A Unifying Approach to HTML Wrapper Representation and Learning

Gunter Grieser[1], Klaus P. Jantke[2], Steffen Lange[3], and Bernd Thomas[4]

[1] Technische Universität Darmstadt, FB Informatik
Alexanderstraße 10, 64283 Darmstadt, Germany
e-mail: grieser@informatik.tu-darmstadt.de

[2] Deutsches Forschungszentrum für Künstliche Intelligenz
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
e-mail: jantke@dfki.de

[3] Universität Leipzig, Institut für Informatik
Augustusplatz 10-11, 04109 Leipzig, Germany
e-mail: slange@informatik.uni-leipzig.de

[4] Universität Koblenz, FB Informatik
Rammsweg 1, 56070 Koblenz, Germany
e-mail: bthomas@uni-koblenz.de

Abstract. The number, the size, and the dynamics of Internet information sources bears abundant evidence of the need for automation in information extraction. This calls for representation formalisms that match the World Wide Web reality and for learning approaches and learnability results that apply to these formalisms.
The concept of elementary formal systems is appropriately generalized to allow for the representation of wrapper classes which are relevant to the description of Internet sources in HTML format. Related learning results prove that those wrappers are automatically learnable from examples. This is setting the stage for information extraction from the Internet by exploitation of inductive learning techniques.

1 Motivation

Today's online access to millions or even billions of documents in the World Wide Web is a great challenge to research areas related to knowledge discovery and information extraction (IE). The general task of IE is to locate specific pieces of text in a natural language document.

The authors' approach draws advantage from the fact that all documents prepared for the Internet in HTML, in XML or in any other possibly forthcoming syntax have to be interpreted by browsers sitting anywhere in the World Wide Web. For this purpose, the documents do need to contain syntactic expressions which are controlling its interpretation including its visual appearance and its interactive behaviour. In HTML, these are the text formatting and annotating strings (*tags*), and in LaTeX, for instance, there are numerous commands. The

document's contents is embedded into those syntactic expressions which are usually hidden from the user.

The user is typically not interested in the varying syntactic tricks invoked for information presentation, but in the information itself. Accordingly, the present approach assumes that the user deals exclusively with the desired contents, whereas a system for IE should deal with the syntax.

In a characteristic scenario of system-supported IE, the user is taking a source document and is highlighting representative pieces of information (s)he is interested in. It is assumed that the user's view at a certain document, which might evolve gradually or which might even change over time, can be represented as a certain relation of text strings from the underlying source. Thus, the user's input are just a few sample instances from the relation (s)he is seeing when looking at the given source document.

It is left to the system to understand how the target information is wrapped into syntactic expressions. This is a first learning task posed to the system.

Next, the system has to generalize the wrapper concept hypothesized for coming up with an extraction procedure. Applied to the given source document, this extraction mechanism generates a certain hypothetical relation which is returned to the user. The step of generalization mentioned is a second learning task to be solved by the system.

In response to providing a few samples illustrating the user's view at the source document, (s)he receives a list of extracted tuples. The user may compare the system's guess to the results aimed at and, in dependence on the comparison's outcome, complain about the result – if necessary – by indicating erroneously extracted tuples or by supplementing those tuples missing in the systems output.

When the user is returning this information to the system, a new cycle of learning and information extraction is initiated. Several further interactions may follow, and the extraction mechanism generated may be applied to further source documents. Several cycles of interaction will improve the IE results.

From a theoretical perspective, the system is performing a two-level learning process based on particular positive and negative examples provided by the user.

2 Introduction

Let us explain this approach with the help of the bibliography of this paper. Imagine, the list of referenced papers is presented in a semi-structured form, namely HTML, as follows:

⟨html⟩⟨body⟩⟨h1⟩References⟨/h1⟩
⟨ol⟩⟨li⟩D. Angluin, 'Inductive inference of formal languages from positive data',
⟨i⟩Information and Control⟨/i⟩, ⟨b⟩45⟨/b⟩, 117-135, (1980).⟨/li⟩
⟨li⟩D. Angluin and C.H. Smith, 'A survey of inductive inference: Theory and methods', ⟨i⟩Computing Surveys⟨/i⟩, ⟨b⟩15⟨/b⟩, 237–269, (1983).⟨/li⟩
⟨li⟩S. Arikawa, ...

Figure 1.

In order to extract information, we can assume special text segments to be delimiters (henceforth called anchors) marking the beginning and the end of the relevant information to be extracted.

For instance, to extract the authors of a publication, for example, the text segments $\langle li \rangle$ resp. *,'* are interpreted as the left and the right anchor for the authors' name(s). The year of publication (more exactly, its last two digits) can then be identified by the left anchor *(19* and the right anchor *).*$\langle$*/li*$\rangle$. Using these anchors, the extraction of all authors' names and the relevant years of publication from the bibliography represented as HTML document is straightforward. One first tries to find the left and right anchor for the authors' name(s) and then, to the right, the nearest left and right anchor for the year of publication. This is repeated as long as those pairs of anchors can be found.

Several approaches [8, 14, 15] in the IE community use this basic idea to define extraction procedures (*wrappers or templates*) based on their own description language. Further, investigations showed that wrappers can be classified according to their expressiveness based on several *structural* constraints. This leads to the general question whether or not such regularly structured descriptions can automatically be constructed and furthermore learned. A rule based way to define a wrapper for the previously discussed example is presented in Figure 2.

Here, by capital letters we denote variables, terminal symbols are typeset in italics. The first rule can be interpreted as follows: An author A and the year of publication Y can be extracted from a document D in case that (i) D matches the pattern $\mathtt{X}_1\mathtt{L}_1\mathtt{A}\mathtt{R}_1\mathtt{X}_2\mathtt{L}_2\mathtt{Y}\mathtt{R}_2\mathtt{X}_3$ and (ii) the instantiations of the variables meet certain constraints. For example, the constraint $\mathtt{r}_1(\mathtt{R}_1)$ states that the variable $\mathtt{R}_1$ can only be replaced by the string *,'* or the string *,*$\langle i \rangle$. Further constraints like $\mathtt{nc\text{-}r}_1\mathtt{(A)}$ explicitly state which text segments are not suited to be substituted for the variable A (cf. rules 2–5 and 15–16). In this particular case, text segments that contain the substrings *,'* or *,*$\langle i \rangle$ are not allowed. If a document D matches the pattern $\mathtt{X}_1\mathtt{L}_1\mathtt{A}\mathtt{R}_1\mathtt{X}_2\mathtt{L}_2\mathtt{Y}\mathtt{R}_2\mathtt{X}_3$ and if all specified constraints are fulfilled, then the instantiations of the variables A and Y yield the information required.

```
1  extract(A,Y,X1L1AR1X2L2YR2X3) ← l1(L1), nc-r1(A), r1(R1),
                                    nc-l2(X2),
                                    l2(L2), nc-r2(Y), r2(R2).
2  nc-r1(X) ← not c-r1(X).
3  c-r1(X) ← r1(X).          4  c-r1(XY) ← c-r1(X).      5  c-r1(XY) ← c-r1(Y).
6  nc-l2(X) ← not c-l2(X).
7  c-l2(X) ← l2(X).          8  c-l2(XY) ← c-l2(X).      9  c-l2(XY) ← c-l2(Y).
10 nc-r2(X) ← not c-r2(X).
11 c-r2(X) ← r2(X).          12 c-r2(XY) ← c-r2(X).      13 c-r2(XY) ← c-r2(Y).
14 l1(⟨li⟩).
15 r1(,').
16 r1(,⟨i⟩).
17 l2(*(19*).
18 r2(*).*⟨*/li*⟩).
```

Figure 2.

The aim of this paper is twofold. We propose a uniform way to represent wrappers, namely advanced elementary formal systems (AEFSs, for short). This concept is generalizing SMULLYANs [12] elementary formal systems (EFSs) which have thoroughly been studied in different respects. We investigate the expressive power of AEFSs and show that it is sufficient for wrapper representation though its semantics is still reasonable. To lay a cornerstone of our application-oriented work, we investigate which learnability results achieved for EFSs lift to AEFSs. Additionally, we prototypically show how AEFSs can be used to describe a certain class of HTML wrappers, so-called island wrappers (cf. [14, 15]). We prove the learnability of island wrappers from only positive examples under certain natural assumptions.

3 Advanced Elementary Formal Systems

In this section, we introduce a quite general formalism to describe wrappers, namely advanced elementary formal systems (AEFSs, for short). In addition, we study the expressive power of AEFSs and deal with the question of whether or not AEFSs can be learned from examples.

AEFSs generalize SMULLYAN's [12] elementary formal systems (EFSs, for short) which he introduced to develop his theory of recursive functions. In the last years, the learnability of EFSs has intensively been studied within several formal frameworks (cf. [4, 16, 3, 5, 11, 17, 10]).

3.1 Elementary Formal Systems

Next, we provide notions and notations that allow for a formalization of EFSs.

Assume three mutually disjoint sets – a finite set Σ of characters, a finite set Π of predicates, and an enumerable set X of variables. We call every element in $(\Sigma \cup X)^+$ a pattern and every string in Σ^+ a ground pattern. For a pattern π, we let $v(\pi)$ be the set of variables in π.

Let $p \in \Pi$ be a predicate of arity n and let $\pi_1, \ldots, \pi_n$ be patterns. Let $A = p(\pi_1, \ldots, \pi_n)$. Then, A is said to be an atomic formula (an atom, for short). A is ground, if all the patterns π_i are ground. Moreover, $v(A)$ denotes the set of variables in A.

Let A and $B_1, \ldots, B_n$ be atoms. Then, $r = A \leftarrow B_1, \ldots, B_n$ is a rule, A is the head of r, and all the B_i form the body of r. Then, r is a ground rule, if all atoms in r are ground. Additionally, if $n = 0$, then r is called a fact and sometimes we write A instead of $A \leftarrow$.

Let σ be a non-erasing substitution, i.e., a mapping from X to Σ^+. For any pattern π, $\pi\sigma$ is the pattern which one obtains when applying σ to π. Let $C = p(\pi_1, \ldots, \pi_n)$ be an atom and let $r = A \leftarrow B_1, \ldots, B_n$ be a rule. Then, we set $C\sigma = p(\pi_1\sigma, \ldots, \pi_n\sigma)$ and $r\sigma = A\sigma \leftarrow B_1\sigma, \ldots, B_n\sigma$. If $r\sigma$ is ground, then it is said to be a ground instance of r.

Definition 1 ([5]) *Let Σ, Π, and X be fixed, and let Γ be a finite set of rules. Then, $S = (\Sigma, \Pi, \Gamma)$ is said to be an EFS.*

EFSs can be considered as particular logic programs without negation. There are two major differences: (i) patterns play the role of terms and (ii) unification has to be realized modulo the equational theory

$$E = \{\circ(x, \circ(y, z)) = \circ(\circ(x, y), z)\},$$

where $\circ$ is interpreted as concatenation of patterns.

As for logic programs (cf., e.g., [9]), the semantics of an EFS S can be defined via the following operator T_S. In the corresponding definition, we use the following notations. For any EFS $S = (\Sigma, \Pi, \Gamma)$, $B(S)$ denotes the set of all well-formed ground atoms over Σ and Π, and $G(S)$ denotes the set of all ground instances of rules in Γ.

Definition 2 *Let S be an EFS and let $I \subseteq B(S)$. Then, we let $T_S(I) = I \cup \{A \mid A \leftarrow B_1, \ldots, B_n \in G(S)$ for some $B_1 \in I, \ldots, B_n \in I\}$.*

Note that, by definition, the operator T_S is idempotent and monotonic.

As usual, we let $T_S^{n+1}(I) = T_S(T_S^n(I))$, where $T_S^0(I) = I$, by convention.

Definition 3 *Let S be an EFS. Then, we let $Sem(S) = \bigcup_{n \in \mathbb{N}} T_S^n(\emptyset)$.*

In general, $Sem(S)$ is semi-decidable, but not decidable. However, as we will see below, $Sem(S)$ turns out to be decidable in case that S meets several natural syntactical constraints (cf. Theorem 5).

Finally, by $\mathcal{E}$ we denote the collection of all EFSs.

When using EFSs to describe wrappers, a problem that usually occurs is that one has to put constraints on the patterns that form admissible substitutions for particular variables (cf., e.g., the wrapper in Figure 2). One approach to cope with this problem is to explicitly describe the admissible patterns. In some cases, it is more convenient to explicitly describe the exceptions and to postulate that every pattern that does not serve as exception is admissible. In contrast to EFSs, AEFSs provide enough flexibility to realize the latter approach, as well.

3.2 Beyond Elementary Formal Systems

Informally speaking, an AEFS is an EFS that may additionally contain rules of the form $A \leftarrow not\ B_1$, where A and B_1 are atoms and *not* stands for negation as finite failure. The underlying meaning is as follows. If, for instance, $A = p(x_1, \ldots, x_n)$ and $B_1 = q(x_1, \ldots, x_n)$, then the predicate p succeeds iff the predicate q fails.

However, taking the conceptual difficulties into consideration that occur when defining the semantics of logic programs with negation as finite failure (cf., e.g., [9]), AEFSs are constrained to meet several additional syntactic requirements (cf. Definition 4). The requirements posed guarantee that, similarly to stratified logic programs (cf., e.g., [9]), the semantics of AEFSs can easily be described. Moreover, as a side-effect, the requirements posed guarantee that AEFSs inherit some of the convenient properties of EFSs (cf., e.g., Theorem 5).

Before formally defining how AEFSs look like, we need some more notations. Let Γ be a set of rules (including rules of the form $A \leftarrow not\ B_1$). Then, $hp(\Gamma)$ denotes the set of predicates that appear in the head of any rule in Γ.

Definition 4 *AEFSs and their semantics are defined according to the following rules. Let* $S' = (\Sigma, \Pi', \Gamma')$ *be an EFS. Moreover, let* $S_1 = (\Sigma, \Pi_1, \Gamma_1)$ *and* $S_2 = (\Sigma, \Pi_2, \Gamma_2)$ *be AEFSs.*

1. $S = S'$ *is an AEFS. Moreover, we let* $Sem(S) = Sem(S')$.
2. *If* $\Pi_1 \cap \Pi_2 = \emptyset$, *then* $S = (\Sigma, \Pi_1 \cup \Pi_2, \Gamma_1 \cup \Gamma_2)$ *is an AEFS. Moreover, we let* $Sem(S) = Sem(S_1) \cup Sem(S_2)$.
3. *Let* $p \notin \Pi_1$ *and* $q \in \Pi_1$ *be predicates of arity* n. *Then,* $S = (\Sigma, \Pi_1 \cup \{p\}, \Gamma_1 \cup \{p(x_1, \ldots, x_n) \leftarrow not\ q(x_1, \ldots, x_n)\})$ *is an AEFS. Moreover, we let* $Sem(S) = Sem(S_1) \cup \{p(s_1, \ldots, s_n) \mid s_1 \in \Sigma^+, \ldots, s_n \in \Sigma^+, q(s_1, \ldots, s_n) \notin Sem(S_1)\}$.
4. *Let* $hp(\Gamma') \cap \Pi_1 = \emptyset$. *Then,* $S = (\Sigma, \Pi' \cup \Pi_1, \Gamma' \cup \Gamma_1)$ *is an AEFS. Moreover, we let* $Sem(S) = \bigcup_{n \in \mathbb{N}} T^n_{S'}(Sem(S_1))$.

Finally, by $\mathcal{AE}$ *we denote the collection of all AEFSs.*

Having a closer look at Figure 2, one realizes that AEFSs can be used to describe interesting wrappers.

3.3 Expressiveness

In the following, we show how AEFSs can be used to describe formal languages and relate the resulting language classes to the language classes of the classical CHOMSKY hierarchy (cf. [7]). Although we are mainly interested in using AEFSs for describing wrappers, we strongly believe that the established relations are quite helpful to better understand the expressive power of AEFSs.

Definition 5 *Let* $S = (\Sigma, \Pi, \Gamma)$ *be an AEFS and let* $p \in \Pi$ *be a unary predicate. We let* $L(S, p) = \{s \mid p(s) \in Sem(S)\}$.

Intuitively speaking, $L(S, p)$ is the language which the AEFS S defines via the unary predicate p.

Definition 6 *Let* $M \subseteq \mathcal{AE}$. *Then, the set* $\mathcal{L}(M)$ *of all languages that are definable with AEFSs in* M *contains every language* L *for which there are an AEFS* $S = (\Sigma, \Pi, \Gamma)$ *in* M *and some unary predicate* $p \in \Pi$ *such that* $L = L(S, p)$.

For example, $\mathcal{L}(\mathcal{AE})$ is the class of all languages that are definable by AEFSs.

Our first result puts the expressive power of AEFSs into the right perspective. Let $\mathcal{L}_{r.e.}$ be the class of all recursively enumerable languages.

Theorem 1 $\mathcal{L}_{r.e.} \subset \mathcal{L}(\mathcal{AE})$.

Moreover, the following closedness properties can be shown.

Theorem 2 $\mathcal{L}(\mathcal{AE})$ *is closed under the operations union, intersection, complement, and concatenation.*

To elaborate a more accurate picture, similarly to [3], we next introduce several constraints on the structure of the rules an EFS resp. AEFS may contain.

Let r be a rule of form $A \leftarrow not\ B_1$ or $A \leftarrow B_1, \ldots, B_n$. Then, r is said to be variable-bounded iff, for all $i \leq n$, $v(B_i) \subseteq v(A)$. Moreover, r is said to be

length-bounded iff, for all substitutions σ, $|A\sigma| \geq |B_1\sigma| + \cdots + |B_n\sigma|$. Clearly, if r is length–bounded, then r is also variable–bounded. Note that, in general, the opposite does not hold. Finally, r is said to be simple iff A is of form $p(\pi)$ and π is a pattern in which every variable occurs at most once.

Definition 7 *Let $S = (\Sigma, \Pi, \Gamma)$ be an AEFS. Then, S is said to be variable–bounded iff all $r \in \Gamma$ are variable–bounded.*
Moreover, S is said to be length–bounded iff all $r \in \Gamma$ are length–bounded.
Next, S is said to be regular iff Π contains exclusively unary predicates and all $r \in \Gamma$ are length-bounded as well as simple.
Finally, by vb-$\mathcal{AE}$ (vb-$\mathcal{E}$), lb-$\mathcal{AE}$ (lb-$\mathcal{E}$), and reg-$\mathcal{AE}$ (reg-$\mathcal{E}$) we denote the collection of all variable–bounded, length–bounded, and regular AEFSs (EFSs).

Now, similarly to Theorem 2, the following result can be established.

Theorem 3 *$\mathcal{L}$(vb-$\mathcal{AE}$), $\mathcal{L}$(lb-$\mathcal{AE}$), and $\mathcal{L}$(reg-$\mathcal{AE}$) are closed under the operations union, intersection, complement, and concatenation.*

The next theorem summarizes the announced relations to some important language classes of the CHOMSKY hierarchy (cf. [7]). Here, $\mathcal{L}_{cs}$ and $\mathcal{L}_{cf}$ denote the class of all context sensitive and context free languages, respectively.

Theorem 4
1. $\mathcal{L}_{r.e.} \subset \mathcal{L}$(vb-$\mathcal{AE}$).
2. $\mathcal{L}_{cs} = \mathcal{L}$(lb-$\mathcal{AE}$).
3. $\mathcal{L}_{cf} \subset \mathcal{L}$(reg-$\mathcal{AE}$) $\subset \mathcal{L}_{cs}$.

Assertion (2) of the latter theorem allows for the following insight:

Theorem 5 *If $S \in$ lb-$\mathcal{AE}$, then Sem(S) is decidable.*

Note that, for length–bounded EFSs, the equivalent of Theorem 5 has already been shown in [5]. Moreover, for EFSs, Theorem 4 rewrites as follows:

Theorem 6 ([5])
1. $\mathcal{L}_{r.e.} = \mathcal{L}$(vb-$\mathcal{E}$).
2. $\mathcal{L}_{cs} = \mathcal{L}$(lb-$\mathcal{E}$).
3. $\mathcal{L}_{cf} = \mathcal{L}$(reg-$\mathcal{E}$).

3.4 Learnability

At the end of this section, we present some basic results concerning the question of whether or not AEFSs can be learned from examples. To be more precise, we study the learnability of several language classes that are definable by AEFSs within the learning model introduced by GOLD [6]. As we shall see, the results obtained provide a firm basis to derive answers to the question to what extent, if ever, HTML wrappers can automatically be synthesized from examples.

Let us briefly review the necessary basic concepts. We refer the reader to [2] and [18] which contain all missing details concerning GOLD's [6] model.

There are several ways to present information about formal languages to be learned. The basic approaches are defined via the concept *text* and *informant*, respectively. A text is just any sequence of words exhausting the target language. An informant is any sequence of words labelled alternatively either by 1 or 0 such that all the words labelled by 1 form a text whereas the remaining words labelled by 0 constitute a text of the complement of the target language.

Now, an algorithmic learner (henceforth, called IIM) receives as its inputs larger and larger initial segments of a text t [an informant i] for a target language L and generates as its outputs hypotheses. In our setting, an IIM is supposed to generate AEFSs resp. EFSs as hypotheses. An IIM learns a target language L from text t [informant i], if the sequence of its outputs stabilizes on an AEFS which correctly describes L. Now, an IIM is said to learn L from text [informant], if it learns L from every text [every informant] for it. Furthermore, some language class $\mathcal{C}$ is said to be learnable from text [informant], if there is an IIM which learns every language $L \in \mathcal{C}$ from text [informant].

Next, we summarize the established learnability and non-learnability results.

Theorem 7

1. *$\mathcal{L}(vb\text{-}\mathcal{AE})$ is not learnable from informant.*
2. *$\mathcal{L}(lb\text{-}\mathcal{AE})$ is learnable from informant.*

Proof. By Theorem 4, $\mathcal{L}_{r.e.} \subseteq \mathcal{L}(vb\text{-}\mathcal{AE})$. Since there is no learning algorithm which is capable to learn the class $\mathcal{L}_r$ of all recursive languages from informant (cf. [6]), we obtain (1) because of $\mathcal{L}_r \subseteq \mathcal{L}_{r.e.}$. Furthermore, since $\mathcal{L}(lb\text{-}\mathcal{AE}) = \mathcal{L}_{cf}$, we know that $\mathcal{L}(lb\text{-}\mathcal{AE})$ constitutes an effectively enumerable class of recursive languages. Hence, we get (2), since every effectively enumerable class of recursive languages is learnable from informant (cf. [6]). □

Based on weaker information, if exclusively positive examples are available, only relatively small language classes turn out to be learnable at all. Next, for all $k \geq 1$, we let $lb\text{-}\mathcal{AE}^k$ ($lb\text{-}\mathcal{E}^k$) denote the collection of all AEFSs (EFSs) which consists of at most k rules.

Theorem 8

1. *For all $k \geq 2$, $\mathcal{L}(lb\text{-}\mathcal{AE}^k)$ is not learnable from text.*
2. *$\mathcal{L}(lb\text{-}\mathcal{AE}^1)$ is learnable from text.*

Proof. Since, by definition, $\mathcal{L}(lb\text{-}\mathcal{AE}^1) = \mathcal{L}(lb\text{-}\mathcal{E}^1)$, (2) is a special case of Assertion (2) in Theorem 9. It remains to verify (1).

Let $\Sigma = \{a\}$, let $L_0 = \{a\}^+$ and, for all $j \geq 1$, let $L_j = L_0 \setminus \{a^j\}$. It is folklore that there is no learning algorithm that is able to learn all languages in $\mathcal{C} = \{L_j \mid j \in \mathbb{N}\}$ from positive data (cf., e.g., [18], for the relevant details). It suffices to show that $\mathcal{C} \subseteq \mathcal{L}(lb\text{-}\mathcal{AE}^2)$. This can be seen as follows. In case of $j = 0$, let $S_0 = (\Sigma, \{p\}, \{\, p(x) \leftarrow \,\})$. Clearly, $L(S_0, p) = L_0$. Next, let $j \geq 1$. Then, $S_j = (\Sigma, \{p, q\}, \{\, q(a^j) \leftarrow,\ p(x) \leftarrow \mathit{not}\ q(x) \,\})$. Clearly, $L(S_j, p) = L_j$. □

For EFSs, the equivalent of Theorem 7 holds, as well. In contrast, Theorem 8 rewrites as follows.

Theorem 9 ([11])
1. *$\mathcal{L}(lb\text{-}\mathcal{E})$ is not learnable from text.*
2. *For all $k \geq 1$, $\mathcal{L}(lb\text{-}\mathcal{E}^k)$ is learnable from text.*

4 Wrappers for HTML Documents

Next, we demonstrate that AEFSs provide an appropriate framework to describe wrappers of practical relevance. Henceforth, the corresponding wrappers are called island wrappers. As a main result, we show under which assumptions learning techniques can be invoked to automatically generate island wrappers from positive examples.

Semi-structured documents carry different information and, moreover, the relevance of the information naturally depends on the users' perspective. As in most IE approaches, we assume that the content of a semi-structured document D (more formally, its semantics) is a set of tuples which D contains. For example, the tuples (`D. Angluin`, `80`), (`D. Angluin and C.H. Smith`, `83`), ... form relevant information in the list of references of the present paper (cf. Figure 1). Now, the aim of the IE task is to provide a wrapper which allows one to extract all tuples of this kind from this document as well as from all similar documents. In contrast to rather traditional approaches to IE in which it is the users' task to construct the relevant wrappers, we are interested in algorithms that automatically synthesize appropriate wrappers from examples.

4.1 Semantics of HTML Documents

Let $D \in \Sigma^+$ be a document. The information which D contains is a finite set of tuples of strings $(s_1, \ldots, s_n)$ where, as a rule, each of these strings must occur in D. Together with a tuple $(s_1, \ldots, s_n)$, it is important to know to which subword in D a string s_i is referring to. For instance, consider the tuple $(s_1, s_2) =$ (`B. Thomas`, `99`) in the list of references. Then, it might be intended that (s_1, s_2) has its origin either completely in reference [14] or completely in [15]. It is rather unlikely that s_1 belongs to [14] and that s_2 belongs to [15].

More formally speaking, the semantics of a document D is more than a set of tuples that describe the information contained. The semantics of D depends on an *interpretation* I which relates the strings in the tuples to subwords in D. More formally speaking, a function $S : \Sigma^+ \to \wp((\Sigma^+)^n)$ is a semantics iff there is an interpretation I such that for all $D \in \Sigma^+$ and for all $(s_1, \ldots, s_n) \in S(D)$ the Conditions (1) and (2) are fulfilled, where

(1) $I((s_1, \ldots, s_n), D)$ describes at which positions the s_i begin in D.
(2) s_{i+1} begins in D after s_i ends in D.

Intuitively speaking, if examples are provided that illustrate a certain semantics S, a learning algorithm is supposed to learn a wrapper that implements S, i.e., the wrapper must allow for the extraction of all tuples in $S(D)$ for any document D.

4.2 Island Wrappers

Island wrappers generalize the structure of the wrapper that has been designed to extract certain information from the HTML representation of the list of references (cf. Figure 2).

Island wrappers are AEFSs that consist of several basic length–bounded AEFSs describing anchor languages and of a couple of fixed top level rules that determine the interplay between these basic AEFSs (cf. Figure 3). To be more precise, consider an island wrapper that is designed to extract n-tuples from a given HTML document. For every extraction variable V_i, there have to be basic AEFSs S_{ℓ_i} and S_{r_i} that define the left and right anchor languages L_{ℓ_i} and L_{r_i} via the unary predicates $\mathtt{p}_{\ell_i}$ and $\mathtt{p}_{r_i}$, i.e., $L(S_{\ell_i}, \mathtt{p}_{\ell_i}) = L_{\ell_i}$ and $L(S_{r_i}, \mathtt{p}_{r_i}) = L_{r_i}$. As a rule, it is assumed that the set of predicate symbols used in the basic AEFSs S_{ℓ_i} and S_{r_i} have to be mutually disjoint. Now, intuitively, a string s_i can be substituted for the extraction variable V_i only in case that the actual document D contains a substring $u_i \circ s_i \circ w_i$ that meets $u_i \in L_{\ell_i}$ and $w_i \in L_{r_i}$. Thus, the anchor languages put constraints on the surroundings in which the relevant strings s_i are embedded in D. Moreover, as argued at the beginning of the last subsection, further minimality constraints are necessary. The strings substituted for the extraction variables must be as short as possible, while the distance between them has to be as small as possible. The top level rules needed are depicted in Figure 3. Let $\mathtt{w}$, $\mathtt{c\text{-}p}_{\ell_1}$, $\mathtt{nc\text{-}p}_{\ell_1}$, $\mathtt{c\text{-}p}_{r_1}$, $\mathtt{nc\text{-}p}_{r_1}$, ..., $\mathtt{c\text{-}p}_{\ell_n}$, $\mathtt{nc\text{-}p}_{\ell_n}$, $\mathtt{c\text{-}p}_{r_n}$, and $\mathtt{nc\text{-}p}_{r_n}$ be predicates not occurring in all the basic AEFSs S_{ℓ_i} and S_{r_i}.

$$
\begin{array}{l}
1\ \mathtt{w}(\mathtt{V}_1,\mathtt{V}_2,\ldots,\mathtt{V}_n,\mathtt{X}_1\mathtt{L}_1\mathtt{V}_1\mathtt{R}_1\mathtt{X}_2\mathtt{L}_2\mathtt{V}_2\mathtt{R}_2\cdots\mathtt{X}_n\mathtt{L}_n\mathtt{V}_n\mathtt{R}_n\mathtt{X}_{n+1}) \leftarrow \\
\qquad\qquad \mathtt{p}_{\ell_1}(\mathtt{L}_1),\ \mathtt{nc\text{-}p}_{r_1}(\mathtt{V}_1),\ \mathtt{p}_{r_1}(\mathtt{R}_1), \\
\qquad\qquad \mathtt{nc\text{-}p}_{\ell_2}(\mathtt{X}_2),\ \mathtt{p}_{\ell_2}(\mathtt{L}_2),\ \mathtt{nc\text{-}p}_{r_2}(\mathtt{V}_2),\ \mathtt{p}_{r_2}(\mathtt{R}_2), \\
\qquad\qquad \ldots \\
\qquad\qquad \mathtt{nc\text{-}p}_{\ell_n}(\mathtt{X}_n),\ \mathtt{p}_{\ell_n}(\mathtt{L}_n),\ \mathtt{nc\text{-}p}_{r_n}(\mathtt{V}_n),\ \mathtt{p}_{r_n}(\mathtt{R}_n). \\
2\ \mathtt{nc\text{-}p}_{\ell_1}(\mathtt{X}) \leftarrow \mathtt{not}\ \mathtt{c\text{-}p}_{\ell_1}(\mathtt{X}). \\
3\ \mathtt{c\text{-}p}_{\ell_1}(\mathtt{X}) \leftarrow \mathtt{p}_{\ell_1}(\mathtt{X}). \qquad 4\ \mathtt{c\text{-}p}_{\ell_1}(\mathtt{XY}) \leftarrow \mathtt{c\text{-}p}_{\ell_1}(\mathtt{X}). \qquad 5\ \mathtt{c\text{-}p}_{\ell_1}(\mathtt{XY}) \leftarrow \mathtt{c\text{-}p}_{\ell_1}(\mathtt{Y}). \\
\quad\vdots \\
6\ \mathtt{nc\text{-}p}_{r_n}(\mathtt{X}) \leftarrow \mathtt{not}\ \mathtt{c\text{-}p}_{r_n}(\mathtt{X}). \\
7\ \mathtt{c\text{-}p}_{r_n}(\mathtt{X}) \leftarrow \mathtt{p}_{r_n}(\mathtt{X}). \qquad 8\ \mathtt{c\text{-}p}_{r_n}(\mathtt{XY}) \leftarrow \mathtt{c\text{-}p}_{r_n}(\mathtt{X}). \qquad 9\ \mathtt{c\text{-}p}_{r_n}(\mathtt{XY}) \leftarrow \mathtt{c\text{-}p}_{r_n}(\mathtt{Y}).
\end{array}
$$

Figure 3.

Formally speaking, an island wrapper is an AEFS $S_{iw} = (\Sigma, \Pi, \Gamma)$, where Π is the collection of all predicates that occur either in rules in Figure 3 or in rules belonging to the basic AEFSs S_{ℓ_i} and S_{r_i} and Γ contains all rules in Figure 3 and all rules in the basic AEFSs S_{ℓ_i} and S_{r_i}. As a matter of fact, note that every S_{iw} is length–bounded. Hence, by Theorem 5, $\mathit{Sem}(S_{iw})$ is decidable. The latter makes island wrappers particularly tailored for IE.

In order to use an island wrapper S_{iw} to extract information from a document D, all the n-tuples $(s_1, \ldots, s_n)$ that meet $w(s_1, \ldots, s_n, D) \in Sem(S_{iw})$ have to be computed. Since all the s_i are substrings of D, this can be done effectively. An island wrapper S_{iw} defines a particular view at the document $D \in \Sigma^+$, namely $view(S_{iw}, D) = \{(s_1, \ldots, s_n) \mid w(s_1, \ldots, s_n, D) \in Sem(S_{iw})\}$. Furthermore, the island wrapper S_{iw} implements a particular semantics S iff $view(S_{iw}, D) = S(D)$.

Having the non-learnability results from Subsection 2.4 in mind its is unrealistic to assume that the class of all island wrappers is learnable from positive examples. For a better understanding of the principal power and limitations of the learning approach under consideration, we provide some finer look at island wrappers by putting some natural constraints on the admissible anchor languages. By $\mathcal{AE}_{iw}$ we denote the collection of all length-bounded island wrappers. For all $k \in \mathbb{N}$, $\mathcal{AE}_{iw}^k$ is the set of all island wrappers in $\mathcal{AE}_{iw}$ that are built upon anchor languages L with $card(L) \leq k$.

4.3 Learning Island Wrappers from Marked Text

Now, we are ready to study the question under which assumptions island wrappers can be learned from positive examples. As defined above, island wrappers differ in their anchor languages only. Hence, the overall learning task reduces to the problem to find AEFSs which describe the relevant anchor languages.

However, a potential user does not provide elements of the anchor languages to the system. Instead, the user marks interesting information in the HTML documents under inspection. To illustrate this, consider again the semi-structured representation of the list of references. In Figure 4, the information the user is interested in is underlined.

⟨html⟩⟨body⟩⟨h1⟩References⟨/h1⟩
⟨ol⟩⟨li⟩D. Angluin, 'Inductive inference of formal languages from positive data',
⟨i⟩Information and Control⟨/i⟩, ⟨b⟩45⟨/b⟩, 117-135, (1980).⟨/li⟩
⟨li⟩D. Angluin and C.H. Smith, 'A survey of inductive inference: Theory
and methods', ⟨i⟩Computing Surveys⟨/i⟩, ⟨b⟩15⟨/b⟩, 237–269, (1983).⟨/li⟩
⟨li⟩S. Arikawa, ...

Figure 4.

Clearly, the marked document provides only implicit information concerning the anchor languages to be learned. For instance, it can easily be deduced that the left anchor language L_{ℓ_1} contains a string that forms a suffix of the text segment *⟨html⟩⟨body⟩⟨h1⟩References⟨/h1⟩⟨ol⟩⟨li⟩* and a (possibly) different string that is a suffix of the text segment *.⟨/li⟩⟨li⟩*. Moreover, the right anchor language L_{r_2} must contain at least one string that is a prefix of the text segment *.⟨/li⟩⟨li⟩*.

To be precise, the learner does not receive a text for any of the relevant anchor languages as input. Therefore, the results from Subsection 2.4 do only translate indirectly – after an appropriate adaptation – to our setting of learning wrappers from *marked texts*.

Let S be a semantics under some fixed interpretation I. A marked text t for S is any sequence of pairs (D, P) fulfilling the following conditions:

(1) Each D is a document from Σ^+.
(2) P is a finite set of pairs (s, p), where $s = (s_1, \ldots, s_n)$ is an n-tuple in $S(D)$ and $p = (p_1, \ldots, p_n)$ describes where the s_i begin in D, i.e. $p = I(s, D)$.
(3) The sequence is exhaustive, i.e., for every document $D \in \Sigma^+$, every n-tuple in $S(D)$ eventually appears in t.

Now, a wrapper learner (henceforth called WIM) receives as input larger and larger initial segments of a marked text for some target semantics and generates as outputs AEFSs describing wrappers. A WIM is said to learn a semantics if the sequence of its outputs stabilizes on a wrapper which implements the target semantics. Finally, a WIM learns a class $\mathcal{C}$ of wrappers if it learns every semantics that is implementable by a wrapper in $\mathcal{C}$.

Our first result points out the general limitations of wrapper induction.

Theorem 10 *The class $\mathcal{AE}_{iw}$ is not learnable from marked text.*

Proof. Consider the following quite simple collection of island wrappers for extracting 2-tuples. Let $\Sigma = \{a, b\}$, let Γ be the set of rules depicted in Figure 3 for the case of $n = 2$, and let Π be the set of all predicates used in Γ. Now, we set $\Gamma_1 = \{p_{l_1}(a)\leftarrow\}$, $\Gamma_3 = \{p_{l_2}(a)\leftarrow\}$, and $\Gamma_4 = \{p_{r_2}(a)\leftarrow\}$. Additionally, for every $j \in \mathbb{N}$, we set $\Gamma_2^0 = \{p_{r_1}(a)\leftarrow, p_{r_1}(aX)\leftarrow p_{r_1}(X)\}$ as well as $\Gamma_2^{j+1} = \{p_{r_1}(a)\leftarrow, \ldots, p_{r_1}(a^{j+1})\leftarrow\}$.

Now, for every $j \in \mathbb{N}$, we let EFS $S_j = (\Sigma, \Pi, \Gamma \cup \Gamma_1 \cup \Gamma_2^j \cup \Gamma_3 \cup \Gamma_4)$. By definition, all island wrappers S_j do only differ in the left anchor language of the first extraction variable V_1.

Finally, we claim that the collection $\{S_j \mid j \in \mathbb{N}\}$ is not learnable from marked text. This can be shown by applying arguments similarly to those used in [6] for proving that superfinite language classes are not learnable from ordinary text. We omit the details. □

In contrast, if there is a uniform bound on the cardinality of the relevant anchor languages, learning becomes possible.

Theorem 11 *For all $k \geq 1$, the class $\mathcal{AE}_{iw}^k$ is learnable from marked text.*

Proof. Let $S_{iw} \in \mathcal{AE}_{iw}^k$ be an island wrapper to extract n-tuples and let $t = (D_1, P_1), (D_2, P_2), \ldots$ be a marked text for S_{iw} under some fixed interpretation. We claim that the following WIM M learns S_{iw} when successively fed t.

When learning an island wrapper from marked text, one may proceed as follows: In a first step, decompose the overall learning problem into several problems of learning anchor languages from ordinary text. In a second step, solve the derived individual learning problems independently and in parallel. In a concluding step, the solutions of the individual problems are combined to formulate a solution for the overall learning problem.

There are three different types of individual learning problems to attack. One problem consists in learning the left anchor language of the first extraction

variable V_1 (type A), another one in learning the right anchor language of the last variable V_n (type C). Moreover, there are $n-1$ learning problems of type B that consists in simultaneously learning the right anchor language of a variable V_i and the left anchor languages of a variable V_{i+1} $(1 \leq i \leq n-1)$.

WIM M: On input $(D_1, P_1), \ldots, (D_m, P_m)$ do the following.
Set $t^0, \ldots, t^n$ to be the empty sequences.
For each $j = 1, \ldots, m$, do the following:
For each pair $((s_1, \ldots, s_n), (p_1, \ldots, p_n))$ in P_j, compute (identified by the given starting positions $p_1, \ldots, p_n$ of $s_1, \ldots, s_n$) the uniquely determined substrings $w_0, w_1, \ldots, w_n$ of D_j such that $D_j = w_0 s_1 w_1 s_2 \ldots w_{n-1} s_n w_n$. Append w_0 to t^0, w_1 to t^1, ..., and w_n to t^n.
Let Γ be the rules depicted in Figure 3 and Π be the set of all predicates used to formulate the rules in Γ. Do in parallel:
- On input t^0, run M_A – an IIM for problems of type A. Fix $\Gamma' = M_A(t^0)$. Γ_0 is built by replacing, everywhere in Γ', the predicate p by p_{l_1}.
- For $i = 1, \ldots, n-1$, compute in parallel:
  On input t^i, run M_B – an IIM for problems of type B. Fix $\Gamma' = M_B(t^i)$. Γ_i is built by replacing, everywhere in Γ', the predicates p and q by p_{r_i} and $p_{l_{i+1}}$, respectively.
- On input t^n, run M_C – an IIM for problems of type C. Fix $\Gamma' = M_C(t^n)$. Γ_i is built by replacing, everywhere in Γ', the predicate p by p_{r_n}.

Output the EFS (Σ, Π, Γ') with $\Gamma' = \Gamma \cup \Gamma_0 \cup \Gamma_1 \cup \ldots \cup \Gamma_n$.

The IIM M_A for learning problems of type A is defined as follows.

IIM M_A: On input $S = u_0, \ldots, u_k$ do the following:
Set $\Gamma' = \emptyset$. Determine the set E of all non-empty suffixes of strings in S. For all strings $e \in E$ check whether or not, for all $a \in \Sigma$, $u = a \circ e$ for some $u \in S$. Let T be the set of all strings e passing this test. Goto $(\alpha 1)$.
$(\alpha 1)$ If $T = \emptyset$, output Γ'. Otherwise, goto $(\alpha 2)$.
$(\alpha 2)$ Determine a shortest string e in T. Set $\Gamma' = \Gamma' \cup \{p(e) \leftarrow\}$ and $T = T \setminus T_e$, where T_e contains all strings in T with the suffix e. Goto $(\alpha 1)$.

The IIM M_C can be obtained from M_A by replacing everywhere the term suffix by prefix. It is not hard to see that M_A and M_C behave as required.

It remains to define an IIM for learning problems of type B. In the definition of M_B, we let $\Gamma'' = \{nc\text{-}q(X) \leftarrow c\text{-}q(X), c\text{-}q(X) \leftarrow q(X), c\text{-}q(XY) \leftarrow c\text{-}q(X), c\text{-}q(XY) \leftarrow c\text{-}q(Y), r(XYZ) \leftarrow p(X), nc\text{-}q(Y), q(Z)\}$ and $\Pi'' = \{p, q, nc\text{-}q, c\text{-}q, r\}$.

IIM M_B: On input $S = u_0, \ldots, u_k$ do the following:
Let B and E be the set of all non-empty prefixes and suffixes of strings in S. Moreover, let $B' = \{p(b) \leftarrow \mid b \in B\}$ and $E' = \{q(e) \leftarrow \mid e \in E\}$.
Let H be the collection of all sets $h \subseteq B' \cup E'$ such that h contains at most k rules from B' and at most k rules from E'.
Search for an $h \in H$ such that, for every $u \in S$, $r(u) \in Sem(S)$ holds, where S is the EFS $(\Sigma, \Pi'', \Gamma'' \cup h)$. If such an h is found, let Γ' be the lexicographically first of them. Otherwise, set $\Gamma' = \emptyset$. Output Γ'.

The verification of M_B's correctness is a little more involved. Due to the space constraints, the details have to be omitted. □

5 Conclusions

The conventional concept of so-called Elementary Formal Systems has been generalized to Advanced Elementary Formal Systems (AEFS) which have been proven sufficiently expressive for representing certain wrappers of practical relevance. So-called island wrappers are appropriate for representing classes of HTML documents under a particular perspective. Island wrappers are characterized as length-bounded AEFSs.

The user is directed to http://LExIKON.dfki.de, where illustrative examples can be found which demonstrate the usefulness of the present approach.

Learnability of formal languages is known to be hard. Even island wrappers are not automatically learnable from examples only. This is throwing some light at the difficulties of invoking learning techniques for information extraction from the Internet.

The authors introduced additional constraints on the families of anchor languages which are induced by island wrappers and investigated the impact of these constraints on learnability. Anchor languages meeting such a constraint turn out to be learnable from positive examples drawn from given semi-structured documents. Thus, their corresponding island wrappers are learnable.

The results about representability and learnability above are setting the stage for some specific approach towards automated information extraction from the Internet. The basic scenario is as follows.

Given a sample Internet document, a user might have a particular view at the document and at the information contained therein which is relevant to him (her). By marking text passages from this document, the user is specifying this view in some detail. Marked text passages result in so-called marked text, which is a formal concept within the underlying theoretic setting. If sufficiently expressive examples have been marked, the intended view can be learned automatically. This is done by inductive inference of island wrappers which are particular AEFSs. Any learned island wrapper does not only allow to extract information from the documents it has been synthesized upon, but it also applies to a potentially unlimited number of further documents not seen so far.

The results of the present paper justify scenarios of this type and prove that information extraction through inductive learning does really work. Several problems are left to future investigations. Among these, there is the question for particularly efficient algorithms and the question for generalization to hierarchically structured document sources.

Finally, note that the representation formalism developed is well suited to describe other wrapper classes of practical relevance. For instance, the wrapper classes from [8] can easily be described and their learnability can be studied within this formal framework.

References

1. D. Angluin, 'Inductive inference of formal languages from positive data', *Information and Control*, **45**, 117–135, (1980).
2. D. Angluin and C.H. Smith, 'A survey of inductive inference: Theory and methods', *Computing Surveys*, **15**, 237–269, (1983).
3. S. Arikawa, S. Miyano, A. Shinohara, T. Shinohara, and A. Yamamoto, 'Algorithmic learning theory with elementary formal systems', *IEICE Trans. Inf. & Syst.*, **E75-D**(4), 405–414, (1992).
4. S. Arikawa, T. Shinohara, and A. Yamamoto, 'Elementary formal systems as a unifying framework for language learning', in *Proc. Second Int. Workshop on Computational Learning Theory*, pp. 312–327. Morgan Kaufmann, (1989).
5. S. Arikawa, T. Shinohara, and A. Yamamoto, 'Learning elementary formal systems', *Theoretical Computer Science*, **95**, 97–113, (1992).
6. M.E. Gold, 'Language identification in the limit', *Information and Control*, **14**, 447–474, (1967).
7. J.E. Hopcroft and J.D. Ullman, *Formal Languages and their Relation to Automata*, Addison-Wesley, (1969).
8. N. Kushmerick, *Wrapper Induction for Information Extraction*, Ph.D. thesis, University of Washington, (1997).
9. V. Lifschitz, 'Foundations of logic programming', in G. Brewka (ed.), *Principles of knowledge representation*, pp. 69–127, CSLI Publications, (1996).
10. S. Miyano, A. Shinohara, and T. Shinohara, 'Polynomial-time learning of elementary formal systems', *New Generation Computing*, **18**, 217–242, (2000).
11. T. Shinohara, 'Rich classes inferable from positive data: Length-bounded elementary formal systems', *Information and Computation*, **108**, 175–186, (1994).
12. R.M. Smullyan, *Theory of Formal Systems*, Annals of Mathematical Studies, No. 47, Princeton University, (1961).
13. S. Soderland, 'Learning information extraction rules from semi-structured and free text', *Machine Learning*, **34**, 233–272, (1999).
14. B. Thomas, 'Anti-unification based learning of T-Wrappers for information extraction', in *Proc. of AAAI Workshop on Machine Learning for IE*, pp. 15–20. AAAI, (1999).
15. B. Thomas, 'Logic programs for intelligent web search', in *Proc. of Int. Symposium on Methodologies for Intelligent Systems*, Lecture Notes in Artificial Intelligence 1609, pp. 190–198. Springer-Verlag, (1999).
16. A. Yamamoto, 'Elementary formal systems as a logic programming language', in *Proc. Logic Programming*, Lecture Notes in Artificial Intelligence 485, pp. 73–86. Springer-Verlag, (1989).
17. C. Zeng and S. Arikawa, 'Applying inverse resolution to EFS language learning', in *Proc. Int. Conference for Young Computer Scientists*, pp. 480–487. Int. Academic Publishers, (1999).
18. T. Zeugmann and S. Lange, 'A guided tour across the boundaries of learning recursive languages', in K.P. Jantke and S. Lange (eds), *Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence 961, pp. 190–258. Springer-Verlag, (1995).