Practical Study of Subclasses of Regular Expressions in DTD and XML Schema

Yeting Li[2], Xiaolan Zhang[1,2], Feifei Peng[1,2], and Haiming Chen[1(✉)]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
{zhangxl,pengff,chm}@ios.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China
liyeting@snnu.edu.cn

Abstract. DTD and XSD are two popular schema languages widely used in XML documents. Most content models used in DTD and XSD essentially consist of restricted subclasses of regular expressions. However, existing subclasses of content models are all defined on standard regular expressions without considering counting and interleaving. Through the investigation on the real world data, this paper introduces a new subclass of regular expressions with counting and interleaving. Then we give a practical study on this new subclass and five already known subclasses of content models. One distinguishing feature of this paper is that the data set is sufficiently large compared with previous relevant work. Therefore our results are more accurate. In addition, based on this large data set, we analyze the different features of regular expressions used in practice. Meanwhile, we are the first to simultaneously inspect the usage of the five subclasses and analyze different reasons dissatisfying the corresponding definitions. Furthermore, since W3C standard requires the content models to be deterministic, the determinism of content models is also tested by our validation tools.

Keywords: XML · DTD · XML schema · Interleaving · Counting

1 Introduction

As a main file format for data exchange, the eXtensible Markup Language (XML) has been widely used on the web [1]. The presence of a schema provides a lot of conveniences and advantages for various applications such as data processing, automatic data integration, static analysis of transformations and so on [3,12,20–23,27,31]. DTD (Document Type Definitions) and XSD (XML Schema Definitions) are two popular schema languages recommended by W3C (World Wide Web Consortium) [30]. Most content models used in DTD and XSD essentially consist of restricted subclasses of regular expressions. Therefore for practical purpose many researches focus on the study of subclasses practically used.

Work supported by the National Natural Science Foundation of China under Grant Nos. 61472405, 61070038.

F. Li et al. (Eds.): APWeb 2016, Part II, LNCS 9932, pp. 368–382, 2016.
DOI: 10.1007/978-3-319-45817-5_29

In 2004, Bex et al. proposed a subclass called the simple regular expressions after analyzing 109 DTDs and 93 XSDs from Cover Pages [6], which became the basis of later work. Martens et al. discussed the complexity of decision problems for eCHARE, an extension of simple regular expression, and still call it simple regular expression [24]. Later eCHARE was called CHARE in [25]. In 2005, Bex et al. discussed the expressiveness of XSDs based on 819 XSDs harvested from the web in [5]. In this corpus only 225 XSDs remained no errors and 85 % were in fact structurally equivalent to a DTD. In 2006, after an analysis of 819 DTDs and XSDs gathered from Cover Pages as well as from the web, Bex et al. proposed two new subclasses: single occurrence regular expression (SORE) and chain regular expression (CHARE) [7]. In [7], CHARE is defined as a subclass of SORE and it has a weaker expressiveness than the CHARE defined in [25]. Using 966 DTDs and XSDs, Feng et al. extended CHARE [7] to Echare to cover more content models [15]. Echare allows two kinds of base symbols a and a^+ where $a \in \Sigma$.

The above discussion reveals some shortcomings of existing work. First, it is clear that the names of different subclasses of regular expressions above are quite confusing in the literature. Therefor we rename these subclasses and use new names in this paper. The relations between the new and the old names are shown in Table 1. Second, the scale of data sets was far from enough for analysis. One distinguishing feature of this paper is that the data set is sufficiently large compared with previous relevant work. So it will be helpful to get a more accurate result. Using techniques such as proxies, disguised as a browser, multi-threading, we gather a large sample of 2427 DTD and 4859 XSD files from Google after removing duplicate schemas by MD5. The data set covers different fields such as education, agriculture, science, economics, engineering, sports and so on. The whole list for our data set and tools used in this paper can be found in http://lcs.ios.ac.cn/~zhangxl/project.html. Besides, in existing work the above subclasses were separately discussed in different papers using relatively small data sets, while in this paper we simultaneously analyze these subclasses using a new larger data set.

Table 1. Relations between new and old restricted subclass names

New names	Old names
CHARE	Simple regular expression [6]
eCHARE	CHARE [25]
SORE	SORE [7]
Simplified CHARE	CHARE [7]
eSimplified CHARE	Echare [15]

In addition, these subclasses are all defined on standard regular expressions. However, counting and interleaving have already been used in XSDs. Björklund

et al. made an incremental evaluation of regular expressions with counters based on a data set from three libraries: RegExLib, Snort and XML Schema on the web in [10]. The results show that almost half of the regular expressions use non-trivial counters which exclude the forms of $a^{[0,0]}$, $a^{[0,1]}$ and $a^{[1,1]}$. They also found that the vast majority is the simple form of CHAINs: $e_1 \cdot e_2 \cdots e_m$ where $e_i = (a_1 + a_2 + \cdots + a_n)^{[k,l]}$. In [17], Ghelli et al. proposed a restricted subclass defined by $T ::= \varepsilon | a^{[m,n]} | T + T | T \cdot T | T \& T$ where $m \in N \backslash \{0\}$ and $n \in N \backslash \{0\} \cup \{^*\}$. In $L(T)$, each alphabet symbol can appear at most once. Counters can only occur as a constraint for terminal symbols. For example, $(a? \cdot (b + c + d)^{[1,100]})$ is not allowed. In 2008 they introduced a linear-time membership algorithm [18] for this subclass. Furthermore, determinism of regular expressions with counting and interleaving has been discussed by many researchers [13,16,19,28]. In this paper, we extend CHARE with counting and interleaving operators. We conduct a series of experiments with result of a more popular use of this new subclass (94 %) in real world. We also analyze the determinism of it together with other five commonly used subclasses based on our data set. The main contributions of this paper are listed as follows.

- Considering numerical occurrence constraints and interleaving, we extend CHARE to a new restricted class called *extended CHARE with counting and interleaving* (eCICHARE). To the best of our knowledge, we are the first to analyze the usage of regular expressions with counting and interleaving together through real world data.

- Based on the large data set, we inspect the properties of different subclasses used in practice including eCICHARE by different measures. Particularly, the proportions of all subclasses used in practice are analyzed. In addition, we are the first to analyze the usage of eCHARE using real world data.

The rest of paper is organized as follows. Section 2 gives the definitions used in this paper. Then we introduce the data set and experiments in Sect. 3. The related work is discussed in Sect. 4 and Sect. 5 gives the conclusion.

2 Definitions

2.1 Regular Expression with Counting and Interleaving

Let Σ be a finite alphabet of terminal symbols. Each string consists of a finite sequence of symbols in Σ. Σ^* means the set of all strings over Σ. A *regular expression with counting and interleaving* over Σ is $\emptyset, \varepsilon$, or $a \in \Sigma$, or is the union $r_1 + r_2$, the concatenation $r_1 \cdot r_2$, the interleaving $r_1 \& r_2$, the Kleene-star r_1^*, the choice $r_1?$, the counting $r_1^{[m,n]}$ with $m \leq n$ and $n > 0$, or the plus r_1^+ where r_1 and r_2 are both regular expressions. $r_1 \cdot r_2$ is also written as $r_1 r_2$. Let s be a string in Σ^* and $|s|$ denotes its size. We use $s_1 \& s_2$ to denote the set of strings obtained by s_1 and s_2 in every possible way. For $s \in \Sigma^*$, $s \& \varepsilon = \varepsilon \& s = s$ and $a \cdot s_1 \& b \cdot s_2 = (a \cdot (s_1 \& b \cdot s_2)) \cup (b \cdot (a \cdot s_1 \& s_2))$. For example, strings accepted by $a \& b \& c$ are $\{abc, acb, bac, bca, cab, cba\}$. Counting is the numerical occurrence constraint which defines the minimal and maximal number of times for a certain

symbol in a regular expression. For $E = (a^{[1,3]} + b)^{[2,3]}b$, it means that two-to-three repeats of a choice between a sequence of one-to-three a elements or a single b, followed by a single b. By $RE(\&, \#)$ we represent regular expressions extended with interleaving and counting. We use $RE(\&)$ and $RE(\#)$ to denote the regular expressions with interleaving and counting respectively. # and & operators help us to express the content models more succinctly and concisely.

2.2 eCICHARE

We introduce a new subclass which extends CHARE with interleaving and counting.

Definition 1 ***(extended CHARE with counting and interleaving (eCICHARE)).*** *Base symbols are a, $a?$, a^*, or a^+ where $a \in \Sigma$. A factor is of the form e, $e?$, e^*, $e^{[m,n]}$, or e^+ where e is a disjunction of base symbols of the same kind. An eCICHARE is $\emptyset$, ε, a concatenation of factors, or an unordered sequence of factors with interleaving among them.*

In the definition, each eCICHARE cannot contain both concatenation and interleaving operators at the same time. Numerical occurrence can only be the constraint to factors. This restriction is severe, but the experiment result shows that it is actually met by most of regular expressions in real world data. In addition, two forms of eCICHARE: $a^{[0,1]}$ and $a^{[1,1]}$ are always substituted by $a?$ and a in practice.

For instance, $E_1 = (a^* + b^*)?(a + b)^*(c^+ + d^+)e?$, $E_2 = a^{[1,3]}b?$ and $E_3 = (a^* + b^*)?\&(a + b)^*\&(c^+ + d^+)\&e?$ are both eCICHAREs while $E_4 = (a^* + b^*)?(a + b^{[0,3]})^*(c^+ + d^+)\&e?$ is not.

2.3 Determinism

Determinism is required by W3C specification for content models of DTDs and XSDs. It is also called the unique particle attribution (UPA) property by W3C Recommendation. It has the same meaning with one-ambiguity [11], and weak determinism [16,29]. Suppose that we match a string s against a given regular expression E from left to right. If we always know definitely the next symbol we will match without looking ahead in the string, E is deterministic. The formal definition of determinism is as follows.

Definition 2 ***(Determinism*** [4]**).** *An expression E is deterministic if and only if for all words $uxv, uyv \in L(E')$ where $|x| = |y| = 1$, if $x \neq y$ then $x' \neq y'$.*

In this definition, E', x' and y' are the marked forms of E, x and y. For example. $E = (a + b)^*a?b^*$, then $E' = (a_1 + b_2)^*a_3?b_4^*$ and $(E')' = E$. $E = aa^*$ is a deterministic regular expression while $E = a^*a$ is not.

2.4 Definitions for Other Five Subclasses

Definition 3 ***(CHARE*** [6]**).** *Base symbols are a, $a?$, a^*, a^+ where $a \in \Sigma$. A factor is of the form e, $e?$, e^*, e^+ where e is a disjunction of base symbols of the same kind. A simple regular expression is $\emptyset$, ε, or a concatenation of factors.*

For instance, $(a^* + b^*)(a + b)?b^*(a + b)^*$ is a CHARE while $(abc + c?)(a^* + b^*)d?$ is not.

Definition 4 ***(eCHARE*** [25]**).** *Base symbols are s, $s?$, s^*, s^+ where s is a non-empty string. A factor is of the form e, $e?$, e^*, e^+ where e is a disjunction of base symbols of the same kind. That is of the form $(s_1 + \cdots + s_n)$, $(s_1^* + \cdots + s_n^*)$, $(s_1? + \cdots + s_n?)$, $(s_1^+ + \cdots + s_n^+)$, where $n \geq 1$ and s_i is non-empty string. An eCHARE is $\emptyset$, ε or a concatenation of factors.*

For example, $((abc)^* + b)(a + b)?(ab)^+(ac + b)^*$ is an eCHARE.

Definition 5 ***(SORE*** [7]**).** *Let Σ be a finite alphabet. A single-occurrence regular expression (SORE) is a regular expression over Σ in which every terminal symbol occurs at most once.*

For instance, $(((a + b), c)?d^+)^*e$ is a SORE while $(a^* + b^*)a$ is not.

Definition 6 ***(Simplified CHARE*** [7]**).** *A Simplified CHARE is a SORE over Σ of the form $f_1 \cdots f_n$ where $n \geq 1$. Every factor f_i is an expression of the form $(a_1 + \cdots + a_n)$, $(a_1 + \cdots + a_n)?$, $(a_1 + \cdots + a_n)^*$, $(a_1 + \cdots + a_n)^+$ where $n \geq 1$ and every a_i is a terminal symbol.*

For example, $(a + b)?c^*$ is a Simplified CHARE while $(a + b?)a$ is not.

Definition 7 ***(eSimplified CHARE*** [15]**).** *An eSimplified CHARE is a SORE over Σ of the form $f_1 \cdots f_n$ where $n \geq 1$. Every factor f_i is an expression of the form $(a_1 + \cdots + a_n)$, $(a_1 + \cdots + a_n)?$, $(a_1 + \cdots + a_n)^*$, $(a_1 + \cdots + a_n)^+$ where $n \geq 1$ and every a_i is a terminal symbol or the form of a_i^+.*

For example, $(a + b^+)c?$ is an eSimplified CHARE.

CHARE and eCHARE require the base symbols in each factor must be the same kind while eSimplified CHARE can be different.

3 Experiments

3.1 Data Set

Data Preprocess. There are two steps in our data preprocess. First, get the DTD and XSD files. Repositories of data set such as GSML, DMTF, DSML, DWML, FACETMAP, GITHUB, GRAPHML, HAPMAP, IOP, KAIST, NCBI, BIOXSD, CORBA, CSML and so on are well-formed. We can harvest DTDs and XSDs from their official websites directly. But others need to be crawled through Google by queries: filetype:dtd or filetype:xsd. Using these two queries, we get many URLs for DTDs (or XSDs). However, not all these URLs point to

a DTD (or XSD) directly but to some HTML files. Such HTML file may have a link or path to a DTD (or XSD), or just some information on the website. For links, we make a recursive resolving to download these related DTDs and XSDs. For information on the websites, we use a specific script tool: *html_parser* which is written in Python to analyze the information and obtain the schema. Second, do data cleaning. We remove duplicate files with same URLs. More precisely, we use the technique MD5 to analyze whether two files with different URLs have the same content. If so, redundant files are also removed. At last, we obtain 2427 DTD files and 4859 XSD files and extract 64249 and 67255 regular expressions from DTDs and XSDs files respectively.

For the convenience of discussion, we introduce a uniform syntax to denote subclasses of restricted regular expression by specifying the allowed factors used in [25]. The base symbols are denoted by a, $a?$, a^*, a^+. s means a string in Σ^*. The disjuncts are denoted by $(a_1 + a_2 + \cdots + a_n)$, $(a_1? + a_2? + \cdots + a_n?)$, $(a_1^* + a_2^* + \cdots + a_n^*)$, $(a_1^+ + a_2^+ + \cdots + a_n^+)$ which can be also extended with choice, Kleene-star, and plus respectively. These factors can be abbreviated by the form of $(+\cdots)$. We use $RE((+a?), a^*)$ to illustrate the subclass whose factors can be in the form of $(a_1? + a_2? + \cdots + a_n?)$ where $a_i \in \Sigma$ and $n \geq 1$ or the form of a^* for some $a \in \Sigma$. We list some possible factors in Table 2.

Table 2. Possible factors in subclasses of regular expressions and their abbreviations [25]

Factor	a	$a?$	a^*	a^+	$s?$	s^*	s^+
Abbr.	a	$a?$	a^*	a^+	$s?$	s^*	s^+

Factor	Abbr.	Factor	Abbr.
$(a_1 + \cdots + a_n)$	$(+a)$	$(s_1 + \cdots + s_n)$	$(+s)$
$(a_1 + \cdots + a_n)?$	$(+a)?$	$(s_1 + \cdots + s_n)?$	$(+s)?$
$(a_1 + \cdots + a_n)^*$	$(+a)^*$	$(s_1 + \cdots + s_n)^*$	$(+s)^*$
$(a_1 + \cdots + a_n)^+$	$(+a)^+$	$(s_1 + \cdots + s_n)^+$	$(+s)^+$
$(a_1^* + \cdots + a_n^*)$	$(+a^*)$	$(s_1^* + \cdots + s_n^*)$	$(+s^*)$
$(a_1^+ + \cdots + a_n^+)$	$(+a^+)$	$(s_1^+ + \cdots + s_n^+)$	$(+s^+)$

Based on 64249 and 67255 regular expressions from DTDs and XSDs, we analyze the occurrence types of regular expressions in practice. We treat the form of a^+ operator as aa^*. The result is shown in Table 3. From Table 3, we can find that the form of $RE(a, (+a)^*)$ accounts for the most proportion (34.6 %) for DTDs while in XSDs, forms of $RE(a, a?)$ and $RE(a, a^*)$ are more popular. In addition, the vast majority of regular expressions belongs to the subclass of eCICHARE with proportion of 90.3 % for DTDs and 94.1 % for XSDs respectively.

Table 3. Occurrence of types

	% of DTDs	% of XSDs
RE(a)	18.71	19.28
RE(a,a?)	6.18	22.56
RE(a,a*)	17.02	23.34
RE(a,a?,a*)	3.16	9.49
RE(a,(+a))	1.74	3.07
RE(a,(+a)?)	0.36	1.09
RE(a,(+a)*)	34.63	5.33
RE(a,(+a)?,(+a)*)	3.72	1.60
RE(a,(+a*)*)	3.59	0.16
RE(#)	0	4.06
RE(&)	0	2.07
RE(#,&)	0	≈ 0
Others	0.97	1.96
Total eCICHARE	90.08	94.00

3.2 Definitions for Measures

In this section, we first introduce some definitions of measures used in the experiment such as star height, nesting depth, density and so on. Then, using these measures, we analyze the properties and complexity of regular expressions from different aspects.

Definition 8 ***(Star Height*** [2]**).** *The star height of a regular expression E over the alphabet Σ, denoted by $h(E)$, is a nonnegative integer defined recursively as follows:*

1. $h(E) = 0$, *if* $E = \emptyset$ *or* a *for* $a \in \Sigma$,
2. $h(E) = max\{h(E_1), h(E_2)\}$, *if* $E = (E_1 + E_2)$ *or* $E = (E_1 \cdot E_2)$, *where* E_1 *and* E_2 *are regular expressions over* Σ,
3. $h(E) = h(E_1) + 1$ *if* $E = (E_1)^*$ *and* E_1 *is a regular expression over* Σ.

The star height [2] reflects the maximum nesting depth of Kleene-star occurring in a regular expression. It is an illustration for the complexity of DTDs and XSDs. We give the star height of DTDs and XSDs respectively. We use some substitutions in our experiment of computing the star height. For example, $E_1 = a^+$ and $E_2 = a^{[m,\infty]}$ can be rewritten as the form of $E_1' = aa^*$ and $E_2' = a^*$. We treat interleaving similarly as the operator of concatenation. In fact, other forms of counting and interleaving do not influence the results of star height. From Table 4, we observe that the result of distributions for DTDs and XSDs have no significant differences. Content models with star height larger than 2 are very

Table 4. Star height observed in DTDs and XSDs

Star height	0	1	2	3	4
% of DTDs	27.71	70.66	1.58	0.05	0
% of XSDs	53.62	44.94	1.15	0.29	≈ 0

rare. For XSDs, the proportion with the star height equal to 1 is higher in our result than that in [6] which is 38 % and 17 % respectively.

Besides, we also consider the counter height for regular expressions of XSDs. Counter height is defined just like the star height. The result indicates that the counter height with 1 of regular expressions with counting accounts for almost 89.3 %. Star height and counter height are both illustration of iteration depth of regular expressions. In this paper, we introduce Nesting depth to measure the complexity of a regular expression. For example, the star height for regular expressions $E_1 = (a + b?)?c^{[2,4]}d$ and $E_2 = a$ are both zero. But the complexity for E_1 and E_2 should be different. Nesting depth considers all operators possible in regular expressions including counting and interleaving. For example, let $E = 1^+ + (2? \cdot 3^+)^* + 4^{[1,3]}$. Its corresponding nesting depth is 2. We show proportions of DTDs and XSDs by nesting depth in Table 5. From Table 5, we can find that both DTDs and XSDs with nesting depth lower than 2 are more than 95 %. That means schemas with complex structures are very rare.

Definition 9 ***(Nesting Depth).*** *The nesting depth of a regular expression E over Σ, denoted by $ND(E)$, is a nonnegative integer defined recursively as follows:*

1. $ND(E) = 0$, *if* $E = \emptyset$, ε *or* a *for* $a \in \Sigma$,
2. $ND(E) = max\{ND(E_1), ND(E_2)\}$, *if* $E = (E_1 + E_2)$, $E = (E_1 \& E_2)$ *or* $E = (E_1 \cdot E_2)$, *where* E_1 *and* E_2 *are regular expressions over* Σ,
3. $ND(E) = ND(E_1) + 1$, *if* $E = (E_1)^*$, $E = (E_1)^+$, $E = (E_1)?$ *or* $E = (E_1)^{[m,n]}$ *for* E_1 *is a regular expression over* Σ.

Then we consider the density distribution of DTDs and XSDs respectively based on the real world data. The density [6] can be another measure to illustrate the complexity degree of rules in content models.

Definition 10 ***(Density*** [6]**).** *The density of a schema is defined as the number of elements occurring in the right hand side of its rules divided by the number of elements. The formula is* $d = \frac{1}{N}\sum_{i=1}^{N}|A_i|$ *where* N *is the total number of*

Table 5. Nesting depth observed in DTDs and XSDs

Nesting depth	0	1	2	3	4
% of DTDs	20.59	76.38	2.49	0.50	0.04
% of XSDs	22.86	72.82	3.91	0.37	0.04

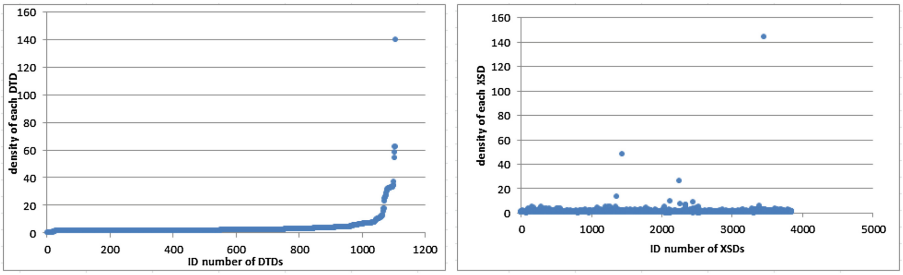

Fig. 1. ID number of DTDs (left) and XSDs (right) versus their density

element definitions occurring in this schema, A_i is the string in the right hand of a rule, $|A_i|$ denotes the size of A_i.

The density [6] can be a measure to illustrate the complexity degree of regular expressions. From Definition 10, we can easily find that the larger the value is, the more sophisticated rules a schema will have. In our experiment, we treat counting as unary operator and the interleaving operator as concatenation. Theses processings do not influence the results of density. Based on the large data set, we give the distributions of density in Fig. 1 for DTDs and XSDs respectively. From Fig. 1, it is easy to find that the density less than 10 for DTDs and XSDs is 95.8 % and 94.4 %.

3.3 XML Schema Features Used in Practical

Although DTD is simple, it develops with some shortcomings such as no modularity, limited expressiveness for new domains, limited basic types and so on. The content model of an element in DTD depends only on the element name. In contrast, XML Schema is based on type definitions and allows the content model to depend on the context in which the element is used. With stronger expressiveness, the usage of XML Schema grows gradually though it is complicated to some extent. Table 6 shows the features of XSDs used in practice.

3.4 Determinism

We consider the determinism of the regular expressions on our large data set. We use our own tools to check the determinism of regular expressions for DTD and XSD respectively. The result is shown in Table 7. From Table 7, we observe that these subclasses almost all satisfy the deterministic requirement. Take the first number 58536/64249 for example. 64249 means the total number of regular expressions for DTDs in the whole data. 58536 means the number of deterministic regular expressions in the whole data set.

Table 6. XML schema features used in the corpus

Features	% of XSDs
SimpleType extension	5.62
SimpleType restriction	13.85
ComplexType extension	10.27
ComplexType restriction	1.69
Abstract attribute	9.57
Final attribute	0.86
Block attribute	0.29
Fixed attribute	2.94
SubstitutionGroup	8.87
Redefine	1.54
xs:all	3.56
Occurrence	4.48
Namespace	96.29
Import	19.16
Key/keyref	0.95
Unique	1.61

Table 7. Determinism of regular expressions

Google	Whole data	CHARE	eCHARE	eCICHARE
DTDs	58536/64249	58343/58404	58513/58579	58343/58404
XSDs	67218/67255	59061/59078	59222/59247	60492/60509

3.5 Usage of Subclasses of Regular Expressions in Practice

In this section, we mainly investigate the usage and proportions of six subclasses in practice based on our large data set. In particular, *k-ORE* [4] is discussed with different values of k at the same time. *k-ORE* means each alphabet symbol in regular expression occurs at most k times. The reasons and proportions dissatisfying the corresponding definitions are also discussed. In our experiment, we call regular expressions with counting and interleaving as non-standard regular expressions. They are analyzed separately because counting and interleaving are specific features for XSDs. This is a little different from that in [6,7]. In [6,7], they rewritten regular expressions with counting as a new form using choice operator ?. For example, $E = a^{[1,3]}$ is transformed to $E' = aa?a?$. This transformation is not reasonable enough which influences the results of CHARE and eCHARE. The result in Table 8 indicates that the existing five subclasses still have been used in a large scale (more that 80 %). However, the usage of regular expressions with counting and interleaving increases gradually. They are called

Table 8. Proportions of subclasses of regular expressions

Subclasses	% of DTDs	% of XSDs
CHARE	90.08	87.88
Non-CHARE: non-terminal symbols	8.84	3.52
Non-CHARE: different unary operators	1.08	1.93
Non-CHARE: non-standard expressions	0	6.67
eCHARE	90.37	88.13
Non-eCHARE: not a string	8.31	3.26
Non-eCHARE: different unary operators	1.32	1.94
Non-eCHARE: non-standard expressions	0	6.67
SORE	95.01	92.45
Non-standard expressions	0	6.67
2-ORE	3.62	0.65
3-ORE	0.37	0.10
4-ORE	0.31	0.07
5-ORE	0.52	0.01
6-ORE	0.10	0.05
$k \geq 7$	0.07	≈ 0
Simplified CHARE	88.10	86.10
Non-Simplified CHARE: not a SORE	4.99	0.88
Non-Simplified CHARE: non-terminal symbols	5.00	2.82
Non-Simplified CHARE: different unary operators	1.91	3.52
Non-Simplified CHARE: non-standard expressions	0	6.67
eSimplified CHARE	89.12	86.73
Non-eSimplified CHARE: not a SORE	4.99	0.88
Non-eSimplified CHARE: non-terminal symbols	5.00	2.82
Non-eSimplified CHARE: the unary operator * or ?	0.89	2.90
Non-eSimplified CHARE: non-standard expressions	0	6.67
eCICHARE	90.08	94.00
Non-eCICHARE: non-terminal symbols	8.84	3.68
Non-eCICHARE: different unary operators	1.08	1.94
Non-eCICHARE: regular expressions with counting	0	0.37

non-standard expressions in our experiment and accounts for 6.67 %. After we extend CHARE to eCICHARE, the proportion rises to 94.00 % from 87.88 %.

Bex et al. concluded that 92 % of DTDs and 97 % of XSDs were CHARE based on 109 DTDs and 93 XSDs [6]. In [7], Bex et al. found that 99 % regular expressions were Simplified CHARE, which were also SORE based on 819 DTDs and XSDs. While in our experiment, the proportions for CHARE, SORE,

Simplified CHARE, eSimplified CHARE are lower. The main reason is the difference of data sets. Our data set is larger and more comprehensive, which leads to more accurate results and is closer to the real situation. Another reason is the different process of counting and interleaving operators. The transformation of counting in [6,7] such as the substitution $aa?a?$ of $a^{[1,3]}$, increases the proportions of CHARE and eCHARE. Besides, we are the first to analyze the usage of eCHARE through real world data. The proportions of eCHARE are 90.37 % for DTDs and 88.13 % for XSDs. Reasons dissatisfying the corresponding definitions are shown in the table clearly. For example, non-terminal symbols mean using expressions such as the forms of ab and $a?b^*$ where $a, b \in \Sigma$ as base symbols in a disjunction. They are not allowed in four subclasses: CHARE, Simplified CHARE, eSimplified CHARE and eCICHARE. For CHARE, they account for 8.84 % and 3.52 % for DTDs and XSDs respectively. The proportions for other three subclasses can be found in Table 8. However, strings like the form of ab as the base symbols is valid for eCHARE. The form of $(a?b^* + c)$ is not allowed in eCHARE. We call it as *extended strings*. The proportion of *extended strings* is not small and they will be considered in our future work. The proportion of SORE (95.01 % for DTDs and 92.45 % for XSDs) means that symbols in the vast majority of regular expressions only occur once most of the time. Simplified CHARE and eSimplified CHARE are both subclasses of SOREs. In [15], eSimplified CHARE accounts for 84.8 % based on 2009 regular expressions while in our experiment the proportion is 89.12 % for DTDs (64249 regular expressions) and 86.73 % for XSDs (67255 regular expressions). The unary operators $*$ and ? cannot be constraints for base symbols in eSimplified CHARE, which account for 0.89 % and 2.9 % for DTDs and XSDs respectively. For all six subclasses, proportions of other reasons are not large. For example, different unary operators depict the use of the form $(a? + b^*)c$ as factors which is not valid in the six subclasses. The unary operator * or ? is the form of $(a^* + b^*)$ which is not allowed in eSimplified CHARE. In the definition of eCICHARE, numerical occurrence constraints cannot be nested. The reason of regular expressions with counting is the form of $(a^{[1,3]} + b)^{[1,2]}$ which is not valid for eCICHARE.

4 Related Work

Early in 2002, Choi [14] made an experiment about DTDs. 60 DTDs were extracted from the XML.org DTD repository [32]. He analyzed the features of DTDs and proposed measures to make a deep study of their structural properties such as local properties including syntactic complexity, determinism, ambiguity and global properties including reachability, recursion, simple path and simple cycle, chain of stars, hubs. He found that the majority of DTDs used in real world is the form of chain. This result has provided important suggestion for later study in this field.

Based on 109 DTDs and 93 XSDs crawled from Cover Pages, Bex et al. [6] proposed simple regular expression which was named CHARE in this paper. They found that 92 % and 97 % of all element definitions in DTDs and XSDs

are CHAREs. Martens et al. extended CHARE to eCHARE in [25]. Using an improved data set crawled from Cover Pages and the web, about 819 DTDs and XSDs, Bex et al. introduced two new subclasses of regular expressions: SORE and Simplified CHARE in [7]. The results revealed that more than 99 % of the regular expressions occurring in practical schemas are Simplified CHARE (therefore also SORE). In 2006, Bex et al. in [8] proposed the concept of *k-occurrence*. A regular expression is *k-occurrence* if every alphabet symbol occurs at most k times in it [8]. According to the same data set in [7], they concluded that regular expressions occurring in practical schemas are such that every alphabet symbol occurs at most k times and actually, in 98 % of the cases $k = 1$. Martens et al. in [26] concluded that in more that 98 % of the XSDs occurring in practice the content model of an element depends only on the label of the element itself, the label of its parent, and (sometimes) the label of its grand-parent after an examination of 225 XSDs gathered from the Cover Pages. Later in 2007, Bex et al. [9] introduced the concept of *k-local*. An XSD is *k-local* if its content models depend only on labels up to the *k-th* ancestor [9]. For most cases in real world, the value of k is 1. This result is conformed with that in [8]. In 2014, inspired by Simplified CHARE, Feng et al. proposed eSimplified CHARE whose base symbol allows the forms of a and a^+ where $a \in \Sigma$. The data set used in [15] consists of 966 valid DTDs and XSDs which were rewritten as 2009 regular expressions. Based on this data set, the cover ratio of eSimplified CHARE reached 84.8 % from 79.5 % for Simplified CHARE. In addition, two inference algorithms for eSimplified CHARE were given.

In addition, regular expressions with counting and interleaving have been studied by many researchers. Björklund et al. were the first to make an incremental evaluation of regular expressions with counters. They gathered about 3024, 458 and 8830 regular expressions respectively from three libraries: RegExLib, Snort, and XML Schema on the web in [10]. The results show that there are 1705 out of 3024 (about 56.3 %), 270 out of 458 (about 58.9 %) regular expressions use non-trivial counters in RegExLib and Snort. Regular expressions with non-trivial counters are the forms excluding three specific ones: $a^{[0,0]}$, $a^{[0,1]}$ and $a^{[1,1]}$. In addition, the proportions of CHAINs are 73.3 %, 85.1 % and 86 % for the data of three libraries above. In [17], Ghelli et al. proposed a restricted subclass defined by $T ::= \varepsilon | a^{[m,n]} | T+T | T \cdot T | T\&T$ where $m \in N \backslash \{0\}$ and $n \in N \backslash \{0\} \cup \{^*\}$. In $L(T)$, each alphabet symbol can appear at most once. Counter can only occur as a constraint for terminal symbols. For example, $(a? \cdot (b+c+d+e+f)^{[1,100]})$ is not allowed. Based on this subclass, they first proposed a linear-time translation algorithm of the translation of each regular expression into a set of constraints in [18]. Then they introduced a linear-time membership algorithm to check whether a word satisfies the resulting constraints.

5 Conclusion and Future Work

In this paper, we introduce a new restricted subclass of regular expression with counting and interleaving. The experiment results show that this subclass can

cover more content models in real world. Then different features of five subclasses of content models, i.e., CHARE, eCHARE, SORE, Simplified CHARE, eSimplified CHARE together with eCICHARE are also analyzed using different measures. We inspect their usages and give the corresponding proportions based on the real world data. Our data set is much larger than previous work which leads to more accurate results. We believe that our work will be helpful to the applications and further study of DTDs and XSDs. One future work is the study of inference algorithms and complexity problems related to eCICHARE. The strong and weak determinism of eCICHARE will also be considered. Besides, based on our experiment results, it is possible to propose other useful subclasses of content models.

References

1. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann, Burlington (2000)
2. Bala, S.: Intersection of regular languages and star hierarchy. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 159–169. Springer, Heidelberg (2002)
3. Benedikt, M., Fan, W., Geerts, F.: XPath satisfiability in the presence of DTDs. J. ACM (JACM) **55**(2), 8 (2008)
4. Bex, G.J., Gelade, W., Neven, F., Vansummeren, S.: Learning deterministic regular expressions for the inference of schemas from XML data. ACM Trans. Web (TWEB) **4**(4), 14 (2010)
5. Bex, G.J., Martens, W., Neven, F., Schwentick, T.: Expressiveness of XSDs: from practice to theory, there and back again. In: Proceedings of the 14th International Conference on World Wide Web, pp. 712–721. ACM (2005)
6. Bex, G.J., Neven, F., Van den Bussche, J.: DTDs versus XML schema: a practical study. In: Proceedings of the 7th International Workshop on the Web and Databases: Colocated with ACM SIGMOD/PODS 2004, pp. 79–84. ACM (2004)
7. Bex, G.J., Neven, F., Schwentick, T., Tuyls, K.: Inference of concise DTDs from XML data. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 115–126. VLDB Endowment (2006)
8. Bex, G.J., Neven, F., Schwentick, T., Vansummeren, S.: Inference of concise regular expressions and DTDs. ACM Trans. Database Syst. (TODS) **35**(2), 11 (2010)
9. Bex, G.J., Neven, F., Vansummeren, S.: Inferring XML schema definitions from XML data. In: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 998–1009. VLDB Endowment (2007)
10. Björklund, H., Martens, W., Timm, T.: Efficient incremental evaluation of succinct regular expressions. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 1541–1550. ACM (2015)
11. Brüggemann-Klein, A., Wood, D.: One-unambiguous regular languages. Inf. Comput. **140**(2), 229–253 (1998)
12. Che, D., Aberer, K., Özsu, M.T.: Query optimization in XML structured-document databases. VLDB J. **15**(3), 263–289 (2006)
13. Chen, H., Lu, P.: Checking determinism of regular expressions with counting. Inf. Comput. **241**, 302–320 (2015)
14. Choi, B.: What are real DTDs like. Technical reports (CIS), p. 17 (2002)

15. Feng, X.Q., Zheng, L.X., Chen, H.M.: Inference algorithm for a restricted class of regular expressions. Comput. Sci. **41**(4), 178–183 (2014)
16. Gelade, W., Gyssens, M., Martens, W.: Regular expressions with counting: weak versus strong determinism. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 369–381. Springer, Heidelberg (2009)
17. Ghelli, G., Colazzo, D., Sartiani, C.: Efficient inclusion for a class of XML types with interleaving and counting. In: Arenas, M. (ed.) DBPL 2007. LNCS, vol. 4797, pp. 231–245. Springer, Heidelberg (2007)
18. Ghelli, G., Colazzo, D., Sartiani, C.: Linear time membership in a class of regular expressions with interleaving and counting. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 389–398. ACM (2008)
19. Kilpeläinen, P.: Checking determinism of XML schema content models in optimal time. Inf. Syst. **36**(3), 596–617 (2011)
20. Koch, C., Scherzinger, S., Schweikardt, N., Stegmaier, B.: Schema-based scheduling of event processors and buffer minimization for queries on structured data streams. In: Proceedings of the 30th International Conference on Very Large Data Bases, vol. 30, pp. 228–239. VLDB Endowment (2004)
21. Manolescu, I., Florescu, D., Kossmann, D.: Answering XML queries on heterogeneous data sources. VLDB **1**, 241–250 (2001)
22. Martens, W., Neven, F.: Typechecking top-down uniform unranked tree transducers. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 64–78. Springer, Heidelberg (2002)
23. Martens, W., Neven, F.: Frontiers of tractability for typechecking simple XML transformations. In: Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 23–34. ACM (2004)
24. Martens, W., Neven, F., Schwentick, T.: Complexity of decision problems for simple regular expressions. In: Fiala, J., Koubek, V., Kratochvíl, J. (eds.) MFCS 2004. LNCS, vol. 3153, pp. 889–900. Springer, Heidelberg (2004)
25. Martens, W., Neven, F., Schwentick, T.: Complexity of decision problems for XML schemas and chain regular expressions. SIAM J. Comput. **39**(4), 1486–1530 (2009)
26. Martens, W., Neven, F., Schwentick, T., Bex, G.J.: Expressiveness and complexity of XML schema. ACM Trans. Database Syst. (TODS) **31**(3), 770–813 (2006)
27. Papakonstantinou, Y., Vianu, V.: DTD inference for views of XML data. In: Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 35–46. ACM (2000)
28. Peng, F., Chen, H., Mou, X.: Deterministic regular expressions with interleaving. In: Rueda, C., et al. (eds.) ICTAC 2015. LNCS, vol. 9399, pp. 203–220. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25150-9_13
29. Sperberg-McQueen, C.: Applications of Brzozowski derivatives to XML schema processing. In: Extreme Markup Languages®, Citeseer (2005)
30. Thompson, H.S., Beech, D., Maloney, M., Mendelsohn, N.: XML schema part 1: structures. 2nd edn. W3C Recommendation (2004)
31. Wang, G., Liu, M., Yu, G., Sun, B., Yu, G., Lv, J., Lu, H.: Effective schema-based XML query optimization techniques. In: 2003 Proceedings of Seventh International Database Engineering and Applications Symposium, pp. 230–235. IEEE (2003)
32. XML: XML.org Registry (2002). http://www.xml.org/xml/registry.jsp/