

LARS: a Learning Algorithm for Rewriting Systems^{*}

Rémi Eyraud, Colin de la Higuera and Jean-Christophe Janodet

EURISE, Université Jean Monnet de Saint-Etienne,

23 rue Paul Michelon, 42023 Saint-Etienne, France,

Tel: 33 4 77 48 15 83, Fax: 33 4 77 48 15 84

e-mails: remi.eyraud,cdlh,janodet@univ-st-etienne.fr

Abstract. Whereas there is a number of methods and algorithms to learn regular languages, moving up the Chomsky hierarchy is proving to be a challenging task. Theoretical barriers make the class of context-free languages hard to learn. To tackle these barriers, we choose to change the way we represent these languages. Among the formalisms that allow the definition of classes of languages, the one of string-rewriting systems (SRS) has outstanding properties. Indeed, we introduce a new type of SRS's, called Delimited SRS (DSRS), that are expressive enough to define, in a uniform way, a noteworthy and non trivial class of languages that contains all the regular languages, $\{a^n b^n : n \geq 0\}$, $\{w \in \{a, b\}^* : |w|_a = |w|_b\}$, the parenthesis languages of Dyck, the language of Lukasiewicz, and many others. Moreover, DSRS's constitute an efficient (often linear) parsing device for strings, and are thus promising and challenging candidates in forthcoming applications of grammatical inference. In this paper, we pioneer the problem of their learnability. We propose a novel and sound algorithm (called LARS) which identifies a large subclass of them in polynomial time (but not data). We illustrate the execution of our algorithm through several examples, discuss the position of that class in the Chomsky hierarchy and finally raise some open questions and research directions.

Keywords: Learning Context-Free Languages, Rewriting Systems.

^{*} This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.



1. Introduction

Grammatical inference is concerned with finding some grammatical description of a language when given only examples of strings from this language, with perhaps some additional information about the structure of the strings, some counter-examples or the possibility of interrogating an oracle. Most work in grammatical inference has taken place in the area of learning regular grammars or finite state automata. Some positive learning results have been obtained, and there are now several good algorithms to deal with the case of learning from an informant (both positive and negative examples are provided) [35], or of learning a regular distribution from positive examples [8, 54].

On the other hand things tend to get harder when the larger class of context-free languages is considered [38, 49]. In that case there are known negative results showing the difficulty of the task: for instance characteristic samples (needed for identification) may not be of polynomial size [12] and it is believed that context-free grammars cannot be identified, in the framework of active learning, from a *minimum adequate teacher* consisting of equivalence and membership queries [3]. But as the problem of extracting some representation other than that which is provided by a regular grammar or an automaton is of crucial importance in a number of fields (natural language processing or computational biology, for instance) there have been several attempts to try to solve a relaxed version of the learning problem.

A first line of research has consisted of limiting the class of context-free grammars to be learned: even linear grammars [53], deterministic linear grammars [14], or very simple grammars [57] have been proved learnable.

If to the idea of simplifying the class of grammars we add that of using queries there are positive results concerning the class of simple deterministic languages. A language is simple deterministic when it can be recognized by a deterministic push-down automaton by empty store, that only uses one state. All languages in this class are thus necessarily deterministic, λ -free and prefix. Ishizaka [28] learns these languages using 2-standard forms: his algorithm makes use of membership queries and extended equivalence queries.

If one accepts the loss of theoretical proofs of convergence, then heuristics using genetic algorithms or artificial intelligence ideas [55, 23, 2, 45], or compression techniques [56, 42] have been proposed.

In the field of computational linguistics efforts have been made to learn context-free grammars from more informative data, such as trees [10] following theoretical results by Sakakibara [48]. Learning from structured data has been a line followed by many: learning tree

automata [31, 20, 26], or context-free grammars from bracketed data [47] allows one to obtain better results, either with queries [48], regular distributions [33, 9, 46], or negative information [22]. This has also led to different studies concerning the probability estimation of such grammars [37, 6].

In 2003 there has been renewed interest in the topic [13]: the OM-PHALOS context-free language learning competition [52] was launched, where state of the art techniques were unable to solve even the easiest tasks. The method [11] that obtained best results used a variety of information about the distributions of the symbols, substitution graphs and context. The approach is mainly empirical and does not provide a convergence theorem.

The progress made contrasts with the theoretical barriers: most theoretical results are negative and show that the entire class of context-free languages is hard to learn in a variety of settings [38, 49, 15].

An attractive alternative when blocked by negative results is to change the representation mode. In this line, little work has been done for the context-free case: one exception is *pure context-free grammars*, which are grammars where both the non-terminals and the terminals come from the same alphabet [32, 19].

In this paper, we investigate string-rewriting systems (SRS's) as an alternative way to describe and manipulate context-free languages. Invented in 1914 by Axel Thue, the theory of SRS's (also called semi-Thue systems) and its extension to trees and to graphs was paid a lot of attention throughout the 20th century (see [5, 17]). Rewriting a string consists of replacing substrings by others, as far as possible, following laws called rewrite rules. For instance, consider strings made of a and b , and the single rewrite rule $ab \mapsto \lambda$. Using this rule consists in replacing some substring ab by the empty string, thus in erasing ab . It allows $abaabbab$ to rewrite as follows:

$$abaabbab \mapsto abaabb \mapsto abab \mapsto ab \mapsto \lambda$$

Other rewriting derivations may be considered but they all lead to λ . Actually, it is rather clear on this example that a string will rewrite to λ *iff* it is a “parenthetic” string, *i.e.*, a string from the Dyck language. More precisely, the Dyck language is completely characterized by this single rewrite rule and the string λ , which is reached by rewriting all other strings of the language. This property was first noticed in a seminal paper by Nivat [43], which was the starting point of a large amount of work during the last three decades. We use this property, and others to introduce a class of rewriting systems that is powerful enough to represent in an economical way all regular languages and some typical context-free languages: $\{a^n b^n : n \geq 0\}, \{w \in \{a, b\}^* : |w|_a = |w|_b\}$, the

parenthesis languages of Dyck, the language of Lukasiewicz, and many others. We also provide a learning algorithm called LARS (Learning Algorithm for Rewriting Systems) that can learn systems representing a subclass of these languages from string examples and counter-examples of the language.

In section 2 we give the general notations relative to the languages we consider and discuss the notion of learning. We introduce our rewriting systems and their expressiveness in section 3 and develop the properties they must fulfill to be learnable in section 4. The general learning algorithm is presented in section 5 and justified in section 6. We report in section 7 some experimental results and conclude.

2. Learning Languages

An *alphabet* Σ is a finite nonempty set of symbols called *letters*. A *string* w over Σ is a finite sequence $w = a_1a_2 \dots a_n$ of letters. Let $|w|$ denote the length of w and $|w|_x$ the number of occurrences of letter x in w . In the following, letters will be indicated by a, b, c, \dots , strings by u, v, \dots, z , and the empty string by λ . Let Σ^* be the set of all strings. We assume a fixed but arbitrary total order $<$ on the letters of Σ . As usual, we extend $<$ to Σ^* by defining the *hierarchical order* [44], denoted by \triangleleft , as follows:

$$\forall w_1, w_2 \in \Sigma^*, w_1 \triangleleft w_2 \text{ iff } \begin{cases} |w_1| < |w_2| \text{ or} \\ |w_1| = |w_2| \text{ and } \exists u, v_1, v_2 \in \Sigma^*, \exists x_1, x_2 \in \Sigma \\ \text{s.t. } w_1 = ux_1v_1, w_2 = ux_2v_2 \text{ and } x_1 < x_2. \end{cases}$$

\triangleleft is a total strict order over Σ^* , and if $\Sigma = \{a, b\}$ and $a < b$, then $\lambda \triangleleft a \triangleleft b \triangleleft aa \triangleleft ab \triangleleft ba \triangleleft bb \triangleleft aaa \triangleleft \dots$

By a language L over Σ we mean every subset $L \subseteq \Sigma^*$. Many classes of languages were investigated in the literature. In general, the definition of a class \mathbb{L} relies on a class \mathbb{R} of abstract machines, here called *representations*, that characterize all and only the languages of \mathbb{L} . The relationship is given by the *naming function* $\mathcal{L} : \mathbb{R} \rightarrow \mathbb{L}$ such that: (1) $\forall R \in \mathbb{R}, \mathcal{L}(R) \in \mathbb{L}$ and (2) $\forall L \in \mathbb{L}, \exists R \in \mathbb{R}$ such that $\mathcal{L}(R) = L$. Two representations R_1 and R_2 are *equivalent* iff $\mathcal{L}(R_1) = \mathcal{L}(R_2)$. In this paper, we will investigate the class REG of *regular* languages characterized by the class DFA of *deterministic finite automata* (dfa), and the class CFL of *context-free languages* represented by the class CFG of *context-free grammars* (cfg).

A deterministic finite automaton (dfa) is a quintuple $A = \langle \Sigma, Q, q_0, F, \delta \rangle$ where Q is a finite set of *states*, $q_0 \in Q$ is an *initial state*, $F \subseteq Q$ is a set of *accepting states* and $\delta : Q \times \Sigma \rightarrow Q$ is a *transition function*.

The *language recognized by* A is $\mathcal{L}(A) = \{w \in \Sigma^* : \delta(q_0, w) \in F\}$, where δ denotes the extended transition function defined over $Q \times \Sigma^*$. We say that a language is *regular* if there exists a *dfa* that recognizes it. Let us remember that given a *dfa* A , one can compute efficiently an equivalent *dfa* B that is minimal in the number of states.

A context-free grammar (*cfg*) is a quadruple $G = \langle \Sigma, V, P, S \rangle$ where Σ is a finite alphabet of *terminal symbols*, V is a finite alphabet of *variables* or *non-terminals*, $P \subseteq V \times (\Sigma \cup V)^*$ is a finite set of *production rules*, and $S \in V$ is the *axiom* (start symbol). We will denote $uTv \Rightarrow uvw$ when $(T, w) \in P$. \Rightarrow^* is the reflexive and transitive closure of \Rightarrow . If there exist u_0, \dots, u_k such that $\forall i, 0 \leq i < k, u_i \Rightarrow u_{i+1}$ we will write $u_0 \xRightarrow{k} u_k$. We denote by $\mathcal{L}(G)$ the language $\{w \in \Sigma^* : S \Rightarrow^* w\}$.

We now turn to our learning problem. The *size* of a representation R , denoted by $\|R\|$, is polynomially related to the size of its encoding.

DEFINITION 1. *Let \mathbb{L} be a class of languages represented by some class \mathbb{R} of representations.*

1. *A sample S for a language $L \in \mathbb{L}$ is a pair $S = \langle S_+, S_- \rangle$ of two finite sets $S_+, S_- \subseteq \Sigma^*$ such that if $w \in S_+$ then $w \in L$ and if $w \in S_-$ then $w \notin L$. The size of S is the sum of the lengths of all the strings in S_+ and S_- .*
2. *An (\mathbb{L}, \mathbb{R}) -learning algorithm \mathfrak{A} is a program that takes as input a sample and outputs a representation from \mathbb{R} .*

Finally, let us discuss what “learning” means. Obviously extracting some consistent grammar is insufficient and therefore some type of convergence towards an ideal result is wanted. The convergence can be statistical (which leads to PAC-learnability or similar definitions) or not if we make no assumption about the way the data is obtained. We choose to base ourselves on the paradigm of polynomial identification, as defined in [24, 12], since several authors showed that it was both relevant and tractable for grammatical inference problems.

In this paradigm we first demand that the learning algorithm has a running time polynomial in the size of the data from which it has to learn from. Next we want the algorithm to converge in some way to a chosen target. Ideally the convergence point should be met very quickly, after having seen a polynomial number of examples only. As this constraint is usually too hard, we want the convergence to take place in the limit, *i.e.*, after having seen a finite number of examples. The polynomial aspects are then taken into account of by the size of a minimal *learning* or *characteristic* sample, whose presence should ensure identification. For more details on these models we refer the reader to [24, 12]. This yields the following definition:

DEFINITION 2. (Polynomial Identification). *A class \mathbb{L} of languages is identifiable in polynomial time and data for a class \mathbb{R} of representations if and only if there exist an algorithm \mathfrak{A} and two polynomials $\alpha()$ and $\beta()$ such that:*

1. *Given a sample $S = \langle S_+, S_- \rangle$ for $L \in \mathbb{L}$ of size m , \mathfrak{A} returns a hypothesis $H \in \mathbb{R}$ in $\mathcal{O}(\alpha(m))$ time and H is consistent with S ;*
2. *For each representation R of size k of a language $L \in \mathbb{L}$, there exists a finite characteristic sample $CS = \langle CS_+, CS_- \rangle$ of size at most $\mathcal{O}(\beta(k))$ such that, on all samples $S = \langle S_+, S_- \rangle$ for L that verify $CS_+ \subseteq S_+$ and $CS_- \subseteq S_-$, \mathfrak{A} returns a hypothesis $H \in \mathbb{R}$ which is equivalent to R .*

3. Defining Languages with String-Rewriting Systems

String-rewriting systems are usually defined as sets of rewrite rules. These rules replace substrings by others in strings. However, as we feel that this mechanism is not flexible enough, we would like to extend it. Indeed, a rule that one would like to use at the beginning or at the end of a string could also be used in the middle of this string and then have undesirable side effects.

Therefore, we introduce two new symbols $\$$ and \mathcal{L} that do not belong to the alphabet Σ and will respectively mark the beginning and the end of each string. In other words, we are going to consider strings from the set $\$\Sigma^*\mathcal{L}$. As for the rewrite rules, they will be *partially* marked, and thus belong to $\overline{\Sigma^*} = (\lambda + \$)\Sigma^*(\lambda + \mathcal{L})$. Their forms will constrain their use either to the beginning, or to the end, or to the middle, or even to the string taken as a whole. Notice that this solution is more permissive than the usual (undelimited) approaches but more restrictive than the string-rewriting systems with variables introduced in [39].

DEFINITION 3. (Delimited SRS).

– *A rewrite rule R is an ordered pair of strings $R = (l, r)$, generally written $R = l \vdash r$. l is called the left-hand side of R and r its right-hand side.*

– *We say that $R = l \vdash r$ is a delimited rewrite rule iff l and r satisfy one of the four following constraints:*

1. $l, r \in \$\Sigma^*$ (used to rewrite prefixes) or
2. $l, r \in \$\Sigma^*\mathcal{L}$ (used to rewrite whole strings) or

3. $l, r \in \Sigma^*$ (used to rewrite substrings) or
4. $l, r \in \Sigma^* \mathcal{L}$ (used to rewrite suffixes).

Rules of type 1 and 2 will be called $\$$ -rules and rules of type 3 and 4 will be called non- $\$$ -rules.

—By a delimited string-rewriting system (DSRS), we mean any finite set \mathcal{R} of delimited rewrite rules.

Let $|\mathcal{R}|$ be the number of rules of \mathcal{R} and $\|\mathcal{R}\|$ the sum of the lengths of the strings \mathcal{R} is defined by: $\|\mathcal{R}\| = \sum_{(l \vdash r) \in \mathcal{R}} |lr|$.

Given a DSRS \mathcal{R} and two strings $w_1, w_2 \in \overline{\Sigma}^*$, we say that w_1 rewrites in one step into w_2 , written $w_1 \vdash_{\mathcal{R}} w_2$ or simply $w_1 \vdash w_2$, iff there exist a rule $(l \vdash r) \in \mathcal{R}$ and two strings $u, v \in \overline{\Sigma}^*$ such that $w_1 = ulv$ and $w_2 = urv$. A string w is *reducible* iff there exists w' such that $w \vdash w'$, and *irreducible* otherwise. E.g., the string $\$aabb\mathcal{L}$ is rewritten to $\$aaa\mathcal{L}$ with rule $bb\mathcal{L} \vdash a\mathcal{L}$.

We immediately get the following property that states that $\$$ and \mathcal{L} cannot appear, nor move nor disappear in a string by rewriting:

PROPOSITION 1. *The set $\$\Sigma^* \mathcal{L}$ is stable w.r.t. $\vdash_{\mathcal{R}}$, i.e., if $w_1 \in \$\Sigma^* \mathcal{L}$ and $w_1 \vdash_{\mathcal{R}} w_2$, then $w_2 \in \$\Sigma^* \mathcal{L}$.*

Let $\vdash_{\mathcal{R}}^*$ (or simply \vdash^*) denote the reflexive and transitive closure of $\vdash_{\mathcal{R}}$. We say that w_1 reduces to w_2 or that w_2 is derivable from w_1 iff $w_1 \vdash_{\mathcal{R}}^* w_2$.

DEFINITION 4. (Language Induced by a DSRS). *Given a DSRS \mathcal{R} and an irreducible string $e \in \Sigma^*$, we define the language $\mathcal{L}(\mathcal{R}, e)$ as the set of strings that reduce to e using the rules of \mathcal{R} :*

$$\mathcal{L}(\mathcal{R}, e) = \{w \in \Sigma^* : \$w\mathcal{L} \vdash_{\mathcal{R}}^* \$e\mathcal{L}\}.$$

Deciding whether a string w belongs to a language $\mathcal{L}(\mathcal{R}, e)$ or not consists in trying to obtain e from w by a rewriting derivation. However, w may be the starting point of numerous derivations, thus such a task may not be tractable. We will tackle these problems in the next section but present some examples first.

Example. Let $\Sigma = \{a, b\}$.

- $\mathcal{L}(\{ab \vdash \lambda\}, \lambda)$ is the Dyck language. Indeed, since this single rule erases substring ab , we get the following example of a derivation:

$$\$aabbab\mathcal{L} \vdash \$aabb\mathcal{L} \vdash \$ab\mathcal{L} \vdash \$\mathcal{L}$$

- $\mathcal{L}(\{ab \vdash \lambda; ba \vdash \lambda\}, \lambda)$ is the language $\{w \in \Sigma^* : |w|_a = |w|_b\}$, because every rewriting step erases one a and one b .
- $\mathcal{L}(\{aabb \vdash ab; \$ab\mathcal{L} \vdash \$\mathcal{L}\}, \lambda) = \{a^n b^n : n \geq 0\}$. For instance,

$$\$aaaabbbb\mathcal{L} \vdash \$aaabbb\mathcal{L} \vdash \$aabb\mathcal{L} \vdash \$ab\mathcal{L} \vdash \$\mathcal{L}$$

Notice that the rule $\$ab\mathcal{L} \vdash \\mathcal{L} is necessary for deriving λ (last derivation step).

- $\mathcal{L}(\{\$ab \vdash \$\}, \lambda)$ is the regular language $(ab)^*$. Indeed,

$$\$_{ababab}\mathcal{L} \vdash \$_{abab}\mathcal{L} \vdash \$_{ab}\mathcal{L} \vdash \$\mathcal{L}$$

Actually, we will see that all regular languages can be induced by a DSRS.

4. On the Expected Properties of the DSRS's

The aim of this section is to examine the properties that we should demand in order to manipulate *tractable* DSRS's. Two (usual) properties are particularly interesting: the termination property (Subsec.4.1) and the confluence property (Subsec.4.2). Finally, we will see that both do not restrict too much the expressivity of the DSRS's *w.r.t.* the language they induce (Subsec.4.3).

4.1. THE TERMINATION PROPERTY

As already mentioned, a string w belongs to a language $\mathcal{L}(\mathcal{R}, e)$ *iff* one can build a derivation from w to e . However this definition is too loose and raises many difficulties. Firstly, it is easy to imagine a DSRS such that a string can be rewritten indefinitely. *E.g.*, the DSRS $\mathcal{R} = \{a \vdash b; b \vdash a; c \vdash cc\}$ induces the following derivations :

$$\begin{aligned} aa \vdash ba \vdash aa \vdash ba \vdash aa \dots \\ aca \vdash acca \vdash accca \vdash acccca \vdash accccca \vdash \dots \end{aligned}$$

Actually, the termination of SRS is undecidable in general. The decidability of termination on subclasses of string-rewriting systems is still a research topic (see [40] for instance). Of course, in the context of the DSRS, there is no reason to believe that the problem is simpler than in the general setting.

On the other hand, although all the derivations induced by a DSRS are finite, they could be of exponential lengths and thus computationally intractable. Indeed, consider the following DSRS:

$$\mathcal{R} = \left\{ \begin{array}{ll} 1\mathcal{L} \vdash 0\mathcal{L}, & 0\mathcal{L} \vdash c1d\mathcal{L}, \\ 0c \vdash c1, & 1c \vdash 0d, \\ d1 \vdash 1d, & dd \vdash \lambda \end{array} \right\}$$

All the derivations induced by \mathcal{R} are finite. Indeed, assuming that $d > 1 > 0 > c$, the left-hand side l is lexicographically greater than the right-hand side r for all rules $l \vdash r$, so this DSRS is strongly normalizing [17]. However, if one uses it to rewrite $\underbrace{11\dots 1}_n \mathcal{L} = 1^n \mathcal{L}$

into $0^n \mathcal{L}$, then all encodings of non-negative integers between $2^n - 1$ and 0 will be encountered at least once, so the corresponding derivation will be of exponential length. *E.g.*, when $n = 4$, we get:

$$\begin{aligned} & \$1111\mathcal{L} \\ \vdash & \$1110\mathcal{L} \\ \vdash & \$11\underline{c}1d\mathcal{L} \vdash \$110\underline{d}1d\mathcal{L} \vdash \$1101\underline{dd}\mathcal{L} \vdash \$1101\mathcal{L} \\ \vdash & \$1100\mathcal{L} \\ \vdash & \$110\underline{c}1d\mathcal{L} \vdash \$11\underline{c}11d\mathcal{L} \vdash \$10\underline{d}11d\mathcal{L} \vdash \$101\underline{d}1d\mathcal{L} \vdash \$1011\underline{dd}\mathcal{L} \vdash \$1011\mathcal{L} \\ \vdash & \$1010\mathcal{L} \\ \vdash^* & \$1001\mathcal{L} \\ \vdash & \$1000\mathcal{L} \\ \vdash^* & \$0111\mathcal{L} \dots \\ \vdash^* & \$0000\mathcal{L}. \end{aligned}$$

In order to tackle these problems, we first extend the hierarchical order \triangleleft to the strings of Σ^* , by defining the *extended hierarchical order*, denoted \prec , as follows:

$$\forall w_1, w_2 \in \Sigma^*, \text{ if } w_1 \triangleleft w_2 \text{ then } w_1 \prec \$w_1 \prec w_1\mathcal{L} \prec \$w_1\mathcal{L} \prec w_2.$$

Therefore, if $a < b$, then $\lambda \triangleleft a \triangleleft b \triangleleft aa \triangleleft ab \triangleleft ba \triangleleft bb \triangleleft aaa \triangleleft \dots$, so $\lambda \prec \$ \prec \mathcal{L} \prec \$\mathcal{L} \prec a \prec \$a \prec a\mathcal{L} \prec \$a\mathcal{L} \prec b \prec \dots$. Notice that \prec conveys $\overline{\Sigma^*}$ the structure of a well-ordered set, that is to say every subset of $\overline{\Sigma^*}$ has a minimum.

The following technical definition ensures that all the rewriting derivations induced by a DSRS are finite and tractable in polynomial time.

DEFINITION 5. (Hybrid DSRS). *We say that a rule $R = l \vdash r$ is hybrid iff*

1. *R is a $\$$ -rule (i.e., $l, r \in \$\Sigma^*(\lambda + \mathcal{L})$) and is length-lexicographic: $r \prec l$, or*

2. R is a non-\$-rule (i.e., $l, r \in \Sigma^*(\lambda + \mathcal{L})$) and is length-reducing:
 $|r| < |l|$.

A DSRS \mathcal{R} is hybrid iff all its rules are hybrid.

For instance, the rules $aa \vdash a$ and $\$ba \vdash \ab are hybrid but $ba \vdash ab$ is not (since it is a non-\$-rule that is not length-reducing). Notice that hybridness is a syntactic property on each rule of a DSRS, so checking whether a DSRS is hybrid or not is straightforward.

THEOREM 1. *All the derivations induced by a hybrid DSRS \mathcal{R} are finite. Moreover, every derivation starting from a string w has a length that is at most $|w| \cdot |\mathcal{R}|$.*

Proof. Let $w_1 \vdash w_2$ be a single rewriting step. There exist a rule $l \vdash r$ and two strings $u, v \in \overline{\Sigma}^*$ such that $w_1 = ulv$ and $w_2 = urv$. Notice that if $|l| > |r|$ then $l \succ r$. Moreover, if $l \succ r$, then we deduce that $w_1 \succ w_2$. So if one considers any derivation $u_0 \vdash u_1 \vdash u_2 \vdash \dots$, then $u_0 \succ u_1 \succ u_2 \succ \dots$. As $\overline{\Sigma}^*$ is a well-ordered set, there is no infinite and strictly decreasing chain of the form $u_0 \succ u_1 \succ u_2 \succ \dots$. So every derivation induced by \mathcal{R} is finite. Now let $n \geq 0$. Assume that for all strings w' such that $|w'| < n$, the lengths of the derivations starting from w' are at most $|w'| \cdot |\mathcal{R}|$. Let w be a string of length n . We claim that the maximum length of a derivation that would preserve the length of w cannot exceed $|\mathcal{R}|$ rewriting steps. Indeed, all rules that can be used along such a derivation are of the form $\$l \vdash \r , with $|l| = |r|$ and $l \succ r$; when such a rule is used once, then it cannot be used a second time in the same derivation. Otherwise, there would exist a derivation $\$lu\mathcal{L} \vdash \$ru\mathcal{L} \vdash \dots \vdash \$lv\mathcal{L}$ with $|u| = |v|$ (since the length is preserved). As $\$ru\mathcal{L} \vdash^* \$lv\mathcal{L}$ and $|l| = |r|$ and $|u| = |v|$, we deduce that $r \succeq l$ which is impossible since $r \prec l$. So there are at most $|\mathcal{R}|$ rewriting steps that preserve the length of w , and then the application of a rule produces a string w' whose length is $< n$. So by the induction hypothesis, the length of a derivation starting from w is no more than $|\mathcal{R}| + |w'| \cdot |\mathcal{R}| \leq |w| \cdot |\mathcal{R}|$. \square

4.2. THE CHURCH-ROSSER PROPERTY

A hybrid DSRS induces finite and tractable derivations. Nevertheless, many different irreducible strings may be reached from one given string by rewriting. Therefore, answering the problem “ $w \in \mathcal{L}(\mathcal{R}, e)$?” requires computing *all* the derivations that start with w and checking if one of them ends with e . In other words, such a DSRS is a kind of “nondeterministic” (thus inefficient) parsing device. A usual way to

circumvent this difficulty is to impose our hybrid DSRS's to also be Church-Rosser [17].

DEFINITION 6. (Church-Rosser DSRS). *We say that a DSRS \mathcal{R} is Church-Rosser iff for all strings $w, u_1, u_2 \in \overline{\Sigma}^*$ such that $w \vdash^* u_1$ and $w \vdash^* u_2$, there exists $w' \in \overline{\Sigma}^*$ such that $u_1 \vdash^* w'$ and $u_2 \vdash^* w'$ (see Fig. 1).*

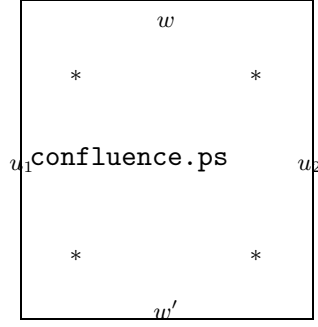


Figure 1. The Church-Rosser property is also called the Diamond property.

In the definition above, if $w \vdash^* u_1$ and $w \vdash^* u_2$ and u_1 and u_2 are irreducible strings, then $u_1 = u_2 (= w')$. So given a string w , there is no more than *one* irreducible string that can be reached by a derivation starting with w , whatever the derivation is considered. However, the Church-Rosser property is undecidable in general, so we constrain our DSRS's to fulfill a restrictive condition:

DEFINITION 7. (ANo DSRS).

– Two (non necessarily distinct) rules $R_1 = l_1 \vdash r_1$ and $R_2 = l_2 \vdash r_2$ are almost nonoverlapping (ANo) iff:

1. Either $l_1 = l_2$ and then $r_1 = r_2$;
2. Or l_1 is strictly included in l_2 (see Fig. 2 (a)):

$$\exists u, v \in \overline{\Sigma}^*, ul_1v = l_2, uv \neq \lambda,$$

and then $ur_1v = r_2$;

3. Or l_2 is strictly included in l_1 :

$$\exists u, v \in \overline{\Sigma}^*, l_1 = ul_2v, uv \neq \lambda,$$

and then $r_1 = ur_2v$;

4. Or a strict suffix of l_1 is a strict prefix of l_2 (see Fig. 2 (b)):

$$\exists u, v \in \overline{\Sigma}^*, l_1 u = v l_2, 0 < |v| < |l_1|,$$

and then $r_1 u = v r_2$;

5. Or a strict suffix of l_2 is a strict prefix of l_1 :

$$\exists u, v \in \overline{\Sigma}^*, u l_1 = l_2 v, 0 < |v| < |l_1|,$$

and then $u r_1 = r_2 v$;

6. Or none of the previous cases occurs.

– We say that a DSRS \mathcal{R} is almost nonoverlapping (ANo) iff its rules are pairwise almost nonoverlapping.



Figure 2. Two rules overlap when their left-hand sides overlap; part (a) of the diagram above shows Case 2 (and symmetrically, Case 4) of the previous definition; part (b) illustrates Case 3 (and symmetrically, Case 5).

Less formally, a system is ANo if whenever two rules that overlap can be applied on a string w , they immediately rewrite w into the same string.

Example.

- The rules $R_1 = ab \vdash \lambda$ and $R_2 = ba \vdash \lambda$ are almost nonoverlapping since aba reduces to a with both rules and bab reduces to b with both rules.
- On the contrary, the rules $R_3 = ab \vdash a$ and $R_4 = ba \vdash a$ are not ANo; indeed, aba rewrites to aa with both rules but bab reduces to ba with R_3 and to ab with R_4 .
- The single rule $R_5 = aa \vdash b$ forms a non ANo DSRS since aaa can be rewritten into ab and ba by using R_5 .

We get the following result:

THEOREM 2. *Every ANo DSRS is Church-Rosser.*

Proof. Let us show that an ANo DSRS \mathcal{R} induces a rewriting relation $\vdash_{\mathcal{R}}$ that is *subcommutative*: let us write $w_1 \vdash_{\varepsilon} w_2$ iff $w_1 \vdash_{\mathcal{R}} w_2$ or $w_1 = w_2$; we claim that for all w, u_1, u_2 , if $w \vdash_{\mathcal{R}} u_1$ and $w \vdash_{\mathcal{R}} u_2$, then there exists a string w' such that $u_1 \vdash_{\varepsilon} w'$ and $u_2 \vdash_{\varepsilon} w'$ [30]. Indeed, assume that $w \vdash_{\mathcal{R}} u_1$ uses a rule $R_1 = l_1 \vdash r_1$ and $w \vdash_{\mathcal{R}} u_2$ uses a rule $R_2 = l_2 \vdash r_2$. If both rewriting steps are independent, i.e., $w = xl_1yl_2z$ for some strings x, y, z , then $u_1 = xr_1yl_2z$ and $u_2 = xl_1yr_2z$; obviously, $u_1 \vdash_{\mathcal{R}} w'$ and $u_2 \vdash_{\mathcal{R}} w'$ with $w' = xr_1yr_2z$. Otherwise, R_1 overlaps R_2 (or vice-versa), and so $u_1 = u_2$, since \mathcal{R} is ANo. By an easy induction one can generalize this property to derivations: if $w \vdash_{\mathcal{R}}^* u_1$ and $w \vdash_{\mathcal{R}}^* u_2$ then there exists w' such that $u_1 \vdash_{\varepsilon}^* w'$ and $u_2 \vdash_{\varepsilon}^* w'$, where \vdash_{ε}^* is the reflexive and transitive closure of \vdash_{ε} . Finally, as $u_1 \vdash_{\varepsilon}^* w'$ and $u_2 \vdash_{\varepsilon}^* w'$, we deduce that $u_1 \vdash_{\mathcal{R}}^* w'$ and $u_2 \vdash_{\mathcal{R}}^* w'$. \square

4.3. ON THE HYBRID ANO DSRS'S

In the rest of this paper, we will now consider only hybrid ANo DSRS's, which allows us to get the following properties:

1. For any string w , there is no more than *one* irreducible string that can be reached by a derivation which starts with w , whatever derivation is considered. This irreducible string will be called the *normal form* of w and denoted w_{\downarrow} (or $w_{\downarrow\mathcal{R}}$ whenever there is an ambiguity on the rewriting system \mathcal{R}).
2. No derivation can be prolonged indefinitely, so every string w has at least one normal form. And whatever the way a string w is reduced, the rewriting steps produce strings that are ineluctably closer and closer to w_{\downarrow} .

An important consequence is that one has an immediate algorithm to check whether $w \in \mathcal{L}(\mathcal{R}, e)$ or not: one only needs to (i) compute the normal form w_{\downarrow} of w and (ii) check if w_{\downarrow} and e are *syntactically* equal. As all the derivations have polynomial lengths, this algorithm is polynomial in time.

Last but not least, notice that all the DSRS's we used as examples at the end of Section 3, that is to say $\langle \{ab \vdash \lambda\}, \lambda \rangle$, $\langle \{ab \vdash \lambda; ba \vdash \lambda\}, \lambda \rangle$, $\langle \{aabb \vdash ab; \$ab\mathcal{L} \vdash \$\mathcal{L}\}, \lambda \rangle$ and $\langle \{\$ab \vdash \$\}, \lambda \rangle$ satisfy the hybrid and ANo constraints. Therefore such conditions are not too restrictive. In particular, the last DSRS induces the regular language $(ab)^*$ and the following result shows that *all* regular languages can be described with a hybrid ANo DSRS:

THEOREM 3. *For each regular language L , there exist a hybrid ANo DSRS \mathcal{R} and a string e such that $L = \mathcal{L}(\mathcal{R}, e)$.*

Proof. One way to prove this result would consist in establishing the equivalence between 1) the DSRS's that are only made of $\$$ -rules and 2) the *prefix grammars* [21], since it is known that such grammars generate all and only the regular languages. Below, we provide a direct proof, by using the characterization of regular languages through automata. Let $A = \langle \Sigma, Q, q_0, F, \delta \rangle$ be the minimal *dfa* of L . For all states $q \in Q$, we define the minimum string w_q that allows q to be reached by parsing, *i.e.*, w_q is the string of Σ^* such that $\delta(q_0, w_q) = q$ and $\forall w' \triangleleft w_q, \delta(q_0, w') \neq q$. Let e be the minimum string that reaches a final state (*w.r.t.* \triangleleft), *i.e.*, $e = \min_{q \in F} w_q$. Let $\mathcal{R}_1 = \{ \$w_q x \vdash \$w_{q'} : q, q' \in Q, x \in \Sigma, \delta(q, x) = q', w_{q'} \neq w_q x \}$ and $\mathcal{R}_2 = \{ \$w_q \mathcal{L} \vdash \$e \mathcal{L} : q \in F, w_q \neq e \}$ and $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$. It is clear that $\mathcal{L}(\mathcal{R}, e) = L$ since $\forall w \in \Sigma^*, \delta(q_0, w) = q \iff \$w \vdash^* \$w_q$ (by induction). An example of this construction is given at the end of the proof.

We claim that \mathcal{R} is a hybrid DSRS, *i.e.*, the $\$$ -rules of \mathcal{R} are all length-lexicographic. On the one hand, if there exists a rule $\$w_q x \vdash \$w_{q'}$ in \mathcal{R}_1 then $\delta(q_0, w_q x) = w_{q'}$; as $w_{q'}$ is the minimum string that reaches q' and $\delta(q_0, w_q x) = q'$, we get $\$w_{q'} \prec \$w_q x$. On the other hand, as $e = \min_{q \in F} w_q$, all the rules of \mathcal{R}_2 are length-lexicographic.

Finally we claim that \mathcal{R} is an ANo DSRS. Indeed, consider two rules $R_1 = \$w_q x \vdash \w_p and $R_2 = \$w_{q'} y \vdash \$w_{p'}$ of \mathcal{R}_1 . The only possible situation of overlapping is the one where the left-hand side of one rule, say R_1 , contains the left-hand side of the other, say R_2 , that is to say, there exists $m \in \Sigma^*$ such that $w_{q'} y m x = w_q x$. There are two cases:

1. Either $m = \lambda$ and then $w_{q'} y = w_q$, that yields $\delta(q', y) = q$. But, by the definition of R_2 , $\delta(q', y) = p'$ and then $p' = q$ (since the *dfa* is deterministic), so $w_{p'} = w_q$. Therefore, rule R_2 is $\$w_{q'} y \vdash \w_q with $w_{q'} y = w_q$, that is in contradiction with the definition of the rules.
2. Or $m \neq \lambda$ and then $w_{q'} y m = w_q$. We deduce that $q = \delta(q_0, w_q) = \delta(q_0, w_{q'} y m) = \delta(q', y m) = \delta(p', m) = \delta(q_0, w_{p'} m)$. However, the definition of R_2 yields $\$w_{p'} \prec \$w_{q'} y$ that implies $\$w_{p'} m \prec \$w_{q'} y m = \$w_q$. Therefore, we get $\delta(q_0, w_{p'} m) = q$ and $\$w_{p'} m \prec \w_q , that is impossible since w_q is the minimum string that reaches state q .

Concerning the rules of \mathcal{R}_2 , none of them can overlap another rule of \mathcal{R}_2 since their left-hand sides are delimited with both $\$$ and \mathcal{L} . So the only possible case is that of an overlapping between a rule of \mathcal{R}_1 and a rule of \mathcal{R}_2 . However, the reader may check that this case is also impossible for the same reason as in Case 2 above. \square

Example. Consider the minimal *dfa* A that recognizes the language $(a+b)^*a(a+b)$. A has four states $\{0, 1, 2, 3\}$, 0 is initial, 2 and 3 are final, and $\delta(0, b) = \delta(3, b) = 0$, $\delta(0, a) = \delta(3, a) = 1$, $\delta(1, a) = \delta(2, a) = 2$, $\delta(1, b) = \delta(2, b) = 3$. Therefore, $w_0 = \lambda$, $w_1 = a$, $w_2 = aa$ and $w_3 = ab$. So we get $e = aa$, $\mathcal{R}_1 = \{\$b \vdash \$; \$aaa \vdash \$aa; \$aab \vdash \$ab; \$aba \vdash \$a; \$abb \vdash \$\}$ and $\mathcal{R}_2 = \{\$ab\mathcal{L} \vdash \$aa\mathcal{L}\}$. One can easily check that $\mathcal{R}_1 \cup \mathcal{R}_2$ is a hybrid ANo DSRS.

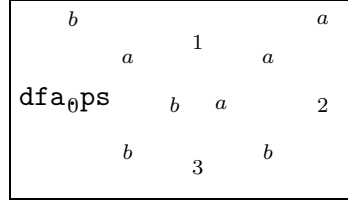



Figure 3. The minimal *dfa* that recognizes the language $(a+b)^*a(a+b)$.

5. Algorithm

In this section we present our learning algorithm and its properties. The idea is to enumerate the rules following the order \preceq . We discard those that are useless or inconsistent *w.r.t.* the data, and those that break the ANo condition.

The first thing LARS does is to compute all the substrings of S_+ and to sort them *w.r.t.* \preceq . Left and right-hand sides of the rules will be chosen in this set. This assumption reduces dramatically the search space without challenging LARS's learning capacities. Then LARS enumerates the elements of this set thanks to two **for** loops, which build the candidate rules.

Function **does_appear** verifies that the candidate left-hand side is a substring of at least one string in the current I_+ . This precaution allows to discard rules that cannot be used to rewrite at least one string of I_+ , thus rules that seem to have no effect but that could be incorrect. Function **is_DSRS** (resp. **is_hybrid**, **is_ANo**) checks if the candidate rule is syntactically correct according to Def.3 (resp. Def.5, Def.7). The last thing to check is that the rule is consistent with the data, *i.e.*, that it does not produce a string belonging to both I_+ and I_- . This is easily done by computing the normal forms of the strings of I_+ and I_- , which is the aim of function **normalize** (*i.e.*, $\text{normalize}(X, \mathcal{S}) = \{w \downarrow_{\mathcal{S}} : w \in X\}$).



algo.eps

Figure 4. The pseudo-code of LARS.

Before running LARS on an example, we establish the following theorem:

THEOREM 4. *Given a sample $S = \langle S_+, S_- \rangle$ (with $S_+ \cap S_- = \emptyset$) of size m , algorithm LARS returns a hybrid ANo DSRS \mathcal{R} and an irreducible string e such that $S_+ \subseteq \mathcal{L}(\mathcal{R}, e)$ and $S_- \cap \mathcal{L}(\mathcal{R}, e) = \emptyset$. Moreover, its execution time is a polynomial of m .*

Proof. The termination and polynomiality of LARS are straightforward (because the number of substrings in F is polynomial in the number of positive examples). Moreover, the following four invariant properties are maintained all along the double “for” loops: (1) \mathcal{R} is a hybrid ANo DSRS, (2) I_+ contains all and only the normal forms of the strings of S_+ w.r.t. \mathcal{R} , (3) I_- contains all and only the normal forms of the strings of S_- w.r.t. \mathcal{R} and (4) $I_+ \cap I_- = \emptyset$. Clearly, these properties remain true before the “foreach” loop. The rules inferred during the “foreach” loop are delimited by both a $\$$ and a \mathcal{L} and so they are not pairwise overlapping; moreover, as their left-hand sides are in normal form w.r.t. \mathcal{R} , they are not overlapped by any rule of \mathcal{R} . So, at the end of the last “foreach” loop, it is clear that \mathcal{R} is a hybrid ANo DSRS. Moreover, (1) e is the normal form of all the strings of S_+ , so $S_+ \subseteq \mathcal{L}(\mathcal{R}, e)$ and (2) the normal forms of the strings of S_- are all in I_- and $e \notin I_-$, so $S_- \cap \mathcal{L}(\mathcal{R}, e) = \emptyset$. \square

To clarify the way the algorithm behaves, we run it on a toy example (see the summary in Table I). Suppose that LARS is fed with $S_+ = \{\$b\mathcal{L}, \$abb\mathcal{L}, \$abaabbb\mathcal{L}, \$ababb\mathcal{L}\}$, $S_- = \{\$ \lambda \mathcal{L}, \$a\mathcal{L}, \$ab\mathcal{L}, \$aa\mathcal{L}, \$baab\mathcal{L}, \$bab\mathcal{L}, \$aab\mathcal{L}, \$abab\mathcal{L}, \$aabb\mathcal{L}\}$. This sample corresponds to the context-free language of Lukasiewicz that can be described by the grammar $\langle \{a, b\}, \{S\}, P, S \rangle$ with $P = \{S \rightarrow aSS; S \rightarrow b\}$. The first step consists in building the sorted set F of substrings of I_+ .

LARS starts with the substring λ , but no hybrid rule can be built using this left-hand side. For the same reason the substrings $\$$ and \mathcal{L} are discarded. As for string $\$ \mathcal{L}$, it does not appear in F .

So, the first relevant substring that LARS deals with is a . It appears in I_+ and the only rule that can be made from it is $a \vdash \lambda$. This rule is rejected as, for instance, the string $\$ab\mathcal{L}$ is reduced to $\$b\mathcal{L}$ which then belongs to both E_+ and E_- .

The next substring is $\$a$ and can be used only in the rule $\$a \vdash \lambda$ that is rejected because it generates the same inconsistency as the previous one.

As the substrings $a\mathcal{L}$ and $\$a\mathcal{L}$ do not appear in F , the next one is b . The first rule that is built from this substring is $b \vdash \lambda$. It is rejected because the positive example $\$b\mathcal{L}$ is then reduced to a negative one, $\$ \lambda \mathcal{L}$. $b \vdash a$ is length-lexicographic but not a $\$$ -rule, and is thus not hybrid.

The substrings $\$b, b\mathcal{L}, \$b\mathcal{L}$ appear in I_+ but the rules that can be made from them generate a non empty intersection of E_+ and E_- (the same examples previously described can be used to show their inconsistency).

Then LARS looks for rules made with aa as left-hand side. The rule $aa \vdash \lambda$ reduces the negative example $\$aab\mathcal{L}$ into $\$b\mathcal{L}$ that is a positive one. Similarly, the rule $aa \vdash a$ (resp. $aa \vdash b$) reduces $\$aabb\mathcal{L}$ into $\$abb\mathcal{L}$ (resp. $\$aa\mathcal{L}$ into $\$b\mathcal{L}$) and so they are discarded.

The next substring of F that appear in I_+ is ab . The rule $ab \vdash \lambda$ cannot be accepted as it reduces, for example, the negative string $\$bab\mathcal{L}$ into the positive one $\$b\mathcal{L}$. For the same reason, rule $ab \vdash a$ (which reduces both the positive example $\$abb\mathcal{L}$ and the negative one $\$ab\mathcal{L}$ into $\$a\mathcal{L}$) and rule $ab \vdash b$ ($\$ab\mathcal{L}$ belongs to I_- and $\$b\mathcal{L}$ to I_+) are discarded. $ab \vdash aa$ is rejected because it is a non- $\$$ -rule that is not length-reducing (thus the system would not be hybrid).

We then consider the substring $\$ab$ that appears in I_+ . The rule $\$ab \vdash \lambda$ is accepted. The normalization process gives $I_+ = \{\$b\mathcal{L}, \$abaabbb\mathcal{L}\}$ and $I_- = \{\$ \lambda \mathcal{L}, \$a\mathcal{L}, \$aa\mathcal{L}, \$baab\mathcal{L}, \$bab\mathcal{L}, \$aab\mathcal{L}, \$aabb\mathcal{L}\}$. The rule is then added to \mathcal{R} that becomes $\{\$ab \vdash \lambda\}$.

Table I. Summary of an execution of LARS. The first column contains the substrings in F : they are vertically ordered as F is sorted. The second column contains the output of the function **does_appear** on the current substring and the current set I_+ . The third column contains the current rule and the fourth one the output of the evaluation of the functions **is_hybrid** and **is_ANo**. The fifth column contains the result of the test “ $E_+ \cap E_- = \emptyset$?”. The last two columns correspond to the content of the sets I_+ and I_- . The two inferred rules are written in bold.

$\mathbf{F}[i]$	appear?	current rule	hybrid and ANo?	$E_+ \cap E_- = \emptyset$?	I_+	I_-
a	yes	$a \vdash \lambda$	yes	no	S_+	S_-
\$a	yes	$\$a \vdash \$$	yes	no		
b	yes	$b \vdash \lambda$ $b \vdash a$	yes yes	no no		
\$b	yes	$\$b \vdash \$$ $\$b \vdash \a	yes yes	no no		
$b\mathcal{L}$	yes	$b\mathcal{L} \vdash \mathcal{L}$ $b\mathcal{L} \vdash a\mathcal{L}$	yes yes	no no		
$\$b\mathcal{L}$	yes	$\$b\mathcal{L} \vdash \\mathcal{L} $\$b\mathcal{L} \vdash \$a\mathcal{L}$	yes yes	no no		
aa	yes	$aa \vdash \lambda$ $aa \vdash a$ $aa \vdash b$	yes yes no	no no -		
\$aa	yes	$\$aa \vdash \$$ $\$aa \vdash \a $\$aa \vdash \b	yes yes yes	no no no		
ab	yes	$ab \vdash \lambda$ $ab \vdash a$ $ab \vdash b$ $ab \vdash aa$	yes yes yes no	no no no -		
\$ab	yes	$\\$ab \vdash \\$	yes	yes	$\{\$b\mathcal{L}, \$aabb\mathcal{L}\}$	$\{\$ \mathcal{L}, \$a\mathcal{L}, \$aa\mathcal{L}, \$baab\mathcal{L}, \$bab\mathcal{L}, \$aab\mathcal{L}, \$aabb\mathcal{L}\}$
ba	no	-	-	-		
bb	yes	$bb \vdash \lambda$ $bb \vdash a$ $bb \vdash b$ $bb \vdash aa$ $bb \vdash ab$ $bb \vdash ba$	yes no yes no no no	no - no - - -		
aab	yes	$aab \vdash \lambda$ $aab \vdash a$	yes yes	no yes	$\{\$b\mathcal{L}\}$	$\{\$ \mathcal{L}, \$a\mathcal{L}, \$aa\mathcal{L}, \$ba\mathcal{L}\}$

All the other possible rules made with $\$ab$ as left-hand side are rejected because they cannot generate an ANo DSRS: their left-hand sides are equal to that of the rule of \mathcal{R} but not their right-hand sides.

The next substring in F is ba but it does not appear in I_+ anymore (because of the normalization process).

The substring bb can still be found in I_+ . But no rule can be induced from it as it would break the ANo condition. Indeed, suppose that we are looking for a rule $bb \vdash r$, for some r . The substring $\$abb$ can then be reduced into $\$ar$ and also into $\$b$ using the only rule of \mathcal{R} . Both these strings can definitely not be equal, so the ANo condition is broken by every rule whose left-hand side is bb .

The next substring is aab that still appears in I_+ . The rule $aab \vdash \lambda$ is discarded whereas the rule $aab \vdash a$ satisfies the needed conditions and is consistent, so the latter is accepted. The normalization process yields $I_+ = \{\$b\mathcal{L}\}$ and $I_- = \{\$ \lambda \mathcal{L}, \$a\mathcal{L}, \$aa\mathcal{L}, \$ba\mathcal{L}\}$.

As there is only one string in I_+ , LARS does not infer any new rule, and thus ends and outputs $\langle \{\$ab \vdash \$ \lambda, aab \vdash a\}, \$b\mathcal{L} \rangle$. Although it is not immediate, the reader may check that this system does induce the expected language.

The above example of an execution of the algorithm LARS is summarized in Table I. Notice that, as no hybrid rule can be constructed with $\lambda, \$, \mathcal{L}$ or $\$ \mathcal{L}$ as left handside, we have not put them in the first column of the table (they correspond to the elements of indices one to four in the sorted tabular F).

6. Learning Hybrid ANo DSRS's

In this section, we study the languages that LARS can learn. On the one hand, we provide an identification result for a restricted class of languages, those that may be defined thanks to *closed DSRS's*. On the other hand, we show that LARS is able to learn a richer class and address the position of this class in the Chomsky hierarchy.

6.1. AN IDENTIFICATION RESULT

LARS is a greedy algorithm that infers a DSRS incrementally. However, it does not consider *every* hybrid rule. Indeed, a candidate rule is built from the substrings of the positive sample only, so it necessarily rewrites at least one string of the target language; we will say that such a rule is *applicable* to the language. Moreover, LARS keeps a rule only if it does not generate a contradiction between the positive and negative examples; we will say that such a rule is *consistent w.r.t.* the language.

DEFINITION 8. (Applicable and Consistent Rule). *Let $L \subseteq \Sigma^*$ be a language and $R = l \vdash r$ a hybrid rule. We say that:*

– R is applicable to L iff there exist $w \in \$L\mathcal{L}$ and $u, v \in \overline{\Sigma}^$ such that $w = ulv$.*

– R is consistent w.r.t. L iff $\forall u, v \in \overline{\Sigma}^, (ulv \in \$L\mathcal{L} \iff urv \in \$L\mathcal{L})$.*

E.g., with respect to $L = \{a^n b^n : n \geq 0\}$, the rule $bba \vdash ba$ is not applicable since no string of $\$L\mathcal{L}$ contains bba as a substring. Actually, describing L by using such a rule is not relevant. On the other hand, the rule $ab \vdash a$ is not consistent w.r.t. L since it rewrites $\$aabb\mathcal{L} \in \$L\mathcal{L}$ into $\$aab\mathcal{L} \notin \$L\mathcal{L}$. Actually, a rule is consistent w.r.t. L if $\$L\mathcal{L}$ and $(\overline{\Sigma}^ \setminus \$L\mathcal{L})$ are both stable by rewriting with this rule.*

All the hybrid rules that could be used to describe a language L are necessarily consistent w.r.t. L (and should be applicable to L). But such systems would probably not be ANo, so LARS would not learn them. This is the reason why we introduce the following definition:

DEFINITION 9. (Closed DSRS). *Let $L = \mathcal{L}(\mathcal{R}, e)$ be a language and R_{max} the greatest¹ rule of \mathcal{R} w.r.t. \preceq . We say that \mathcal{R} is closed iff*

1. \mathcal{R} is hybrid and ANo, and

2. for any hybrid rule R , if (i) $R \preceq R_{max}$ and (ii) R is applicable to L and (iii) R is consistent w.r.t. L , then $R \in \mathcal{R}$.

We say that a language is closed if it can be induced by a closed DSRS.

Given a hybrid ANo DSRS, it is probably not possible to decide whether this system is closed or not; the problem comes from the consistency property of a rule that seems to be undecidable. Beyond these drawbacks, the closedness property yields the following result:

THEOREM 5. *Given a language $L = \mathcal{L}(\mathcal{T}, e)$ such that \mathcal{T} is closed, there exists a finite characteristic sample $CS = \langle CS_+, CS_- \rangle$ such that, on $S = \langle S_+, S_- \rangle$ with $CS_+ \subseteq S_+$ and $CS_- \subseteq S_-$, algorithm LARS finds e and returns a hybrid ANo DSRS \mathcal{R} such that $\mathcal{L}(\mathcal{R}, e) = \mathcal{L}(\mathcal{T}, e)$.*

Notice that the polynomiality of the characteristic samples is not established.

We first define the characteristic sample for a closed language:

¹ \preceq is basically extended to ordered pairs of strings, thus to rules, as follows: $\forall u_1, u_2, v_1, v_2 \in \overline{\Sigma}^*, (u_1, u_2) \preceq (v_1, v_2)$ iff $u_1 \prec v_1$ or $(u_1 = v_1$ and $u_2 \preceq v_2)$.

DEFINITION 10. Let $L = \mathcal{L}(\mathcal{T}, e)$ be the target language. \mathcal{T} is assumed closed. We define the characteristic sample $CS = \langle CS_+, CS_- \rangle$ as follows:

1. $\$e\mathcal{L} \in CS_+$.
2. For any rule $R = l \vdash r \in \mathcal{T}$, there exist two strings $ulv, u'rv' \in \$L\mathcal{L} \cap CS_+$ for some $u, v, u', v' \in \overline{\Sigma}^*$.
3. For any hybrid rule $R = l \vdash r$ such that $R \preceq R_{max}$ and $R \notin \mathcal{T}$, (R is not consistent since \mathcal{T} is closed), there exists $u, v \in \overline{\Sigma}^*$ such that $ulv \in (\Sigma^*\mathcal{L} \setminus \$L\mathcal{L}) \cap CS_-$ and $urv \in \$L\mathcal{L} \cap CS_+$.
4. For any $\mathcal{R} \subseteq \mathcal{T}$ and all $R = l \vdash r \in \mathcal{T}$ such that $\mathcal{L}(\mathcal{R}, e) \neq L$ but $\mathcal{L}(\mathcal{R} \cup \{R\}, e) = L$, there exists $w \in \$L\mathcal{L} \setminus \$\mathcal{L}(\mathcal{R}, e)\mathcal{L}$ such that $w = ulv$, w is in normal form w.r.t. \mathcal{R} and $w \in CS_+$.

Before the proof of Theorem 5, we discuss informally the definition of the characteristic sample above. The two first items ensure that LARS has all the elements it needs in the sample: the smallest string in the language and all the left and right hand sides of the rules of the target are in the positive set of the characteristic sample. As the algorithm checks only the rules whose left and right hand sides appear in the positive examples of the learning sample, this is a necessary condition for identification. The third item concerns the hybrid rules that are not consistent w.r.t the target language. For such a rule $l \vdash r$, we need two strings ulv and urv in the characteristic sample, one positive, one negative, such that LARS rejects the rule (the intersection between the set of normalized positive examples and the one of normalized negative examples is then not empty). Although consistency is an equivalence property, we only need the rule to rewrite either a positive example in a negative one, or a negative example in a positive one, to reject it. The first solution may have an undesirable side effect: it may unnecessarily increase the size of the set of substrings F LARS works on. We then choose to put a negative example ulv and a positive example urv in the characteristic sample. The last item is more technical: its goal is to prevent LARS to erase, during a normalization step of the evaluation, the left hand side of a needed rule, that is to say a rule such that without it the target language cannot be induced. Notice that the definition (and the following proof) show the existence of a characteristic sample, but not the constructivity of it: the last item requires the comparison of the languages induced by two different hybrid ANo DSRS's, which, as far as we know, seems to be undecidable. This is not as crucial as the reader may think: in their seminar paper [25], which is the major starting point of identification in the limit with the use of a

characteristic sample, Goldman and Mathias define the “teachability” as the existence of a characteristic sample (later in the article, they call these problems “semi-poly teachable” in comparison to “teachable” problems where the characteristic sample is effectively constructible).

Proof. Let $L = \mathcal{L}(\mathcal{T}, e)$ be the target language. \mathcal{T} is assumed closed and let R_{\max} denotes the greatest rule of \mathcal{T} w.r.t. \preceq .

We now prove that if $S_+ \supseteq CS_+$ and $S_- \supseteq CS_-$ then LARS returns a correct system. By construction of the characteristic set, F contains all the left and right-hand sides of the rules of the target (Case 2 above). Assume now that LARS has been running during a certain number of steps. Let \mathcal{R} be the current hybrid ANo DSRS. By the closedness of \mathcal{T} , $\mathcal{R} \subseteq \mathcal{T}$.

Let $R = l \vdash r$ be the next rule to be checked (i.e. l still appear in I_+). We assume that $\mathcal{R} \cup \{R\}$ is hybrid ANo. Otherwise LARS discards it. Notice that this is not a problem since if $\mathcal{R} \cup \{R\}$ is not hybrid ANo, then $\mathcal{T} \cup \{R\}$ cannot be hybrid ANo, so R cannot belong to \mathcal{T} (by the closedness property). There are two cases:

1. If R is inconsistent, then by Case 3 of the characteristic sample, there exist $m \in (\$ \Sigma^* \mathcal{L} \setminus \$ L \mathcal{L}) \cap CS_-$ and $m' \in \$ L \mathcal{L} \cap CS_+$ such that $m \vdash_R m'$. By the definition of I_+ and I_- , we get $u = m \downarrow_{\mathcal{R}} \in I_-$ and $u' = m' \downarrow_{\mathcal{R}} \in I_+$. As $\mathcal{R} \cup \{R\}$ is ANo, it is Church-Rosser, so there exists a string z such that $u \vdash_{\mathcal{R} \cup \{R\}} z$ and $u' \vdash_{\mathcal{R} \cup \{R\}} z$. Therefore $z \in E_+ \cap E_-$, thus LARS discards R .
2. If R is consistent, then consider the system $\mathcal{S} = \mathcal{R} \cup \{T \in \mathcal{T} : R \prec T\}$. Notice that \mathcal{S} is made by the rules of \mathcal{T} except the rule R and the consistent rules that could have belonged to \mathcal{R} but were discarded because they appeared to be useless when LARS considered them. There are two subcases:
 - a) $\mathcal{L}(\mathcal{S}, e) = L$ and thus the rule R is not needed to get L . In this case, LARS adds R to \mathcal{R} since there is no way to reject it. (But notice that this rule could also be rejected with no harm.)
 - b) $\mathcal{L}(\mathcal{S}, e) \neq L$ and $\mathcal{L}(\mathcal{S} \cup \{R\}, e) = L$. Then, by Case 4 of the characteristic sample, there is a string w in CS_+ , that is in normal form w.r.t. \mathcal{S} . As $\mathcal{R} \subseteq \mathcal{S}$, it is clear that $w \in I_+$. As w can be rewritten with R , R can be used at least once on a positive string. So LARS adds R to \mathcal{R} .

At the end of the execution of LARS, the current DSRS \mathcal{R} contains all the rules of \mathcal{T} except those whose left-hand sides did not appear in I_+ when LARS considered them. Nevertheless, such rules were not needed to identify L since otherwise, Case 4 of the characteristic sample would

have ensured that their left-hand side appears in I_+ . Finally, notice that e is the only string that remains in I_+ , and so LARS returns the pair $\langle \mathcal{R}, e \rangle$ that does satisfy $\mathcal{L}(\mathcal{R}, e) = L$. \square

6.2. LARS AND THE CHOMSKY HIERARCHY

The results of this section are given in Fig. 5. The following remarks can be made.

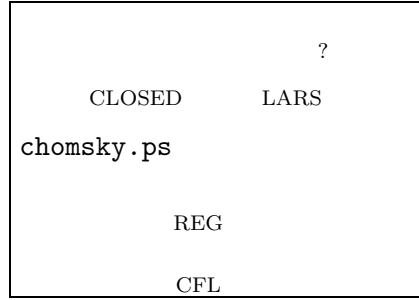


Figure 5. LARS and the languages in the Chomsky hierarchy. LARS denotes the class of languages learnable by Algorithm 1 (LARS)

First of all, we have seen that all closed languages are identifiable with LARS. Although checking whether a DSRS is closed or not seems to be undecidable, this can be done by hand on particular DSRS's. For instance, it is easy to check that $\langle \{ab \vdash \lambda, ba \vdash \lambda\}, \lambda \rangle$ that induces the context-free language $\{w \in \{a, b\}^* : |w|_a = |w|_b\}$ is closed. In addition, one can also check that all the context-free languages described in Section 7.2 admit a closed DSRS. So the closedness property is not too restrictive.

Secondly, LARS can also learn languages that are not closed because it is able to infer DSRS's that are not closed. For instance, consider $\langle \mathcal{R}, ab \rangle$ with $\mathcal{R} = \{\$b \vdash \$, a\mathcal{L} \vdash \mathcal{L}, abab \vdash ab\}$, that induces the regular language $L_0 = b^*(ab)^+a^*$. The rule $R = aa\mathcal{L} \vdash \mathcal{L}$ is consistent and applicable to L_0 ; moreover, $R \prec (abab \vdash ab)$ and $\mathcal{R} \cup \{R\}$ is not ANo; so \mathcal{R} is not closed. However, LARS finds \mathcal{R} ! Actually the rule R erases a 's at the end of the strings, that is also done by the rule $a\mathcal{L} \vdash \mathcal{L} \in \mathcal{R}$, so R is not needed to induce L_0 . Notice that all DSRS's that induce L_0 need a rule that deals with the substrings made of ab 's and so, whatever is their greatest rule R_{\max} , $R' = a\mathcal{L} \vdash \mathcal{L}$ and R are smaller than R_{\max} . Therefore, there exists no closed DSRS that induces L_0 (as R and R' are both consistent and applicable but not ANo). Hence, this

example shows that the class of closed languages is strictly contained in the class of languages that LARS is able to learn.

Concerning the regular languages, we have shown in Theorem 3 that they were all induced with at least one hybrid ANo DSRS. Unfortunately, LARS is not able to learn all of them. For example, consider the language $L_1 = \{w \in \{a, b\}^* : |w|_a \bmod 3 = |w|_b \bmod 3\}$ (see Fig. 6). L_1 is induced by $e = \lambda$ and, for instance, one of the following hybrid

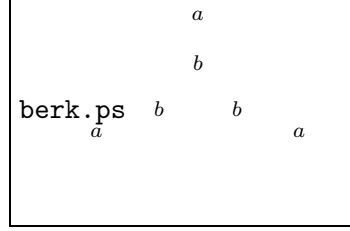


Figure 6. The minimal dfa of $\{w \in \{a, b\}^* : |w|_a \bmod 3 = |w|_b \bmod 3\}$.

DSRS's:

$$\begin{aligned}\mathcal{R}_1 &= \{aa \vdash b; ab \vdash \lambda; ba \vdash \lambda; bb \vdash a\} \\ \mathcal{R}_2 &= \{\$aa \vdash \$b; \$ab \vdash \$; \$ba \vdash \$; \$bb \vdash \$a\}\end{aligned}$$

Notice that \mathcal{R}_2 is ANo ; as for \mathcal{R}_1 , it is Church-Rosser but not ANo. Notice that L_1 is not a closed languages because the rule $aa \vdash b$ is the smallest consistent and applicable rule but is not ANo. On this language, by using a large amount of examples that should allow LARS to learn it, the algorithm begins to infer the following system:

$$\mathcal{R} = \{\$aa \vdash \$b; \$ab \vdash \$; ba \vdash \lambda\}.$$

\mathcal{R} is a hybrid ANo DSRS made of rules coming from \mathcal{R}_1 and \mathcal{R}_2 . Moreover, \mathcal{R} does not induce L , in particular because a lot of bb 's still appear as substrings of the sample. However, there is definitely no rule that would erase them and could be added to \mathcal{R} without breaking the ANo condition, so no super hybrid ANo DSRS of \mathcal{R} induces L . In other words, L is a regular language that LARS cannot learn.

As a consequence, it is clear that every context-free language is not learnable either. However, we will see in the following section that LARS is still able to identify a lot of important context-free languages, including some non-linear ones. At last but not least, we guess that LARS is not able to learn any pure context-sensitive language, simply because such languages can probably not be described with DSRS's.

7. Experimental Results

An implementation of LARS in Java is available online at the URL <http://eurise.univ-st-etienne.fr/~eyraud/LARS>: different fields must be filled with positive and negative examples, and then LARS can be run on them. The result of the evaluation is then given on the webpage.

7.1. ABOUT GRAMMATICAL INFERENCE EXPERIMENTS

Experiments in grammatical inference can be separated into three families:

- Identification experiments on large-scale competition-type benchmarks;
- Learning experiments on real-world data;
- Identification experiments on small (toy) examples.

The first case includes well known benchmarks based on the ABBADINGO, OMPHALOS competitions [36, 52]. Other large benchmarks have been made available by de Oliveira [16] or the GOWACHIN system [34]. Sizes of the alphabets, the grammars, the training sets can vary, but correspond to *limit / state of the art* situations; in the case of the ABBADINGO competition some of the problems have not been solved seven years after the end of the competition! In all these cases the idea has been to get the researchers to push their algorithms as far as possible in order to obtain the best possible classification rates. In doing so other issues may have been forgotten such as the correctness of the algorithms (do they converge?) or their intelligibility (in order to get better results some fine tuning often becomes necessary).

In the second case the situation has not changed that much since the seventies: if there is no target grammar to be exactly identified, we are in a situation of noisy learning, which to date cannot be solved by grammatical inference with deterministic methods. Statistical methods perform better, but there are many other questions that arise from the choice of learning a distribution instead of a language. And if a language is sought for, then learning a stochastic automaton or grammar, or another type of language model fails to solve the problem as it is addressed.

So when a new algorithm is presented it is reasonable to show (not prove!) that positive results can be obtained and that in the proposed setting (learning with a teacher or with the help of a characteristic sample implies that positive results depend on the fact that specific

pieces of data are present). On the other hand most new ideas in grammatical inference are presented through experiments on smaller and more manageable benchmarks. The experiments do not in that case have the same meaning: the point is to show what is happening, not to infer that on some random data set the algorithms perform well.

We have chosen to test LARS mainly in this context. We have experimented with toy examples only: these correspond nevertheless to languages that have been considered hard by different authors [41, 50].

Nevertheless, the system has also been tested on the OMPHALOS competition training sets and the results have been bad. There are two explanations for this: on one hand LARS is a greedy algorithm that needs a restrictive learning sample to converge (data or evidence driven methods would be more robust and still need to be investigated), and on the other hand, there is no means to know if the target languages admit rewriting systems with the desired properties. Essentially LARS is provable: it makes its decisions (to create rules) on the basis of the fact that *there is no reason not to create the rule*. A more pragmatic view is to base such a decision on *the fact that it seems a good idea to do so*. But convergence, in the second setting, needs a statistical decision, and changes quite drastically the type of results one can achieve.

Let us see, on an example, what can happen when running LARS on the first problem of the OMPHALOS competition: it is given by a learning sample composed of 266 positive examples and 535 negative ones. The alphabet is $\Sigma_1 = \{a, b, c, d, e, f\}$. LARS infers first some trivial rules: $\mathcal{R}_{current} = \{\$b \vdash \$, c \vdash \lambda, f \vdash \lambda, be \vdash e, db \vdash d, dde \vdash ed, bbde \vdash bde\}$. The second and the third rules completely erase two letters of the alphabet. The others contribute to decreasing drastically the number of b 's. We do not know what the target language is, but it is not reasonable to erase half of the letters if one wants to learn it. This result is due to the small size of the learning sample in comparison to the complexity of the target language: there do not exist pairs of positive and negative examples that would allow the algorithm to reject these probably inconsistent rules. One of the consequences is that the current learning set I_+ contains 215 positives examples that are too close (*w.r.t* the edit distance) to the negative ones: LARS is not able to infer any new interesting rule and then adds 214 rules of the form $\$w\mathcal{L} \vdash \$e\mathcal{L}$ that rewrite each positive example w into the smallest one e .

7.2. LARS ON SMALL GRAMMARS

We present in this section some specific languages for which rewriting systems exist, and on which the algorithm LARS has been tested. In each case we describe the task and the learning sample to which the

algorithm has been applied. We do not report any runtimes here as all computations took less than one second: both the systems and the learning samples were small.

Dyck Languages.

The language of all bracketed strings or balanced parentheses is classical in formal language theory. It is usually defined by the rewriting system $\langle \{ab \vdash \lambda\}, \lambda \rangle$. The language is context-free and can be generated by the grammar $\langle \{a, b\}, \{S\}, P, S \rangle$ with $P = \{S \rightarrow aSbS; S \rightarrow \lambda\}$. The language is learned in [50] from all positive strings of length up to 10 and all negative strings of length up to 20. In [41] the authors learn it from all positive and negative strings within a certain length, typically from five to seven. Algorithm LARS learns the correct grammar from both types of learning samples but also from much smaller samples of about 20 strings. Alternatively, [45] have tested their GRIDS system on this language, but when learning from positive strings only. They do not identify the language. It should also be noted that the language can be modified to deal with more than one pair of brackets and remains learnable.

Language $\{a^n b^n : n \in \mathbb{N}\}$.

Language $\{a^n b^n : n \in \mathbb{N}\}$ is a language often used as a context-free language that is not regular. The corresponding system is $\langle \{aabb \vdash ab; \$ab\mathcal{L} \vdash \$\lambda\mathcal{L}\}, \lambda \rangle$. Variants of this language are $\{a^n b^n c^m : m, n \in \mathbb{N}\}$ which is studied in [50], and $\{a^m b^n : 1 \leq m \leq n\}$ from [41]. In all cases algorithm LARS has learned the intended system from as few as 20 examples, which is much less than for previous methods.

Regular languages.

We have run algorithm LARS on benchmarks for regular language learning tasks. There are several such benchmarks. Those related to the ABBADINGO [35] tasks were considered too hard for LARS: as we have constructed a greedy algorithm (in the line for instance of RPNI [44]), results when the required strings are not present are bad. We turned to smaller benchmarks, as used in earlier regular inference tasks [18]. These correspond to small automata, and thus from 1 to 6 rewriting rules. In most cases LARS found a correct system, but when it did not, the language induced by the inferred DSRS had little to do with the target language.

Other languages and properties.

The language $\{a^p b^q : p \geq 0, q \geq 0, p \neq q\}$ is not a NTS language [4] but LARS outputs the correct system $\langle \{\$b\mathcal{L} \vdash \$a\mathcal{L}; aa\mathcal{L} \vdash a\mathcal{L}; \$bb \vdash \$b; aab\mathcal{L} \vdash a\mathcal{L}; \$abb \vdash \$b; aabb \vdash ab\}, a \rangle$ from as few as 20 examples. Notice that this language could not have been described without the use of delimiters.

Languages $\{w \in \{a, b\}^* : |w|_a = |w|_b\}$ and $\{w \in \{a, b\}^* : 2|w|_a = |w|_b\}$ are used in [41]. In both cases the languages can be learned by LARS from less than 30 examples.

The language of Lukasiewicz is generated by the grammar $\langle \{a, b\}, \{S\}, P, S \rangle$ with $P = \{S \rightarrow aSS; S \rightarrow b\}$. The intended system is $\langle \{abb \vdash b\}, b \rangle$ but what LARS returned was $\langle \{\$ab \vdash \$\lambda; aab \vdash a\}, b \rangle$, which is correct.

The language $\{a^m b^m c^n d^n : m, n \geq 0\}$ is not linear (but neither is the Dyck language) and is recognized by the system $\langle \{aabb \vdash ab; ccdd \vdash cd, \$abcd\mathcal{L} \vdash \$\mathcal{L}\}, \lambda \rangle$.

On the other hand the language of palindromes ($\{w : w = w^R\}$) does not admit a DSRS, unless the center is some special character. [41] identifies this language, whereas LARS cannot.

System $\langle \{ab^k \vdash b\}, b \rangle$ requires an exponential characteristic sample so learning this language with LARS is a hard task.

8. Conclusion and Future Work

In this paper, we have investigated the problem of learning languages that can be defined with string-rewriting systems (SRS's). We have first tailored a definition of “hybrid almost nonoverlapping delimited SRS's”, proved that they were efficient (often linear) parsing devices and showed that they define all regular languages as well as important context-free languages (Dyck, Lukasiewicz, $\{a^n b^n : n \geq 0\}$, $\{w \in \{a, b\}^* : |w|_a = |w|_b\}$, ...). Then we have provided an algorithm to learn them, LARS, and proved that it could identify, in polynomial time (but not data), the languages whose DSRS had some “closedness” property. Finally, after a discussion on the position of the class of “closed languages” in the Chomsky hierarchy, we have shown that LARS was capable of learning several languages, both regular and not.

However, much remains to be done on this topic. On the one hand, LARS suffers from its simplicity, as it failed in solving the (hard) problems of the OMPHALOS competition. We think that we could improve our algorithm either by pruning our exploration of the search space (this is work in progress), or by studying more restrictive SRS's (*e.g.*, special or monadic SRS [5]), or by investigating more sophisticated

properties (such as *basicity* [51]). On the other hand, other kinds of SRS's can be used to define languages, such as the CR-languages of McNaughton [39], or the DOL systems (that can generate deterministic *context-sensitive* languages). All these SRS's may be the source of new attractive learning results in Grammatical Inference.

Acknowledgements

We thank Géraud Sénizergues from LaBRI (Bordeaux, France) for providing us with pointers to the rewriting systems literature, as well as Alexander Clark (Royal Holloway University of London, UK) for fruitful discussions.

References

1. P. Adriaans, H. Fernau, and M. van Zaannen, editors. *Grammatical Inference: Algorithms and Applications, Proceedings of ICGI '02*, volume 2484 of *LNAI*, Berlin, Heidelberg, 2002. Springer-Verlag.
2. P. Adriaans and M. Vervoort. The EMILE 4.1 grammar induction toolbox. In Adriaans et al. [1], pages 293–295.
3. D. Angluin. Queries revisited. In N. Abe, R. Khardon, and T. Zeugmann, editors, *Proceedings of ALT 2001*, number 2225 in *LNCS*, pages 12–31, Berlin, Heidelberg, 2001. Springer-Verlag.
4. L. Boasson. Grammaire à non-terminaux séparés. In *Proc. 7th ICALP*, pages 105–118. *LNCS* 85, 1980.
5. R. Book and F. Otto. *String-Rewriting Systems*. Springer-Verlag, 1993.
6. J. Calera-Rubio and R. C. Carrasco. Computing the relative entropy between regular tree languages. *Information Processing Letters*, 68(6):283–289, 1998.
7. R. C. Carrasco and J. Oncina, editors. *Grammatical Inference and Applications, Proceedings of ICGI '94*, number 862 in *LNAI*, Berlin, Heidelberg, 1994. Springer-Verlag.
8. R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In *ICGI'94* [7], pages 139–150.
9. R. C. Carrasco, J. Oncina, and J. Calera-Rubio. Stochastic inference of regular tree languages. *Machine Learning Journal*, 44(1):185–197, 2001.
10. E. Charniak. Tree-bank grammars. In *AAAI/IAAI, Vol. 2*, pages 1031–1036, 1996.
11. A. Clark. Learning deterministic context free grammars: the omphalos competition. *Published in this special issue*, 2006.
12. C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning Journal*, 27:125–138, 1997.
13. C. de la Higuera, P. Adriaans, M. van Zaanen, and J. Oncina, editors. *Proceedings of the Workshop and Tutorial on Learning Context-free grammars*. ISBN 953-6690-39-X, 2003.

14. C. de la Higuera and J. Oncina. Learning deterministic linear languages. In Kivinen and Sloan [29], pages 185–200.
15. C. de la Higuera and J. Oncina. Learning context-free languages. *To appear in Artificial Intelligence Reviews*, 2006.
16. A. L. de Oliveira and J. P. M. Silva. Efficient algorithms for the inference of minimum size DFAs. *Machine Learning Journal*, 44(1):93–119, 2001.
17. N. Dershowitz and J. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science : Formal Methods and Semantics*, volume B, chapter 6, pages 243–320. North Holland, Amsterdam, 1990.
18. P. Dupont. Regular grammatical inference from positive and negative samples by genetic search: the GIG method. In Carrasco and Oncina [7], pages 236–245.
19. J. D. Emerald, K. G. Subramanian, and D. G. Thomas. Learning a subclass of context-free languages. In Honavar and Slutski [27], pages 223–231.
20. H. Fernau. Learning tree languages from text. In Kivinen and Sloan [29], pages 153–168.
21. M. Frazier and C.D. Page Jr. Prefix grammars: An alternative characterisation of the regular languages. *Information Processing Letters*, 51(2):67–71, 1994.
22. P. García and J. Oncina. Inference of recognizable tree sets. Technical Report DSIC-II/47/93, Departamento de Lenguajes y Sistemas Informáticos, Universidad Politécnica de Valencia, Spain, 1993.
23. J. Y. Giordano. Inference of context-free grammars by enumeration: Structural containment as an ordering bias. In Carrasco and Oncina [7], pages 212–221.
24. E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.
25. S. A. Goldman and M. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.
26. A. Habrard, M. Bernard, and F. Jacquenet. Generalized stochastic tree automata for multi-relational data mining. In Adriaans et al. [1], pages 120–133.
27. V. Honavar and G. Slutski, editors. *Grammatical Inference, Proceedings of ICGI '98*, number 1433 in LNAI, Berlin, Heidelberg, 1998. Springer-Verlag.
28. H. Ishizaka. Polynomial time learnability of simple deterministic languages. *Machine Learning Journal*, 5:151–164, 1995.
29. J. Kivinen and R. H. Sloan, editors. *Proceedings of COLT 2002*, number 2375 in LNAI, Berlin, Heidelberg, 2002. Springer-Verlag.
30. J. W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–112. Oxford University Press, 1992.
31. T. Knuutila and M. Steinby. Inference of tree languages from a finite sample: an algebraic approach. *Theoretical Computer Science*, 129:337–367, 1994.
32. T. Koshiha, E. Mäkinen, and Y. Takada. Inferring pure context-free languages from positive data. *Acta Cybernetica*, 14(3):469–477, 2000.
33. S. C. Kremer. Parallel stochastic grammar induction. In *Proceedings of the 1997 International Conference on Neural Networks (ICNN '97)*, volume I, pages 612–616, 1997.
34. K. Lang, B. A. Pearlmutter, and F. Coste. The Gowachin automata learning competition, 1998.
35. K. Lang, B. A. Pearlmutter, and R. A. Price. The Abbadingo one DFA learning competition. In *Proceedings of ICGI'98*, pages 1–12, 1998.

36. K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In Honavar and Slutski [27], pages 1–12.
37. K. Lari and S. J. Young. The estimation of stochastic context free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.
38. L. Lee. Learning of context-free languages: A survey of the literature. Technical Report TR-12-96, Center for Research in Computing Technology, Harvard University, Cambridge, Massachusetts, 1996.
39. R. McNaughton, P. Narendran, and F. Otto. Church-Rosser Thue systems and formal languages. *Journal of the Association for Computing Machinery*, 35(2):324–344, 1988.
40. W. Moczydlowski and A. Geser. Termination of single-threaded one-rule semi-thue systems. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications*, pages 338–352. LNCS 3467, 2005.
41. K. Nakamura and M. Matsumoto. Incremental learning of context-free grammars based on bottom-up parsing and search. *Pattern Recognition*, 38(9):1384–1392, 2005.
42. C. Nevill-Manning and I. Witten. Identifying hierarchical structure in sequences: a linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997.
43. M. Nivat. On some families of languages related to the dyck language. In *Proc. 2nd Annual Symposium on Theory of Computing*, 1970.
44. J. Oncina and P. García. Identifying regular languages in polynomial time. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, pages 99–108. World Scientific, 1992.
45. G. Petasis, G. Paliouras, V. Karkaletsis, C. Halatsis, and C. Spyropoulos. E-grids: Computationally efficient grammatical inference from positive examples. *Grammars*, 7:69–110, 2004.
46. J. R. Rico-Juan, J. Calera-Rubio, and R. C. Carrasco. Stochastic k -testable tree languages and applications. In Adriaans et al. [1], pages 199–212.
47. Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, 76:223–242, 1990.
48. Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97:23–60, 1992.
49. Y. Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185:15–45, 1997.
50. Y. Sakakibara and M. Kondo. Ga-based learning of context-free grammars using tabular representations. In *Proceedings of 16th International Conference on Machine Learning (ICML-99)*, pages 354–360, 1999.
51. G. Sénizergues. A polynomial algorithm testing partial confluence of basic semi-thue systems. *Theor. Comput. Sci.*, 192(1):55–75, 1998.
52. B. Starkie, F. Coste, and M. van Zaanen. Omphalos context-free language learning competition, 2004.
53. Y. Takada. Grammatical inference for even linear languages based on control sets. *Information Processing Letters*, 28(4):193–199, 1988.
54. F. Thollard, P. Dupont, and C. de la Higuera. Probabilistic DFA inference using kullback-leibler divergence and minimality. In *Proc. 17th International Conf. on Machine Learning*, pages 975–982. Morgan Kaufmann, San Francisco, CA, 2000.

- 55. K. Vanlehn and W. Ball. A version space approach to learning context-free grammars. *Machine Learning Journal*, 2:39–74, 1987.
- 56. G. Wolf. Grammar discovery as data compression. In *Proceedings of AISB/GI Conference on Artificial Intelligence*, pages 375–379, Hamburg, 1978.
- 57. T. Yokomori. Polynomial-time identification of very simple grammars from positive data. *Theor. Comput. Sci.*, 1(298):179–206, 2003.