Computational Aspects of Resilient Data Extraction from Semistructured Sources (Extended Abstract)

Hasan Davulcu Guizhen Yang* Michael Kifer I.V. Ramakrishnan
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794, U.S.A.
{davulcu,guizyang,kifer,ram}@cs.sunysb.edu

February 3, 2002

Abstract

Automatic data extraction from semistructured sources such as HTML pages is rapidly growing into a problem of significant importance, spurred by the growing popularity of the so called "shopbots" that enable end users to compare prices of goods and other services at various web sites without having to manually browse and fill out forms at each one of these sites.

The main problem one has to contend with when designing data extraction techniques is that the contents of a web page changes frequently, either because its data is generated dynamically, in response to filling out a form, or because of changes to its presentation format. This makes the problem of data extraction particularly challenging, since a desirable requirement of any data extraction technique is that it be "resilient", i.e., using it we should always be able to locate the object of interest in a page (such as a form or an element in a table generated by a form fill-out) in spite of changes to the page's content and layout.

In this paper we propose a formal computation model for developing resilient data extraction techniques from semistructured sources. Specifically we formalize the problem of data extraction as one of generating unambiguous extraction expressions, which are regular expressions with some additional structure. The problem of resilience is then formalized as one of generating a maximal extraction expression of this kind. We present characterization theorems for maximal extraction expressions, complexity results for testing them, and algorithms for synthesizing them.

*Contact author: (Phone)+1-516-632-7943, (Fax)+1-516-632-8334, (Email)guizyang@cs.sunysb.edu

1 Introduction

The World Wide Web is becoming the dominant medium for information delivery and electronic commerce. The number of users who routinely use the web to buy goods and services continues to increase at a rapid pace. In response, software robots (called "shopbots") that allow consumers to quickly find out the best prices for comparable goods and services are beginning to emerge. Information about prices and other attributes of products are typically obtained by filling out forms at a vendor's site. Software robots retrieve such information by automatically navigating to relevant sites, locating the correct forms, filling them out and extracting the data of interest from web pages returned as the result. (Junglee and Jango [17] are two examples of such shopbots.) Currently almost all web information is stored as semi-structured data, mostly as HTML pages, and shopbots need to retrieve data from such sources. Hence automatic data extraction from semistructured data sources is a problem of significant importance especially in the context of web-based electronic commerce.

This problem has attracted a lot of research attention recently. The techniques proposed so far, are by and large centered around creating a *wrapper* [3, 6, 9, 12, 13, 15, 18, 14, 7, 21, 22], that parses an HTML source and maps it into a set of structured or semi-structured database objects that can be readily queried and manipulated by applications.

The central difficulty in designing data extraction techniques is the volatile nature of HTML pages, in the sense that they change very frequently. Changes occur either because they have to accommodate new services and content offerings or because they are *dynamically* generated in response to form-based queries. Such variations can give rise to "brittleness" in data extractors, i.e. they may no longer be able to locate the objects of interest in the page (such as a form or a table element). Thus developing data extraction techniques that are resilient to changes in the data source structure is both desirable and important. To the best of our knowledge this problem has not yet been fully explored.

In this paper, we make the first step towards developing a formal framework for creating resilient data extraction wrappers for semistructured sources. As an example, Web pages can be represented as sequences of tokens (HTML tags and strings), and the extraction problem is usually reduced to the problem of parsing using regular grammars [9, 12, 13, 21, 18, 22, 14, 15], context-free grammars [6, 3] or specialized languages [7].

We propose the notion of *extraction expressions*, which are tag-marked regular expressions, as a formalization of the informal concept of the "target object that can be identified by its local or global context."

The initial extraction expression is usually derived from a set of examples (i.e., of HTML pages and target objects) using one of the known heuristic approaches [13, 21, 22, 9], or one of the learning algorithms [18, 4, 5], or even manually [14, 15, 12].

We first introduce the notion of "unambiguity" as a consistency requirement for extraction expressions. Unambiguous extraction expressions must identify the target objects uniquely within any page. We show that ambiguity of any given extraction expressions is decidable in polynomial time.

If an extraction expression can correctly identify the target object in many variations of the same document, we shall call such an expression *resilient* to changes. Unambiguous extraction expressions can be ordered according to the languages which they parse. The larger the language — the more resilient the expression. We thus reduce the problem of resilience against variations to the task of synthesizing *maximal* unambiguous extraction expressions.

We show that some unambiguous expressions can be maximized in several different — even infinite

number (!) of — ways and the problem of deciding maximality is PSPACE-complete. We do not know whether every unambiguous extraction expression can be maximized (although we conjecture that this is possible). However, we propose algorithms for maximizing a large, non-trivial class of extraction expressions that arise frequently in practical situations.

Our preliminary experiments with the maximization algorithms proposed here indicate that they are sufficient to provide resilient extraction capabilities for a web-based information harvesting system that we have developed recently.

The remainder of the paper is organized as follows. Section 2 surveys the related work. In Section 3, we motivate the problem of resilient extraction and our approach using a concrete and realistic example. Section 4 introduces the main definitions, including the notions of ambiguity and resilience. In Section 5, we resolve the complexity of various decision problems related to resilient extraction. In Section 6, we present our maximization algorithms and prove their correctness. Section 7 illustrates how the techniques developed in this paper apply to the example of Section 3. Section 8 concludes our paper.

2 Related Work

Semi-structured data [10, 1] has recently emerged as an important area of study. Querying semi-structured data through a query language [11, 2] is one way to extract data. This technique is applicable when the structure of the data and the location of the desired object is known to a large degree and is not subject to drastic changes.

Wrapper based extraction techniques from semi-structured data include [3, 6, 9, 13, 15, 18, 22, 14, 7, 21]. The works [14, 15] provide a powerful framework for manual extraction of complex data objects based on the application of sequences of patterns. An expressive specialized language for data extraction is proposed in [7]. The language is based on searches using a subset of regular expressions and other page restructuring instructions. The works in [3, 9, 13, 20, 22] propose various graphical tools and heuristics for semi-automatic derivation of regular extraction expressions. Such techniques could be used at the initial learning stage of our framework (see Sections 3 and 7). However, none of these works addresses the problems inherent in resilient data extraction.

The works [18, 21] describe fully automated wrapper generation techniques that use machine learning induction algorithms (similar to [8, 4, 5]). Such techniques are useful in our framework since they could supply us with initial extraction expressions that could then be generalized using our techniques. A limitation of these approaches, however, is that the extracted data must be representable as a set of tuples.

An ontology-based approach to extracting data is presented in [12]. This technique requires that suitable ontologies of context keywords, relationships, and regular expressions for lexicons must be constructed in advance (and manually).

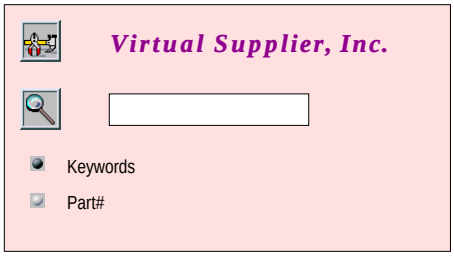
In [6], a heuristic algorithm for automatic generation of context-free grammar for data extraction purposes is presented. However this work does not address the issue robustness of the wrappers in the presence of variations in source documents.

Overall, with the exception of [6], all data extraction techniques we are aware of rely on regular expressions. Hence, our results enhance these techniques by providing both the objective criteria for sorting out the good expressions from the bad ones, and techniques for making the good extraction expressions more robust.

3 Motivating Example

One of the main problems with data extraction from Web documents is that the structure of the data might change due to the dynamic nature of the document or because the document was redesigned manually and, thus, the location of the object of interest might be hard to pinpoint. The most typical changes are insertion or deletion of HTML elements before or after the object of interest and embedding of the object inside some other HTML element.

```
<P>
<H1><IMG SRC="supplier.gif">Virtual Supplier, Inc.</H1>
<P>
<FORM METHOD="POST" ACTION="search.cgi">
<INPUT TYPE="image" ALIGN=left SRC="search.gif">
<INPUT TYPE="text" SIZE="15" NAME="VALUE"> <BR>
<P>
<INPUT TYPE="radio" NAME="ATTR" VALUE="1" checked>
Keywords<BR>
<INPUT TYPE="radio" NAME="ATTR" VALUE="2">
Manufacturer Part#
</FORM>
```




```
<TABLE>
<TR> <TD><IMG SRC="supplier.gif"></TD>
     <TD><H1>Virtual Supplier, Inc.</H1></TD>
</TR>
<TR> <TD><IMG SRC="cust.gif"></TD>
     <TD><A HREF="cust.html">Customer Service</A><BR></TD>
</TR>
<FORM METHOD="POST" ACTION="search.cgi">
<TR> <TD><INPUT TYPE="image" SRC="search.gif"></TD>
     <TD><INPUT TYPE="text" SIZE="15" NAME="VALUE"></TD>
     <TD> <INPUT TYPE="radio" NAME="ATTR" VALUE="1" checked>
             Keywords<BR>
          <INPUT TYPE="radio" NAME="ATTR" VALUE="2">
             Manufacturer Part# </TD>
</TR>
</FORM>
</TABLE>
```

Figure 1: Original and Rearranged Page

To illustrate, consider the two documents in Figure 1, which represent the main elements of a typical catalog shopping site. The user can fill out a form to search for a product by keywords or part number and out come the results. The top document consists of some header information and a form. The bottom document is similar, but the form is now embedded in a table and a new table row, a link to customer service, is added. If we are building a web robot that goes around and compares prices, the most likely object of interest would be the form. Suppose now that we train our robot to find the needed form using the top document. We want to make sure that the robot does not fail even if the site administrator reorganizes the document as shown at the bottom of Figure 1, if more rows are added to the second document before or after the form, and (hopefully) if other forms or tables are added to that document.

To begin, we can try to represent our documents as strings of objects. For instance, the top

document can be represented as

```
P H1 /H1 P FORM /FORM
```

In this representation, we assume that the contents of the objects H1 and FORM is of no interest. If the insides of, say, the form are of interest (usually they are), we could expand the representation as follows:

```
P H1 /H1 P FORM INPUT INPUT P INPUT INPUT /FORM
```

Likewise, the second document in the figure can be represented as:

```
TABLE TR TD /TD TD /TD /TR TR TD /TD TD /TD /TR FORM TR TD INPUT /TD TD INPUT /TD TD INPUT /TD /TR /FORM /TABLE
```

The exact details of this representation is of no significance here. It is easy to enrich this model to take the tag attributes into account, and so on. The point is that documents can be represented as sequences of tags plus some additional symbols (*e.g.*, to incorporate tag attributes).

Each of the above strings is an abstract representation of a semi-structured Web document, but alone they do not yet specify the objects of interest. To tell the robot which object we want, we can enclose it in angle brackets. To indicate that we are interested in the second INPUT-element of the form, we can write the following:

```
P H1 /H1 P FORM INPUT <INPUT> P INPUT INPUT /FORM
TABLE TR TD /TD TD /TD /TR TR TD /TD TD /TD /TR FORM TR TD INPUT /TD TD <INPUT> /TD TD INPUT /TD /TR /FORM /TABLE
```

Since the above expressions are obviously different and each matches only one of our candidate documents, they are still not very useful to our robot. However, we can try to generalize them, say, using the syntax of regular expressions:

$(\mathcal{T}$ags- FORM)* FORM ($\mathcal{T}$ags- INPUT)* INPUT ($\mathcal{T}$ags- INPUT)* <INPUT> $\mathcal{T}$ags*

In the above expression, $\mathcal{T}$ags is a regular expression that matches any HTML tag, ($\mathcal{T}$ags − FORM) matches any tag except FORM), and "*" says that the corresponding expression can be repeated zero or more times.

Since this regular expression matches both documents and identifies the object of interest correctly, we could give it to our robot and it will extract the information properly even if the document changes as shown in Figure 1.

Regular expressions with a marked occurrence of a symbol will be called *extraction expressions* in this paper. If an extraction expression can correctly identify the desired object in many variations of the same document, we shall say that such an expression is *resilient* to changes.

The above discussion suggests the following strategy for finding resilient extraction expressions. In the first stage, a small number of sample variants of the desired document can be obtained by filling out the same form in different ways. If the document is static but might change as a result of a redesign, heuristics can be used to automatically create several perturbations of the document. Next, the variations of the document in question are represented as strings and the object of interest is marked in each document (this object must be of the same kind in each case, since we assumed that they are perturbations of the same document), which leaves us with a small number of very rigid extraction expressions. Finally, these expressions are generalized into a single extraction expression that matches all the instances of our document.

Once the problem of data extraction is reduced to the problem of generalizing marked regular expressions, many interesting questions arise. The first of these is: how far should (and can) we generalize before the robot gets confused? Indeed, the original extraction expressions were not resilient because they matched only one document. However, when they *did* match, the object of interest was identified correctly and uniquely. Can it happen that a generalized extraction expression might match two or more objects and thus confuse our robot (because it will not know which is the true object of interest)? We do not need to think hard to find such a generalization: $\mathcal{T}$`ags* <INPUT>` $\mathcal{T}$`ags*`.

This leads to the question of *ambiguity* of extraction expressions. For instance, the expression `((qp)*|(`$\mathcal{T}$`ags − p)*)<p>p*`, where "`|`" denotes an alternative (union of regular languages) and ($\mathcal{T}$ags − `p`) matches anything but `p`, is *unambiguous* in the sense that whenever it matches a string, the marked occurrence of the symbol `p` falls into a unique place on the string (even though the prefix `((qp)*|(`$\mathcal{T}$`ags` $-$ p`)*)` can match the prefix of a string in more than one way). In contrast, `(qp)*p*<p>p*` is an *ambiguous* expression, because there are many ways to match the marked occurrence of `p` on *some* strings. For instance, consider `qppp`. The prefix `(qp)*p*` can match `qp` and `qpp`, so the marked occurrence of `p` can match the second *or* the third occurrence of `p`.

We believe that (un)ambiguity is an important property of extraction expressions that places limits on how much we can generalize from a set of examples. We shall see that ambiguity can be decided in polynomial time.

While unambiguity is important, we should not loose sight of the fact that our goal is to find generalizations that are as resilient as possible. It turns out that extraction expressions of the form $\mathtt{E}_1$`<p>`$\mathtt{E}_2$ can be ordered according to how vast the regular languages $L(\mathtt{E}_1)$ and $L(\mathtt{E}_2)$ are. The bigger they are — the more resilient the extraction expression is. Ideally, we would like to find generalizations that are *maximal* with respect to the above order among the unambiguous expressions that match the object of interest on our sample pages. We show that maximality of any given unambiguous regular expressions is decidable, albeit it is a PSPACE-complete problem. Surprisingly, the question of whether every unambiguous extraction expression has a maximal generalization seems to be a hard problem, and it remains open at this point. However, we shall see that maximal generalizations do exist for large classes of extraction expressions.

4 Extraction Expressions, Unambiguity, and Maximality

The previous section provided the intuition behind the notions of extraction expressions, ambiguity, and resilience. We are now going to define these concepts formally and prove several of their important properties.

Definition 4.1 **(Extraction Expression)** Given a finite alphabet Σ, $p \in \Sigma$, and regular expressions E_1 and E_2 over Σ, $E_1\langle p\rangle E_2$ is called an *extraction expression* over Σ.

In other words, an extraction expression is simply an ordinary regular expression of a special form $E_1\langle p\rangle E_2$, with one *marked* occurrence, $\langle p\rangle$, of an alphabet symbol. □

The *language parsed by* $E_1\langle p\rangle E_2$ is defined as $L(E_1\langle p\rangle E_2) \stackrel{\text{def}}{=} \{\rho | \rho \in L(E_1 \cdot p \cdot E_2)\}$. Given a string $\rho \in \Sigma^*$, we say that $E_1\langle p\rangle E_2$ *parses* ρ if $\rho \in L(E_1\langle p\rangle E_2)$. We also say that $E_1\langle p\rangle E_2$ *extracts* p from ρ if there exist $\alpha, \beta \in \Sigma^*$, such that $\rho = \alpha \cdot p \cdot \beta$ and $\alpha \in L(E_1), \beta \in L(E_2)$.

Suppose ρ is a string. We can use $E_1\langle p\rangle E_2$ to extract information from ρ as follows: First we can try to split ρ into a prefix α, followed by p, followed by a suffix β. If α is recognized by E_1 and β is

recognized by E_2, then p is the extracted object. We try such splits until we either succeed on some split or fail on all candidates.

As illustrated in Section 3, the expression $E_1\langle p\rangle E_2$ might be *ambiguous*, *i.e.*, there might be a string, ρ, where two different splits can succeed. For instance, $p^*\langle p\rangle q$ parses $pppq$, but any one of three p's in $pppq$ can be returned as the extracted object.

Definition 4.2 **(Unambiguous Extraction Expression)** Extraction expression $E_1\langle p\rangle E_2$ is *unambiguous* iff for all $\alpha, \alpha' \in L(E_1)$ and $\beta, \beta' \in L(E_2)$, if $\alpha \cdot p \cdot \beta = \alpha' \cdot p \cdot \beta'$ then $\alpha = \alpha'$ and $\beta = \beta'$. □

Example 4.3 **(Ambiguous and Unambiguous Extraction Expressions)**

$(pq)^*\langle p\rangle\Sigma^*$ and $(p|pp)\langle p\rangle(p|pp)$ are ambiguous extraction expressions, whereas $(qp)^*\langle p\rangle\Sigma^*$ and $(p|pp)\langle p\rangle(p|ppp)$ are unambiguous extraction expressions. For instance, $pqpq$ can be parsed by $(pq)^*\langle p\rangle\Sigma^*$ as $\epsilon \cdot p \cdot qpq$ and as $pq \cdot p \cdot q$. Likewise, $(p|pp)\langle p\rangle(p|pp)$ can parse $pppp$ in two different ways. On the other hand, it can be proved that the last two expressions always parse their matching strings uniquely. □

Since, as explained in Section 3, we are mostly interested in unambiguous extraction expressions, from now on the term "extraction expression" will refer to unambiguous expressions, unless explicitly specified otherwise.

Definition 4.4 **(Partial Order among Extraction Expressions)** $F_1\langle p\rangle F_2 \preceq E_1\langle p\rangle E_2$ iff $L(F_1) \subseteq L(E_1)$ and $L(F_2) \subseteq L(E_2)$. We shall also say that $E_1\langle p\rangle E_2$ *generalizes* $F_1\langle p\rangle F_2$. □

Informally, extraction expressions that are "larger" with respect to $\preceq$ are more resilient because they can uniquely parse larger sets of strings. Moreover if $F_1\langle p\rangle F_2 \preceq E_1\langle p\rangle E_2$ then the two expressions parse the strings in $L(F_1\langle p\rangle F_2)$ the same way. It is easy to see that $\preceq$ is reflexive, antisymmetric and transitive. Therefore, $\preceq$ is a partial order on the set of all unambiguous extraction expressions.

Note that $F_1\langle p\rangle F_2 \preceq E_1\langle p\rangle E_2$ implies $L(F_1\langle p\rangle F_2) \subseteq L(E_1\langle p\rangle E_2)$, but not the other way around! Indeed, the two expressions $p\langle p\rangle ppp$ and $pp\langle p\rangle pp$ *parse* exactly the same language, but they extract different objects from that language: $p\langle p\rangle ppp$ extracts the second occurrence of p, while $pp\langle p\rangle pp$ extracts the third.

Definition 4.5 **(Maximal Extraction Expression)** An unambiguous extraction expression $\mathcal{E}$ over a finite alphabet Σ is maximal iff for any unambiguous extraction expression $\mathcal{E}'$ over Σ, if $\mathcal{E} \preceq \mathcal{E}'$ then $L(\mathcal{E}) = L(\mathcal{E}')$. □

For the following examples, we use the expression $E_1 - E_2$ to represent the regular expression that recognizes the regular set $L(E_1) - L(E_2)$.

Example 4.6 **(Maximal Extraction Expressions)** Although it might not be immediately obvious, both $(\Sigma - p)^*\langle p\rangle\Sigma^*$ and $(qp)^* \cdot ((\Sigma - p)^* - q)\langle p\rangle\Sigma^*$ are maximal extraction expressions. This follows from Proposition 5.7 in the next section. □

Example 4.7 **(Unambiguity and Maximality)** Given a non-maximal unambiguous extraction expression $\mathcal{E}$ over Σ, if there exists a maximal extraction expression $\mathcal{E}'$ over Σ such that $\mathcal{E} \preceq \mathcal{E}'$, we say that extraction expression $\mathcal{E}$ can *be maximized to* $\mathcal{E}'$.

It is not known whether every unambiguous extraction expression $\mathcal{E}$ can be maximized. However, even when maximization is known to exist then it might not be unique. For example, $qp\langle p\rangle\Sigma^*$ can be maximized to $(\Sigma - p)^* \cdot p \cdot (\Sigma - p)^*\langle p\rangle\Sigma^*$ and $((qp(\Sigma - p)^*)|((\Sigma - p)^* - q))\langle p\rangle\Sigma^*$. (The latter expression is obtained using Algorithm 6.2 in Section 6, when $qp\langle p\rangle\Sigma^*$ is used as input.)

In fact, it can be shown that the above expression has an *infinite* number of maximal expressions that generalize it. □

5 Complexity of the Ambiguity and Maximality Problems

Definition 5.1 **(Prefix and Suffix Factoring)** Given a pair of regular expressions E_1 and E_2 over a finite alphabet Σ, the *prefix factorization* of E_1 by E_2 is defined as ${}_{E_2}\backslash E_1 \stackrel{\text{def}}{=} \{\alpha | \exists \beta \in L(E_2), \beta \cdot \alpha \in L(E_1)\}$. The *suffix factorization* of E_1 by E_2 is $E_1/_{E_2} \stackrel{\text{def}}{=} \{\alpha | \exists \beta \in L(E_2), \alpha \cdot \beta \in L(E_1)\}$. □

It is known that ${}_{E_2}\backslash E_1$ and $E_1/_{E_2}$ are regular languages, if both E_1 and E_2 are regular expressions [19]. Thus factors can be represented as regular expressions. Since every regular language corresponds to a regular expression and vice versa, we shall use E and $L(E)$ interchangeably to denote the regular language recognized by E.

Lemma 5.2 *Given regular expressions E_1 and E_2 over a finite alphabet Σ, the regular expressions corresponding to ${}_{E_2}\backslash E_1$ or $E_1/_{E_2}$ can be computed in polynomial time in the size of E_1 and E_2.*

Proof: To be finished. □

Lemma 5.3 *$E_1\langle p\rangle E_2$ is ambiguous iff there exist $\alpha, \beta, \gamma \in \Sigma^*$ such that $\alpha \cdot p \cdot \gamma \cdot p \cdot \beta \in L(E_1\langle p\rangle E_2)$, α, $\alpha \cdot p \cdot \gamma \in L(E_1)$ and β, $\gamma \cdot p \cdot \beta \in L(E_2)$.*

Proposition 5.4 (Necessary and Sufficient Condition for Unambiguity) *An extraction expression $E_1\langle p\rangle E_2$ over a finite alphabet Σ is unambiguous iff ${}_{(E_1 \cdot p)}\backslash E_1 \cap E_2/_{(p \cdot E_2)} = \phi$.*

Proof: By contradiction. Suppose $E_1\langle p\rangle E_2$ is unambiguous but ${}_{(E_1 \cdot p)}\backslash E_1 \cap E_2/_{(p \cdot E_2)} \neq \phi$. Then there must exist $\gamma \in {}_{(E_1 \cdot p)}\backslash E_1 \cap E_2/_{(p \cdot E_2)}$, $\alpha \in L(E_1)$, and $\beta \in L(E_2)$ such that $\alpha \cdot p \cdot \gamma \in L(E_1)$, and $\gamma \cdot p \cdot \beta \in L(E_2)$. But $\alpha \cdot p \cdot \gamma \cdot p \cdot \beta \in L(E_1\langle p\rangle E_2)$ and thus by Lemma 5.3 contradicts the fact that $E_1\langle p\rangle E_2$ is unambiguous. On the other hand, if ${}_{(E_1 \cdot p)}\backslash E_1 \cap E_2/_{(p \cdot E_2)} = \phi$ but $E_1\langle p\rangle E_2$ is ambiguous then, again, by Lemma 5.3 there must exist $\gamma \in \Sigma^*$ such that $\gamma \in {}_{(E_1 \cdot p)}\backslash E_1 \cap E_2/_{(p \cdot E_2)}$, which also leads to a contradiction. □

Proposition 5.5 (Necessary and Sufficient Condition for Unambiguity) *Let $E_1\langle p\rangle E_2$ be an extraction expression over a finite alphabet Σ. It is unambiguous iff $(E_1 \cdot c \cdot E_2) \cap (E_1 \cdot p \cdot (E_2)_p^{(p|c)}) = \phi$, where $c \notin \Sigma$ and $(E_2)_p^{(p|c)}$ is a regular expression obtained from E_2 by simultaneously replacing every occurence of p in E_2 with $(p|c)$.*

Proof: By contradiction. If $E_1\langle p\rangle E_2$ is unambiguous but $(E_1 \cdot c \cdot E_2) \cap (E_1 \cdot p \cdot (E_2)_p^{(p|c)}) \neq \phi$, then there exist α, $\alpha' \in L(E_1)$, $\beta \in L(E_2)$, $\beta' \in L((E_2)_p^{(p|c)})$ such that $\alpha \cdot c \cdot \beta = \alpha' \cdot p \cdot \beta'$. Since E_1 is a regular expression over Σ and $c \notin \Sigma$, there must exist $\gamma \in \Sigma^*$ such that $\alpha = \alpha' \cdot p \cdot \gamma$. Thus $\alpha' \cdot p \cdot \gamma \in L(E_1)$

and $\beta' = \gamma \cdot c \cdot \beta$. Because $\beta' \in L((E_2)_p^{(p|c)})$, we know that $\gamma \cdot p \cdot \beta \in L(E_2)$, which contradicts Lemma 5.3. On the other hand, if $(E_1 \cdot c \cdot E_2) \cap (E_1 \cdot p \cdot (E_2)_p^{(p|c)}) = \phi$ but $E_1\langle p\rangle E_2$ is ambiguous, then by Lemma 5.3 there exist α, β, $\gamma \in \Sigma^*$ such that α, $\alpha \cdot p \cdot \gamma \in L(E_1)$ and β, $\gamma \cdot p \cdot \beta \in L(E_2)$. Thus $\alpha \cdot p \cdot \gamma \cdot c \cdot \beta \in L(E_1 \cdot c \cdot E_2)$. Since $\gamma \cdot p \cdot \beta \in L(E_2)$ we know that $\gamma \cdot c \cdot \beta \in L((E_2)_p^{(p|c)})$, which means that $\alpha \cdot p \cdot \gamma \cdot c \cdot \beta \in L(E_1 \cdot p \cdot (E_2)_p^{(p|c)})$, also a contradiction. □

Theorem 5.6 (Complexity of Testing Ambiguity) *Let $E_1\langle p\rangle E_2$ be an extraction expression over a finite alphabet Σ. Then testing whether $E_1\langle p\rangle E_2$ is ambiguous can be done in time quadratic in the size of $E_1\langle p\rangle E_2$.*

Proof: To be finished. □

Proposition 5.7 (Necessary and Sufficient Condition for Maximality) *An unambiguous extraction expression $E_1\langle p\rangle E_2$ over a finite alphabet Σ is maximal iff $\overline{E_1} \subseteq (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$ and $\overline{E_2} \subseteq {}_{(E_1 \cdot p)}\backslash(E_1 \cdot p \cdot E_2)$.*

Proof: First we prove that if $E_1\langle p\rangle E_2$ is maximal and unambiguous then $\overline{E_1} \subseteq (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$.

Suppose, to the contrary, that $\overline{E_1} \not\subseteq (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$ and so there is ρ such that $\rho \in \overline{E_1} - (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$. Because $\rho \notin (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$, it follows that $\rho \cdot p \cdot \beta \notin L(E_1 \cdot p \cdot E_2)$, for every $\beta \in L(E_2)$. Therefore, $E_1\langle p\rangle E_2 \preceq (\rho|E_1)\langle p\rangle E_2$ but $L(E_1\langle p\rangle E_2) \neq L((\rho|E_1)\langle p\rangle E_2)$. Furthermore, $(\rho|E_1)\langle p\rangle E_2$ is unambiguous, because if not, then by Lemma 5.3 there must exist α, β, $\gamma \in \Sigma^*$ such that α, $\alpha \cdot p \cdot \gamma \in L(\rho|E_1)$ and β, $\gamma \cdot p \cdot \beta \in L(E_2)$. Then because $E_1\langle p\rangle E_2$ is unambiguous it follows that either $\rho = \alpha$ or $\rho = \alpha \cdot p \cdot \gamma$. In either case, we can derive $\rho \in (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$, a contradiction. Thus we have just shown that $(\rho|E_1)\langle p\rangle E_2$ is unambiguous and $E_1\langle p\rangle E_2 \preceq (\rho|E_1)\langle p\rangle E_2$ but $L(E_1\langle p\rangle E_2) \neq L((\rho|E_1)\langle p\rangle E_2)$, which contradicts the fact $E_1\langle p\rangle E_2$ is maximal. Therefore, if $E_1\langle p\rangle E_2$ is maximal then $\overline{E_1} \subseteq (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$. Similarly we can prove that if $E_1\langle p\rangle E_2$ is maximal then $\overline{E_2} \subseteq {}_{(E_1 \cdot p)}\backslash(E_1 \cdot p \cdot E_2)$.

Next we show that if $\overline{E_1} \subseteq (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$, $\overline{E_2} \subseteq {}_{(E_1 \cdot p)}\backslash(E_1 \cdot p \cdot E_2)$ and $E_1\langle p\rangle E_2$ is unambiguous, then $E_1\langle p\rangle E_2$ is maximal. Let $E_1'\langle p\rangle E_2'$ be an arbitrary unambiguous extraction expression such that $E_1\langle p\rangle E_2 \preceq E_1'\langle p\rangle E_2'$. Clearly $L(E_1) \subseteq L(E_1')$. Next we shall prove that $L(E_1') \subseteq L(E_1)$.

Suppose, to the contrary, that $L(E_1') \not\subseteq L(E_1)$, then there must exist $\rho \in L(E_1') - L(E_1)$. Clearly $\rho \in L(\overline{E_1})$, it follows that $\rho \in (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$. Therefore there exist $\alpha \in L(E_1)$ and β, $\beta' \in L(E_2)$, such that $\alpha \cdot p \cdot \beta = \rho \cdot p \cdot \beta'$. But α, $\rho \in L(E_1')$, $\alpha \neq \rho$, which means that $E_1'\langle p\rangle E_2'$ is ambiguous, a contradiction. Therefore, $L(E_1') \subseteq L(E_1)$ must hold. Similarly, we can prove $L(E_2') \subseteq L(E_2)$. Thus $L(E_1') = L(E_1)$ and $L(E_2') = L(E_2)$, *i.e.*, $E_1\langle p\rangle E_2$ is maximal. □

Corollary 5.8 (Necessary and Sufficient Condition for Maximality) *An unambiguous extraction expression $E_1\langle p\rangle E_2$ over a finite alphabet Σ is maximal iff $(E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)} = \Sigma^*$ and ${}_{(E_1 \cdot p)}\backslash(E_1 \cdot p \cdot E_2) = \Sigma^*$.*

Proof: Directly follows from Proposition 5.7 and the obvious fact that $E_1 \subseteq (E_1 \cdot p \cdot E_2)/_{(p \cdot E_2)}$ and $E_2 \subseteq {}_{(E_1 \cdot p)}\backslash(E_1 \cdot p \cdot E_2)$. □

Lemma 5.9 *Given a regular expression E over a finite alphabet Σ, the problem of testing whether $L(E) = \Sigma^*$ is PSPACE-complete.*

Proof: See [16] $\square$

Lemma 5.10 *For any regular expression E over a finite alphabet Σ, the extraction expression $(\Sigma - p)^*\langle p\rangle E$ is unambiguous.*

Proof: Because ${}_{(\Sigma-p)^*\cdot p}\backslash(\Sigma - p)^* = \phi$, we know that ${}_{(\Sigma-p)^*\cdot p}\backslash(\Sigma - p)^* \cap E/_{(p\cdot E)} = \phi$, for any E. Therefore, $(\Sigma - p)^*\langle p\rangle E$ is unambiguous by Proposition 5.4. $\square$

Proposition 5.11 *For any regular expression E over a finite alphabet Σ, the extraction expression $(\Sigma - p)^*\langle p\rangle E$ is maximal iff $L(E) = \Sigma^*$.*

Proof: First $(\Sigma - p)^*\langle p\rangle\Sigma^*$ is unambiguous by Lemma 5.10. Because ${}_{((\Sigma-p)^*\cdot p)}\backslash((\Sigma - p)^* \cdot p \cdot \Sigma^*) = \Sigma^*$ and $((\Sigma - p)^* \cdot p \cdot \Sigma^*)/_{(p\cdot\Sigma^*)} = (\Sigma - p)^* \cup ((\Sigma - p)^* \cdot p \cdot \Sigma^*) = \Sigma^*$, it follows from Corollary 5.8 that $(\Sigma - p)^*\langle p\rangle\Sigma^*$ is maximal. Thus, again by Lemma 5.10, we conclude that $(\Sigma - p)^*\langle p\rangle E$ is maximal iff $L(E) = \Sigma^*$. $\square$

Theorem 5.12 (Complexity of Testing Maximality) *For any extraction expression $E_1\langle p\rangle E_2$ over a finite alphabet Σ, the problem of testing whether $E_1\langle p\rangle E_2$ is maximal is PSPACE-complete.*

Proof: From Lemma 5.9 and Proposition 5.11, we know that the problem is PSPACE-hard. Testing whether $E_1\langle p\rangle E_2$ is unambiguous only takes polynomial time according to Theorem 5.6. Then by Corollary 5.8, to test if $E_1\langle p\rangle E_2$ is maximal it suffices to test whether $(E_1 \cdot p \cdot E_2)/_{(p\cdot E_2)} = \Sigma^*$ and ${}_{(E_1\cdot p)}\backslash(E_1 \cdot p \cdot E_2) = \Sigma^*$. According to Lemma 5.2 both $(E_1 \cdot p \cdot E_2)/_{(p\cdot E_2)}$ and ${}_{(E_1\cdot p)}\backslash(E_1 \cdot p \cdot E_2)$ can be computed in polynomial time. After applying Lemma 5.9 again, we conclude that the problem of testing maximality is in PSPACE. $\square$

6 Synthesizing Maximal Extraction Expressions

In this section we first propose an algorithm, called *left-filtering maximization*, that can maximize a large class of unambiguous extraction expressions. Then we develop a *pivot maximization framework*, which can be used to enhance maximization algorithms. In particular, when applied to the left-filtering maximization algorithm, it yields a much more powerful maximization technique.

Left-Filtering Maximization. Consider an extraction expression $E_1\langle p\rangle E_2$ over a finite alphabet Σ. If ${}_{(E_1\cdot p)}\backslash E_1 = \phi$, then by Proposition 5.4 we can replace E_2 with Σ^* and obtain a more general unambiguous extraction expression: $E_1\langle p\rangle E_2 \preceq E_1\langle p\rangle\Sigma^*$. (Similarly, if $E_2/_{(p\cdot E_2)} = \phi$ then we can generalize $E_1\langle p\rangle E_2$ to $\Sigma^*\langle p\rangle E_2$.) The problem is that $E_1\langle p\rangle\Sigma^*$ might not be maximal, so we must do some work on E_1 to make our expression maximal. The algorithm, below, is one way to maximize such an extraction expression.

Definition 6.1 **(Finite Sequence Filtering Operator)** Given a regular expression E over a finite alphabet Σ, a symbol $p \in \Sigma$, and an integer $n \geq 0$, the finite sequence filtering operator $E\|_n^p$ is defined as follows:

$$E\|_n^p \stackrel{\text{def}}{=} E \ \cap \ (\Sigma - p)^* \cdot \underbrace{p \cdot (\Sigma - p)^* \ \cdots \ p \cdot (\Sigma - p)^*}_{n}$$

where the suffix $p \cdot (\Sigma - p)^*$ repeats n times. $\square$

Informally, $E\|_n^p$ consists of exactly those strings recognized by E that contain precisely n occurrences of the symbol p. Since the intersection of two regular expressions can be computed in polynomial time, $E\|_n^p$ can be computed in polynomial time.

Algorithm 6.2 (Left-Filtering Maximization)

```
Input:   an unambiguous extraction expression E⟨p⟩Σ*, where Σ is a finite alphabet and
         E/(p·Σ*)||ⁿᵖ = φ, for some n ≥ 0.
Output:  a maximal, unambiguous extraction expression E'⟨p⟩Σ* that generalizes E⟨p⟩Σ*.

BEGIN
1       F := E/(p·Σ*);
2       S := (Σ − p)* − (F||⁰ᵖ);
3       n := 0;
4       while F||ⁿᵖ ≠ φ {
5               S := S + (F||ⁿᵖ · p · (Σ − p)* − F||ⁿ⁺¹ᵖ);
6               n := n + 1;
7       }
8       E' := E + S;
9       return E'⟨p⟩Σ*
END
```

In proving the correctness of the above algorithm, we use the following simple facts.

Lemma 6.3 *For any regular expressions E, E_1 and E_2 over a finite alphabet Σ and $p \in \Sigma$, the following statements are true:*

1. $(E_1 + E_2)/_E = E_1/_E + E_2/_E$
2. ${}_E\backslash(E_1 + E_2) = {}_E\backslash E_1 + {}_E\backslash E_2$
3. $E/_{(E_1+E_2)} = E/_{E_1} + E/_{E_2}$
4. ${}_{(E_1+E_2)}\backslash E = {}_{E_1}\backslash E + {}_{E_2}\backslash E$
5. $(E_1 \cdot E_2)/_{(p\cdot\Sigma^*)} = E_1/_{(p\cdot\Sigma^*)} + E_1 \cdot (E_2/_{(p\cdot\Sigma^*)})$
6. ${}_{(\Sigma^*\cdot p)}\backslash(E_1 \cdot E_2) = {}_{(\Sigma^*\cdot p)}\backslash E_2 + ({}_{(\Sigma^*\cdot p)}\backslash E_1) \cdot E_2$
7. *If* $E_1 \subseteq E_2/_{(p\cdot\Sigma^*)}$ *then* $E_1/_{(p\cdot\Sigma^*)} \subseteq E_2/_{(p\cdot\Sigma^*)}$.
8. *For any* $\alpha \in \Sigma^*$, $\alpha \in (E \cdot p \cdot \Sigma^*)/_{(p\cdot\Sigma^*)}$ *iff either* $\alpha/_{(p\cdot\Sigma^*)} \cap E \neq \phi$ *or* $\alpha \in E + E/_{(p\cdot\Sigma^*)}$.

Proof: By direct application of the definition of factorization. □

Lemma 6.4 *For any regular expression E over a finite alphabet Σ and $p \in \Sigma$:*

1. ${}_{(E\cdot p)}\backslash E = \phi$ *iff* $E\langle p\rangle\Sigma^*$ *is unambiguous.*
2. ${}_{(E\cdot p)}\backslash E = \phi$ *iff* $E/_{(p\cdot\Sigma^*)} \cap E = \phi$.
   Thus, in view of (1), $E/_{(p\cdot\Sigma^*)} \cap E = \phi$ *and unambiguity of* $E\langle p\rangle\Sigma^*$ *are equivalent.*

3. *$E\langle p\rangle\Sigma^*$ is maximal iff* $(E \cdot p \cdot \Sigma^*)/_{(p\cdot\Sigma^*)} = \Sigma^*$.

4. *If $E/_{(p\cdot\Sigma^*)}\|_n^p = \phi$ for some $n \geq 0$, then $E/_{(p\cdot\Sigma^*)}\|_m^p = \phi$ for all $m > n$.*

5. *If $E/_{(p\cdot\Sigma^*)}\|_n^p \neq \phi$ for some $n \geq 0$, then $E/_{(p\cdot\Sigma^*)}\|_m^p \neq \phi$ for all $0 \leq m \leq n$.*

Proof: Part (1) follows from Proposition 5.4, because $\Sigma^*/_{p\cdot\Sigma^*} = \Sigma^*$. Part (2) is proved by direct application of the definition of factoring. Part (3) follows from Corollary 5.8 and the fact that ${}_{(E\cdot p)}\backslash(E \cdot p \cdot \Sigma^*) = \Sigma^*$. For Part (4), suppose $s \cdot p \cdot s' \in E/_{(p\cdot\Sigma^*)}\|_{n+1}^p$, where s contains exactly n occurrences of p and s' contains none. Then $s \in E/_{(p\cdot\Sigma^*)}\|_n^p$. Hence if $E/_{(p\cdot\Sigma^*)}\|_{n+1}^p$ is non-empty then so must be $E/_{(p\cdot\Sigma^*)}\|_n^p$. Part (5) follows from Part (4). □

If $E/_{(p\cdot\Sigma^*)}\|_n^p = \phi$ we shall be informally saying that $E/_{(p\cdot\Sigma^*)}$ *matches a bounded number of p's.*

Proposition 6.5 (Correctness of Left-Filtering Maximization Algorithm) *Let $E\langle p\rangle\Sigma^*$ be an unambiguous extraction expression over a finite alphabet Σ, such that $E/_{(p\cdot\Sigma^*)}\|_n^p = \phi$ for some $n \geq 0$. Then the extraction expression $E'\langle p\rangle\Sigma^*$ computed by Algorithm 6.2 is a maximal and unambiguous generalization of $E\langle p\rangle\Sigma^*$.*

Proof: If there exists integer $n \geq 0$ such that $E/_{(p\cdot\Sigma^*)}\|_n^p = \phi$, then the **while**-loop of Algorithm 6.2 obviously terminates after n iterations. It also follows from Line 8 of the algorithm that $E\langle p\rangle\Sigma^* \preceq E'\langle p\rangle\Sigma^*$. It thus remains to be shown that $E'\langle p\rangle\Sigma^*$ is unambiguous and maximal.

To simplify the analysis of Algorithm 6.2, let us define:

$$F_i = E/_{(p\cdot\Sigma^*)}\|_i^p \qquad (i \geq 0) \tag{1}$$

$$R_0 = (\Sigma - p)^* - F_0 \tag{2}$$

$$R_i = F_{i-1} \cdot p \cdot (\Sigma - p)^* - F_i \qquad (i \geq 1) \tag{3}$$

Unambiguity: By Lemma 6.4(2), to show that $E'\langle p\rangle\Sigma^*$ is unambiguous it suffices to show that $E'/_{(p\cdot\Sigma^*)} \cap E' = \phi$. It follows from the above equations (2) and (3) that when Algorithm 6.2 terminates $E' = E + S = E + \bigcup_{0\leq i\leq n} R_i$. By (3), $R_i \subseteq F_{i-1} \cdot p \cdot (\Sigma - p)^*$ for any $R_i (i \geq 1)$. Thus it follows that $R_i/_{(p\cdot\Sigma^*)} \subseteq (F_{i-1} \cdot p \cdot (\Sigma - p)^*)/_{(p\cdot\Sigma^*)} = F_{i-1} + F_{i-1}/_{(p\cdot\Sigma^*)}$. Also, by (1), for any $F_i (i \geq 0)$, $F_i \subseteq E/_{(p\cdot\Sigma^*)}$. This together with Lemma 6.3(7) implies that $F_i/_{(p\cdot\Sigma^*)} \subseteq E/_{(p\cdot\Sigma^*)}$. Therefore, it follows that for all $i \geq 0$, $R_i/_{(p\cdot\Sigma^*)} \subseteq E/_{(p\cdot\Sigma^*)}$. By combining this with Lemma 6.3(1) we know that $E'/_{(p\cdot\Sigma^*)} = E/_{(p\cdot\Sigma^*)} + \bigcup_{0\leq i\leq n}(R_i/_{(p\cdot\Sigma^*)}) = E/_{(p\cdot\Sigma^*)}$. By Definition 6.1 and the above equations (2) and (3), every string in $R_i (i \geq 0)$ contains exactly i occurrences of p and $R_i \cap E/_{(p\cdot\Sigma^*)}\|_i^p = \phi$. Also by Definition 6.1, all prefixes of E that have precisely i occurrences of p are in $E/_{(p\cdot\Sigma^*)}\|_i^p$, it follows that $R_i \cap E/_{(p\cdot\Sigma^*)} = \phi$. Because $E\langle p\rangle\Sigma^*$ is unambiguous by assumption, it follows from Lemma 6.4(1) that $E/_{(p\cdot\Sigma^*)} \cap E = \phi$. Therefore, $E'/_{(p\cdot\Sigma^*)} \cap E' = E/_{(p\cdot\Sigma^*)} \cap (E + \bigcup_{0\leq i\leq n} R_i) = \phi$. Thus, by Lemma 6.4(2), $E'/_{(p\cdot\Sigma^*)}$ is unambiguous.

Maximality: By Lemma 6.4, Part (3), to show that $E'\langle p\rangle\Sigma^*$ is maximal it suffices to show that $(E' \cdot p \cdot \Sigma^*)/_{(p\cdot\Sigma^*)} = \Sigma^*$. First, we prove the following claim:

For any $\alpha \in \Sigma^*$, either $\alpha/_{(p\cdot\Sigma^*)} \cap \bigcup_{0\le i\le n} R_i \neq \phi$ or $\alpha \in R_s + F_s$, for some $0 \le s \le n$.

The proof is by induction on k, the number of occurrences of p in α. If $k = 0$, then $\alpha \in (\Sigma - p)^*$ and by (2) we have $(\Sigma - p)^* = R_0 + F_0$.

Suppose that the claim is true for $k = m$ and let $k = m+1$. Then $\alpha = \beta \cdot p \cdot \gamma$, where $\gamma \in (\Sigma - p)^*$ and p occurs in β exactly m times. By induction hypothesis, $\beta/_{(p\cdot\Sigma^*)} \cap \bigcup_{0\le i\le n} R_i \neq \phi$ or $\beta \in R_s + F_s$, $s \le n$. If $\beta/_{(p\cdot\Sigma^*)} \cap \bigcup_{0\le i\le n} R_i \neq \phi$, then $\alpha/_{(p\cdot\Sigma^*)} \cap \bigcup_{0\le i\le n} R_i \neq \phi$, and we are done. Otherwise, $\beta \in R_s$ or $\beta \in F_s$, $s \le n$. If $\beta \in R_s$, then $\beta \in \alpha/_{(p\cdot\Sigma^*)}$, so $\alpha/_{(p\cdot\Sigma^*)} \cap \bigcup_{0\le i\le n} R_i \neq \phi$, and we are done again. If, on the other hand, $\beta \in F_s$, it must be the case that $F_s \neq \phi$, so $0 \le s < n$, and $\alpha = \beta \cdot p \cdot \gamma \in F_s \cdot p \cdot (\Sigma - p)^*$. Then by (3) above, we have $\alpha \in R_{s+1} + F_{s+1}$. Since $s + 1 \le n$, the claim holds once again. Thus, the claim has been shown to hold for $k = m + 1$, which completes the proof of the claim.

During the proof of unambiguity, we have shown that $E' = E + \bigcup_{0\le i\le n} R_i$, $E'/_{(p\cdot\Sigma^*)} = E/_{(p\cdot\Sigma^*)}$, and that $F_i \subseteq E/_{(p\cdot\Sigma^*)}$, for all $0 \le i \le n$. In particular, $\bigcup_{0\le i\le n} R_i \subseteq E'$. It now follows from the above claim that for any $\alpha \in \Sigma^*$, either $\alpha/_{(p\cdot\Sigma^*)} \cap E' \neq \phi$ or $\alpha \in E' + E'/_{(p\cdot\Sigma^*)}$. Therefore, by Lemma 6.3(8), we have $\alpha \in (E' \cdot p \cdot \Sigma^*)/_{(p\cdot\Sigma^*)}$. Since α has been chosen arbitrarily, we obtain that $\Sigma^* = (E' \cdot p \cdot \Sigma^*)/_{(p\cdot\Sigma^*)}$. Finally, by Lemma 6.4(3), $E'\langle p\rangle\Sigma^*$ is maximal. □

Pivot Maximization Framework. Consider an extraction expression $E\langle p\rangle\Sigma^*$ and suppose we can find an equivalent representation for E of the form:

$$E_1 \cdot q_1 \cdot E_2 \cdot q_2 \cdots E_n \cdot q_n \cdot E_{n+1} \tag{4}$$

such that

- $E_1\langle q_1\rangle\Sigma^*$, $\cdots$, $E_n\langle q_n\rangle\Sigma^*$, $E_{n+1}\langle p\rangle\Sigma^*$ are all unambiguous extraction expressions; and
- $E_1\langle q_1\rangle\Sigma^*$, $\cdots$, $E_n\langle q_n\rangle\Sigma^*$, $E_{n+1}\langle p\rangle\Sigma^*$ can be maximized to $E'_1\langle q_1\rangle\Sigma^*$, $\cdots$, $E'_n\langle q_n\rangle\Sigma^*$, $E'_{n+1}\langle p\rangle\Sigma^*$, respectively.

In such a case, we shall call each q_i a *pivot* and say that $E\langle p\rangle\Sigma^*$ is *pivot-maximizable*. It turns out (by Proposition 6.8) that given a pivot-maximizable expression as above, the expression

$$(E'_1 \cdot q_1 \cdot E'_2 \cdot q_2 \cdots E'_n \cdot q_n \cdot E'_{n+1})\langle p\rangle\Sigma^* \tag{5}$$

is a maximal and unambiguous generalization of $E\langle p\rangle\Sigma^*$.

Pivot maximization is a powerful framework for harnessing the various specialized maximization algorithms. For instance, the left-filtering maximization algorithm can be used in this framework as follows. Suppose E can be represented as Expression (4), where E_1 matches only a bounded number of q_1's (*i.e.*, $E_1/_{(q_1\cdot\Sigma^*)}\|_n^{q_1} = \phi$ for some $n \ge 0$), E_2 matches only a bounded number of q_2's, etc., and E_{n+1} matches only a bounded number of p's. Then conditions of left-filtering maximization apply to each of the E_i's, so we can maximize the corresponding extraction expressions using that method.

By Proposition 6.8, Expression (5) is a maximal generalization of the original expression. Note that this technique is strictly more powerful than plain left-filtering maximization. Indeed, to maximize $E\langle p\rangle\Sigma^*$ through left-filtering, it must be the case that E matches only a bounded number of p's. In contrast, pivot maximization requires that *only* E_{n+1} must match a bounded number of p's, so E itself can potentially match an unbounded number of p's.

The proof of correctness of pivot maximization relies on Proposition 6.7 (and indirectly on Proposition 6.6).

Proposition 6.6 (Composition of Unambiguous Extraction Expressions) *If $E_1\langle q\rangle\Sigma^*$ and $E_2\langle p\rangle\Sigma^*$ are unambiguous extraction expressions over a finite alphabet Σ, where $q, p \in \Sigma$ (and $q = p$ is possible), then the extraction expression $(E_1 \cdot q \cdot E_2)\langle p\rangle\Sigma^*$ is unambiguous over Σ.*

Proof: By Lemma 6.4(2), to show that $(E_1 \cdot q \cdot E_2)\langle p\rangle\Sigma^*$ is unambiguous it suffices to show that $(E_1 \cdot q \cdot E_2)/_{(p\cdot\Sigma^*)} \cap E_1 \cdot q \cdot E_2 = \phi$. It follows from Lemma 6.3(5) that $(E_1 \cdot q \cdot E_2)/_{(p\cdot\Sigma^*)}$ is the union of $(E_1 \cdot q)/_{(p\cdot\Sigma^*)}$ and $E_1 \cdot q \cdot (E_2/_{(p\cdot\Sigma^*)})$. The proof that $(E_1 \cdot q)/_{(p\cdot\Sigma^*)} \cap E_1 \cdot q \cdot E_2 = \phi$ is by contradiction. Suppose, on the contrary, there exist $\alpha \in L(E_1), \beta \in L(E_2), \rho \in L((E_1 \cdot q)/_{(p\cdot\Sigma^*)})$ such that $\rho = \alpha \cdot q \cdot \beta$. Since there exists $\gamma \in \Sigma^*$ such that $\rho \cdot p \cdot \gamma \in L(E_1 \cdot q)$, then it follows that there exists $\lambda \in L(E_1)$ such that $\alpha \cdot q \cdot \beta \cdot p \cdot \gamma = \lambda \cdot q$, which means $\alpha \cdot q$ must be a prefix of λ. Therefore we have $\alpha \in E_1/_{(q\cdot\Sigma^*)} \cap E_1 \neq \phi$. By Lemma 6.4(2), this contradicts the assumption that $E_1\langle q\rangle\Sigma^*$ is unambiguous.

To show that $E_1 \cdot q \cdot (E_2/_{(p\cdot\Sigma^*)}) \cap E_1 \cdot q \cdot E_2 = \phi$, suppose, to the contrary, that there are $\alpha, \alpha' \in L(E_1)$, $\beta \in L(E_2)$, and $\beta' \in L(E_2/_{(p\cdot\Sigma^*)})$, such that $\alpha \cdot q \cdot \beta = \alpha' \cdot q \cdot \beta'$. Since $E_2\langle p\rangle\Sigma^*$ is unambiguous, Lemma 6.4(2) guarantees that $E_2/_{(q\cdot\Sigma^*)} \cap E_2 \neq \phi$ and, thus, $\beta \neq \beta'$. It then follows that either $\alpha \cdot q$ is a prefix of α', or $\alpha' \cdot q$ is a prefix of α. Thus, either α or α' belongs to $E_1/_{(q\cdot\Sigma^*)} \cap E_1$. By Lemma 6.4(2), this contradicts the fact that $E_1\langle q\rangle\Sigma^*$ is unambiguous. □

Proposition 6.7 (Composition of Maximal Extraction Expressions) *If $E_1\langle q\rangle\Sigma^*$ and $E_2\langle p\rangle\Sigma^*$ are maximal unambiguous extraction expressions over a finite alphabet Σ, where $q, p \in \Sigma$ (and $q = p$ is possible), then the extraction expression $(E_1 \cdot q \cdot E_2)\langle p\rangle\Sigma^*$ is maximal and unambiguous over Σ.*

Proof: *Unambiguity:* Directly follows from Proposition 6.6.

Maximality: By Lemma 6.4(3), to show that $(E_1 \cdot q \cdot E_2)\langle p\rangle\Sigma^*$ is maximal it suffices to show that $(E_1 \cdot q \cdot E_2 \cdot p \cdot \Sigma^*)/_{(p\cdot\Sigma^*)} = \Sigma^*$. By Lemma 6.3(5), we have:

$$\begin{aligned}(E_1 \cdot q \cdot E_2 \cdot p \cdot \Sigma^*)/_{(p\cdot\Sigma^*)} \;=\; & E_1/_{(p\cdot\Sigma^*)} + \\ & E_1 \cdot (q/_{(p\cdot\Sigma^*)}) + \\ & E_1 \cdot q \cdot (E_2/_{(p\cdot\Sigma^*)}) + \\ & E_1 \cdot q \cdot E_2 + \\ & E_1 \cdot q \cdot E_2 \cdot p \cdot \Sigma^*\end{aligned} \tag{6}$$

Given an arbitrary string $\alpha \in \Sigma^*$, we must show that $\alpha \in (E_1 \cdot q \cdot E_2 \cdot p \cdot \Sigma^*)/_{(p\cdot\Sigma^*)}$, or in other words, α must fall into one of the subsets as shown in the above Equation (6). Because $E_1\langle q\rangle\Sigma^*$ is maximal, from Lemma 6.3(8) there are two cases to consider:

(i) $\alpha/_{(q\cdot\Sigma^*)} \cap E_1 \neq \phi$. Then there exist $\beta \in L(E_1), \gamma \in \Sigma^*$ such that $\alpha = \beta \cdot q \cdot \gamma$. Because $E_2\langle p\rangle\Sigma^*$ is maximal, it follows from Lemma 6.3(8) that either $\gamma/_{(p\cdot\Sigma^*)} \cap E_2 \neq \phi$ or $\gamma \in E_2 + E_2/_{(p\cdot\Sigma^*)}$. If $\gamma/_{(p\cdot\Sigma^*)} \cap E_2 \neq \phi$, then there must exist $\rho \in L(E_2),\ \lambda \in \Sigma^*$ such that $\gamma = \rho \cdot p \cdot \lambda$. It then follows that $\alpha = \beta \cdot q \cdot \rho \cdot p \cdot \lambda$. Therefore $\alpha \in L(E_1 \cdot q \cdot E_2 \cdot p \cdot \Sigma^*)$ (the last component of Equation (6) above). On the other hand, if $\gamma \in E_2 + E_2/_{(p\cdot\Sigma^*)}$, then, clearly, either $\alpha \in L(E_1 \cdot q \cdot E_2)$ or $\alpha \in L(E_1 \cdot q \cdot (E_2/_{(p\cdot\Sigma^*)}))$ (the 4th and the 3rd component of Equation (6), respectively).

(ii) $\alpha/_{(q\cdot\Sigma^*)} \cap E_1 = \phi$ and $\alpha \in E_1 + E_1/_{(q\cdot\Sigma^*)}$. If $(\alpha \cdot p)/_{(q\cdot\Sigma^*)} \cap E_1 = \phi$, then by Lemma 6.4(2), $\alpha \cdot p \in E_1$ or $\alpha \cdot p \in E_1/_{(q\cdot\Sigma^*)}$. In either case, $\alpha \in E_1/_{(p\cdot\Sigma^*)}$(the first component of Equation (6)). If, on the other hand, $(\alpha \cdot p)/_{(q\cdot\Sigma^*)} \cap E_1 \neq \phi$, then by Lemma 6.3(5), $(\alpha \cdot p)/_{(q\cdot\Sigma^*)} = \alpha/_{(q\cdot\Sigma^*)} + \alpha \cdot (p/_{(q\cdot\Sigma^*)})$ and so $(\alpha \cdot p)/_{(q\cdot\Sigma^*)} \cap E_1 = \alpha \cdot (p/_{(q\cdot\Sigma^*)}) \cap E_1 \neq \phi$. This implies that $p/_{(q\cdot\Sigma^*)} = \epsilon$(the empty string) and thus $q = p$. In turn, this means that $\alpha \in E_1 = E_1 \cdot (q/_{(p\cdot\Sigma^*)})$ (the 2nd component of Equation (6)). □

Proposition 6.8 (Correctness of Pivot Maximization) *If an extraction expression $E\langle p\rangle\Sigma^*$ over a finite alphabet Σ, represented as in Expression (4), is pivot-maximizable, then Expression (5) is a maximal and unambiguous generalization of $E\langle p\rangle\Sigma^*$.*

Proof: By induction on n, the number of pivots in E. The case of $n = 0$ is trivial.

Suppose the claim is true for $n = k, k \geq 0$. Then for $n = k + 1$, we have:

$$E\langle p\rangle\Sigma^* = (E_1 \cdot q_1 \cdots E_k \cdot q_k \cdot E_{k+1} \cdot q_{k+1} \cdot E_{k+2})\langle p\rangle\Sigma^*. \tag{7}$$

Let F be $F = E_1 \cdot q_1 \cdots E_{k-1} \cdot q_{k-1} \cdot E_k \cdot q_k \cdot E_{k+1}$. Since the number of pivots in F is k, by induction hypothesis we know that

$$(E'_1 \cdot q_1 \cdots E'_{k-1} \cdot q_{k-1} \cdot E'_k \cdot q_k \cdot E'_{k+1})\langle q_{k+1}\rangle\Sigma^* \tag{8}$$

is a maximal and unambiguous generalization of $F\langle q_{k+1}\rangle\Sigma^*$.

Because $E'_{k+2}\langle p\rangle\Sigma^*$ is a maximal expression that generalizes $E_{k+2}\langle p\rangle\Sigma^*$ (due to the assumption that E is pivot-maximizable), Proposition 6.7 implies that the result of composing Expression (8) and $E'_{k+2}\langle p\rangle\Sigma^*$:

$$(E'_1 \cdot q_1 \cdot E'_2 \cdot q_2 \cdots E'_{k+1} \cdot q_{k+1} \cdot E'_{k+2})\langle p\rangle\Sigma^* \tag{9}$$

is unambiguous and maximal. Clearly, Expression (9) generalizes Expression (7), since Expression (7) is $(F \cdot q_{k+1} \cdot E_{k+2})\langle p\rangle\Sigma^*$ and Expression (8) generalizes $F\langle q_{k+1}\rangle\Sigma^*$. □

7 Putting it All Together

We are now going to revisit our motivating example of Section 3 and apply the tools and techniques we developed in Sections 4 through 6.

Consider the two HTML pages from Sections 4, in their tag-sequence representation. In both cases, we are interested in the second `INPUT`-element of the form. The corresponding extraction expressions (each one works only for one of the two pages) are as follows:

```
P H1 /H1 P FORM INPUT <INPUT> P INPUT INPUT /FORM
TABLE TR TD /TD TD /TD /TR TR TD /TD TD /TD /TR FORM TR TD INPUT /TD TD <INPUT> /TD TD INPUT /TD /TR /FORM /TABLE
```

Our strategy is to first generalize these strings into an extraction expression. Learning techniques of [18, 3, 8, 4, 5] could be utilized at this stage. For the sake of this example, we will use a simple left-to-right merging heuristic, which tries to find a sequence of tags common to the two strings and takes the union of everything in-between. This yields the extraction expression below, where we replaced `TR TD /TD TD /TD /TR TR TD /TD TD /TD /TR` with `TR ... /TR`, to save space:

$$((\texttt{P H1 /H1 P}) + (\texttt{TABLE TR ... /TR}))\ \texttt{FORM}\ (\texttt{TR TD})?\ \texttt{INPUT}\ (\texttt{/TD TD})?\ \texttt{<INPUT>}\ \mathcal{T}\texttt{ags*} \tag{10}$$

In this expression, the symbol "?" means that the corresponding subexpression can occur zero or more times, and "+" stands for the union, as before.

By Proposition 5.4, this expression is unambiguous, but it is not maximal. If none of the heuristics succeeds in producing an unambiguous expression, then the algorithm fails. An interesting problem is to develop heuristics for guiding the disambiguation process for extraction expressions, as mentioned in Section 8.

The left-merging heuristics used for the above example is geared towards the pivot maximization framework. In our case, we can use the symbols `FORM` and *INPUT* as pivots. It turns out that the conditions for pivot maximization are satisfied and that each of the three expressions

$$((\texttt{P H1 /H1 P}) + (\texttt{TABLE TR ... /TR}))\ \texttt{<FORM>}\ \mathcal{T}\texttt{ags*}$$
$$(\texttt{TR TD})?\ \texttt{<INPUT>}\ \mathcal{T}\texttt{ags*}$$
$$(\texttt{/TD TD})?\ \texttt{<INPUT>}\ \mathcal{T}\texttt{ags*}$$

can be maximized using the left-filtering algorithm (Algorithm 6.2). The result is the following maximal and unambiguous extraction expression:

$$(\mathcal{T}\texttt{ags- FORM})*\ \texttt{FORM}\ (\mathcal{T}\texttt{ags- INPUT})*\ \texttt{INPUT}\ (\mathcal{T}\texttt{ags- INPUT})*\ \texttt{<INPUT>}\ \mathcal{T}\texttt{ags*}$$

It is worth noting that Expression (10) can also be maximized by a direct application of Algorithm 6.2. However, this will produce a different (much larger) extraction expression. The semantics of the two expressions will also be different: while the above expression always finds the second `INPUT`-element in the *first* form, the expression produced by a direct application of Algorithm 6.2 will be looking for a second `INPUT`-element on the page, even if the first and the second `INPUT`-elements come from different forms.

8 Conclusion and Future Work

In this paper, we have made the first few steps towards a theory of resilient data extraction from semi-structured documents. To this end, we defined the notion of extraction expression, provided a correctness criterion for it (unambiguity), and formalized the intuitive notion of such an expression being "robust" in the presence of changes in the source document (maximality). We provided complexity results for deciding ambiguity and maximality and proposed powerful algorithms that can maximize very large, practical classes of unambiguous extraction expressions.

Several problems still remain. First, it is still unknown whether the general problem of maximization is decidable. Second, there still is a need for learning techniques that generate good initial unambiguous expressions that could be used by our maximization algorithms. The works discussed in Section 2 do not address this issue. One way how the existing algorithms can help the task of learning unambiguous extraction expressions is as follows: we can use them to generate ambiguous

expressions first. Then we could feed this expression to a "disambiguation procedure" along with a number of counterexamples. Developing such disambiguation techniques is a topic for future research. The third line of work is to explore classes of regular expressions that can be maximized with lower computational complexity.

Another interesting issue is to explore data extraction from XML. Although XML documents are generally better structured than HTML, automatic extraction from such documents calls for the creation of ontologies in order to be able to generate suitable queries automatically. We believe that the techniques presented here along with the works on learning regular expressions discussed in Section 2 can provide a simpler solution. However, XML can make this task simpler and more reliable. One interesting issue here is using DTDs to guide the learning algorithms.

Finally, we should mention that, like all extraction techniques that are based on regular expressions, our framework has limitations. For instance, we cannot learn or generalize extraction expressions that can be expressed only using context-free grammars. A typical example here is extracting the *middle row* from dynamically generated tables. Indeed, the training set for such a learning system would consist of extraction expressions of the form $\mathtt{TR} < \mathtt{TR} > \mathtt{TR}$, $\mathtt{TR}\,\mathtt{TR} < \mathtt{TR} > \mathtt{TR}\,\mathtt{TR}$, etc. The desired pattern to learn here is $\mathtt{TR}^n < \mathtt{TR} > \mathtt{TR}^n$, but the language recognized by this expression is not regular, so this extraction problem cannot be solved using regular expression based techniques. It would therefore be interesting to extend our framework to include more general patterns.It would then be possible to apply our results to works like [6] and, thus, enhance their results.

References

[1] S. Abiteboul. Querying semi-structured data. In *Int'l Conference on Database Theory*, volume 1186, pages 1–18, Delphi, Greece, 1997. Springer.

[2] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The lorel query language for semistructured data. *Int. J. on Digital Libraries*, 1(1):68–88, 1997.

[3] B. Adelberg. NoDoSe: A tool for semi-automatically extracting structured and semi-structured data from text documents. In *ACM SIGMOD Conference on Management of Data*, pages 283–294, Washington, 1998. ACM.

[4] D. Angluin. Finding patterns common to a set of strings. In *ACM Symposium on Theory of Computing*, pages 130–141, 1979.

[5] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[6] N. Ashish and C. Knoblock. Wrapper generation for semi-structured internet sources. *ACM SIGMOD Record*, 26(4):8–15, 1997.

[7] P. Atzeni and G. Mecca. Cut & paste. In *ACM Symposium on Principles of Database Systems*, pages 117–121, Arizona, June 1997. ACM.

[8] R.C. Berwick and S. Pilato. Learning syntax by automata induction. *Machine Learning*, 2:9–38, 1987.

[9] B.Ribeiro-Neto, A.H.L. Laender, and A.S. da Silva. Extracting semi-structured data through examples. In *Proceedings of the International Conference on Knowledge Management*, November 1999.

[10] P. Buneman. Semistructured data. In *ACM Symposium on Principles of Database Systems*, pages 117–121, Tucson, Arizona, June 1997.

[11] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *ACM SIGMOD Conference on Management of Data*, Montreal, Canada, 1996. ACM.

[12] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle aand D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Journal of Data and Knowledge Engineering*, November 1999.

[13] Jean-Robert Gruser, L. Raschid, M. E. Vidal, and L. Bright. Wrapper generation for web accessible data sources. In *Proceedings of the Third International Conference on Cooperative Information Systems (CoopIS98)*, pages 14–23, New York City, New York, 1998.

[14] J. Hammer, H. Garcia-Molina, J. Cho, A. Crespo, and R. Aranha. Extracting semistructured information from the web. In *In Proceedings of the Workshop on Management of Semistructured Data*, pages 18–25, Tucson, Arizona, May 1997.

[15] J. Hammer, Hector Garcia-Molina, S. Nestorov, Ramana Yerneni, Markus M. Breunig, and Vasilis Vassalos. Template-based wrappers in the tsimmis system. In *ACM SIGMOD Conference on Management of Data*, pages 532–535. ACM, 1997.

[16] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

[17] http://www.jango.com. Jango Corporation.

[18] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *Int'l Joint Conference on Artificial Intelligence*, volume 1, pages 729–737, Nagoya, Japan, 1997.

[19] H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation.* Prentice Hall, Englewood Cliffs, NJ, 1981.

[20] Seung-Jin Lim and Yiu-Kai Ng. An automated approach for retrieving hierarchical data from html table. In *Proceedings of the International Conference on Knowledge Management*, November 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, March 1997.

[22] A. Sahuguet and F. Azavant. Web Ecology: Recycling HTML pages as XML documents using W4F. In *ACM SIGMOD Workshop on Database the Web and Databases (WebBD'99)*, pages 31–35, Philadelphia, Pennsylvania, June 1999.