# 确定型语言的相关研究综述

陈海明 陆 平

（中国科学院软件研究所计算机科学国家重点实验室 北京 100190）

**摘 要** 随着互联网应用的不断普及，XML（可扩展标记语言）在数据库、数据传输等方面的作用也越来越大。一般而言，XML 文档的结构都是由 XML 模式语言来定义，比如 DTD 和 XML Schema。文章主要对 DTD 和 XML Schema 的内容模式约束，即确定型正则表达式进行研究，分别详细介绍确定型表达式、确定型语言、及相关问题的研究现状。文章首先给出确定型表达式的定义及相关概念。接着，介绍几种确定型表达式的判定算法。然后，分析确定型语言的判定、对应确定型表达式的生成、及近似确定型表达式的生成等问题。 文章最后还列举一些其他相关问题的研究。

**关键词** 正则表达式；确定型语言；算法；复杂度

## Review on Deterministic Languages

CHEN Haiming LU Ping

( *State Key Laboratory of Computer Science*, *Institute of Software Chinese Academy of Sciences*, *Beijing* 100190, *China* )

**Abstract** With the development of network applications, XML has become more and more important. In general, the structures of XML documents are specified by schema languages, e.g., DTD and XML Schema. In this paper, the definition of deterministic regular expressions was introduced and an overview of current research was provided. At first, the definition of deterministic regular expressions and some relative notations were given. Next, some algorithms checking determinism of regular expressions were presented in details. Then, some results about deciding, learning, approximating deterministic languages were shown. Finally, some other relative topics were discussed.

**Keywords** regular expression; deterministic language; algorithm; complexity

## 1 引 言

XML（可扩展标记语言）在数据库[1]、网络数据传输[2]等方面有着重要的应用。XML 文档的类型和结构都是通过 XML 模式语言进行定义。常见的模式语言有 DTD 和 XML Schema。然而，在实际使用中，有很多 XML 模式定义都不符合 W3C 标准[3,4]。其中一个重要原因是因为 W3C 要求元素的内容模式必须是确定型正则表达式[5]。例如， 以下 DTD 描述的是一个 DVD 音像店的货存情况[4]：

$$store \rightarrow \quad dvd \quad dvd^*$$

$$dvd \rightarrow \quad title \quad price$$

箭头右边的部分称为内容模式。它们可以分别抽象为如下两个正则表达式：$aa^*$、$bc$。 据 W3C 的标准，要求 $aa^*$ 和 $bc$ 必须是确定型正则表达式。

由于确定型正则表达式的定义基于语义[6]，且目前还没有一个简单的文法定义，因此给设计正确的内容模式带来了很大的困难。对确定型正则表达式进行深入研究，将有助于设计人员书写正确的内容模式。

**基金项目**：国家自然科学基金资助项目（61070038）。

**作者简介**：陈海明（通讯作者），博士，研究员，博士生导师，研究方向为软件设计、计算模型、程序语言，E-mail：chm@ios.ac.cn；陆平，博士研究生，研究方向为正则表达式、自动机、算法。

## 2 确定型表达式的定义

假设 $\Sigma$ 是一个字母表。标准正则表达式 $E$ 可以递归地定义如下：$\varepsilon$ 和 $a\in\Sigma$ 是正则表达式；假设 $E_1$ 和 $E_2$ 是两个任意的标准正则表达式，将他们进行选择运算 $E_1+E_2$、连接运算 $E_1\cdot E_2$ 和星号运算 $E_1^*$ 的结果仍然是标准正则表达式。正则表达式 $E$ 中所有字母出现的次数总和，称为 $E$ 的长度，记为 $|E|$。

给定一个正则表达式 $E$，我们对 $E$ 中出现的每一个字符都进行标号，使每一个带标号的字符在标号后的表达式中只出现一次。我们将标号后的表达式记为 $\overline{E}$，其中 $\overline{E}$ 中所有带标号的字符称为 $E$ 的位置，而 $E$ 的所有位置的集合记为 $pos(E)$。例如，$E=a^*a$。$\overline{E}=a_1^*a_2$ 为 $E$ 的一个标号表达式，则 $E$ 的位置集合 $pos(E)=\{a_1,a_2\}$。与标号操作相对应的是去标号。假设 $F$ 是一个带标号的正则表达式，去掉 $F$ 中所有标号后所得到的表达式记为 $F^\natural$。例如，令 $F=\overline{E}=a_1^*a_2$，则 $F^\natural=a^*a$。对于字和集合中元素进行加标号和去标号操作后的结果，我们使用相同的记号。利用这些记号，我们可以定义确定型表达式如下：

定义1[6] 给定一个正则表达式 $E$，若 $L(\overline{E})$ 中的任意两个字 $uxv$ 和 $uyw(|x|=|y|=1)$，当满足 $x\neq y$ 时，$x^\natural\neq y^\natural$ 成立，则称 $E$ 是确定型表达式。若 $L(E)$能被一个确定型表达式所描述，则 $L(E)$为确定型语言。

例如，正则表达式 $E=a^*a$ 不是确定型表达式。因为存在两个字 $a_2\in L(a_1^*a_2)$，$a_1a_2\in L(a_1^*a_2)$，令 $u=\varepsilon$，$x=a_2$，$y=a_1$，$v=\varepsilon$ 和 $w=a_2$，由于 $x\neq y$ 且 $x^\natural=y^\natural=a$ 成立，按照定义 $E$ 不是确定型表达式。但 $L(a^*a)$是确定型语言，因为 $L(a^*a)=L(aa^*)$，且 $aa^*$ 是确定型表达式。直观地说，对于 $L(aa^*)$中的每一个字，它的第一个字符都匹配 $aa^*$ 中的头一个 $a$，其余的字符都匹配第二个 $a$。

非确定型有限自动机[7](NFA)由满足如下条件的五元组$(Q,\Sigma,\delta,q_0,F)$组成：(1)$Q=\{q_0,q_1,\ldots,q_n\}$是由有限的状态组成的集合；(2)$\Sigma$ 是输入字符的集合；(3)$F\subseteq Q$ 是可接受状态的集合；(4)$\delta:Q\times\Sigma\to 2^Q$ 是迁移函数。

确定型有限自动机(DFA)[7]也是由五元组$(Q,\Sigma,\delta,q_0,F)$ 组成，与 NFA 的唯一区别在于迁移函数定义为 $\delta:Q\times\Sigma\to Q$。其中，状态数 $|Q|$ 称为 DFA 的大小。最小确定型有限自动机定义为接受相同语言的所有 DFA 中状态数最少的 DFA。对于任意正则语言，其对应的最小确定型有限自动机在同构意义下是唯一的[7]。

假设 $E$ 是一个正则表达式，我们定义如下的函数和集合[8]：

$$\lambda(E)=\begin{cases}true & if\ \varepsilon\in L(E),\\ false & otherwise\end{cases}$$

$$first(E)=\{a\mid aw\in L(E),a\in\Sigma,w\in\Sigma^*\}$$

$$last(E)=\{a\mid wa\in L(E),a\in\Sigma,w\in\Sigma^*\}$$

$$follow(E,a)=\{b\mid uabv\in L(E),\ u\in\Sigma^*,v\in\Sigma^*,b\in\Sigma\},a\in\Sigma$$

$$followLast(E)=\{b\mid v,vbw\in L(E),\ v\neq\varepsilon,b\in\Sigma,w\in\Sigma^*\}$$

这些函数和集合都可以在表达式的语法树上按照自底向上的方式进行计算[8]。

## 3 确定型表达式的判定

本节处理如下问题：给定正则表达式 $E$，判定 $E$ 是否为确定型正则表达式。

首先我们介绍一类和确定型正则表达式相对应的有限自动机—Glushkov 自动机。给定正则表达式 $E$，其对应的 Glushkov 自动机 $G_E=(Q_E\cup\{q_I\},\Sigma,q_I,F_E)$[8] 定义如下：

(1)$Q_E=pos(E)$;

(2) $\forall a\in\Sigma$，$\delta_E(q_I,a)=\{x\mid x\in first(\overline{E}),x^\natural=a\}$;

(3)对于 $E$ 的任意位置 $x\in pos(E)$，$a\in\Sigma$，$\delta_E(x,a)=\{y\mid y\in follow(\overline{E},x),y^\natural=a\}$。

$$(4)\ F_E=\begin{cases}last(\overline{E})\cup\{q_I\}, & \varepsilon\in L(E),\\ last(\overline{E}), & \varepsilon\notin L(E).\end{cases}$$

对于 $E=a^*a$，$\overline{E}=a_1^*a_2$ 是 $E$ 的一个标号表达式。利用 $\overline{E}$，我们计算如下相关集合：状态集合为 $Q=\{a_1,a_2\}\cup\{q_I\}$；因为 $first(\overline{E})=\{a_1,a_2\}$，$follow(\overline{E},a_1)=\{a_1,a_2\}$，$follow(\overline{E},a_2)=\varnothing$，所以迁移函数定义为 $\delta_E(q_I,a)=\{a_1,a_2\}$ 和 $\delta_E(a_1,a)=\{a_1,a_2\}$；可接受状态的集合为 $F_E=last(\overline{E})=\{a_2\}$。最后，$E$ 的 Glushkov 自动机 $G_E$ 构造如下：

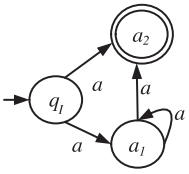


图 1 $E$ 的 Glushkov 自动机

正则表达式 $E$ 的确定性和 $E$ 的 Glushkov 自动机有如下的关系：

定理 1[8]　$E$ 是确定型表达式当且仅当 $G_E$ 是 DFA。

由于图 1 中的有限自动机不是 DFA，所以 $E=a^*a$ 不是确定型表达式。而 $E_1=aa^*$ 是确定型的，因为 $E_1$ 的Glushkov 自动机是 DFA（如图 2 所示）。

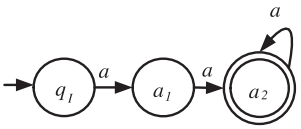


图 2　$E_1$ 的 Glushkov 自动机

我们还可以利用语法树来判定确定型正则表达式。这个方法的基础是确定型表达式的如下刻画：

定理 2[9]　假设 $E$ 是一个正则表达式。

(1) $E=\varepsilon$ 或 $a\in\Sigma$：$E$ 是确定型表达式。

(2) $E=E_1+E_2$：$E$ 是确定型表达式当且仅当 $E_1$ 和 $E_2$ 都是确定型表达式，并且 $first(E_1)\cap first(E_2)=\varnothing$。

(3) $E=E_1E_2$：若 $\varepsilon\in L(E_1)$，则 $E$ 是确定型表达式当且仅当 $E_1$ 和 $E_2$ 都是确定型表达式，$first(E_1)\cap first(E_2)=\varnothing$，且 $followLast(E_1)\cap first(E_2)=\varnothing$。若 $\varepsilon\notin L(E_1)$，则 $E$ 是确定型表达式当且仅当 $E_1$ 和 $E_2$ 都是确定型表达式，且 $followLast(E_1)\cap first(E_2)=\varnothing$。

(4) $E=E_1^*$：$E$ 是确定型表达式当且仅当 $E_1$ 是确定型表达式，并满足如下条件：$\forall x\in followLast(\overline{E_1})$，$\forall y\in first(\overline{E_1})$，$x^\natural=y^\natural\rightarrow x=y$。

由于这个方法既使用了原表达式，又使用了标号表达式，所以在判定过程中需要特殊处理 *followLast* 集合[9]。为了克服这个问题，我们定义了如下的函数，使整个判定的过程仅在原表达式上进行。

定义 2[10]　给定一个正则表达式 $E$，函数 $P(E)$ 定义如下：

$$P(\varepsilon)=P(a)=true\qquad a\in\Sigma$$

$$\begin{aligned}P(E_1+E_2)=&P(E_1)\wedge P(E_2)\\&\wedge(first(E_1)\cap followLast(E_2)=\varnothing)\\&\wedge(followLast(E_1)\cap first(E_2)=\varnothing)\end{aligned}$$

$$\begin{aligned}P(E_1E_2)=&(\neg\lambda(E_1)\wedge\neg\lambda(E_2)\\&\wedge(first(E_1)\cap followLast(E_2)=\varnothing))\vee\\&(\lambda(E_1)\wedge\neg\lambda(E_2)\wedge P(E_2)\\&\wedge(first(E_1)\cap followLast(E_2)=\varnothing))\vee\\&(\neg\lambda(E_1)\wedge\lambda(E_2)\wedge P(E_1)\\&\wedge(first(E_1)\cap followLast(E_2)=\varnothing)\\&\wedge(first(E_1)\cap first(E_2)=\varnothing))\vee\\&(\lambda(E_1)\wedge\lambda(E_2)\wedge P(E_1)\wedge P(E_2)\\&\wedge(first(E_1)\cap followLast(E_2)=\varnothing))\end{aligned}$$

$$P(E_1^*)=P(E_1)$$

定理 2 中关于标号表达式的条件可以完全替换成关于 $P$ 函数的计算。

定理 3[10]　$E^*$ 是确定型表达式当且仅当 $E$ 是确定型表达式并且 $P(E)=true$。

由 *followLast* 的计算需要 $O(|\Sigma_E||E|)$时间（其中，$\Sigma_E$ 是表达式 $E$ 中出现的不同字符的集合），所以这两个判定方法的运行时间都是 $O(|\Sigma_E||E|)$。

最近，Groz 等人给出了第一个判定确定型表达式的线性算法[11]。这个算法的一个最重要思路是：利用后缀数组将表达式的语法树按出现的字母分解成多颗小的语法树，然后在小的语法树上进行判定。

## 4　确定型正则语言的相关问题

由于确定型正则语言是正则语言的一个真子集[6]，一个很自然的问题就是：给定正则表达式 $E$，如何判定 $L(E)$ 是否为确定型语言？本节我们主要详细介绍这个问题的有关结论及一些相关问题。

### 4.1　确定型正则语言的判定

最早的判定算法出现在文献[6]中。给定正则表达式 $E$，该算法首先构造 $L(E)$ 的最小自动机 $M$，然后在 $M$ 上进行递归地判定。由于篇幅有限，感兴趣的读者可以参考文献[6]和[12]。因为构造了最小自动机，所以算法需要 $O(2^{|E|})$ 的空间。

最近，Czerwiński 等人发现这个算法有如下特性[13]：给定正则表达式 $E$，算法递归的深度最多为 $O(|E|^2)$。利用该特性，他们给出了原算法[6]的一个 on-the-fly 版本，从而得到了一个多项式空间的算法。

对于这个问题的复杂度下界，最初的证明是从 Tiling 问题归约过来的[5,13]，然而这个证明比较复杂。在 Groz 的博士论文里[14]，他给出了一个更简洁的证明：将正则表达式的 UNIVERSALITY 问题（给定正则表达式 $E$，判断 $L(E)=\Sigma^*$ 是否成立？）归约到确定型语言的判定。结合[13]，[14]中的结论，我们可以得到：

定理 4[13]　给定正则表达式 $E$，判定 $L(E)$ 是否为确定型语言是 PSPACE 完全的。

### 4.2　确定型正则表达式的构造

本节我们考虑如下问题：给定正则表达式 $E$，如

果 $L(E)$ 是确定型语言，那么我们如何得到一个对应的确定型表达式。

最早实现这个功能的算法同样出现在文献[6]中。其分为如下两个步骤：(1)构造 $L(E)$ 的最小自动机 $M$；(2)从 $M$ 中生成确定型表达式 $E'$。Bex 等[5]对步骤(2)提出了一些优化，可以使 $|E'|$ 尽可能地小。然而，相对于 $M$ 的大小而言，生成的表达式的长度仍然是指数，即 $|E'|=O(2^{|M|})$。由于从表达式 $E$ 中构造最小自动机 $M$ 会引起状态的指数爆炸，所以我们有 $|E'|=O(2^{2^{|E|}})$。

Losemann 等证明存在正则表达式 $E$，满足 $L(E)$ 是确定型语言，且任何一个描述 $L(E)$ 的确定型表达式 $E'$ 的长度总是会出现一次指数爆炸[15]，即 $|E'|=\Omega(2^{|E|})$。目前还不知道在(1)和(2)两个步骤中，哪一步的指数爆炸是可以避免的[15]。

### 4.3 确定型正则语言的近似算法

本节介绍在 $L(E)$ 不是确定型语言的情况下，如何生成近似的确定型表达式 $E'$，使得 $L(E)\subseteq L(E')$。Ahonen[16]给出的方法是：在原始算法递归判定过程中，如果判断条件不能满足，则通过增加迁移使算法能够继续进行[5]。Bex 等[5]在 Ahonen 的方法的基础上增加了一些改进，如不同构自动机的生成等，使得最后生成的确定型表达式尽可能地小。

## 5 带数字出现的确定型正则表达式

这一节我们研究在实际应用中使用更为广泛的一类表达式：带数字出现的正则表达式。带数字出现的表达式对标准表达式进行了扩展，增加了带数字出现的操作符。其定义如下：如果 $E$ 是带数字出现的表达式，则 $E^{[m,n]}$、$E^{[m,\infty]}$也是带数字出现的表达式(其中，$\infty$ 表示无穷)。在 XML Schema 中，使用的就是这类表达式。

对于带数字出现的确定型表达式的判定，我们仍然可以使用文献[11]中的线性算法来判定。所以，本节我们主要研究确定型语言的判定。

### 5.1 确定型语言判定的复杂度

带数字出现的确定型表达式的表达能力较确定型标准表达式的强。例如，$(a^{[2,3]}b^{[0,1]})^{[0,\infty]}$ 是确定型表达式[17]，但不能被任何确定型标准表达式描述。目前，若给定正则表达式 $E$，还不能判定 $L(E)$ 是否能被一个带数字出现的确定型表达式所描述[17]。利用已有结论，我们可以给出这个问题的一个复杂度下界。

Gelade 等人证明[17]：在单字母表的情况下，确定型标准表达式和带数字出现的确定型表达式具有相同的表达能力。因此，通过研究在单字母表下，判定一个正则语言是否对应于一个确定型标准表达式的复杂度，可以给出以上问题的一个复杂度下界。

由于单字母语言和数字集合之间有对应关系，我们首先给出如下函数：

定义 3[18] 函数 $S(E)$ 定义如下：

$$S(\varepsilon)=\{\langle 0,0\rangle\}$$

$$S(a)=\{\langle 0,1\rangle\}\qquad a\in\Sigma$$

$$S(E_1+E_2)=S(E_1)\cup S(E_2)$$

$$S(E_1E_2)=\{\langle \gcd(l_i,l_j),s_i+s_j\rangle \mid \langle l_i,s_i\rangle\in S(E_1)\wedge\langle l_j,s_j\rangle\in S(E_2)\}$$

$$S(E^*)=\begin{cases}\{\langle 0,0\rangle\} & \text{若 } S(E_1)=\{\langle 0,0\rangle\},\\ \{\langle l,0\rangle \mid l=\max\{x \mid \forall l_i\forall s_i(\langle l_i,s_i\rangle\in S(E_1)\wedge(x\mid l_i)\wedge(x\mid s_i))\}.\end{cases}$$

给定正则表达式 $E$，$S(E)$ 包含一些等差数列。当且仅当 $S(E)$ 中的等差数列的并仍然构成一个等差数列[18]时，可以证明 $L(E)$ 是确定型语言。通过从覆盖系问题[19]归约到这个关于等差数列的问题，我们可以证明判定单字母表确定型语言是 co-NP 完全的[18]。

要完全解决带数字出现的确定型正则语言的判定问题，我们需要考虑以下两个问题：(1)对于带数字出现的正则表达式，是否存在一类自动机满足如下性质：表达式是确定型当且仅当对应的自动机是确定型；(2)能被带数字出现的确定型表达式所描述的语言所对应的最小自动机具有怎样的结构特点？

对于第一个问题，Gelade 等[17,20]给出了一类自动机：Counter Automata。这类自动机的构造和 Glushkov 自动机的构造很类似。但这类自动机是确定型的当且仅当对应的表达式是强确定型表达式[17]。Hovland[21]也定义了一类带 counter 的自动机：Finite Automata with Counters。然而，这种自动机的确定性仍然对应于表达式的强确定性。目前还没有找到满足所需要性质的自动机。

第二个问题对于解决整个问题至关重要。目前，这个问题的研究尚无实质性的进展。主要的问题在于带数字出现的操作符对表达式中的字符出现进行了压缩，使得最小自动机的状态和表达式中字符之间的对应关系变得更为复杂。

## 6　其他相关研究

字母单次出现表达式(SOREs)是确定型正则表达式的一个子类。在这类表达式中，每一个字符在表达式中仅出现一次。在实际使用的 XML 文档中，99% 的内容模式都是由 SOREs 描述[4]。针对 SOREs 的研究有很多，但主要集中在通过样本学习表达式[22-24]。Freydenberger 和 Kötzing[23]给出了一个算法，使得学习出来的表达式 $E$ 满足：(1)$L(E)$包含所有的样本；(2)任何包含所有样本的字母单次出现表达式 $E'$ 都满足 $L(E)\subseteq L(E')$。

强确定型表达式主要应用于 XML 数据流的解析[25]。这类表达式是确定型表达式的一种扩展，它不仅要求字匹配正则表达式时匹配的位置唯一，而且要求匹配过程中操作符的使用也是唯一确定的。例如，$(a^*)^*$，因为$(a^*)^*$仅有一个位置，所以$(a^*)^*$是确定型表达式，但这个表达式不是强确定型的，因为其中的两个星号操作符的使用顺序不能唯一确定。Koch 等[25]和 Gelade 等[17]分别给出了不同的判定强确定型表达式的 $O(|E|^3)$算法。之后，我们给出了一个$O(|\Sigma_E||E|)$的判定算法[26]。

文献[27]给出了两种判定确定型表达式之间包含关系的算法。文献[28]提出了一个判定正则表达式之间包含关系的推导系统。基于这个推导系统，文献[28]给出的包含判定算法还可以进一步判定当其中一个表达式不是确定型表达式时的情况。

## 7　结束语

确定型正则表达式在实际中有比较广泛的应用，深入研究这类表达式能够为网络应用的设计提供很大的帮助。本文详细地介绍了确定型表达式及其相关问题的研究现状，希望能对相关研究起到理论参考的作用。

### 参 考 文 献


[1] Bourret R. XML and Databases [R]. Technical University of Darmstadt, 2000, http://www.rpbourret.com/xml/XMLAndDatabases.html.

[2] Amer-Yahia S, Kotidis Y. A web-service architecture for efficient XML data exchange [C] // The 20th International Conference on Data Engineering, 2004: 523-534.

[3] World Wide Web Consortium [EB/OL]. http://www.w3.org/wiki/UniqueParticleAttribution.

[4] Bex GJ, Neven F, Bussche JV. DTDs versus XML schema: a practical study [C] // The 7th International Workshop on the Web and Databases, 2004: 79-84.

[5] Bex GJ, Gelade W, Martens W, et al. Simplifying XML schema: effortless handling of nondeterministic regular expressions [C] // The ACM International Conference on Management of Data, 2009: 731-743.

[6] Brüggemann-Klein A, Wood D. One-unambiguous regular languages regular languages [J]. Information and Computation, 1998, 142(2): 182-206.

[7] Hopcroft JE, Motwani R, Ullman JD. Introduction to Automata Theory, Languages, and Computation [M]. Addison-Wesley, 2007.

[8] Brüggemann-Klein A. Regular expressions into finite automata [J]. Theoretical Computer Science, 1993, 120(2): 197-213.

[9] Kilpeläinen P. Checking determinism of XML schema content models in optimal time [J]. Information Systems, 2011, 36(3): 596-617.

[10] Chen HM, Lu P. Assisting the design of XML schema: diagnosing nondeterministic content models [C] // The 13th Asia-Pacific Web Conference, 2011: 301-312.

[11] Groz B, Maneth S, Staworko S. Deterministic regular expressions in linear time [C] // The 31st ACM Symposium on Principles of Database Systems, 2012: 49-60.

[12] Gelade W. Foundations of XML: regular expressions revisited [D]. Diepenbeek: University Hasselt, 2009.

[13] Czerwiński W, David C, Losemann K, et al. Deciding definability by deterministic regular expressions [C] // The 16th International Conference on Foundations of Software Science and Computation Structures, 2013: 289-304.

[14] Groz B. XML Security views: Queries, Updates and Schemas [D]. Lille: Université de Lille 1, 2012.

[15] Losemann K, Martens W, Niewerth M. Descriptional complexity of deterministic regular expressions [C] // The 37th International Symposium on Mathematical Foundations of Computer Science, 2012: 643-654.

[16] Ahonen H. Disambiguation of SGML content models [C] // The 3rd International Workshop on Principles of Document Processing, 1996: 27-37.

[17] Gelade W, Gyssens M, Martens W. Regular expressions with counting: weak versus strong determinism [J]. SIAM Journal on Computing, 2012, 41(1): 160-190.

[18] Lu P, Peng FF, Chen HM. Deciding determinism of unary languages is coNP-complete [C] // The 17th International Conference on Developments in Language Theory, 2013: 350-361.

[19] Garrey MR, Johnson DS. Computers and Intractability: a Guide to the Theory of NP-completeness [M]. San Francisco: Freeman, 1979.

[20] Sperberg-McQueen CM. Notes on finite state automata with

counters [EB/OL], http://www.w3.org/XML/2004/05/msm-cfa.html, 2004.

[21] Hovland D. The membership problem of regular expressions with unordered concatenation and numerical constraints [C] // The 6th International Conference on Language and Automata Theory and Applications, 2012: 313-324.

[22] Bex GJ, Neven F, Schwentick T, et al. Inference of concise DTDs from XML data [C] // The 32nd International Conference on Very Large Data Bases, 2006: 115-126.

[23] Freydenberger DD, Kötzing T. Fast Learning of restricted regular expressions and DTDs [C] // Joint 2013 EDBT/ICDT Conference, 2013: 45-56.

[24] Bex GJ, Gelade W, Neven F, et al. Learning deterministic regular expressions for the inference of schemas from XML data [J]. ACM Transactions on the Web, 2010, 4(4): 14.

[25] Koch C, Scherzinger S. Attribute grammars for scalable query processing on XML streams [J]. VLDB Journal, 2007, 16(3): 317-342.

[26] Chen HM, Lu P. Checking determinism of regular expressions with counting [C] // The 16th International Conference on Developments in Language Theory, 2012: 332-343.

[27] Chen H, Chen L. Inclusion test algorithms for one-unambiguous regular expressions [C] // The 5th International Colloquium on Theoretical Aspects of Computing, 2008: 96-110.

[28] Hovland D. The inclusion problem for regular expressions [J]. Journal of Computer and System Sciences, 2012, 78(6): 1795-1813.