

On the Complexity of Schema Inference from Web Pages in the Presence of Nullable Data Attributes

Guizhen Yang
Department of Computer
Science and Engineering
University at Buffalo
Buffalo, NY 14260, USA
gzyang@cse.buffalo.edu

I. V. Ramakrishnan
Department of Computer
Science
Stony Brook University
Stony Brook, NY 11794, USA
ram@cs.stonybrook.edu

Michael Kifer
Department of Computer
Science
Stony Brook University
Stony Brook, NY 11794, USA
kifer@cs.stonybrook.edu

ABSTRACT

An increasingly large number of Web pages are machine-generated by filling in templates with data stored in backend databases. These templates can be viewed as the implicit schemas of those Web pages. The ability to infer the implicit schema from a collection of Web pages is important for scalable data extraction, since the inferred schema can be used to automatically identify schema attributes that are “encoded” in Web pages.

However, the task of inferring a “good” schema is complicated due to the existence of nullable (missing) data attributes. Usually if an attribute contains a *null* value, then it will be omitted in the generated Web page, giving rise to different variations and permutations of layout structures in Web pages that are generated from the same template.

In this paper we investigate the complexity of schema inference from Web pages in the presence of nullable data attributes. We introduce the notion of unambiguity as a quality measure for inferred schemas and prove that the problem of inferring “good” (unambiguous) schemas is NP-complete. Our complexity results imply that ambiguity resolution is one of the root causes of the computational difficulty underlying schema inference from Web pages.

Categories and Subject Descriptors

F.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems

General Terms

Theory

Keywords

Data Extraction, Web Mining, Wrapper Induction, Schema Inference, Data Mining, Machine Learning, World Wide Web

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'03, November 3–8, 2003, New Orleans, Louisiana, USA.
Copyright 2003 ACM 1-58113-723-0/03/0011 ...\$5.00.

1. INTRODUCTION

A growing number of Web sites are maintained using content management software and an increasingly large number of Web pages are machine-generated from backend databases. Normally such Web pages possess an implicit, fixed schema, which underlies the templates used for generating these Web pages. By and large the process of generating Web pages can be considered as one of instantiating the implicit schema with the attribute data stored in backend databases.

Knowing the implicit schema can help uncover the attribute data “encoded” in Web pages and thus automate the process of data extraction. Along with learning-based data extraction techniques [21, 16, 6, 20, 4, 7], schema inference approaches to data extraction are becoming important since they exhibit a high degree of automation and scalability. Recently interesting works on inferring the schema of a collection of template-driven Web pages were reported in [13, 8, 3]. Grumbach and Mecca [13] first introduced the schema inference problem and established several complexity results assuming the collection of Web pages satisfy certain properties such as *data richness*. In [8] the inferred schema is represented as a union-free regular expression with optionals (*i.e.*, the “?” operator). But the sophisticated algorithm proposed in [8] for discovering a “good” schema can suffer from exponential blowup. Recently the work of Arasu and Garcia-Molina [3] describes efficient heuristics for the schema inference problem.

However, the above proposals all assume that the Web pages from which the schema is to be inferred should be of “good quality”. In practice the schema inference problem is complicated by all kinds of variations in Web pages. One such complication can be traced to *nullable* data attributes. In a typical template, if an attribute’s value is *null*, then the attribute is omitted in the generated Web page. This omission gives rise to a multitude of possible variations in the layout structures of the Web pages generated from the same template (see Section 2 for a motivating example).

Therefore, in the presence of nullable data attributes it is required that the inferred schema be able to accommodate variations in Web pages. Moreover, when used to extract attribute values, the inferred schema should also be able to *disambiguate* different occurrences of different attributes. Such a property is highly desirable since an *unambiguous* schema can boost the precision level of the extracted data.

The quality of inferred schemas has been a relatively unexplored subject in the literature. In this paper we intro-

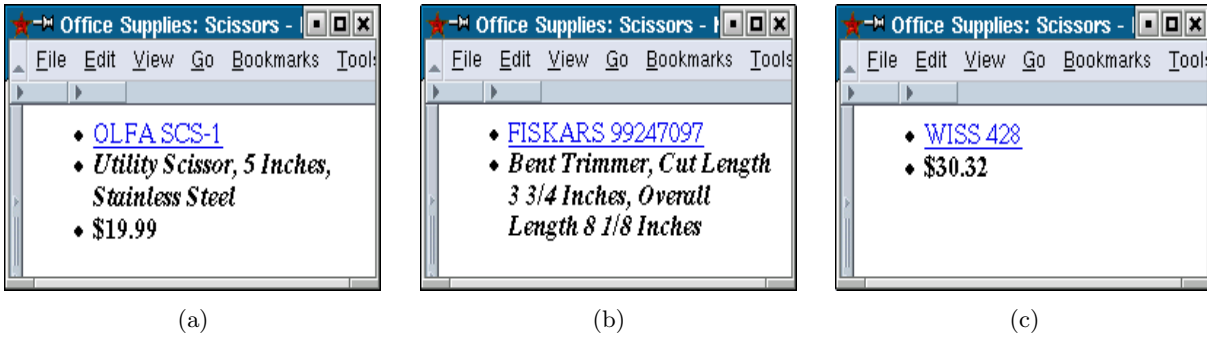


Figure 1: Web Pages Listing Product Information

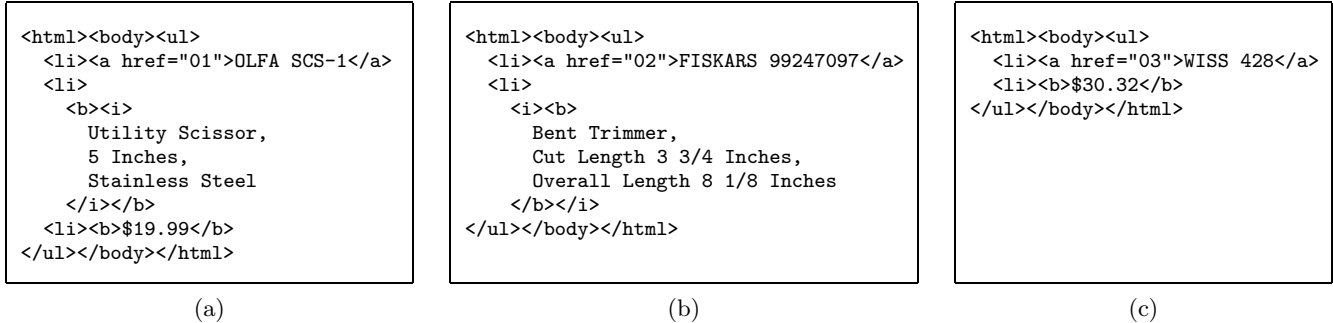


Figure 2: HTML Documents

duce the notion of *unambiguity* as a quality measure for inferred schemas. Intuitively, the notion of a good schema corresponds to our notion of an unambiguous schema. We formally study the complexity of schema inference from Web pages in the presence of nullable data attributes. In particular, we prove that the problem of inferring unambiguous schemas¹ is NP-complete. Our results point to ambiguity resolution as one of the root causes of the computational difficulty underlying schema inference from Web pages.

The rest of this paper is organized as follows. Section 2 presents a motivating example to illustrate the significance of unambiguous schema inference. In Section 3 we formalize the problem of unambiguous schema inference and present our complexity results. Section 4 discusses related work and Section 5 concludes this paper.

2. A MOTIVATING EXAMPLE

Web pages of data-intensive Web sites are usually automatically generated from backend databases. Consider the three Web pages shown in Figure 1. They list product description and pricing information about scissors. Each Web page corresponds to a database record about a scissor item. In our example the schema of the hidden relational database can be considered as a triple as follows

$$\langle ModelNumber, Description, Price \rangle$$

Here *ModelNumber* is a key attribute containing the manufacture model number, *Description* is a short text about product information, and *Price* is the unit sales price.

¹As in [13, 8], here we use union-free regular expressions to represent schemas of HTML documents.

However, observe that while Figure 1(a) lists all three attributes of a scissor item, the *Price* attribute is missing in Figure 1(b) and *Description* missing in Figure 1(c). Normally if an attribute contains a null value, then it is *completely* omitted in the generated HTML document.² To reflect the fact that the *Description* and *Price* attributes are *nullable*, we represent the implicit database schema as follows

$$\langle ModelNumber, Description?, Price? \rangle$$

Here a question mark (“?”) following an attribute name denotes that this attribute is nullable.

Despite the fact that some attributes may be missing in the generated Web pages, there still is a high degree of consistency in the presentation style for encoding different attributes of a database record. Note that in all three Web pages of Figure 1: (i) manufacture model number is always followed by product description (if present) followed by unit sales price (if present); (ii) manufacture model number is always marked by a hyperlink while product description is presented using bold, italic font and unit sales price using bold font. Such consistency can be verified from the HTML documents shown in Figures 2(a), 2(b), and 2(c), which correspond to the Web pages shown in Figures 1(a), 1(b), and 1(c), respectively.³

²Note that in the problem settings of [13] some form of markup encoding is *always* generated for a null data value. This difference in handling nullable data attributes has led to tractable schema inference problems [13]. See Section 4 for a detailed account of related work.

³Note that the two sequences of HTML tags, `<i>` and `<i>` in Figures 2(a) and 2(b), respectively, render the same bold, italic font in a Web browser.

is *ambiguous*. Clearly, an ambiguous schema exhibits “bad” quality since it compromises precision of the extracted data. Therefore, a highly desirable property that we want to require of an inferred schema is that it be *unambiguous*.

So the second important question is:

Can we infer an unambiguous schema that generalizes a collection of Web pages?

This is the problem that we will study in the rest of this paper. Observe that ambiguity of the schema in Figure 4 is due to its failure in identifying the `<i>` tag as the *distinguishing* presentation style for the *Description* attribute. If such a feature is identified, then we can obtain an *unambiguous* schema as shown in Figure 5. To this end, we can claim that the schema in Figure 5 has “good” quality.

3. PROBLEM FORMALIZATION AND COMPLEXITY RESULTS

In this section we formalize the problem of unambiguous schema inference from Web pages. We will show that the problem of inferring an unambiguous schema from Web pages can be cast as a problem of inferring union-free regular expressions (possibly with optionals) that are consistent with respect to positive and negative examples.

Firstly, we can view the generalized schemas in Figures 4 and 5 as two regular expressions, S_1 and S_2 , respectively. But these expressions are not arbitrary regular expressions. Instead, their syntax is restricted to union-free regular expressions with optionals, which are formally defined below.

Definition 1 (Union-Free Regular Expressions) *Let Σ be a finite alphabet. A union-free regular expression (abbr. UFRE) over Σ is defined inductively as follows:*

- Any symbol $c \in \Sigma$ is a UFRE.
- If E_1 and E_2 are UFREs, so is $E_1 \cdot E_2$.
- If E is a UFRE, so are E^* and $E?$.

In the above definition, E^* means E can be repeated zero or multiple times while $E?$ means zero or once, *i.e.*, optional. For instance, $(ab)^* \cdot d?$ is a valid UFRE. Note that $E? = \varepsilon|E$, where ε denotes the empty string. Therefore, the above definition actually allows a very restricted use of the union operator (“|”). But for simplicity we will still call such expressions union-free regular expressions.

We should point out that the union-free regular expressions described here are exactly the same as those used in [13, 8] for representing schemas of HTML documents. Following the standard convention, we will use $\mathcal{L}(E)$ to denote the set of strings recognized by a regular expression E .

Secondly, we can view the specialized schemas in Figure 3 as strings. Since these strings basically mirror the collection of HTML documents in Figure 2, they serve as *positive* examples from which a schema needs to be inferred. Let us denote this set of strings as POS . Then clearly $POS \subseteq \mathcal{L}(S_1)$ and $POS \subseteq \mathcal{L}(S_2)$ (S_1 and S_2 represent the generalized schemas in Figures 4 and 5, respectively). This condition means that a *schema should generalize all the positive examples*.

But we only see Web pages coming from a particular Web site as positive examples. Where do the *negative* examples

come from? In fact, negative examples can be *implicitly* derived from positive examples. For instance, from the positive examples in Figure 3 we can derive the two negative examples shown in Figure 6. Intuitively, these two negative examples imply that the *Price* attribute is not presented using bold, italic font and the *Description* attribute is not presented using bold font alone. Clearly, if an inferred schema accepts either of these two examples, then ambiguity can arise when this schema is used to extract attribute values from an HTML document.

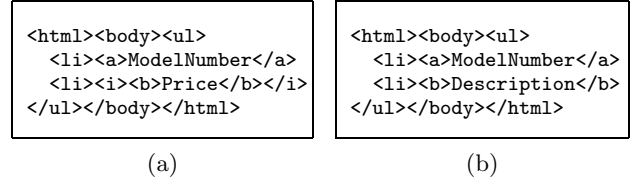


Figure 6: Implicitly Derived Negative Examples

Therefore, assuming that the strings in Figure 6 serve as negative examples, we would expect that a *good schema should exclude all the negative examples*. Observe that the ambiguous schema in Figure 4 accepts the negative example in Figure 6(b) whereas the unambiguous schema in Figure 5 does not. Technically, if we denote the set of strings in Figure 6 as NEG , then $NEG \cap \mathcal{L}(S_1) \neq \emptyset$, $NEG \cap \mathcal{L}(S_2) = \emptyset$. This is basically what distinguishes a good schema (S_2) from a bad one (S_1).

In summary, an unambiguous schema can be considered as a *union-free regular expression which accepts all the positive examples but excludes all the negative examples*. To formalize the above discussion, we have the following definition.

Definition 2 (Consistency) *Let POS and NEG be two sets of strings. Given a union-free regular expression E , we say that E is consistent w.r.t. $\langle POS, NEG \rangle$, iff $\mathcal{L}(E) \supseteq POS$ and $\mathcal{L}(E) \cap NEG = \emptyset$.*

In the above Definition 2, the strings in POS serve as positive examples while those in NEG serve as negative examples. Intuitively, a consistent UFRE generalizes all positive examples but excludes all negative examples. For instance, aab^* is consistent w.r.t. $\{\{aa, aab\}, \{ab, cd\}\}$ whereas a^*b^* is not, since the negative example $ab \in \mathcal{L}(a^*b^*)$.

Given a collection of Web pages as positive examples, our goal is to infer an unambiguous schema (represented using a UFRE) from these Web pages. Although we are not going to define how the set of implicit negative examples can be derived, we will informally say that an inferred schema is unambiguous iff it is consistent w.r.t. the positive examples and the (implicitly derived) negative examples.

Now our unambiguous schema inference problem can be stated in the following form.

Problem 1 (Consistent UFRE) *Given two sets of strings POS and NEG , is there a UFRE that is consistent w.r.t. $\langle POS, NEG \rangle$?*

Is there *always* a UFRE that is consistent w.r.t. two sets of positive and negative examples? The answer turns out to be “No”. For instance, it can be shown that there is no UFRE that is consistent w.r.t. $\{\{a, b\}, \{ab, ba\}\}$.

It turns out that the consistent UFRE problem is intractable in general.

Theorem 1 *The consistent UFRE problem is NP-complete.*

Proof. See Appendix A. \square

Interestingly, we can also look at the problem of unambiguous schema inference from a slightly different perspective. Recall that in the motivating example of Section 2 the database schema is represented as

$$\langle \text{ModelNumber}, \text{Description?}, \text{Price?} \rangle$$

where the attributes *Description* and *Price* are nullable. Therefore, it is not surprising to see that the corresponding schema for these Web pages is roughly in the form of $\alpha \cdot M \cdot (D)? \cdot (P)? \cdot \beta$, where α, β are some strings and M, D, P represent the subschemas for the attributes *ModelNumber*, *Description*, and *Price*, respectively.

Recall the motivating example in Section 2. Let S_1 and S_2 represent the two generalized schemas in Figures 4 and 5, respectively. Moreover, let us define⁵

$$\begin{aligned} \alpha &= \langle \text{html} \rangle \langle \text{body} \rangle \langle \text{ul} \rangle \\ \beta &= \langle \text{/ul} \rangle \langle \text{/body} \rangle \langle \text{/html} \rangle \\ M &= \langle \text{li} \rangle \langle \text{a} \rangle \# \text{PCDATA} \langle \text{/a} \rangle \\ P &= \langle \text{li} \rangle \langle \text{b} \rangle \# \text{PCDATA} \langle \text{/b} \rangle \\ D_1 &= \langle \text{li} \rangle \langle \text{i} \rangle? \langle \text{b} \rangle \langle \text{i} \rangle? \# \text{PCDATA} \langle \text{/i} \rangle? \langle \text{/b} \rangle \langle \text{/i} \rangle? \\ D_2 &= \langle \text{li} \rangle \langle \text{b} \rangle? \langle \text{i} \rangle \langle \text{b} \rangle? \# \text{PCDATA} \langle \text{/b} \rangle? \langle \text{/i} \rangle \langle \text{/b} \rangle? \end{aligned}$$

Then S_1 can be represented as $\alpha \cdot M \cdot (D_1)? \cdot (P)? \cdot \beta$ and S_2 as $\alpha \cdot M \cdot (D_2)? \cdot (P)? \cdot \beta$.

Also let POS_D and POS_P be the sets of occurrences⁶ of the attributes *Description* and *Price*, respectively (see Figure 2):

$$\begin{aligned} POS_D &= \left\{ \langle \text{li} \rangle \langle \text{b} \rangle \langle \text{i} \rangle \# \text{PCDATA} \langle \text{/i} \rangle \langle \text{/b} \rangle \right\} \\ POS_P &= \left\{ \langle \text{li} \rangle \langle \text{i} \rangle \langle \text{b} \rangle \# \text{PCDATA} \langle \text{/b} \rangle \langle \text{/i} \rangle \right\} \\ POS_P &= \left\{ \langle \text{li} \rangle \langle \text{b} \rangle \# \text{PCDATA} \langle \text{/b} \rangle \right\} \end{aligned}$$

We can think of POS_D and POS_P as two sets of positive examples for the attributes *Description* and *Price*, respectively. Now looking inside the structures of S_1 and S_2 , we can see that both D_1 and D_2 generalize the set of examples in POS_D , i.e., $POS_D \subseteq \mathcal{L}(D_1)$, $POS_D \subseteq \mathcal{L}(D_2)$.

However, if we take POS_P (the set of positive examples for the attribute *Price*) as the set of negative examples for the attribute *Description*, then it is straightforward to verify that D_1 is *not* consistent w.r.t. $\langle POS_D, POS_P \rangle$ whereas D_2 is. Clearly, the ambiguity of the schema S_1 is due to its inconsistency w.r.t. the negative examples (observe that in this case the set of positive examples for one attribute is a set of negative examples for another attribute).

In principle, to infer a schema in which multiple attributes, A_1, \dots, A_n , are nullable, we need to infer a subschema for each attribute A_i . So given a list of sets of positive examples, P_1, \dots, P_n , for the attributes, A_1, \dots, A_n , respectively, it is highly desirable to infer a subschema, E_i , for each attribute A_i such that E_i accepts all the positive examples in A_i but excludes all the positive examples of other attributes combined. This notion is formalized in the following definition.

⁵We have replaced all the attributes names with `#PCDATA`, assuming that they all match text strings.

⁶In our example all attribute values are just text strings. For simplicity we use `#PCDATA` to denote any text string.

Definition 3 (Unambiguity) *Given a list of sets of strings, (P_1, \dots, P_n) , and a list of UFREs, (E_1, \dots, E_n) , we say that (E_1, \dots, E_n) is unambiguous w.r.t. (P_1, \dots, P_n) , iff E_i is consistent w.r.t. $\langle P_i, \bigcup_{j \neq i} P_j \rangle$ for all $1 \leq i \leq n$.*

However, even when a list of UFREs is unambiguous w.r.t. the given examples, the sets of strings recognized by these UFREs may overlap. In such a case ambiguity can still arise since the overlapping UFREs may match the same text string in a given Web page. Therefore, a even more desirable, better quality that we want to impose is that the sets of strings recognized by these UFREs be pairwise disjoint. If so, then we say the list of UFREs is *inherently unambiguous*.

Definition 4 (Inherent Unambiguity) *Let (P_1, \dots, P_n) be a list of sets of strings and (E_1, \dots, E_n) be a list of UFREs. We say that (E_1, \dots, E_n) is inherently unambiguous w.r.t. (P_1, \dots, P_n) , iff $\mathcal{L}(E_i) \supseteq P_i$ for all $1 \leq i \leq n$, and $\mathcal{L}(E_i) \cap \mathcal{L}(E_j) = \emptyset$ for all $1 \leq i \leq n, 1 \leq j \leq n, i \neq j$.*

Inherently unambiguous UFREs are able to retain more precision than those that are only unambiguous w.r.t. the given examples. Therefore, when we cast the problem of unambiguous schema inference as the problem of inferring multiple subschemas for multiple (nullable) data attributes, we obtain two different types of unambiguity — either unambiguity w.r.t. the given examples or inherent unambiguity. We formally state our new schema inference problems and complexity results below.

Problem 2 (Unambiguous UFREs) *Given a list of sets of strings, (P_1, \dots, P_n) , is there a list of UFREs, (E_1, \dots, E_n) , such that (E_1, \dots, E_n) is unambiguous w.r.t. (P_1, \dots, P_n) ?*

Problem 3 (Inherently Unambiguous UFREs) *Given a list of sets of strings, (P_1, \dots, P_n) , is there a list of UFREs, (E_1, \dots, E_n) , such that (E_1, \dots, E_n) is inherently unambiguous w.r.t. (P_1, \dots, P_n) ?*

Theorem 2 *The unambiguous UFREs problem is NP-complete.*

Theorem 3 *The inherently unambiguous UFREs problem is decidable.*

Proof. [Sketch]

We can show that given a list of sets of strings, (P_1, \dots, P_n) , if there exists a list of UFREs, (E_1, \dots, E_n) , that is inherently unambiguous w.r.t. (P_1, \dots, P_n) , then the size of each E_i is bounded by the size of P_i . Therefore, We can construct a naive algorithm to enumerate each E_i and check whether the resulting list of UFREs is inherently unambiguous w.r.t. (P_1, \dots, P_n) . \square

We should point out that Problems 1 and 2 are *not* equivalent problems. Given a pair of sets of strings, (P_1, P_2) , if there is a pair of UFREs, (E_1, E_2) , which is unambiguous w.r.t. (P_1, P_2) , then clearly E_1 is consistent w.r.t. $\langle P_1, P_2 \rangle$. Therefore, a “yes” answer to Problem 2 implies a “yes” answer to Problem 1. However, the converse is not necessarily true. The existence of a UFRE that is consistent w.r.t. $\langle P_1, P_2 \rangle$ does not necessarily imply the existence of a pair of UFREs that is unambiguous w.r.t. (P_1, P_2) . For example, $(ab)?(ba)?$ is consistent w.r.t. $\langle \{ab, ba\}, \{a, b\} \rangle$.

But there is no pair of UFREs that is unambiguous w.r.t. $(\{ab, ba\}, \{a, b\})$.

Therefore, Problem 1 cannot directly reduce to Problem 2, although Theorem 1 immediately implies that Problem 2 is in NP. Hence it requires a separate treatment to establish the complexity result in Theorem 2. The proof of Theorem 2 is similar to that of Theorem 1 but is omitted here due to space limitation.

Problems 2 and 3 are *not* equivalent problems either. Problem 2 cannot directly reduce to Problem 3. For example, the following pair of UFREs

$$((a^?)(b^?)(a^?)(cbbc)^?(bc^*b)^?, (c^?)(b^?)(c^?)(abba)^?(ba^*b)^?)$$

is unambiguous w.r.t. the following pair of sets of strings

$$(\{ab, ba, cbbc, bccb, bcb\}, \{cb, bc, abba, baab, bab\})$$

But it can be shown that there is no pair of UFREs that is inherently unambiguous w.r.t. these two sets of strings above. In fact, we can show that for any pair of UFREs, (E_1, E_2) , that is unambiguous w.r.t. the two sets of strings above, it must be true that $b \in \mathcal{L}(E_1) \cap \mathcal{L}(E_2)$.

Because it can be done in polynomial time to check whether two regular expressions are disjoint, it follows that Problem 3 is in NP. Although we do not know yet exactly the complexity of the inherently unambiguous UFREs problem, our conjecture is that this problem is also NP-complete.

Finally, summarizing all the main results of this section, we make the following claim.

Corollary 4 *It is an NP-complete problem to decide whether there exists an unambiguous schema, represented using union-free regular expressions, that generalizes an arbitrary collection of Web pages with multiple nullable data attributes.*

4. RELATED WORK

Our work is closely related to the grammar inference problem which was first addressed in the seminal works of Gold and Angluin. Gold [11, 12] proved that the problem of inferring a DFA of *minimum* size from positive examples is NP-complete. In [1] Angluin showed that the problem of inferring a regular expression (with no restriction on the use of unions) of *minimum* size from positive and negative examples is NP-complete. In this paper, however, we restrict the syntax to union-free regular expressions and do not impose any constraint on the size of the inferred UFREs. Our problems of inferring consistent UFREs and unambiguous lists of UFREs do not have equivalent counterparts in the classical works on grammar inference and hence none of the known results there is applicable.

In [2] Angluin proposed a polynomial time algorithm for *actively* learning the minimum DFA of a regular language from a *teacher* who knows the true identity of this regular language. We should point out that such an active learning framework is different from ours, which is passive.

There is also a large body of work on learning subsequences and supersequences from sets of strings. The following problems are all NP-complete: (1) finding either the shortest common supersequence or the longest common subsequence of an arbitrary number of strings over a binary alphabet [17, 22]; (2) finding a sequence which is a common subsequence/supersequence of a set of positive examples but not a subsequence/supersequence of any string

in a set of negative examples [15, 18]. The syntax of UFREs is much more expressive than plain strings and hence much better suited for representing schematic information. Moreover, our problem of inferring an unambiguous *set* of UFREs (one per nullable attribute) has no counterpart in the area of sequence learning.

The XTRACT system for inferring schemas of XML documents was reported in [10]. However, in the problem settings of [10], the examples are labeled trees instead of strings. Moreover, the main constraint there is minimization of the size of the inferred schema. Consequently, all input examples in [10] are positive examples and our notion of unambiguity w.r.t. positive and negative examples is not explored in [10].

The complexity results presented in this paper are inspired by our recent work on multi-attribute data extraction from Web sources [23]. But in [23] our focus is on learning extraction patterns (using different syntax from UFREs) from examples and on the precision and recall metrics of the extracted data. In fact, the issue of ambiguity resolution can commonly arise in learning-based approaches to data extraction and schema inference [4, 16, 14, 7, 8, 3] but has remained relatively unexplored in the literature.

A different notion of unambiguity was introduced in our earlier work [9] to ascribe a quality measure to extraction patterns learned (using different syntax from UFREs). However, [9] is mainly concerned with the computational complexity of checking various properties (including unambiguity) of extraction patterns rather than inferring extraction patterns from examples.

Our results have direct bearing on the works reported in [14, 13, 8, 3]. In [13], several schema inference problems were introduced and shown to be solvable in polynomial time. However, the lower complexity results reported in [13] are due to the assumption of *explicit* encoding of null data values. Our results have shown that if null data values are completely omitted (which can be commonly observed in practice), then the (unambiguous) schema inference problem quickly becomes intractable in general. Although exponential-time and polynomial-time heuristics have been proposed in [8, 3], respectively, to infer schemas from unlabeled Web pages, our complexity results imply that the schema inference problem is still computationally difficult even when the labels are already known.

5. CONCLUSION

Research on wrapper construction for Web sources has made a transition from its early focus on manual and semi-automatic approaches to fully automated techniques based on machine learning and schema inference. In this paper we have formalized the problem of schema inference from Web pages in the presence of nullable data attributes. We introduced the notion of unambiguity as an important quality measure for inferred schemas and studied the schema inference problem from several different perspectives, proving that the problem of inferring a good schema is NP-complete.

Our results provide a theoretical basis for the complexity-related aspects of unambiguous data extraction and schema inference. However, it should be noted that our complexity results, although pessimistic, deal with *worst-case* scenarios. In practice many Web sources exhibit a rather high degree of regularity in presentation styles, which is, in fact, a hallmark of a good Web site design. For such Web sources efficient

heuristics can still be developed to construct robust wrapper systems. The notion of unambiguity and its impact on precision of the extracted data can provide important guidance in building these systems.

An important issue raised by our complexity results has to do with training examples. Recall that it was assumed that the schema inference problem can begin with an *arbitrary* set of examples. This assumption is a major reason for the exponential blowup in worst-case complexity. We believe that certain “good” properties in the set of training examples can lower the computation cost of schema inference even if unambiguity is sought. Therefore, it is very important to be able to obtain “clean” training data to improve the performance of fully automated data extraction techniques. We believe that good domain ontologies will be able to play a significant role in boosting the quality of training data. Generation of training examples from such ontologies appears to be a promising direction of future research.

6. ACKNOWLEDGMENTS

This work was supported in part by NSF grants IIS-0072927, CCR-0205376, and CCR-0311512. The authors would like to thank the anonymous referees for their comments and suggestions that help improve the quality of this paper.

7. REFERENCES

- [1] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39(3):337–350, 1978.
- [2] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.
- [3] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *ACM International Conference on Management of Data (SIGMOD)*, 2003.
- [4] N. Ashish and C. Knoblock. Wrapper generation for semi-structured internet sources. *SIGMOD Record*, 26(4):8–15, 1997.
- [5] P. Atzeni, G. Mecca, and P. Merialdo. To weave the web. In *International Conference on Very Large Data Bases (VLDB)*, 1997.
- [6] B. Chidlovskii. Wrapping web information providers by transducer induction. In *European Conference on Machine Learning*, 2001.
- [7] W. Cohen, M. Hurst, and L. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *International World Wide Web Conference (WWW)*, 2002.
- [8] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards automatic data extraction from large web sites. In *International Conference on Very Large Data Bases (VLDB)*, 2001.
- [9] H. Davulcu, G. Yang, M. Kifer, and I. V. Ramakrishnan. Computational aspects of resilient data extraction from semistructured sources. In *ACM International Symposium on Principles of Database Systems (PODS)*, 2000.
- [10] M. N. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: A system for extracting document type descriptors from XML documents. In *ACM International Conference on Management of Data (SIGMOD)*, 2000.
- [11] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.
- [12] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.
- [13] S. Grumbach and G. Mecca. In search of the lost schema. In *International Conference on Database Theory (ICDT)*, 1999.
- [14] C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.
- [15] T. Jiang and M. Li. On the complexity of learning strings and sequences. *Theoretical Computer Science*, 119(2):363–371, 1993.
- [16] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.
- [17] D. Maier. The complexity of some problems on subsequences and supersequences. *Journal of ACM*, 25(2):322–336, 1978.
- [18] M. Middendorf. On finding various minimal, maximal, and consistent sequences over a binary alphabet. *Theoretical Computer Science*, 145(1-2):317–327, 1995.
- [19] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [20] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Third International Conference on Autonomous Agents (Agents’99)*, 1999.
- [21] M. Perkowitz, R. B. Doorenbos, O. Etzioni, and D. S. Weld. Learning to understand information on the Internet: An example-based approach. *Journal of Intelligent Information Systems*, 8(2):133–153, 1997.
- [22] K.-J. Räihä and E. Ukkonen. The shortest common supersequence problem over binary alphabet is NP-complete. *Theoretical Computer Science*, 16:187–198, 1981.
- [23] G. Yang, S. Mukherjee, and I. V. Ramakrishnan. On precision and recall of multi-attribute data extraction from semistructured sources. In *IEEE International Conference on Data Mining (ICDM)*, 2003.

APPENDIX

A. PROOF OF NP-COMPLETENESS

In this section we will formally prove the complexity result in Theorem 1. In the sequel, we will use ε to denote either the empty string or the empty expression. Its intended usage should be clear from the context. We will also use the notation α^k , where α is a string and k an integer, to represent the string obtained by repeating k times the string α . In particular, $\alpha^0 = \varepsilon$.

Let POS and NEG be two sets of strings. First, deciding whether or not a string is accepted by a regular expression can be done in polynomial time. Second, we can show that the size of the shortest UFRE that is consistent w.r.t. $\langle POS, NEG \rangle$ is bounded by the size of POS and NEG . Therefore, this problem is in NP.

To prove that this problem is NP-hard, we will reduce SAT to our problem. We will also assume the alphabet $\Sigma = \{\$, 0, 1\}$.

Let F be a propositional formula in *conjunctive* normal form which has m clauses C_1, C_2, \dots, C_m and n variables V_1, V_2, \dots, V_n . For $1 \leq i \leq m$ and $1 \leq j \leq n$, let us define:

$$F_{ij} = \begin{cases} \$10, & \text{if } V_j \text{ appears positively in } C_i; \\ \$01, & \text{if } V_j \text{ appears negatively in } C_i; \\ \$0, & \text{if } V_j \text{ does not appear in } C_i. \end{cases}$$

The idea is that in a string we will use $\$01$ and $\$10$ to represent the logical values *true* and *false*, respectively. Thus for all $1 \leq i \leq m$, the string $F_{i1}F_{i2} \dots F_{in}$ encodes the *only* assignment of truth values to the variables, V_1, V_2, \dots, V_n , which makes the clause C_i *false*. Moreover, define:

$$\begin{aligned} POS &= \{(\$0)^n, (\$1)^n\} \\ NEG &= N_1 \cup N_2 \cup N_3 \\ N_1 &= \{\$^k \mid 0 \leq k \leq n-1\} \\ N_2 &= \{\$^k 00\$^{n-k}, \$^k 11\$^{n-k} \mid 1 \leq k \leq n\} \\ N_3 &= \{F_{i1}F_{i2} \dots F_{in} \mid 1 \leq i \leq m\} \end{aligned}$$

Next we will show that the formula F is satisfiable iff there is a UFRE that is consistent w.r.t. $\langle POS, NEG \rangle$.

We will also use the following two UFREs

$$\begin{aligned} E_t &= \$(0?)(1?) \\ E_f &= \$(1?)(0?) \end{aligned}$$

to represent the logical values *true* and *false*, respectively. First, given an assignment of truth values to the variables, V_1, V_2, \dots, V_n , in the formula F , we can construct a UFRE, $E = E_1E_2 \dots E_n$, where for all $1 \leq j \leq n$,

$$E_j = \begin{cases} E_t, & \text{iff the truth value assigned to } V_j \text{ is } \textit{true}; \\ E_f, & \text{iff the truth value assigned to } V_j \text{ is } \textit{false}. \end{cases}$$

So if the formula F is satisfiable, then there must be an assignment of truth values to the variables, V_1, V_2, \dots, V_n , which satisfies F . It can be verified that if we construct a UFRE, E , as defined above, then E is consistent w.r.t. $\langle POS, NEG \rangle$.

Now suppose that there is a UFRE, E , which is consistent w.r.t. $\langle POS, NEG \rangle$. Then it follows that $\mathcal{L}(E) \supseteq POS$ and $\mathcal{L}(E) \cap NEG = \emptyset$. We will show that from E we can obtain an assignment of truth values to the variables, V_1, V_2, \dots, V_n , which satisfies the formula F .

Let $E = A_1A_2 \dots A_i$, where each A_k ($1 \leq k \leq i$) is a single symbol ($\$, 0$, or 1), or in the form of $(X)?$ or $(X)^*$. Since $\mathcal{L}(E) \supseteq POS$, it follows that for each A_k , if A_k is a single symbol, then $A_k = \$$. Because $\mathcal{L}(E) \cap N_1 = \emptyset$, so the number of A_k 's that are the single symbol $\$$ must be exactly n . It follows that E must have the form of $B_0\$B_1\$B_2 \dots \$B_n$, where each B_k is a concatenation of expressions in the form of $(X)?$ or $(X)^*$. Moreover, if B_0 is not an empty expression, then we can remove B_0 from E , because the resulting expression would still be consistent w.r.t. $\langle POS, NEG \rangle$. Therefore, in the following we will assume that $E = \$B_1\$B_2 \dots \$B_n$.

Next we will show that each B_k in E can be transformed into either $(0?)(1?)$ or $(1?)(0?)$ and the resulting new expression E is still consistent w.r.t. $\langle POS, NEG \rangle$.

We define two transformation operations on the B_k 's as follows: (i) remove a “?” operator; (ii) remove a “.” operator together with one of its operands. We will keep performing either of these two operations on each B_k unless it gives rise to inconsistency. So when this process stops, we cannot remove any operator from any of the B_k 's and still get a new consistent expression.

We claim that when the above process ends, each B_k must be either $(0?)(1?)$ or $(1?)(0?)$. Because $\mathcal{L}(E) \supseteq POS$, it must be true that $\{\varepsilon, 0, 1\} \subseteq \mathcal{L}(B_k)$ for each B_k . Since $\mathcal{L}(E) \cap N_2 = \emptyset$, it follows that $00 \notin \mathcal{L}(B_k)$, $11 \notin \mathcal{L}(B_k)$, for each B_k . Therefore, B_k cannot have the form of $(X)^*$. Moreover, B_k cannot have the form of $(X)?$ either; otherwise we could remove the “?” operator and the resulting new expression would still be consistent, because $\{0, 1\} \subseteq \mathcal{L}(X) \subseteq \mathcal{L}((X)?)$.

Therefore, it must be true that $B_k = (C_{k1}) \cdot (C_{k2})$. Note that $\mathcal{L}(C_{k1})$ and $\mathcal{L}(C_{k2})$ must contain only one of 0 and 1 but not both; otherwise the “.” operator together with one of C_{k1} and C_{k2} could be removed. Since $\varepsilon \in \mathcal{L}(B_k)$, it follows that $\varepsilon \in \mathcal{L}(C_{k1})$, $\varepsilon \in \mathcal{L}(C_{k2})$.

Let us suppose $0 \in \mathcal{L}(C_{kj})$ ($j = 1$ or $j = 2$). Since $00 \notin \mathcal{L}(B_k)$, it follows that $00 \notin \mathcal{L}(C_{kj})$. So C_{kj} cannot be a single symbol, or have the form of $(X)^*$ or $(X) \cdot (Y)$. Therefore C_{kj} must have the form of $(D)?$. By a similar argument, we can show that D must be the single symbol 0. So C_{kj} must be $(0)?$. Similarly, we can show that if $1 \in \mathcal{L}(C_{kj})$ ($j = 1$ or $j = 2$), then C_{kj} must be $(1)?$. Therefore, B_k must be either $(0?)(1?)$ or $(1?)(0?)$.

We have shown that we can obtain a consistent UFRE, $E = \$B_1\$B_2 \dots \$B_n$, where each B_k is either $(0?)(1?)$ or $(1?)(0?)$. We define an assignment of truth values to the variables, V_1, V_2, \dots, V_n , as follows: if $B_k = (0?)(1?)$, then assign *true* to V_k ; if $B_k = (1?)(0?)$, then assign *false* to V_k . Because $\mathcal{L}(E) \cap N_3 = \emptyset$, we can verify that this truth value assignment must satisfy the formula F .

Clearly, $|POS| + |NEG| = O(mn)$. Therefore this problem is NP-hard.