FlashSchema: Achieving High Quality XML Schemas with Powerful Inference Algorithms and Large-scale Schema Data

Yeting Li[†§], Jialun Cao[‡], Haiming Chen[†], Tingjian Ge[♯], Zhiwu Xu[♮], Qiancheng Peng[†§]
[†] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[§] University of Chinese Academy of Sciences, Beijing, China
[‡] The Hong Kong University of Science and Technology, Hong Kong, China
[♯] University of Massachusetts, Lowell, United States
[♮] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[†§]{liyt,chm,pengqc}@ios.ac.cn, [‡]jcaoap@cse.ust.hk, [♯]ge@cs.uml.edu, [♮]xuzhiwu@szu.edu.cn

***Abstract*—Getting high quality XML schemas to avoid or reduce application risks is an important problem in practice, for which some important aspects have yet to be addressed satisfactorily in existing work. In this paper, we propose a tool FlashSchema for high quality XML schema design, which supports both one-pass and interactive schema design and schema recommendation. To the best of our knowledge, no other existing tools support interactive schema design and schema recommendation. One salient feature of our work is the design of algorithms to infer k-occurrence interleaving regular expressions, which are not only more powerful in model capacity, but also more efficient. Additionally, such algorithms form the basis of our interactive schema design. The other feature is that, starting from large-scale schema data that we have harvested from the Web, we devise a new solution for type inference, as well as propose schema recommendation for schema design. Finally, we conduct a series of experiments on two XML datasets, comparing with 9 state-of-the-art algorithms and open-source tools in terms of running time, preciseness, and conciseness. Experimental results show that our work achieves the highest level of preciseness and conciseness within only a few seconds. Experimental results and examples also demonstrate the effectiveness of our type inference and schema recommendation methods.**

I. Introduction

As a main data exchange format, XML has been widely used in various applications on the Web. In recent years, although some competing formats such as JSON appeared, XML has advantages in many aspects and still serves as a common data exchange format—for this reason, major database management systems such as Oracle, SQL Server, DB2, and MySQL all support XML. XML schemas have always played a crucial role in XML data management. Low quality schemas, however, may pose serious risks—including security risks. For example, over-permissive schemas could allow invalid values with potential safety hazards.

Obtaining high quality schemas to avoid or reduce application risks is an important problem in practice. However, for this problem, many challenging issues have not been well addressed by existing work. This problem is closely related to *schema inference*—this is because many XML documents in practice are not accompanied by a (valid) schema [1], thus making schema inference an essential task.

Inference of regular expressions (REs) is an essential issue in the XML schema inference problem [2]–[8]. Presently, the most powerful model for XML is the k-occurrence regular expression (k-ORE), which means each symbol can occur at most k times in an expression. According to [9], its coverage in DTD and XSD is up to almost 100%. However, it does not support the unordered operator *interleaving*, whereas this operator plays a contributing role in RNG, making its coverage only 76.11% in RNG. The statistics in [9] also show that when supporting both k-occurrence ($1 \leq k \leq 7$) and interleaving, the coverage in RNG achieves 99.85%. This indicates the need for the interleaving operator, and a more powerful model than k-ORE.

Supporting both k-occurrence and interleaving makes the problem challenging. In the literature, there have been only a few attempts to address either k-OREs or interleaving. Inferring ***k*-OREs** is much more complicated than the single-occurrence ones (i.e. each symbol occurs at most once) [4]. Furthermore, learning single-occurrence REs (SOREs) with **interleaving** is already intractable, as this unordered operator drastically increases the possible number of REs, let alone k-occurrence with interleaving. Currently there is only one paper addressing k-OREs with *limited* interleaving [10].

Further, presently almost all XML inference algorithms only support learning from *positive samples*. On the other hand, it has been proved that learning from positive and negative samples is more powerful than only from positive samples [11]. Indeed, negative samples are useful in many applications—e.g., schema evolution [12] can be done incrementally with little feedback needed from the user, when we also allow negative samples. Therefore, we aim to learn from both *positive* and *negative* samples.

Besides RE inference, *type inference* is also an important aspect of XML schema inference. One difficulty lies in how to infer types as accurately as possible. Unfortunately, most of the existing approaches either do not take into account

type inference, or omit the types and consider all values as strings. To our knowledge, only two methods [13], [14] can infer simple types only by a pattern-matching process on the format of the values; i.e., the characteristics that make unique a type, which is called the *lexical space* according to the W3C Recommendation. There are two main limitations of lexical space based methods: (i) they cannot infer all built-in simple types, since there are intersections between types' lexical spaces; (ii) they cannot infer user-defined types derived from existing simple types.

The main contributions of the paper are as follows:

- We develop a tool FlashSchema for high quality schema design based on several powerful yet efficient schema inference algorithms and large-scale schema data. Specifically, it supports interactive schema design and schema recommendation, which cannot be supported by existing tools.
- We propose a new subclass of REs, k-occurrence REs with interleaving (k-OIREs). It covers almost 100% DTD, XSD, and RNG schemas. We devise powerful and efficient algorithms to infer k-OIREs from both positive and negative samples, which also form the basis of interactive schema design.
- Based on large-scale schema data, we provide a new solution for type inference, which overcomes the limitations of existing methods by using word embedding techniques to take the semantics of the context into consideration; we also propose schema recommendation for schema design, which is suited for situations where the users only know some elements of the schema under design, or the users do not want to design from scratch.
- Experimental results show that our approach achieves the highest level of preciseness and conciseness, outperforming all previous state-of-the-art results. Experimental results and examples also reveal the effectiveness of our type inference method.

II. Main Techniques

In this section, we present the main techniques of our approach. Our framework FlashSchema gives three application scenarios (i.e. *one-pass schema design*, *interactive schema design* and *schema recommendation*) for different requirements. In particular, *one-pass schema design* means the process only needs one run, and is suited for users who are clear of their own needs and also can provide enough samples. In such situations, our framework can give satisfactory results in one run. On the other hand, if users are not fully clear of their needs, and they have some samples at hand, then an *interactive schema design* would be suitable. The users use the interactive process to adjust input, i.e., to add, update, or delete positive or negative samples based on the current results returned by our framework. In this way, the users ultimately fully understand their needs, and the process repeats until the result returned by FlashSchema satisfies the users. If users only know some element names of a schema under design, they can use *schema recommendation*. FlashSchema may return the most used schemas in practice that contain the user-provided element names, based on our schema library, which certainly are of great value for the schema under design. Next we will illustrate the three scenarios through several examples.

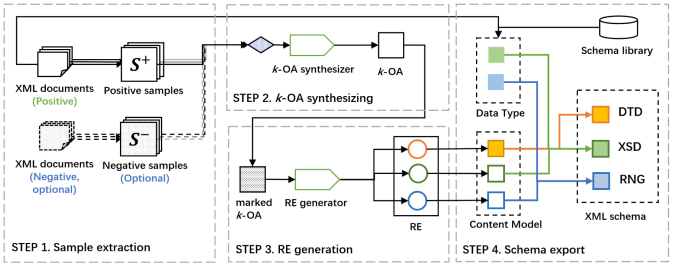


Fig. 1: An overview of the inference engine.

A. One-pass Schema Design

There are four main steps in our inference engine (as shown in Fig. 1). Firstly, from several XML files, we extract multiple groups of samples. Then, each group of samples is fed into a *k-OA synthesizer* to produce a corresponding deterministic k-OA[1]. Next, we mark the synthesized deterministic k-OAs and generate possible REs from the marked k-OAs, by using an *RE generator*. We convert the generated REs to the corresponding content models and get the data types corresponding to the content models, based on the lexical space analysis and the semantics of the context. Finally, we combine content models and corresponding data types into an XML schema.

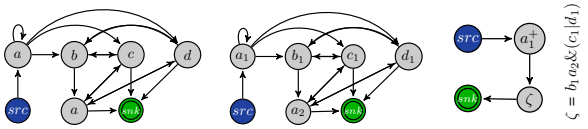
The step *RE generation* has multiple choices according to users' needs. Users can call the procedure $Soa2Sore$ [6] used in Freydenberger and Kotzing's work to get the REs for DTDs/XSDs content models; users can also execute our proposed algorithm $Soa2Soire$ [15] to get REs with interleaving for RNG content models.

Example 1. Here is an example of inferring an RNG content model. Suppose we extract positive samples S^+=$\{aacba, abda, abca, aabad, adba, abac\}$ and negative samples S^-=$\{acabd, adad, adab\}$ from XML files. Then, we execute the k-OA synthesizing algorithm to produce a 2-OA (Fig. 2(a)) and we marked the synthesized 2-OA (Fig. 2(b)). Fig. 2(c) gives an intermediate SOA obtained in $Soa2Soire(\overline{\mathcal{A}})$, from which a marked RE $a_1^+b_1a_2\&(c_1|d_1)$ is built by $Soa2Soire$. Therefore, the final RE we extract for $\mathcal{A}$ is $r_2 = a^+(ba\&(c|d))$. We convert r_2 into an RNG content model, in which we use the *interleave* indicator in RNG to replace $\&$, *choice* for $|$, *group* for $\cdot$ and *oneOrMore* for $+$. The inferred RNG content model is shown in Fig. 3.

Since the data types of RNG are not as rich as those of XSD, we next see an example of inferring XSD data types.

Example 2. Here is an example of inferring XSD data types. As shown in Fig. 4, there are four positive XML documents that record location information, from which we infer the

[1]A deterministic k-OA, a specific kind of deterministic automaton with (i) labels placed on states rather than on edges in which (ii) each alphabet symbol occurs at most k times.



(a) k-OA $\mathcal{A}$ (b) marked k-OA $\overline{\mathcal{A}}$ (c) marked k-OA

Fig. 2: The main process of k-OA synthesizing.

RE $name \cdot (latitude \cdot longitude|coordinates)$ by using *RE generator*. Note that existing work can only use text content for the lexical space analysis, e.g., 55.95 is regarded as an float type; while we believe that not only from the text (e.g., 55.95), but also the semantics of the element (e.g., latitude) and its context are useful for type inference. So naturally we think of using natural language processing techniques to analyze context semantics. Specifically, first we embed the inferred RE and the REs in our library into vectors by using the *CBOW* model [16]. Then, we find the most similar RE $(longitude \cdot latitude|coordinates) \cdot precision^?$ by vector similarity computation, and return the corresponding types {*"longitude": "LongitudeType"; "latitude": "LatitudeType"; "coordinates": "CoordinatesStructure"*} to the inferred RE. For the $name$ element, we give it a *string* type by lexical space analysis. Similarly, We convert $name \cdot (latitude \cdot longitude|coordinates)$ into an XSD content model, in which we use *choice* to replace "|", and *sequence* for ".". Finally, we combine the content model and the corresponding types into an XSD. Our method can also be applied to other type inferences, such as RDF.

B. Interactive Schema Design

Most existing approaches focus on purely automatic inference of an XML schema. The problem is that the resulting schema may be unsatisfactory (i.e., over-fitting or under-fitting) when the samples in the user's hand are not sufficient. Hence, a natural improvement is to exploit user interactions, in which the user may adjust the input until the result is satisfactory. Let us look at an example that illustrates the process of interacting with users.

Example 3. Suppose the user has three positive XML files, as shown in Fig. 5. Firstly, FlashSchema returns the resulting schema <!ELEMENT root (a^+b^+) >. Suppose she is not satisfied with the resulting schema, because she does not want the element a to appear more than twice. So the user constructs a negative XML file, as shown in Fig. 5, and feeds the negative sample back to FlsahSchema. Then, FlsahSchema

```
<element name="root">
  <oneOrMore> <element name="a"/> </oneOrMore>
  <interleave>
    <group> <element name="b"/> <element name="a"/> </group>
    <choice> <element name="c"/> <element name="d"/> </choice>
  </interleave>
</element>
```

Fig. 3: The inferred RNG content model.

```
< location >
  <name> Edinburgh </name>
  <latitude> 55.95 </latitude>
  <longitude> -3.19 </longitude>
</location >
```

```
<location>
  <name> Beijing </name>
  <latitude> 39.91 </latitude>
  <longitude> 116.40 </longitude>
</location>
```

```
<location>
  <name> Boston </name>
  <coordinates> 42°21' N,71°3' W
  </coordinates>
</location >
```

```
<location>
  <name> Melbourne </name>
  <coordinates> 37°48' S, 144°56' E
  </coordinates>
</location>
```

Fig. 4: Four XML files that record location information.

get a new schema <!ELEMENT root $(aa^?b^+)$ > based on the new input. The user finds that the new resulting schema is consistent with her intention—she is now satisfied, and the interactive process stops.

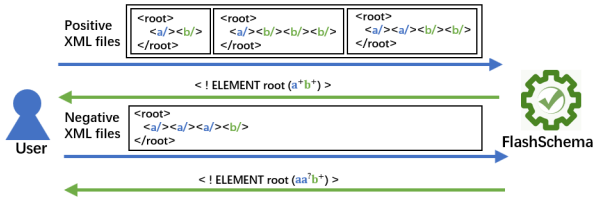


Fig. 5: An interaction example.

C. Schema Recommendation

We have collected 532,230 XML schema files from the real world [9] and built the largest XML schema library so far through a series of processing (such as schema normalization [9], splitting the schema files to get the content models and their corresponding REs). We can return the most practical schemas in reality through SchemaRank [9], which contain the XML elements that the user provides to our tool. These widely used schemas in reality have been tested by practice, and hence are of great value to non-expert users who have limited knowledge on XML, and even to experts who do not wish to design schemas from scratch. In addition, *schema recommendation* also helps automatic schema repair and cleaning.

III. Experiments

All our experiments are conducted on a machine with 16 cores Intel Xeon CPU E5620 @ 2.40GHz with 12MB Cache, 24GB RAM, and Windows 10 operating system, implemented in Python.

A. Datasets, Algorithms and Tools

We use two publicly available datasets, *incollection* and *www*, as XML inference sources. They were extracted from DBLP, a frequently-used data repository for XML schema inference. *incollection* is a collection of $45,985$ books, with an alphabet of size 14 and *www* is the large dataset, containing $2,000,226$ URLs with an alphabet of size 10. The experimental comparisons involve five state-of-the-art learning algorithms and four publicly available schema inference tools used in the field. The learning algorithms include Soa2Sore [6], Soa2Chare [6], iDREGEX [4], GenESIRE [8], and iKOIRE [10]. We abbreviate them as *S2S, S2C, IDR,*

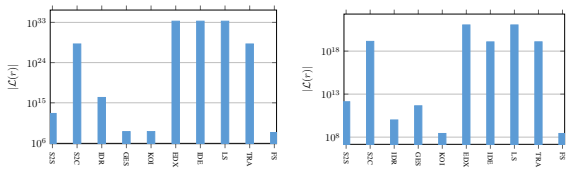
GES and *KOI*, respectively. While the public schema inference tools include EditiX, IntelliJ IDEA, Liquid Studio, and Trang. We abbreviate them as *EDX, IDE, LS*, and *TRA*, respectively. FlashSchema is abbreviated as *FS*.

B. Running Time

The experiments are conducted on the large dataset *www*. The result is displayed in TABLE I. We can see that *IDR* took almost one day and *KOI* over five hours, while others only need seconds to produce the REs. Remark that only *IDR* and *KOI* and *FS* support k-ORE inference, this may explain why they cost much longer time to get a result. It is noteworthy that *FS* can finish the task within a few seconds.

TABLE I: Running time of the related work (in seconds).

Name	S2S	S2C	IDR	GES	KOI	EDX	IDE	LS	TRA	**FS**
Time	1.3	0.9	86307.9	3.4	18762.1	13.3	7.1	3.5	21.9	**5.2**



(a) incollection (b) www

Fig. 6: The $|\mathcal{L}(r)|$ of learning algorithms/inference tools.

C. Preciseness and Conciseness

We choose *Language Size* ($|\mathcal{L}(r)|$) [4], [10] as the criteria to evaluate the balance between **preciseness** and **conciseness**. The results are shown in Fig. 6. Overall, two datasets share similar trends with only slight differences. It is clear that the largest figure ($1.86 * 10^{33}$) is achieved by *EDX, IDE*, and *LS* on the first dataset, whereas the smallest $|\mathcal{L}(r)|$ ($2.21 * 10^{8}$) is reached by *FS*, meaning that *FS* strikes the best balance between preciseness and conciseness. Note that the reason why our *FS* achieves such good results especially in inferring RNGs is due to its unrestricted support of *interleaving*. By contrast, the tools that claim the support of RNG inference (i.e., *EDX*, *IDE*, and *TRA*) give the $|\mathcal{L}(r)|$ of RNGs that is the same as that of DTDs/XSDs. Upon examining the reasons, we find that their RNGs are in fact *structurally equivalent* to the DTDs/XSDs except for the grammar itself. That is, the RNGs that they produce cannot reflect the power of RNG.

IV. Conclusion

In this paper, we devise algorithms along with a tool called FlashSchema for high quality schema design, which supports both one-pass and interactive schema design, as well as schema recommendation. One salient feature of our work is that we devise powerful and efficient algorithms to infer k-OIREs from both positive and negative samples. These algorithms also form the basis of an interactive schema design. Another feature is that, starting from the large-scale schema data that we have harvested from the Web, we provide a new solution for type inference, as well as propose schema recommendation for schema design. We have also conducted a series of experiments on two XML datasets, comparing with 9 state-of-the-art algorithms and open-source tools in terms of running time, preciseness, and conciseness. Experimental results show that our work achieves the highest level of preciseness and conciseness within only a few seconds. Experimental results and examples also demonstrate the effectiveness of our type inference and schema recommendation methods. To the best of our knowledge, no other existing tools can support interactive schema design and schema recommendation.

Acknowledgment

Haiming Chen, Yeting Li and Qiancheng Peng are supported in part by the National Natural Science Foundation of China under Grant Nos. 61872339, 61472405. Tingjian Ge is supported in part by NSF grant IIS-1633271. Zhiwu Xu is supported by the National Natural Science Foundation of China under Grant No. 61972260 and Guangdong Basic and Applied Basic Research Foundation under Grant No. 2019A1515011577.

References

[1] S. Grijzenhout and M. Marx, "The quality of the XML web," *J. Web Semant.*, vol. 19, pp. 59–68, 2013.
[2] M. Garofalakis, A. Gionis, K. Shim, K. Shim, and K. Shim, "XTRACT: Learning document type descriptors from XML document collections," *Data Mining and Knowledge Discovery*, vol. 7, no. 1, pp. 23–56, 2003.
[3] G. J. Bex, F. Neven, T. Schwentick, and K. Tuyls, "Inference of concise dtds from XML data," in *Proceedings of the 32nd VLDB*, 2006, pp. 115–126.
[4] G. J. Bex, W. Gelade, F. Neven, and S. Vansummeren, "Learning deterministic regular expressions for the inference of schemas from XML data," *TWEB*, vol. 4, no. 4, pp. 14:1–14:32, 2010.
[5] R. Ciucanu and S. Staworko, "Learning schemas for unordered XML," in *Proceedings of the 14th DBPL*, 2013, pp. 31–40.
[6] D. D. Freydenberger and T. Kötzing, "Fast learning of restricted regular expressions and DTDs," *Theory of Computing Systems*, vol. 57, no. 4, pp. 1114–1158, 2015.
[7] Y. Li, X. Mou, and H. Chen, "Learning concise Relax NG schemas supporting interleaving from XML documents," in *Proceedings of the 14th ADMA*, 2018, pp. 303–317.
[8] Y. Li, X. Zhang, H. Xu, X. Mou, and H. Chen, "Learning restricted regular expressions with interleaving from XML data," in *Proceedings of the 37th ER*, 2018, pp. 586–593.
[9] Y. Li, X. Chu, X. Mou, C. Dong, and H. Chen, "Practical study of deterministic regular expressions from large-scale XML and schema data," in *Proceedings of the 22nd IDEAS*, 2018, pp. 45–53.
[10] Y. Li, X. Zhang, J. Cao, H. Chen, and C. Gao, "Learning k-occurrence regular expressions with interleaving," in *Proceedings of the 24th DASFAA*, 2019, pp. 70–85.
[11] E. M. Gold, "Language identification in the limit," *Information and Control*, vol. 10, no. 5, pp. 447–474, 1967.
[12] D. Florescu, "Managing semi-structured data," *ACM Queue*, vol. 3, no. 8, pp. 18–24, 2005.
[13] J. Hegewald, F. Naumann, and M. Weis, "Xstruct: Efficient schema extraction from multiple and large XML documents," in *Proceedings of the 22nd ICDE Workshops*, 2006, p. 81.
[14] B. Chidlovskii, "Schema extraction from XML collections," in *Proceedings of the 2nd JCDL*, 2002, pp. 291–292.
[15] Y. Li, H. Chen, X. Zhang, and L. Zhang, "An effective algorithm for learning single occurrence regular expressions with interleaving," in *Proceedings of the 23rd IDEAS*, 2019, pp. 24:1–24:10.
[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the 1st ICLR*, 2013.