Mach Learn (2007) 66:7–31
DOI 10.1007/s10994-006-9593-8

LARS: A learning algorithm for rewriting systems*

Rémi Eyraud · Colin de la Higuera · Jean-Christophe Janodet

Received: 5 July 2005 / Revised: 2 May 2006 / Accepted: 22 June 2006 / Published online: 4 August 2006

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. Indeed, several 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. 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 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 the 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 Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

Editor: Georgios Paliouras and Yasubumi Sakakibara

R. Eyraud (✉) · C. de la Higuera · J.-C. Janodet
EURISE, Université Jean Monnet de Saint-Etienne, 23 rue Paul Michelon, 42023 Saint-Etienne, France
e-mail: remi.eyraud@univ-st-etienne.fr

C. de la Higuera
e-mail: cdlh@univ-st-etienne.fr

J.-C. Janodet
e-mail: janodet@univ-st-etienne.fr

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) (Lang, Pearlmutter, & Price, 1998), or of learning a regular distribution from positive examples (Carrasco & Oncina, 1994; Thollard, Dupont, & de la Higuera, 2000).

On the other hand things tend to get harder when the larger class of context-free languages is considered (Lee, 1996; Sakakibara, 1997). 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 (de la Higuera, 1997) and it is believed that context-free grammars cannot be identified, in the framework of active learning, from a *minimum adequate teacher*: such a teacher is allowed to make equivalence and membership queries (Angluin, 2001). 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 in limiting the class of context-free grammars to be learned: even linear grammars (Takada, 1988), deterministic linear grammars (de la Higuera & Oncina, 2002), or very simple grammars (Yokomori, 2003) 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 (1995) 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 (Vanlehn & Ball, 1987; Giordano, 1994; Adriaans & Vervoort, 2002; Petasis et al., 2004), or compression techniques (Wolf, 1978; Nevill-Manning, & Witten, 1997) 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 (Charniak, 1996) following theoretical results by Sakakibara (1992). Learning from structured data has been a line followed by many: learning tree automata (Knuutila & Steinby, 1994; Fernau, 2002; Habrard, Bernard, & Jacquenet,), or context-free grammars from bracketed data (Sakakibara, 1990) allows one to obtain better results, either with queries (Sakakibara, 1992), regular distributions (Kremer, 1997; Carrasco, Oncina, & Calera-Rubio, 2001; Rico-Juan, Calera-Rubio, & Carrasco, 2002), or negative information (García & Oncina, 1993). This has also led to different studies concerning the probability estimation of such grammars (Lari & Young, 1990; Calera-Rubio & Carrasco, 1998).

In 2003 there has been renewed interest in the topic (de la Higuera et al., 2003): the OMPHALOS context-free language learning competition (Starkie, Coste, & van Zaanen, 2004) was launched, where state of the art techniques were unable to solve even the easiest tasks. The method (Clark, 2006) 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 (probabilistic or not, with additional information or not) (Lee, 1996; Sakakibara, 1997; de la Higuera & Oncina, 2006).

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 (Koshiba, Mäkinen, & Takada, 2000; Emerald, Subramanian, & Thomas, 1998).

In this paper, we investigate string-rewriting systems (SRS's) as an alternative way to describe and manipulate context-free languages. The theory of SRS's (also called semi-Thue systems) was invented in 1914 by Axel Thue and extended since to trees and graphs, a lot of attention has been paid to it throughout the 20th century (see Book & Otto, 1993; Dershowitz & Jouannaud, 1990). 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 $ab \vdash \lambda$. Using this rule consists of replacing some substring ab by the empty string, thus of erasing ab. It allows $abaabbab$ to rewrite as follows:

$$abaabb\underline{ab} \vdash aba\underline{ab}b \vdash \underline{ab}ab \vdash \underline{ab} \vdash \lambda$$

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 if and only if 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 Nivat's seminal paper (Nivat, 1970), which has been 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 $w = a_1 a_2 \dots a_n$ 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* or *length-lexicographic order* (Oncina & García, 1992), denoted by $\lhd$, as

follows:

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

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

By a language L over Σ we mean every subset $L \subseteq \Sigma^*$. Many classes of languages have been 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* $\mathcal{L} : \mathbb{R} \to \mathbb{L}$ 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 $\delta : Q \times \Sigma \to Q$ 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 $uTv \Rightarrow uwv$ when $(T, w) \in P$. $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$. If there exist $u_0, \ldots, u_k$ such that $\forall i, 0 \leq i < k$, $u_i \Rightarrow u_{i+1}$ we will write $u_0 \overset{k}{\Rightarrow} u_k$. 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 Gold (1978), de la Higuera (1997), 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 Gold (1978) and de la Higuera (1997). 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 £ 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 $\$\Sigma^*£$. As for the rewrite rules, they will be *partially* marked, and thus belong to $\overline{\Sigma^*} = (\lambda + \$)\Sigma^*(\lambda + £)$. 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 McNaughton, Narendran, and Otto (1988).

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. $l, r \in \$\Sigma^*£$ (used to rewrite whole strings) or
  3. $l, r \in \Sigma^*$ (used to rewrite substrings) or
  4. $l, r \in \Sigma^*£$ (used to rewrite suffixes).

  Rules of types 1 and 2 will be called \$-*rules* and rules of types 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 let $\|\mathcal{R}\|$ be 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 $\$aabb\pounds$ is rewritten to $\$aaa\pounds$ with rule $bb\pounds \vdash a\pounds$.

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

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

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 $\mathcal{R}$:

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

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:

$$\$aabb\underline{ab}\pounds \vdash \$a\underline{abb}\pounds \vdash \$\underline{ab}\pounds \vdash \$\pounds$$

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

$$\$aa\underline{aabb}bb\pounds \vdash \$a\underline{aabbb}\pounds \vdash \$\underline{aabb}\pounds \vdash \$\underline{ab}\pounds \vdash \$\pounds$$

  Notice that the rule $\$ab\pounds \vdash \$\pounds$ is necessary for deriving λ (last derivation step).
- $\mathcal{L}(\{\$ab \vdash \$\}, \lambda)$ is the regular language $(ab)^*$. Indeed,

$$\underline{\$ab}abab\pounds \vdash \underline{\$ab}ab\pounds \vdash \underline{\$ab}\pounds \vdash \$\pounds$$

  Actually, we will see, following the mechanism from the above example, 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 (Section 4.1) and the confluence property (Section 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 (Section 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 :

$$aa \vdash ba \vdash aa \vdash ba \vdash aa \ldots$$

$$aca \vdash acca \vdash accca \vdash acccca \vdash accccca \vdash \ldots$$

Actually, the termination of SRS's is undecidable in general and the decidability of termination on subclasses of string-rewriting systems is still an active research topic (see Moczydlowski & Geser, 2005 for instance). And in the context of the DSRS, there is no reason to believe that the problem may be 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:

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

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 (Dershowitz & Jouannaud, 1990). However, if one uses it to rewrite $\$1^n£$ into $\$0^n£$, 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:

$$\begin{array}{ll} & \$1111£ \\ \vdash & \$1110£ \\ \vdash & \$11\underline{1c}1d£ \vdash \$110\underline{d1}d£ \vdash \$1101\underline{dd}£ \vdash \$1101£ \\ \vdash & \$1100£ \\ \vdash & \$11\underline{0c}1d£ \vdash \$1\underline{1c}11d£ \vdash \$10\underline{d1}1d£ \vdash \$101\underline{d1}d£ \vdash \$1011\underline{dd}£ \vdash \$1011£ \\ \vdash & \$1010£ \\ \vdash^* & \$1001£ \\ \vdash & \$1000£ \\ \vdash^* & \$0111£\ldots \\ \vdash^* & \$0000£. \end{array}$$

In order to tackle these problems, we first extend the hierarchical order $\lhd$ to the strings of $\overline{\Sigma^*}$, by defining the *extended hierarchical order*, denoted $\prec$, as follows:

$$\forall w_1, w_2 \in \Sigma^*, \text{ if } w_1 \lhd w_2 \text{ then } w_1 \prec \$w_1 \prec w_1£ \prec \$w_1£ \prec w_2.$$

Therefore, if $a \prec b$, then $\lambda \lhd a \lhd b \lhd aa \lhd ab \lhd ba \lhd bb \lhd aaa \lhd \ldots$, so $\lambda \prec \$ \prec £ \prec \$£ \prec a \prec \$a \prec a£ \prec \$a£ \prec b \prec \cdots$. 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 become 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., $l, r \in \$\Sigma^*(\lambda + £)$) and is length-lexicographic: $r \prec l$, or
2. R is a non-\$-rule (i.e., $l, r \in \Sigma^*(\lambda + £)$) 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 $u_0 \vdash u_1 \vdash u_2 \vdash \cdots$, then $u_0 \succ u_1 \succ u_2 \succ \cdots$. As $\overline{\Sigma^*}$ is a well-ordered set, there is no infinite and strictly decreasing chain of the form $u_0 \succ u_1 \succ u_2 \succ \cdots$. 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 $\$lu£ \vdash \$ru£ \vdash \cdots \vdash \$lv£$ with $|u| = |v|$ (since the length is preserved). As $\$ru£ \vdash^* \$lv£$ 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 $|\mathcal{R}| + |w'| \cdot |\mathcal{R}| \leq |w| \cdot |\mathcal{R}|$. □

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 (Dershowitz & Jouannaud, 1990).

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).

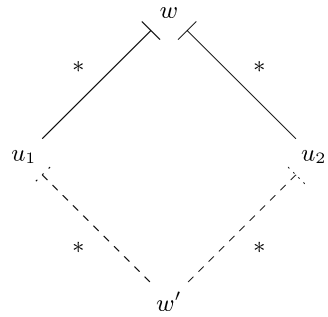

Fig. 1 The Church-Rosser property is also called the Diamond property

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; the condition will be more restrictive than what is needed to be able to obtain positive decidability results, but will be able to be verified easily:

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)):

$$\exists u, v \in \overline{\Sigma^*}, l_1u = vl_2, 0 < |v| < |l_1|,$$

and then $r_1u = vr_2$;

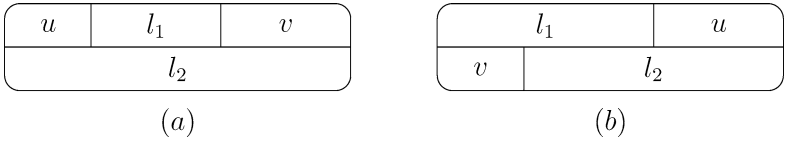

Fig. 2 Two rules overlap when their left-hand sides overlap; part (a) of the diagram above shows Case 2 (and symmetrically, Case 4) of the previous definition; part (b) illustrates Case 3 (and symmetrically, Case 5)

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

$$\exists u, v \in \overline{\Sigma^*}, ul_1 = l_2 v, 0 < |v| < |l_1|,$$

and then $ur_1 = r_2 v$;

6. Or when there is no overlapping, i.e. ($l_i = uv$ and $l_j = vw \Rightarrow v = \lambda$), for $i, j \in \{1, 2\}, i \neq j$.

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

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. Such condition is often used in the framework of constructor-based term rewriting systems (O'Donnell, 1977).

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 $u_2 \vdash_\varepsilon w'$ (Klop, 1992). 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 $\vdash^*_\varepsilon$ is the reflexive and transitive closure of $\vdash_\varepsilon$. Finally, as $u_1 \vdash^*_\varepsilon w'$ and $u_2 \vdash^*_\varepsilon w'$, we deduce that $u_1 \vdash^*_{\mathcal{R}} w'$ and $u_2 \vdash^*_{\mathcal{R}} w'$. □

4.3. On the hybrid ANo DSRS's

In the rest of this paper, we will consider only hybrid ANo DSRS's. This allows the following properties to hold:

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 $\langle\{ab \vdash \lambda\}, \lambda\rangle$, $\langle\{ab \vdash \lambda; ba \vdash \lambda\}, \lambda\rangle$, $\langle\{aabb \vdash ab; \$ab\pounds \vdash \$\pounds\}, \lambda\rangle$ and $\langle\{\$ab \vdash \$\}, \lambda\rangle$ satisfy the hybrid and ANo constraints. 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* (Frazier & Page Jr, 1994), 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 $\forall w' \lhd w_q, \delta(q_0, w') \neq q$. Let e be the minimum string that reaches a final state (w.r.t. $\lhd$) , 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 $\mathcal{R}_2 = \{\$w_q\pounds \vdash \$e\pounds : q \in F, w_q \neq e\}$ 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 $\delta(q_0, w_q x) = q'$; as $w_{q'}$ is the minimum string that reaches q' and $\delta(q_0, w_q x) = q'$, we get $\$w_{q'} \prec \$w_q x$. One 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:

– 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.
– Or $m \neq \lambda$ and then $w_{q'} y m = w_q$. We deduce that $q = \delta(q_0, w_q) = \delta(q_0, w_{q'} y m) = \delta(q', ym) = \delta(p', m) = \delta(q_0, w_{p'} m)$. 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 £. 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. □

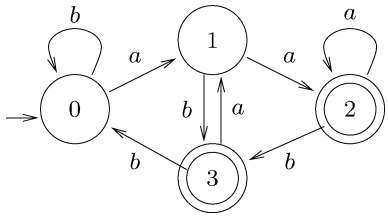

Fig. 3 The minimal *dfa* that recognizes the language $(a+b)^*a(a+b)$

Example . Consider the minimal *dfa* A that recognizes the language $(a+b)^a(a+b)$ (see Fig. 3). 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 $\mathcal{R}_2=\{\$ab£\vdash\$aa£\}$. One can easily check that $\mathcal{R}_1\cup\mathcal{R}_2$ is a hybrid ANo DSRS.

5. Algorithm

In this section we present our learning algorithm (see Fig. 4) 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.

Algorithm 1: LARS (Learning Algorithm for Rewriting Systems)

Data : a sample $S=\langle S_+,S_-\rangle$
Result : $\langle\mathcal{R},e\rangle$ where $\mathcal{R}$ is a hybrid ANo DSRS and e is an irreducible string
begin
  $\mathcal{R}\longleftarrow\emptyset$;$I_+\longleftarrow S_+$;$I_-\longleftarrow S_-$;
  $F\longleftarrow$ `sort`$_\preceq\ \{v\in\overline{\Sigma^*}:\exists u,w\in\overline{\Sigma^*},uvw\in I_+\}$;
  for $i=1$ **to** $|F|$ **do**
    if `does_appear`$(F[i],I_+)$ **then**
      for $j=0$ *to* $i-1$ **do**
        $\mathcal{S}\longleftarrow\mathcal{R}\cup\{F[i]\vdash F[j]\}$;
        if `is_DSRS`$(\mathcal{S})$ **and** `is_hybrid`$(\mathcal{S})$ **and** `is_ANo`$(\mathcal{S})$ **then**
          $E_+\longleftarrow$ `normalize`$(I_+,\mathcal{S})$;$E_-\longleftarrow$ `normalize`$(I_-,\mathcal{S})$;
          if $E_+\cap E_-=\emptyset$ **then**
            $\mathcal{R}\longleftarrow\mathcal{S}$; $I_+\longleftarrow E_+$; $I_-\longleftarrow E_-$;
  $e\longleftarrow$ `min`$_\preceq I_+$;
  foreach $w\in I_+$ **do**
    if $w\neq e$ **then** $\mathcal{R}\longleftarrow\mathcal{R}\cup\{w\vdash e\}$;
  return $\langle\mathcal{R},e\rangle$;
end

Fig. 4 The pseudo-code of LARS

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 Definition 3 (resp. Definition 5 and 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., $normalize(X, \mathcal{S}) = \{w{\downarrow_{\mathcal{S}}}: w \in X\}$).

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 £ 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 $S_- \cap \mathcal{L}(\mathcal{R}, e) = \emptyset$. □

To clarify the way the algorithm behaves, we run it on a toy example (summarized in Table I). Suppose that LARS is fed with $S_+ = \{\$b£, \$abb£, \$abaabbb£, \$ababb£\}$ and $S_- = \{\$\lambda£, \$a£, \$ab£, \$aa£, \$baab£, \$bab£, \$aab£, \$abab£, \$aabb£\}$. 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 $P = \{S \to aSS; S \to b\}$. 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 £ are discarded. As for string \$£, 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 $\$ab£$ is reduced to $\$b£$ 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 $a£$ and $\$a£$ 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 $\$b£$ is then reduced to a negative one, $\$\lambda£$. $b \vdash a$ is length-lexicographic but not a \$-rule, and is thus not hybrid.

The substrings $\$b$, $b£$, $\$b£$ 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).

Table 1 Summary of an execution of LARS. The 1st column contains the substrings in F: they are vertically ordered as F is sorted. The 2nd column contains the output of the function `does_appear` on the current substring and the current set I_+. The 3rd column contains the current rule and the 4th one the output of the evaluation of the functions `is_hybrid` and `is_ANo`. The 5th 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

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

Then LARS looks for rules made with aa as left-hand side. The rule $aa \vdash \lambda$ reduces the negative example $\$aab\pounds$ into $\$b\pounds$ that is a positive one. Similarly, the rule $aa \vdash a$ (resp. $aa \vdash b$) reduces $\$aabb\pounds$ into $\$abb\pounds$ (resp. $\$aa\pounds$ into $\$b\pounds$) 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 $\$bab\pounds$ into the positive one $\$b\pounds$. For the same reason, rule $ab \vdash a$ (which reduces both the positive example $\$abb\pounds$ and the negative one $\$ab\pounds$ into $\$a\pounds$) and rule $ab \vdash b$ ($\$ab\pounds$ belongs to I_- and $\$b\pounds$ 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 step gives $I_+ = \{\$b\pounds, \$aabbb\pounds\}$ and $I_- = \{\$\lambda\pounds, \$a\pounds, \$aa\pounds, \$baab\pounds, \$bab\pounds, \$aab\pounds, \$aabb\pounds\}$. The rule is then added to $\mathcal{R}$ that becomes $\{\$ab \vdash \$\lambda\}$.

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 checking 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 $I_+ = \{\$b\pounds\}$ and $I_- = \{\$\lambda\pounds, \$a\pounds, \$aa\pounds, \$ba\pounds\}$.

At this point there is only one string in I_+, so LARS does not infer any new rule, and thus ends and outputs $\langle\{\$ab \vdash \$\lambda, aab \vdash a\}, \$b\pounds\rangle$. 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 1. Notice that, as no hybrid rule can be constructed with λ, \$ or £ as left-hand side, we have not put them in the first column of the table (they respectively correspond to the elements of indices zero to two 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 question of 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 $w \in \$L\pounds$ and $u, v \in \overline{\Sigma^*}$ such that $w = ulv$.
- R is *consistent* w.r.t. L *iff* $\forall u, v \in \overline{\Sigma^*}, (ulv \in \$L\pounds \iff urv \in \$L\pounds)$.

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 $\$L\pounds$ 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 $\$aabb\pounds \in \$L\pounds$ into $\$aab\pounds \notin \$L\pounds$. Actually, a rule is consistent w.r.t. L if $\$L\pounds$ and $(\overline{\Sigma^*} \setminus \$L\pounds)$ 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 greatest[1] 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 characteristic sample may not be of polynomial size as is required in (de la Higuera, 1997).

We first define the characteristic sample for a closed language:

Definition 10 (Characteristic sample). 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. $\$e\pounds \in CS_+$.
2. For any rule $R = l \vdash r \in \mathcal{T}$, there exist two strings $ulv, u'rv' \in \$L\pounds \cap CS_+$ 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), if there exist α and β in $\overline{\Sigma^*}$ such that $\alpha l \beta \in \$L\pounds$, then there exists $u, v \in \overline{\Sigma^*}$ such

[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 $u_2 \preceq v_2)$.

that $ulv \in (\$\Sigma^*\pounds \setminus \$L\pounds) \cap CS_-$ and $urv \in \$L\pounds \cap CS_+$, or $urv \in (\$\Sigma^*\pounds \setminus \$L\pounds) \cap CS_-$ and $ulv \in \$L\pounds \cap CS_+$.

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 $w \in \$L\pounds \setminus \$\mathcal{L}(\mathcal{R}, e)\pounds$ 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 needed 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, these conditions are necessary for identification. These conditions correspond to those required in most grammatical inference algorithms, when identification in the limit is the issue.

The third item concerns the hybrid rules that are not consistent (but applicable) 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 into a negative one, or a negative example in a positive one, to reject it. The first choice above may have an undesirable side effect: it may unnecessarily increase the size of the set of substrings F LARS works on. But it is not always possible to find a negative example that is rewritten into a positive one using this rule. As the rule is not consistent, at least one of the two cases hold, which allows us to define a characteristic sample from which the rule will be rejected.

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 the target language cannot be induced without it.

Notice that the definition (and the following proof) show the existence of a characteristic sample, but not its constructability: the last item requires the comparison between the languages induced by two different hybrid ANo DSRS's, which is probably an undecidable problem. However, this is not a problem from a theoretical point of view. Indeed, it is known since (de la Higuera, 1997) that polynomial identification in the limit (see Definition 2) is equivalent to "semi-poly teachability", as defined by Goldman & Kearns (1995). This last notion requires the existence of a characteristic sample but not its constructability (that would correspond to "teachability").

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 sample, 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 appears 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 and m', one in S_+, the other in S_- such that $m \vdash_R m'$. Suppose that m is in $(\$\Sigma^*\pounds \setminus \$L\pounds) \cap CS_-$

and m' in $\$L\pounds \cap CS_+$ (the other case is symmetric). 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 $\mathcal{L}(\mathcal{R}, e) = L$. □

6.2. LARS and the Chomsky Hierarchy

In the field of grammar induction, it is usual to discuss the position of the learned class in the Chomsky Hierarchy. This hierarchy, introduced in Chomsky (1956), is composed of four classes of languages: the regular, the context-free, the context-sensitive and the recursively enumerable ones. The regular languages are included in the context-free ones, that are contained in the context-sensitive ones. The class of recursively enumerable languages includes all the others.

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

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 $\langle \{ab \vdash \lambda, ba \vdash \lambda\}, \lambda \rangle$ which induces the context-free language $\{w \in \{a, b\}^* : |w|_a = |w|_b\}$ is closed. In addition, one can

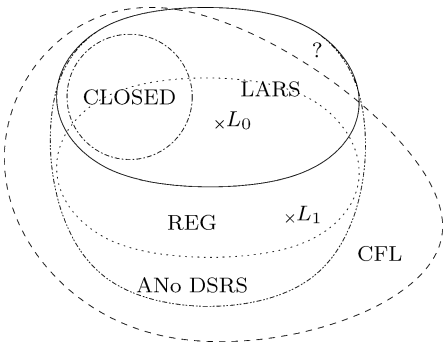

Fig. 5 LARS and the languages in the Chomsky hierarchy. LARS denotes the class of languages learnable by Algorithm 1 (LARS) and ANo DRSR the class of languages representable by hybrid ANo DSRS's

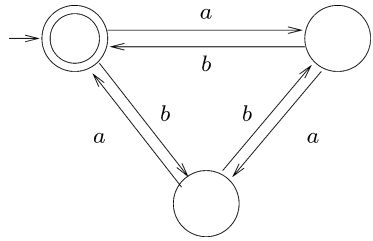

Fig. 6 The minimal *dfa* of $L_1 = \{w \in \{a, b\}^* : |w|_a \bmod 3 = |w|_b \bmod 3\}$

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 $\mathcal{R} = \{\$b \vdash \$, a£ \vdash £, abab \vdash ab\}$, that induces the regular language $L_0 = b^*(ab)^+a^*$. The rule $R = aa£ \vdash £$ 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 $a£ \vdash £ \in \mathcal{R}$, so R is not needed to induce L_0. Notice that all the 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 $R_{\max}$, $R' = a£ \vdash £$ 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. Yet, 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 DSRS's:

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

Notice that $\mathcal{R}_2$ is ANo ; as for $\mathcal{R}_1$, it is Church-Rosser but not ANo. Notice also that L_1 is not a closed language 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 allows LARS to reject inconsistent rules, 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 $\mathcal{R}_2$. No smaller consistent and applicable rule can be inferred during the process. Moreover, $\mathcal{R}$ does not induce L_1, 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_1. In other words, L_1 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. Last but not least, we conjecture 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 on-line 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 or OMPHALOS competitions (Lang, Pearlmutter, & Price, 1998; Starkie, Coste, & van Zaanen, 2004). Other large benchmarks have been made available by Oliveira & Silva (2001) or the GOWACHIN system (Lang, Pearlmutter, & Coste, 1998). Sizes of the alphabets, the grammars, the training sets can vary, but correspond to limit situations w.r.t. the state of the art: 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.

When a new algorithm is presented it is reasonable to show (not prove!) that positive results can be obtained 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 (Nakamura & Matsumoto, 2005; Sakakibara & Kondo, 1999).

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 unknown 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 create a rule*. 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: $\mathcal{R}_{\text{current}} = \{\$b \vdash \$, c \vdash \lambda, f \vdash \lambda, be \vdash e, db \vdash d, dde \vdash ed, bbde \vdash bde\}$. 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 does not seem 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 $\$w\pounds \vdash \$e\pounds$ 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 size of 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 $P = \{S \to aSbS; S \to \lambda\}$. The language is learned in Sakakibara and Kondo (1999) from all positive strings of length up to 10 and all negative strings of length up to 20. In Nakamura and Matsumoto (2005) 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, Petasis et al. (2004) 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 $\langle\{aabb \vdash ab; \$ab\pounds \vdash \$\lambda\pounds\}, \lambda\rangle$. Variants of this language are $\{a^n b^n c^m : m, n \in \mathbb{N}\}$ which is studied in Sakakibara and Kondo (1999), and $\{a^m b^n : 1 \leq m \leq n\}$ from Nakamura and Matsumoto (2005). 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 (Lang, Pearlmutter, & Price, 1998) tasks were considered too hard for LARS: as we have constructed a greedy algorithm (in the line for instance of RPNI (Oncina & García, 1992)), results when the required strings are not present are bad. We turned to smaller benchmarks, as used in earlier regular inference tasks Dupont (). 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 no connection 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 (Boasson, 1980) but LARS outputs the correct system $\langle\{\$b\pounds \vdash \$a\pounds; aa\pounds \vdash a\pounds; \$bb \vdash \$b; aab\pounds \vdash a\pounds; \$abb \vdash \$b; aabb \vdash ab\}, a\rangle$ 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 Nakamura and Matsumoto (2005). In both cases the languages can be learned by LARS from less than 30 examples.

The language of Lukasiewicz is generated for instance by the grammar $\langle\{a, b\}, \{S\}, P, S\rangle$ with $P = \{S \to aSS; S \to b\}$. 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 $\langle\{aabb \vdash ab; ccdd \vdash cd, \$abcd\pounds \vdash \$\pounds\}, \lambda\rangle$.

On the other hand the language of palindromes ($\{w : w = w^R\}$) does not admit a DSRS, unless the center is some special character. Nakamura and Matsumoto (2005) 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, or by studying more restrictive SRS's (e.g., special or monadic SRS (Book & Otto, 1993)), or by investigating more sophisticated properties (such as *basicity* (Sénizergues, 1998)). On the other hand, other kinds of SRS's can be used to define languages, such as the CR-languages of McNaughton, Narendran, and Otto (1988), or the D0L systems (that can generate deterministic *context-sensitive* languages). Notice also that the learnability of term rewriting systems is being investigated (see for instance Rao, 2004; Togashi & Noguchi, 1990; Laird & Gamble, 1990). All these SRS's may be the source of new attractive learning results in Grammatical Inference.

Acknowledgments We thank Géraud Sénizergues (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. The efforts made by the anonymous referees also deserve to be acknowledged.

References

Adriaans, P., Fernau, H., & van Zaannen, M. (Eds.) (2002). *Grammatical inference: Algorithms and applications*, In *Proceedings of ICGI '02*, vol. 2484 of *LNAI*, Berlin, Heidelberg: Springer-Verlag.

Adriaans, P., Vervoort, M. (2002). The EMILE 4.1 grammar induction toolbox. In P., Adriaans, H., Fernau, & van, M. Zaannen (Eds.), *Grammatical inference: Algorithms and applications*, *Proceedings of ICGI '02*, vol. 2484 of *LNAI* (pp. 293-–295). Berlin, Heidelberg: Springer-Verlag.

Angluin, D. (2001). Queries revisited. In N. Abe, R. Khardon, & T. Zeugmann (Eds.), *Proceedings of ALT 2001*, number 2225 in LNCS, (pp. 12–31), Berlin, Heidelberg: Springer-Verlag.

Boasson, L. (1980). Grammaire à non-terminaux séparés. In *Proc. 7th ICALP* (pp. 105–118). LNCS 85.

Book, R., & Otto, F. (1993). *String-rewriting systems*. Springer-Verlag.

Calera-Rubio, J., & Carrasco, R. C. (1998). Computing the relative entropy between regular tree languages. *Information Processing Letters*, *68*(6), 283–289.

Carrasco, R. C., & Oncina, J. (Eds.) (1994). *Grammatical inference and applications*. In *Proceedings of ICGI '94*, number 862 in LNAI, Berlin, Heidelberg: Springer-Verlag.

Carrasco, R. C., & Oncina, J. (1994) Learning stochastic regular grammars by means of a state merging method. In R. C., Carrasco, & J. Oncina (Eds.), *Grammatical inference and applications. Proceedings of ICGI '94*, number 862 in LNAI, Berlin, (pp. 139–150), Heidelberg, Springer-Verlag.

Carrasco, R. C., Oncina, J., & Calera-Rubio, J. (2001). Stochastic inference of regular tree languages. *Machine Learning Journal*, *44*(1), 185–197.

Charniak, E. (1996). Tree-bank grammars. In *AAAI/IAAI,* (vol. 2, pp. 1031–1036).

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *3*, 113–124.

Clark, A. (2006). Learning deterministic context free grammars: the omphalos competition. *Published in this special issue*.

de la Higuera, C. (1997). Characteristic sets for polynomial grammatical inference. *Machine Learning Journal*, *27*, 125–138.

de la Higuera, C., Adriaans, P., van Zaanen, M., & Oncina, J. (Eds.), (2003). In *Proceedings of the Workshop and Tutorial on Learning Context-free Grammars*. ISBN 953-6690-39-X.

de la Higuera, C., & Oncina, J. (2002). Learning deterministic linear languages. In J., Kivinen, & R. H., Sloan, (Eds.), *Proceedings of COLT 2002*, number 2375 in LNAI, (pp. 185–200). Berlin, Heidelberg. Springer-Verlag.

de la Higuera, C., & Oncina, J. (2006). Learning context-free languages. *Artificial Intelligence Reviews*. (To appear).

de Oliveira, A. L., & Silva, J. P. M. (2001). Efficient algorithms for the inference of minimum size DFAs. *Machine Learning Journal*, *44*(1), 93–119.

Dershowitz, N., & Jouannaud, J. (1990). Rewrite systems. In J. van Leeuwen (Ed.), *Handbook of Theoretical Computer Science: Formal Methods and Semantics*, (vol. B, chap. 6, pp. 243–320). North Holland, Amsterdam.

Dupont, P. (1994). Regular grammatical inference from positive & negative samples by genetic search: the GIG method. In R. C., Carrasco, & J. Oncina, (Eds.), *Grammatical inference and applications*, *Proceedings of ICGI '94*, number 862 in LNAI (pp. 236–245). Berlin, Heidelberg: Springer-Verlag.

Emerald, J. D., Subramanian, K. G., & Thomas, D. G. (1998). Learning a subclass of context-free languages. In V., Honavar, & G. Slutski, (Eds.), *Grammatical inference, Proceedings of ICGI '98*, number 1433 in LNAI, (pp. 223–231). Berlin, Heidelberg: Springer-Verlag.

Fernau, H. (2002). Learning tree languages from text. In J., Kivinen, & R. H. Sloan, (Eds.), *Proceedings of COLT 2002*, number 2375 in LNAI, (pp. 153–168). Berlin, Heidelberg. Springer-Verlag.

Frazier, M., & Page, C.D. Jr, (1994). Prefix grammars: An alternative characterisation of the regular languages. *Information Processing Letters*, *51*(2), 67–71.

García, P., & Oncina, J. (1993). Inference of recognizable tree sets. Technical Report DSIC-II/47/93, Departamento de Lenguajes y Sistemas Informáticos, Universidad Politécnica de Valencia, Spain.

Giordano, J. Y. (1994). Inference of context-free grammars by enumeration: Structural containment as an ordering bias. In R. C., Carrasco, & J. Oncina, (Eds.), *Grammatical inference and applications*, *Proceedings of ICGI '94*, number 862 in LNAI, (pp. 212–221). Berlin, Heidelberg, Springer-Verlag.

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, *37*, 302–320.

Goldman, S. A., & Kearns, M. (1995). On the complexity of teaching. *Journal of Computer and System Sciences*, *50*(1), 20–31.

Habrard, A., Bernard, M., & Jacquenet, F. (2002). Generalized stochastic tree automata for multi-relational data mining. In P., Adriaans, H., Fernau, & M. van Zaannen. (Eds.), *Grammatical inference: Algorithms and applications*, *Proceedings of ICGI '02*, vol. 2484 of *LNAI*, (pp. 120–133). Berlin, Heidelberg. Springer-Verlag.

Honavar, V., & Slutski, G. (Eds.) (1998). *Grammatical inference, Proceedings of ICGI '98*, number 1433 in LNAI, Berlin, Heidelberg. Springer-Verlag.

Ishizaka, H. (1995). Polynomial time learnability of simple deterministic languages. *Machine Learning Journal*, *5*, 151–164.

Kivinen, J., & Sloan, R. H. (Eds.), (2002). In *Proceedings of COLT 2002*, number 2375 in LNAI, Berlin, Heidelberg: Springer-Verlag.

Klop, J. W. (1992). Term rewriting systems. In S. Abramsky, D. Gabbay, & T. Maibaum, (Eds.), *Handbook of Logic in Computer Science*, (vol. 2, pp. 1–112). Oxford University Press.

Knuutila, T., & Steinby, M. (1994). Inference of tree languages from a finite sample: an algebraic approach. *Theoretical Computer Science*, *129*, 337–367.

Koshiba, T., Mäkinen, E., & Takada, Y. (2000). Inferring pure context-free languages from positive data. *Acta Cybernetica*, *14*(3), 469–477.

Kremer, S. C. (1997). Parallel stochastic grammar induction. In *Proceedings of the 1997 International Conference on Neural Networks (ICNN '97)*, (vol. I, pp. 612–616).

Laird, P., & Gamble, E. (1990). Ebg and term rewriting systems. In *Algorithmic Learning Theory* (pp. 425–440).

Lang, K., Pearlmutter, B. A., & Coste, F. (1998). The Gowachin automata learning competition.

Lang, K., Pearlmutter, B. A., & Price, R. A. (1998). The Abbadingo one DFA learning competition. In *Proceedings of ICGI'98*, (pp. 1–12). The abbadingo competition can be found at the address: http://abbadingo.cs.unm.edu/

Lang, K. J., Pearlmutter, B. A., & Price, R. A. (1998). Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In V., Honavar, & G. Slutski, (Eds.), *Grammatical Inference, Proceedings of ICGI '98*, number 1433 in LNAI, (pp. 1–12). Berlin, Heidelberg: Springer-Verlag.

Lari, K., & Young, S. J. (1990). The estimation of stochastic context free grammars using the inside-outside algorithm. *Computer Speech and Language*, *4*, 35–56.

Lee, L. (1996). 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.

McNaughton, R., Narendran, P., & Otto, F. (1988). Church-Rosser Thue systems and formal languages. *Journal of the Association for Computing Machinery*, *35*(2), 324–344.

Moczydlowski, W., & Geser, A. (2005). Termination of single-threaded one-rule semi-thue systems. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications*, (pp. 338–352). LNCS 3467.

Nakamura, K., & Matsumoto, M. (2005). Incremental learning of context-free grammars based on bottom-up parsing and search. *Pattern Recognition*, *38*(9), 1384–1392.

Nevill-Manning, C., & Witten, I. (1997). Identifying hierarchical structure in sequences: a linear-time algorithm. *Journal of Artificial Intelligence Research*, *7*, 67–82.

Nivat, M. (1970). On some families of languages related to the dyck language. In *Proc. 2nd Annual Symposium on Theory of Computing*.

O'Donnell, M. J. (1977). *Computing in Systems Described by Equations*, vol. 58 of *LNCS*. Springer.
Oncina, J., & García, P. (1992). Identifying regular languages in polynomial time. In H. Bunke, (Ed.), *Advances in Structural and Syntactic Pattern Recognition*, vol. 5 of *Series in Machine Perception and Artificial Intelligence*, (pp. 99–108). World Scientific.
Petasis, G., Paliouras, G., Karkaletsis, V., Halatsis, C., & Spyropoulos, C. (2004). E-grids: Computationally efficient grammatical inference from positive examples. *Grammars*, *7*, 69–110.
Rao, M. R. K. Krishna. (2004). Inductive inference of term rewriting systems from positive data. In *Agorithmic Learning Theory*, (pp. 69–82).
Rico-Juan, J. R., Calera-Rubio, J., & Carrasco, R. C. Stochastic *k*-testable tree languages and applications. In Adriaans, P., Fernau, H., & van Zaannen, M. (Eds.), (2002). *Grammatical inference: Algorithms and applications*, In *Proceedings of ICGI '02*, vol. 2484 of *LNAI*, (pp. 199–212). Berlin, Heidelberg: Springer-Verlag.
Sakakibara, Y. (1990). Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, *76*, 223–242.
Sakakibara, Y. (1992). Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, *97*, 23–60.
Sakakibara, Y. (1997). Recent advances of grammatical inference. *Theoretical Computer Science*, *185*, 15–45.
Sakakibara, Y., & Kondo, M. (1999). Ga-based learning of context-free grammars using tabular representations. In *Proceedings of 16th International Conference on Machine Learning (ICML-99)* (pp. 354–360).
Sénizergues, G. (1998). A polynomial algorithm testing partial confluence of basic semi-thue systems. *Theor. Comput. Sci.*, *192*(1), 55–75.
Starkie, B., Coste, F., & van Zaanen, M. (2004). Omphalos context-free language learning competition. The Omphalos competition is at the address: http://www.irisa.fr/Omphalos/
Takada, Y. (1988). Grammatical inference for even linear languages based on control sets. *Information Processing Letters*, *28*(4), 193–199.
Thollard, F., Dupont, P., & de la Higuera, C. (2000). Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *Proc. 17th International Conf. on Machine Learning*, (pp. 975–982). San Francisco, CA: Morgan Kaufmann.
Togashi, A., & Noguchi, S. (1990). Inductive inference of term rewriting systems realizing algebras. In *Algorithm Learning Theory*, (pp. 411–424).
Vanlehn, K., & Ball, W. (1987). A version space approach to learning context-free grammars. *Machine Learning Journal*, *2*, 39–74.
Wolf, G. (1978). Grammar discovery as data compression. In *Proceedings of AISB/GI Conference on Artificial Intelligence*, (pp. 375–379), Hamburg.
Yokomori, T. (2003). Polynomial-time identification of very simple grammars from positive data. *Theor. Comput. Sci.*, *1*(298), 179–206.