0, 0\rangle\}\\
&\mathcal{S}(a) = \{\langle 0, 1\rangle\} \quad a \in \Sigma\\
&\mathcal{S}(E_1 + E_2) = \mathcal{S}(E_1) \cup \mathcal{S}(E_2)\\
&\mathcal{S}(E_1E_2) = \{\langle gcd(l_i, l_j), s_i + s_j\rangle | \langle l_i, s_i\rangle \in S(E_1) \wedge \langle l_j, s_j\rangle \in S(E_2)\}\\
&\mathcal{S}(E_1^*) = \begin{cases} \{\langle 0, 0\rangle\} & \text{if } S(E_1) = \{\langle 0, 0\rangle\},\\ \{\langle l, 0\rangle | l = \max\{x | \forall l_i \forall s_i (\langle l_i, s_i\rangle \in S(E_1) \wedge (x|l_i) \wedge (x|s_i))\}\} & \text{otherwise;}\end{cases}
\end{aligned}$$

The intuition behind the construction of $\mathcal{S}(E)$ is Lemma 1. The cases ε, a, and E_1+E_2 are obvious. For the case E_1E_2, let $\langle l_1, s_1\rangle \in \mathcal{S}(E_1)$ and $\langle l_2, s_2\rangle \in \mathcal{S}(E_2)$. Then numbers in $\mathcal{S}(E)$ should be in the form $k_1l_1+s_1+k_2l_2+s_2$, where $k_1, k_2 \in \mathbb{N}$. From Lemma 1, these numbers can be written as $k \cdot gcd(l_1, l_2) + s_1 + s_2$, where $k \in \mathbb{N}$. Moreover, we have added some extra numbers in $\mathcal{S}(E)$, but the number of new natural numbers is finite. The case E_1^* is similar.

Example 1. Let $E = (aaa + aa)^* + (aaa)^*((aa)^*aaa + (aaa)^*aa)$. The process of computing $\mathcal{S}$ is shown in Table 1. E_1 in the table stands for subexpressions of E. At last, $\mathcal{S}(E) = \{\langle 1,0\rangle, \langle 1,3\rangle, \langle 3,2\rangle\}$. It is easy to see that $\mathcal{S}(E)$ contains all natural numbers. However, $a \notin L(E)$ and a is the only word, which is not in $L(E)$.

Table 1. The process of computing $\mathcal{S}$

E_1	$\mathcal{S}(E_1)$	E_1	$\mathcal{S}(E_1)$	E_1	$\mathcal{S}(E_1)$	E_1	$\mathcal{S}(E_1)$
a	$\langle 0,1\rangle$	$aa+aaa$	$\langle 0,2\rangle$ $\langle 0,3\rangle$	$(aa+aaa)^*$	$\langle 1,0\rangle$	$(aa)^*aaa+(aaa)^*aa$	$\langle 2,3\rangle$ $\langle 3,2\rangle$
aa	$\langle 0,2\rangle$	$(aa)^*$	$\langle 2,0\rangle$	$(aa)^*aaa$	$\langle 2,3\rangle$	$(aaa)^*((aa)^*aaa+(aaa)^*aa)$	$\langle 1,3\rangle$ $\langle 3,2\rangle$
aaa	$\langle 0,3\rangle$	$(aaa)^*$	$\langle 3,0\rangle$	$(aaa)^*aa$	$\langle 3,2\rangle$	$(aaa+aa)^*+$ $(aaa)^*((aa)^*aaa+(aaa)^*aa)$	$\langle 1,0\rangle$ $\langle 1,3\rangle$ $\langle 3,2\rangle$

At first, we have the following property about the tuples in $\mathcal{S}(E)$.

Proposition 1. *Let E be an expression. For any $\langle l,s\rangle \in \mathcal{S}(E)$ there exists an L $(L \geq 0)$ such that* (1) : *$L + t \cdot l + s \in L(E)$ for any integer t $(t \geq L)$;* (2) : *if $l \neq 0$, then $l|L$.* (3) : *if $l = 0$, then $L = 0$.*

Proof. We prove it by induction on the structure of E. For simplicity, we denote $\mathcal{Q}(l,s,L,E)$ as the conditions in the proposition. That is $\mathcal{Q}(l,s,L,E) = true$ if and only if (1), (2), and (3) hold for the parameters l, s, L and E.

The cases $E = \varepsilon$ or a, $a \in \Sigma$ are obvious, since $L = 0$ satisfy the conditions.

$E = E_1 + E_2$: Suppose $\langle l,s\rangle \in \mathcal{S}(E)$. From the definition of $\mathcal{S}$, we have $\langle l,s\rangle \in \mathcal{S}(E_1)$ or $\langle l,s\rangle \in \mathcal{S}(E_2)$. If $\langle l,s\rangle \in \mathcal{S}(E_1)$, then by the inductive hypothesis there is an L_1 $(L_1 \geq 0)$ such that $\mathcal{Q}(l,s,L_1,E_1) = true$. Then $L = L_1$ satisfies $\mathcal{Q}(l,s,L,E) = true$. The case $\langle l,s\rangle \in \mathcal{S}(E_2)$ can be proved in a similar way.

$E = E_1E_2$: Suppose $\langle l,s\rangle \in \mathcal{S}(E)$. From the definition of $\mathcal{S}$, there are $\langle l_1,s_1\rangle \in \mathcal{S}(E_1)$ and $\langle l_2,s_2\rangle \in \mathcal{S}(E_2)$ such that $l = gcd(l_1,l_2)$ and $s = s_1 + s_2$. By the inductive hypothesis there are L_1 $(L_1 \geq 0)$ and L_2 $(L_2 \geq 0)$ such that $\mathcal{Q}(l_1,s_1,L_1,E_1) = true$ and $\mathcal{Q}(l_2,s_2,L_2,E_2) = true$. Let $L = L_1 + L_2 + l_1L_1 + l_2L_2 + lcm(l_1,l_2) - l_1 - l_2 + gcd(l_1,l_2)$. Then from Corollary 1, for any integer t $(t \geq L)$ there are $k_1 \geq L_1$ and $k_2 \geq L_2$ such that:

$$\begin{aligned}
&L + t \cdot l + s \\
&= L_1 + L_2 + l_1L_1 + l_2L_2 + lcm(l_1,l_2) - l_1 - l_2 + gcd(l_1,l_2) + t \cdot gcd(l_1,l_2) + s_1 + s_2 \\
&= L_1 + L_2 + k_1l_1 + k_2l_2 + s_1 + s_2 \\
&= L_1 + k_1l_1 + s_1 + L_2 + k_2l_2 + s_2
\end{aligned}$$

From the inductive hypothesis and $E = E_1E_2$, we have $L + t \cdot l + s \in L(E)$. If $l \neq 0$, then $l_1 \neq 0$ or $l_2 \neq 0$. Suppose $l_1 \neq 0$ and $l_2 = 0$. By the inductive hypothesis, $l_1|L_1$ and $L_2 = 0$. Hence $L = L_1 + l_1L_1$. Since $l = gcd(l_1,l_2)$, $l|L$.

The other cases can be proved in a similar way. If $l = 0$, then $l_1 = 0$ and $l_2 = 0$. Therefore $L_1 = 0$ and $L_2 = 0$ from the inductive hypothesis. Hence $L = 0$.

$E = E_1^*$: Suppose $\langle l, 0\rangle \in \mathcal{S}(E)$ and $\mathcal{S}(E_1) = \{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \ldots, \langle l_n, s_n\rangle\}$. Then $l = gcd(l_1, l_2, \ldots, l_n, s_1, s_2, \ldots, s_n)$. Because $\mathcal{S}(E_1) = \{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \ldots, \langle l_n, s_n\rangle\}$, by the inductive hypothesis there are $L_1 \geq 0, L_2 \geq 0, \ldots, L_n \geq 0$ such that $\mathcal{Q}(l_1, s_1, L_1, E_1) = true$, $\mathcal{Q}(l_2, s_2, L_2, E_1) = true$, $\ldots$, $\mathcal{Q}(l_n, s_n, L_n, E_1) = true$. Therefore for any integer $t_1 \geq L_1$, $t_2 \geq L_2, \ldots,$ $t_n \geq L_n$ we have $L_1 + t_1 l_1 + s_1 \in L(E_1)$, $L_2 + t_2 l_2 + s_2 \in L(E_1)$, $\ldots$, and $L_n + t_n l_n + s_n \in L(E_1)$. Let $L = \sum_{i=1}^{n} 2(L_i + l_i L_i + s_i) + 2n \prod_{i=1}^{n} (L_i + l_i L_i + s_i) l_i$ and $g = gcd(l_1, l_2, \ldots, l_n, L_1 + l_1 L_1 + s_1, L_2 + l_2 L_2 + s_2, \ldots, L_n + l_n L_n + s_n)$. If $g = 0$, then $l_1 = 0$, $l_2 = 0, \ldots,$ $l_n = 0$, $L_1 + l_1 L_1 + s_1 = 0$, $L_2 + l_2 L_2 + s_2 = 0, \ldots,$ $L_n + l_n L_n + s_n = 0$, $l = 0$ and $L = 0$. It is easy to see that the statements (1), (2) and (3) hold. Otherwise, suppose $g \neq 0$. Then $g|L$. From Corollary 2, we know that for any integer t $(t \geq L)$, $\sum_{i=1}^{n} 2(L_i + l_i L_i + s_i) + 2n \prod_{i=1}^{n} (L_i + l_i L_i + s_i) l_i + t \cdot g$ can be represented as $\sum_{i=1}^{n} (L_i + l_i L_i + s_i) x_i + \sum_{i=1}^{n} l_i y_i$, with $x_i \geq 2$, $y_i \geq 0$ $(1 \leq i \leq n)$. By the inductive hypothesis, for any $1 \leq i \leq n$, $gcd(l_i, L_i + l_i L_i + s_i) = gcd(l_i, s_i)$. Then $gcd(l_1, l_2, \ldots, l_n, L_1 + l_1 L_1 + s_1, L_2 + l_2 L_2 + s_2, \ldots, L_n + l_n L_n + s_n) = gcd(l_1, l_2, \ldots, l_n, s_1, s_2, \ldots, s_n)$. Therefore for any integer t $(t \geq L)$,

$$\begin{aligned}
&L + t \cdot gcd(l_1, l_2, \ldots, l_n, s_1, s_2, \ldots, s_n)\\
&= \sum_{i=1}^{n} 2(L_i + l_i L_i + s_i) + 2n \prod_{i=1}^{n} (L_i + l_i L_i + s_i) l_i + t \cdot g\\
&= \sum_{i=1}^{n} (L_i + l_i L_i + s_i) x_i + \sum_{i=1}^{n} l_i y_i\\
&= \sum_{i=1}^{n} [(L_i + l_i L_i + s_i) x_i + l_i y_i]
\end{aligned}$$

For all i $(1 \leq i \leq n)$, since $x_i \geq 2$, $(L_i + l_i L_i + s_i) x_i + l_i y_i = (L_i + l_i L_i + s_i)(x_i - 1) + L_i + l_i(L_i + y_i) + s_i$. By the inductive hypothesis, we have $L_i + l_i L_i + s_i \in L(E_1)$ and $L_i + l_i(L_i + y_i) + s_i \in L(E_1)$. Hence $(L_i + l_i L_i + s_i) x_i + l_i y_i \in L(E)$. Therefore $\sum_{i=1}^{n} [(L_i + l_i L_i + s_i) x_i + l_i y_i] \in L(E)$. That is $L + t \cdot gcd(l_1, l_2, \ldots, l_n, s_1, s_2, \ldots, s_n) \in L(E)$. If $l \neq 0$, then $l | l_i$ and $l | s_i$ for any $0 \leq i \leq n$. Hence $l|L$ by the inductive hypothesis. If $l = 0$, then $l_i = 0$ and $s_i = 0$. Therefore $L_i = 0$. Hence $L = 0$. □

This proposition means that for any $\langle l, s\rangle \in \mathcal{S}(E)$ there exists an L such that any word w, satisfying $|w| = L + t \cdot l + s$ for some integer t $(t \geq L)$, is in $L(E)$.

On the other hand, for any word w in $L(E)$ there is a tuple $\langle l, s\rangle$ in $\mathcal{S}(E)$ such that w satisfies $|w| = t \cdot l + s$ for some integer t $(t \in \mathbb{Z})$. This statement can be ensured by the following proposition.

Proposition 2. *Let E be an expression. For all $w \in L(E)$, there exists $\langle l, s\rangle \in \mathcal{S}(E)$ such that if $l \neq 0$, then $|w| \equiv s \pmod{l}$, or if $l = 0$, then $|w| = s$.*

Proof. We prove it by induction on the structure of E. For simplicity, we denote $\mathcal{R}(l, s, w)$ as the conditions in the proposition. That is $\mathcal{R}(l, s, w) = true$ if and only if the following statement holds: if $l \neq 0$, then $|w| \equiv s \pmod{l}$, or if $l = 0$, then $|w| = s$. The cases $E = \varepsilon$ or a, $a \in \Sigma$ are obvious.

$E = E_1 + E_2$: For all $w \in L(E)$, we know that $w \in L(E_1)$ or $w \in L(E_2)$. If $w \in L(E_1)$, then by the inductive hypothesis there exists $\langle l_1, s_1 \rangle \in \mathcal{S}(E_1)$ such that $\mathcal{R}(l_1, s_1, w) = \mathit{true}$. Because $\mathcal{S}(E) = \mathcal{S}(E_1) \cup \mathcal{S}(E_2)$, there exists $\langle l_1, s_1 \rangle \in \mathcal{S}(E)$ such that $\mathcal{R}(l_1, s_1, w) = \mathit{true}$. The case $w \in L(E_2)$ can be proved in a similar way.

$E = E_1E_2$: For all $w \in L(E)$, there are $w_1 \in L(E_1)$ and $w_2 \in L(E_2)$ such that $w = w_1w_2$. By the inductive hypothesis there exists $\langle l_1, s_1 \rangle \in \mathcal{S}(E_1)$ and $\langle l_2, s_2 \rangle \in \mathcal{S}(E_2)$ such that $\mathcal{R}(l_1, s_1, w_1) = \mathit{true}$ and $\mathcal{R}(l_2, s_2, w_2) = \mathit{true}$. Therefore there are $k_1, k_2 \in \mathbb{Z}$ such that $|w_1| = k_1l_1 + s_1$ and $|w_2| = k_2l_2 + s_2$. From the definition of $\mathcal{S}$, we know that $\langle gcd(l_1, l_2), s_1 + s_2 \rangle \in \mathcal{S}(E)$. Hence

$$\begin{aligned}|w| = |w_1w_2| &= k_1l_1 + s_1 + k_2l_2 + s_2\\ &= k_1k_1'gcd(l_1, l_2) + s_1 + k_2k_2'gcd(l_1, l_2) + s_2\\ &= (k_1k_1' + k_2k_2')gcd(l_1, l_2) + s_1 + s_2\end{aligned}$$

That is $\mathcal{R}(gcd(l_1, l_2), s_1 + s_2, w) = \mathit{true}$.

$E = E_1^*$: Suppose $\langle l, 0 \rangle \in \mathcal{S}(E)$. If $w = \epsilon$, then clearly $\mathcal{R}(l, 0, \epsilon) = \mathit{true}$. For all $w \in L(E)$ and $w \neq \epsilon$, there are $w_1, w_2, \ldots, w_n \in L(E_1)$ such that $w = w_1w_2 \ldots w_n$. By the inductive hypothesis there exists $\langle l_1, s_1 \rangle \in \mathcal{S}(E_1)$, $\langle l_2, s_2 \rangle \in \mathcal{S}(E_2)$,..., and $\langle l_n, s_n \rangle \in \mathcal{S}(E_1)$ satisfying $\mathcal{R}(l_1, s_1, w_1) = \mathit{true}$, $\mathcal{R}(l_2, s_2, w_2) = \mathit{true}$,..., and $\mathcal{R}(l_n, s_n, w_n) = \mathit{true}$. From the definition of $\mathcal{S}$, $w \in L(E)$ and $w \neq \epsilon$, it is easy to prove that $l \neq 0$. Then for any $\langle l', s' \rangle \in \mathcal{S}(E_1)$, $l|l'$ and $l|s'$. Therefore there are $k_{11}, k_{12}, k_{21}, k_{22}, \ldots, k_{n1}, k_{n2} \in \mathbb{Z}$ such that $|w_1| = k_{11}l + k_{12}l$, $|w_2| = k_{21}l + k_{22}l$, ..., and $|w_n| = k_{n1}l + k_{n2}l$. Hence

$$\begin{aligned}|w| = |w_1w_2 \ldots w_n| &= k_{11}l + k_{12}l + k_{21}l + k_{22}l + \ldots + k_{n1}l + k_{n2}l\\ &= (k_{11} + k_{12} + k_{21} + k_{22} + \ldots + k_{n1} + k_{n2})l\end{aligned}$$

That is $\mathcal{R}(l, 0, w) = \mathit{true}$. □

The following lemma is straightforward.

Lemma 2. *Let E be an expression. The following statements hold:* (1) *For all $\langle 0, s \rangle \in \mathcal{S}(E)$ we have $s \in L(E)$;* (2) *$l = 0$ for all $\langle l, s \rangle \in \mathcal{S}(E)$ iff $L(E)$ is finite;* (3) *If there is a tuple $\langle 1, s \rangle \in \mathcal{S}(E)$, then there exists an L ($L \geq 0$) such that $w \in L(E)$ for any w ($|w| > L$).*

From the above properties, we build the relations between words in $L(E)$ and tuples in $\mathcal{S}(E)$. Then we can study properties of $L(E)$ by investigating attributes of $\mathcal{S}(E)$.

Now we analyze the time used to compute $\mathcal{S}(E)$. Given an expression E, we compute $\mathcal{S}(E)$ in the following way: We first construct the syntax tree of E, after that we use a bottom-up traversal to compute $\mathcal{S}$ for each node. It is known that for two m-bit numbers, the greatest common divisor can be computed in $O(m^2)$ time [6]. In our computation, each number has $O(\log |E|)$ bits. Then computing $\mathcal{S}$ for each node takes $O(|E|^2 \log^2 |E|)$ time. Therefore the total time to compute $\mathcal{S}(E)$ is $O(|E|^3 \log^2 |E|)$.

Given an NFA $\mathcal{N}$, Sawa [16] also gave an algorithm to construct a set of arithmetic progressions such that the union of these arithmetic progressions is the language accepted by $\mathcal{N}$. The algorithm runs in $O(n^2(n+m))$ time, where n is the number of states in $\mathcal{N}$ and m is the number of transitions in $\mathcal{N}$. The advantage of our method is that it works merely on original expressions and reaches some kind of the lower bound of the algorithm in [16], since there is an expression E_n such that $|E_n| = n$ and any NFA describing $L(E_n)$ has $\Omega(n \cdot (\log n)^2)$ transitions [17,13]. But the price is that we add words in the language. However, we will see later that adding such words does not affect determinism of the language.

4 Determinism of Unary Languages

In the previous section, we derived a set of arithmetic progressions from a given expression E. We will show how to use these arithmetic progressions to check determinism of $L(E)$ in this section.

4.1 Decision Problems for *Covering Systems*

A *covering system* CS is a set of ordered pairs $\{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \dots, \langle l_n, s_n\rangle\}$, with $0 \le s_i < l_i$ $(1 \le i \le n)$ and $\sum_{i=1}^{n}(l_i + s_i) \le p(n)$ for some polynomial function p, such that any integer x satisfies $x \equiv s_i \pmod{l_i}$ for some i $(1 \le i \le n)$ [2]. For example, the set of pairs $\{\langle 2,0\rangle, \langle 4,1\rangle, \langle 4,3\rangle\}$ forms a *covering system*. Since any integer i satisfies one of the following conditions: $i \equiv 0 \pmod 2$; $i \equiv 1 \pmod 4$; $i \equiv 3 \pmod 4$.

The *covering problem* (**CP**) is the following problem: Whether a given set of ordered pairs $\{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \dots, \langle l_n, s_n\rangle\}$, with $0 \le s_i < l_i$ $(1 \le i \le n)$ and $\sum_{i=1}^{n}(l_i + s_i) \le p(n)$ for some polynomial function p, forms a *covering system*? The complexity of **CP** is shown in the following theorem.

Theorem 1 ([19],[8]). CP *is* **coNP**-*complete*[1].

Similarly, an equal difference covering system (**EDCS**) is a set $\mathcal{P}$ of pairs, $\{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \dots, \langle l_n, s_n\rangle\}$, with $0 \le s_i < l_i$ $(1 \le i \le n)$ and $\sum_{i=1}^{n}(l_i + s_i) \le p(n)$ for some polynomial function p, such that there exist two integers (y, x) $(0 \le x < y)$ satisfying the following condition: For any integer k, $x \equiv k \pmod y$ if and only if $k \equiv s_i \pmod{l_i}$ for some i $(1 \le i \le n)$. We define (y, x) as the answer to $\mathcal{P}$. Let $\mathcal{P} = \{\langle 4,1\rangle, \langle 4,3\rangle\}$. It is straightforward to see that $\mathcal{P}$ is an **EDCS**, but is not a **CS**. The answer to $\mathcal{P}$ is $(2,1)$. Intuitively, the union of the

[1] The polynomial bound is not necessary for this theorem [8]. However, we concentrate on unary languages in this paper, and we need this condition for the definition of **CP**. For this restricted case, the theorem also holds [19].

numbers represented by an **EDCS** forms an arithmetic progression, while the union of the numbers represented by a **CS** contains all integers.

The equal difference covering problem (**EDCP**) is defined as follows: Whether a given set of ordered pairs $\{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \ldots, \langle l_n, s_n\rangle\}$, with $0 \leq s_i < l_i$ $(1 \leq i \leq n)$ and $\sum_{i=1}^{n}(l_i + s_i) \leq p(n)$ for some polynomial function p, forms an **EDCS**?

The union of arithmetic progressions is used in the study of the evenly spaced integer topology [18]. However, the complexity of **EDCP**, as far as we know, has not been given.

We have the following properties of an **EDCS**.

Lemma 3. *Suppose $\mathcal{P}$ is an* **EDCS**. *The answer to $\mathcal{P}$ is unique.*

For a set $\mathcal{P}$ of ordered pairs $\{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \ldots, \langle l_n, s_n\rangle\}$, we denote $L = gcd(l_1, l_2, \ldots, l_n)$ and suppose $0 < p_1 < p_2 < p_3 \ldots < p_m$ are the distinct divisors of L.

Lemma 4. *Suppose $\mathcal{P}$ is an* **EDCS** *and the answer to $\mathcal{P}$ is (y, x). Then there is an integer k such that $y = p_k$, $k = \max\{l \mid \forall i \forall j(\langle l_i, s_i\rangle \in \mathcal{P} \wedge \langle l_j, s_j\rangle \in \mathcal{P} \wedge s_i \equiv s_j \pmod{p_l})\}$ and $x = (s_1 \bmod p_k)$.*

Hence given a set $\mathcal{P}$ of ordered pairs, if we know $\mathcal{P}$ is an **EDCS**, we can find the answer to $\mathcal{P}$ from the tuples in $\mathcal{P}$. Since $\sum_{i=1}^{n}(l_i + s_i) \leq p(n)$, this computation is polynomial-time computable. It is easy to see that the converse of the lemma does not hold. Consider the following set of ordered pairs: $\{\langle 3, 0\rangle, \langle 4, 0\rangle\}$. $y = 1$ and $x = 0$ satisfy all the conditions, but obviously this set is not an **EDCS**.

Bickel et al. [2] gave a method to construct a *covering system* from a set $\mathcal{P}$ of ordered pairs, where the union of numbers represented by the pairs contains an arithmetic progression. Inspired by this idea, we can prove that given an **EDCS**, we can construct a *covering system*, and vice versa.

Theorem 2. **EDCP** *is* **coNP**-*complete.*

Proof. At first, we prove that the problem is **coNP**-hard. This can be proved by reduction from **CP**. Given a set $\mathcal{P}$ of ordered pairs $\{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \ldots, \langle l_n, s_n\rangle\}$, we construct the set $\mathcal{P}_1 = \{\langle 3l_1, 3s_1\rangle, \langle 3l_2, 3s_2\rangle, \ldots, \langle 3l_n, 3s_n\rangle\}$. We claim that $\mathcal{P}$ is a **CS** if and only if $\mathcal{P}_1$ is an **EDCS** and the answer to $\mathcal{P}_1$ is $(3, 0)$.

Suppose $\mathcal{P}$ is a **CS**. For any integer k such that $k \equiv 0 \pmod 3$, let $k = 3k_1$. Because $\mathcal{P}$ is a **CS**, there is a tuple $\langle l, s\rangle \in \mathcal{P}$ satisfying $k_1 \equiv s \pmod l$. Hence $k_1 = lt + s$ for some integer t $(t \in \mathbb{Z})$. Therefore $3k_1 = 3lt + 3s$ and $k = 3lt + 3s$. That is $k \equiv 3s \pmod{3l}$, and obviously $\langle 3l, 3s\rangle \in \mathcal{P}_1$. If $k \equiv 3s \pmod{3l}$ for some tuple $\langle 3l, 3s\rangle \in \mathcal{P}_1$, then $k = 3lt + 3s$ for some integer t $(t \in \mathbb{Z})$. Hence $3|k$. That is $k \equiv 0 \pmod 3$. Therefore $\mathcal{P}_1$ is an **EDCS** and $(3, 0)$ is the answer to $\mathcal{P}_1$.

Suppose $\mathcal{P}_1$ is an **EDCS** and $(3, 0)$ is the answer to $\mathcal{P}_1$. For any integer k, since $3k \equiv 0 \pmod 3$, there is a tuple $\langle 3l, 3s\rangle \in \mathcal{P}_1$ satisfying $3k \equiv 3s \pmod{3l}$. Hence $3k = 3lt + 3s$ for some integer t $(t \in \mathbb{Z})$. Therefore $k = lt + s$. That is $k \equiv s \pmod l$. Since $\langle l, s\rangle \in \mathcal{P}$, we can conclude that $\mathcal{P}$ is a **CS**.

From Theorem 1 we conclude that **EDCP** is **coNP**-hard.

Next, we show that **EDCP** is in **coNP**. This can be proved by reduction to **CP**. Suppose $\mathcal{P}$ is the set of ordered pairs $\{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \ldots, \langle l_n, s_n\rangle\}$. Then let $p = gcd(l_1, l_2, \ldots, l_n)$. From the definition of **EDCP**, we know that $p > 0$. Suppose $0 < p_1 < p_2 < \ldots < p_m$ are the distinct divisors of p. We look for an integer k such that $k = \max\{l | \forall i \forall j (\langle l_i, s_i\rangle \in \mathcal{P} \wedge \langle l_j, s_j\rangle \in \mathcal{P} \wedge s_i \equiv s_j \pmod{p_l})\}$. If there is not such k, then from Lemma 4 we know that $\mathcal{P}$ is not an **EDCS**. Hence suppose there is a k satisfying the condition. Denote $s = (s_1 \bmod p_k)$. We construct the following set $\mathcal{Q}$ of ordered pairs, $\{\langle \frac{l_1}{p_k}, \frac{s_1-s}{p_k}\rangle, \langle \frac{l_2}{p_k}, \frac{s_2-s}{p_k}\rangle, \ldots, \langle \frac{l_n}{p_k}, \frac{s_n-s}{p_k}\rangle\}$. We have the following relationship between $\mathcal{P}$ and $\mathcal{Q}$.

Claim. $\mathcal{Q}$ is a **CS** iff $\mathcal{P}$ is an **EDCS**.

Proof. Suppose $\mathcal{Q}$ is a **CS**. Then let $x = s$ and $y = p_k$. For any integer j, such that $j \equiv s_i \pmod{l_i}$ for some i $(1 \leq i \leq n)$, there is j_1 such that $j = j_1 l_i + s_i = j_1 j' y + j'' y + s = (j_1 j' + j'')y + x$. Hence $x \equiv j \pmod{y}$. For any integer j such that $x \equiv j \pmod{y}$, there is j_1 satisfying $j = j_1 y + x$. Since $\mathcal{Q}$ is a **CS**, there is an integer j_2 such that $j_1 = j_2 \frac{l_i}{p_k} + \frac{s_i - s}{p_k}$ for some i $(1 \leq i \leq n)$. Then $j_1 p_k = j_2 l_i + s_i - s$, so $j = j_1 y + x = j_2 l_i + s_i - s + x = j_2 l_i + s_i$. Therefore $\mathcal{P}$ is an **EDCS**.

On the other hand, suppose $\mathcal{P}$ is an **EDCS**. From Lemma 4, we know that (p_k, s) is the answer to $\mathcal{P}$. For any integer j', let $J = j' p_k + s$. Since $s \equiv J \pmod{p_k}$, $J \equiv s_i \pmod{l_i}$ for some i $(1 \leq i \leq n)$. Hence there is an integer j_1 such that $J = j_1 l_i + s_i = j_1 p_k \frac{l_i}{p_k} + p_k \frac{s_i - s}{p_k} + s = (j_1 \frac{l_i}{p_k} + \frac{s_i - s}{p_k}) p_k + s = j' p_k + s$. Since $p_k \neq 0$, $j' = j_1 \frac{l_i}{p_k} + \frac{s_i - s}{p_k}$. Hence $\mathcal{Q}$ is a **CS**. □

Then to decide whether $\mathcal{P}$ is an **EDCS**, we only need to check whether $\mathcal{Q}$ is a **CS**. Since **CP** is in **coNP** and the computations of p, p_k and s take polynomial time, **EDCP** is in **coNP**.

We conclude that **EDCP** is **coNP**-complete. □

4.2 The Complexity of Determinism of Unary Languages

In this section, we will discuss the complexity of deciding determinism of unary languages.

For an expression E, suppose $\mathcal{S}(E) = \{\langle l_1, s_1\rangle, \langle l_2, s_2\rangle, \ldots, \langle l_n, s_n\rangle\}$. Define $L(\mathcal{S}(E)) = \bigcup_{\langle l,s\rangle \in \mathcal{S}(E)} \{kl + s | k \in \mathbb{Z} \wedge kl + s \geq 0\}$.

To build the relations between $L(E)$ and $L(\mathcal{S}(E))$, we need the following lemma.

Lemma 5 ([1]). *For every deterministic language L, the following statements hold:* (1) *If string $w \in L$, then the language $L \setminus \{w\}$ is deterministic;* (2) *If string $w \notin L$, then the language $L \cup \{w\}$ is deterministic.*

Then we can simplify $\mathcal{S}(E)$ in the following ways: (1) Delete all tuples $\langle 0, s\rangle$; (2) For $\langle l, s\rangle$, where $l \leq s$, we replace $\langle l, s\rangle$ with $\langle l, s \bmod l\rangle$. This process just deletes or adds a finite number of words to $L(\mathcal{S}(E))$. From Lemma 5, it does not change determinism of the language. From now on, when we say $\mathcal{S}(E)$, we mean the simplified one. After the simplification, any tuple $\langle l, s\rangle \in \mathcal{S}(E)$ satisfies $0 \leq s < l$. Moreover, from the definition of $\mathcal{S}$, it is easy to see that $\sum_{i=1}^{|\mathcal{S}(E)|} (l_i + s_i) \leq 2|E|^2$, where $\langle l_i, s_i\rangle \in \mathcal{S}(E)$.

From the construction of $\mathcal{S}(E)$, the relations between $L(E)$ and $L(\mathcal{S}(E))$ can be characterized as follows.

Theorem 3. *Let E be an expression. There exists a number $M \in \mathbb{N}$ such that $|L(\mathcal{S}(E)) \setminus L(E)| + |L(E) \setminus L(\mathcal{S}(E))| \leq M$.*

Corollary 3. *Given an expression E, $L(\mathcal{S}(E))$ is deterministic if and only if $L(E)$ is deterministic.*

Then to decide determinism of $L(E)$, we can check determinism of $L(\mathcal{S}(E))$.

For a unary language, the corresponding minimal DFA consists of a chain of states or a chain followed by a cycle [9]. Then to check determinism we need the following characterization of determinism of unary languages.

Lemma 6 ([9]). *Let $\Sigma = \{a\}$, and L be a regular language, then L is a deterministic language if and only if L is finite or the cycle of the minimal DFA of L has at most one final state.*

From the definition of $L(\mathcal{S}(E))$ and Lemma 6, we can easily see that to check determinism of $L(\mathcal{S}(E))$ we only need to check whether $\mathcal{S}(E)$ is an **EDCS**.

Theorem 4. *Suppose $L(\mathcal{S}(E))$ is infinite. $L(\mathcal{S}(E))$ is deterministic if and only $\mathcal{S}(E)$ is an* **EDCS**.

Proof. ($\Rightarrow$) Since $L(\mathcal{S}(E))$ is deterministic, the cycle of the minimal DFA of $L(\mathcal{S}(E))$ has at most one final state. Let the size of the cycle be p and $n = |\mathcal{S}(E)|$. Denote the start state as q_0 and the only final state in the cycle as q_1. Suppose w is the shortest word such that $\delta(q_0, w) = q_1$. Because $l_i > 0$ $(1 \leq i \leq n)$, there is an integer k' $(k' > 0)$ such that $M = k' \prod_{i=1}^{n} l_i$ and $M > w$. Since $L(\mathcal{S}(E))$ is infinite, $p \neq 0$. For any $\langle l_i, s_i\rangle$ $(1 \leq i \leq n)$, since q_1 is the only final state in the cycle, there is an integer k such that $kl_i + s_i > w$, $\delta(q_0, kl_i + s_i) = q_1$ and $\delta(q_0, kl_i + l_i + s_i) = q_1$. Then $p|l_i$, $p|M$, and there is an integer k_1 such that $w + k_1 p = kl_i + s_i$. Hence $w \equiv s_i \pmod p$. Let $s = (w \bmod p)$. We prove that (p, s) is the answer to $\mathcal{S}(E)$. For any integer k satisfying $s \equiv k \pmod p$, there is an integer k_1' such that $k_1' > 0$ and $k + k_1' M > w$ and $(k + k_1' M) \equiv s \pmod p$. So there is an integer k'' satisfying $k + k_1' M = w + k'' p$. That is $k + k_1' M \in L(\mathcal{S}(E))$. Hence $(k + k_1' M) \equiv s_i \pmod{l_i}$ for some i $(1 \leq i \leq n)$. Because $l_i | M$, $k \equiv s_i \pmod{l_i}$. For any integer k satisfying $k \equiv s_i \pmod{l_i}$ for some i $(1 \leq i \leq n)$, there is an integer k_1 such that $k = k_1 l_i + s_i$. So

$$\begin{aligned} k &\equiv k_1 l_i + s_i \pmod p \\ &\equiv s_i \pmod p \end{aligned}$$

$$\equiv w \pmod{p}$$
$$\equiv s \pmod{p}$$

Hence (p, s) is the answer to $\mathcal{S}(E)$. Therefore $\mathcal{S}(E)$ is an **EDCS**.

($\Leftarrow$) Since $\mathcal{S}(E)$ is an **EDCS**, suppose (p, s) is the answer to $\mathcal{S}(E)$. If $L(\mathcal{S}(E))$ is not deterministic, then there are two final states p_1 and p_2 in the cycle of the minimal DFA of $L(\mathcal{S}(E))$. Since p_1, p_2 are not equivalent, there is a word k such that $\delta(q_1, k) \in L(\mathcal{S}(E)) \wedge \delta(q_2, k) \notin L(\mathcal{S}(E))$ or $\delta(q_1, k) \notin L(\mathcal{S}(E)) \wedge \delta(q_2, k) \in L(\mathcal{S}(E))$. Suppose the case $\delta(q_1, k) \in L(\mathcal{S}(E)) \wedge \delta(q_2, k) \notin L(\mathcal{S}(E))$ holds. Denote the start state as q_0. Then there are words k_1 and k_2 such that $\delta(q_0, k_1) = q_1$ and $\delta(q_0, k_2) = q_2$. Since q_1 and q_2 are final states, $k_1 \in L(\mathcal{S}(E))$ and $k_2 \in L(\mathcal{S}(E))$. From $\mathcal{S}(E)$ is an **EDCS**, we have $k_1 \equiv s \pmod{p}$ and $k_2 \equiv s \pmod{p}$. Because $\delta(q_1, k) \in L(\mathcal{S}(E))$, $k + k_1 \equiv s \pmod{p}$. Hence $p|k$. However, from $\delta(q_2, k) \notin L(\mathcal{S}(E))$, we have $k + k_2 \not\equiv s \pmod{p}$. So $p \not| k$, which is a contradiction. The case $\delta(q_1, k) \notin L(\mathcal{S}(E)) \wedge \delta(q_2, k) \in L(\mathcal{S}(E))$ can be proved in a similar way. Hence $L(\mathcal{S}(E))$ is deterministic. □

From Theorem 2 and Theorem 4, we can obtain the main result of the paper.

Theorem 5. *Given a regular expression E over a unary alphabet, the problem of deciding whether $L(E)$ is deterministic is* **coNP***-complete.*

For any expression $E = E_1^*$, we have $|\mathcal{S}(E_1^*)| = 1$ from the definition of $\mathcal{S}$. Then $\mathcal{S}(E_1^*)$ is an **EDCS**. Hence we can easily obtain the following theorem.

Theorem 6 ([15]). *Let L be any language over a unary alphabet. Then L^* is deterministic.*

5 Conclusion and Future Work

In this paper, we give the complexity of deciding determinism of regular languages over a unary alphabet. By studying unary languages, we can conclude that the problem, whether a language can be defined by a deterministic expression with counting, is **coNP**-hard.

There are a few problems for future research. It is easy to see that we have only handled standard regular expressions. What is the complexity when the input is an expression with counting? To solve this problem in the way we used in this paper, we have to handle the following problems: (1) For a word w and an expression E with counting over a unary alphabet, can the membership problem be tested in time $O(\log^k |w|)$ for some integer $k > 0$? (2) For an expression E with counting over a unary alphabet, how to define $\mathcal{S}(E)$? The hardness of these problems mainly comes from the fact that we do not have a good tool to handle expressions with counting.

Acknowledgments. We thank Wim Martens for sending us the full version of the paper [15].

References

1. Bex, G.J., Gelade, W., Martens, W., Neven, F.: Simplifying XML schema: effortless handling of nondeterministic regular expressions. In: SIGMOD 2009, pp. 731–743 (2009)
2. Bickel, K., Firrisa, M., Ortiz, J., Pueschel, K.: Constructions of Coverings of the Integers: Exploring an Erdős problem. Summer Math Institute, Cornell University (2008)
3. Brüggemann-Klein, A.: Regular expressions into finite automata. Theoretical Computer Science 120(2), 197–213 (1993)
4. Brüggemann-Klein, A., Wood, D.: One-unambiguous regular languages. Information and Computation 142(2), 182–206 (1998)
5. Chen, H., Lu, P.: Assisting the Design of XML Schema: Diagnosing Nondeterministic Content Models. In: Du, X., Fan, W., Wang, J., Peng, Z., Sharaf, M.A. (eds.) APWeb 2011. LNCS, vol. 6612, pp. 301–312. Springer, Heidelberg (2011)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. The MIT Press, Cambridge (2001)
7. Erdős, P.: On integers of the form $2^k + p$ and some related problems. Summa Brasil. Math. 2, 113–123 (1950)
8. Garrey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman (1979)
9. Gelade, W., Gyssens, M., Martens, W.: Regular expressions with counting: weak versus strong determinism. SIAM J. Comput. 41(1), 160–190 (2012)
10. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics: a foundation for computer science, 2nd edn. Addison-Wesley (1994)
11. Groz, B., Maneth, S., Staworko, S.: Deterministic regular expressions in linear time. In: PODS 2012, pp. 49–60 (2012)
12. Guy, R.K.: Unsolved problems in Number Theory, 3rd edn. Problem Books in Math. Springer, New York (2004)
13. Holzer, M., Kutrib, M.: The complexity of regular(-like) expressions. In: Gao, Y., Lu, H., Seki, S., Yu, S. (eds.) DLT 2010. LNCS, vol. 6224, pp. 16–30. Springer, Heidelberg (2010)
14. Kilpeläinen, P.: Checking determinism of XML Schema content models in optimal time. Informat. Systems 36(3), 596–617 (2011)
15. Losemann, K., Martens, W., Niewerth, M.: Descriptional complexity of deterministic regular expressions. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) MFCS 2012. LNCS, vol. 7464, pp. 643–654. Springer, Heidelberg (2012)
16. Sawa, Z.: Efficient construction of semilinear representations of languages accepted by unary NFA. In: Kučera, A., Potapov, I. (eds.) RP 2010. LNCS, vol. 6227, pp. 176–182. Springer, Heidelberg (2010)
17. Schnitger, G.: Regular expressions and NFAs without ϵ-transitions. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 432–443. Springer, Heidelberg (2006)
18. Steen, L.A., Seebach, J.A.: Counterexamples in topology, 2nd edn. Springer, New York (1978)
19. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time: preliminary report. In: STOC 1973, pp. 1–9 (1973)
20. van der Vlist, E.: XML Schema. O'Reilly (2002)
21. World Wide Web Consortium, `http://www.w3.org/wiki/UniqueParticleAttribution`
22. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular langauges. Theoretical Computer Science 125(2), 315–328 (1994)