Fundamenta Informaticae XXI (2001) 1001–1026
DOI 10.3233/FI-2012-0000
IOS Press

Grammatical Inference of Lambda-Confluent Context Rewriting Systems

Peter Černo*
Department of Software and Computer Science Education
Charles University in Prague, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 PRAHA 1, Czech Republic
Tel.: +420 221 914 217
Fax: +420 221 914 281
petercerno@gmail.com

Abstract. Although a lot of methods have been proposed to learn regular languages, learning more complex language classes is still a challenging task. One attractive approach is to consider non-classical formalisms that provide alternative ways to represent interesting classes of languages and give rise to new promising learning techniques. To this end we use the so-called context rewriting systems, which are defined as string-rewriting systems extended by contexts, and propose a novel learning algorithm for inferring various restricted classes of context rewriting systems. We show that, under certain conditions, it is possible to identify in polynomial time (but not polynomial data) any target λ-confluent context rewriting system with minimal width of instructions from informant. We discuss the complexity of the learning algorithm, relate the considered language classes to Chomsky hierarchy and finally raise some open questions and further research directions.

Keywords: grammatical inference, context rewriting systems, formal languages, λ-confluence.

Address for correspondence: Department of Software and Computer Science Education
Charles University in Prague, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 PRAHA 1, Czech Republic
*This work was partially supported by the Grant Agency of Charles University under Grant-No. 272111/A-INF/MFF and by the Czech Science Foundation under Grant-No. P103/10/0783 and Grant-No. P202/10/1333. This paper is an extended version of the paper [8].

1. Introduction

Grammatical inference is concerned with finding a description (representation or model) of a language when given only some information about the language, e.g. some words of the language, the structure of the language, counter-examples or access to an oracle. One common root of many of the formalizations used in grammatical inference is Gold's model of *identification in the limit*. E. Mark Gold [18] can be justly called a pioneer in this field thanks to his seminal paper on "Language identification in the limit." The question of how children learn languages was part of his motivation; however, his focus was on theoretical investigations. In Gold's [18] model, a language is a set of strings over some fixed finite alphabet. In the following, we will use the term *target language* to refer to a language which has to be learned. Assume, for a moment, that there is some fixed system of representations of languages, called a *hypothesis space*, such that at least one correct model for the target language is contained in that system. Gold considers a *learner* to be an algorithmic device which is given examples, step by step, and which, in each of the infinitely many steps, returns a hypothesis. A learner is considered successful if after some step the learner returns the same hypothesis over and over again for all future steps and the hypothesis it converges to is a correct representation for the target language in the underlying hypothesis space. Gold considers both *learning from text*, i.e. the case when only positive examples (strings belonging to the target language) are available for the learner, as well as *learning from informant*, i.e. the case when both positive and negative examples (strings labeled according to whether or not they belong to the target language) are available.

When concerned about *efficient* grammatical inference [15] the main definitions of polynomial inference have been proposed by Pitt and Angluin. In his seminal paper [30] Pitt discusses different possible ideas as to what polynomial complexity for the problem of identification in the limit of *deterministic finite automata* (DFA) should be. Pitt proposes that for an identification algorithm to be polynomial it must have polynomial *update time*, and also make a polynomial number of *implicit errors* (in the size of the automaton). An implicit error is made when the current hypothesis does not agree with a new example. This definition alas is shown (by Pitt) to be very restrictive, in the sense that no superclasses of regular languages allow polynomial time inference.

Another model of learning has been proposed by Angluin [1] where the presentation of the language can be controlled by asking queries to an oracle. There are two types of queries: the *membership queries* (a string is proposed to the oracle which returns its correct classification), and *equivalence queries*, where a representation is proposed to the oracle, which either accepts it as a correct representation of the language to be inferred, or returns a counter-example: a string from the symmetrical difference of the proposed language and the target one. This is known as the *MAT* model (*minimally adequate teacher*). With time complexity depending on the size of the automaton to be inferred and the length of the longest counter-example returned by the oracle, Angluin proves that DFA can be identified in polynomial time with membership queries and equivalence queries. However, both of these queries are necessary: neither membership queries alone, nor equivalence queries alone allow polynomial inference.

Since both of these models show that even DFA can't be inferred in polynomial time (unless strong oracles are used), we follow another theoretical framework provided by Gold [19]: he presented a model for identification from given data, where a sample of labeled strings (S^+, S^-), with S^+ a set of positive instances, and S^- a set of negative instances, is presented to the inference algorithm that must return a representation compatible with (S^+, S^-). The further conditions are that for each language there exists a *characteristic sample* with which the algorithm returns a correct representation, and this must be

monotonous in the sense that if correctly labeled examples are added to the characteristic set, then the algorithm infers the same language. These conditions insure identification, and it is easy to see that a class of representations is identifiable in the limit from given data if and only if it is identifiable in the limit from a complete presentation of examples.

Most of the work in grammatical inference has focused on the area of learning regular languages. Now there are several good algorithms to deal with the case of learning from an informant [11]. On the other hand, moving up the Chomsky hierarchy gives rise to some difficult problems. Characteristic samples (needed for identification) may not be of polynomial size, and it is believed, for instance, that context-free grammars cannot be identified, in the framework of active learning, from a minimum adequate teacher. In [12] Alexander Clark emphasized the importance of learnability of the representation classes of formal languages. He proposed that one way to build learnable representations is by making them *objective* or *empiricist*: the structure of the representation should be based on the structure of the language. He illustrated this approach with three classes corresponding to the lowest three levels of the Chomsky hierarchy. All these classes were efficiently learnable under suitable learning paradigms. In defining these representation classes the author followed a simple slogan: "Put learnability first!" It means that we should design representations from the ground to be learnable. Rather than defining a representation, and then defining a function from the representation to the language, we should start by defining the map from the language to the representation. The basic elements of such formalism, whether they are states in an automaton, or non-terminals in a phrase-structure grammar, must have a clear definition in terms of sets of strings. In the conclusive remarks the author suggested that the representations, which are both efficiently learnable and capable of representing mildly context-sensitive languages seem to be good candidates for models of human linguistic competence.

Another promising alternative to tackle the learning barriers is to choose a non-classical representation. One approach is to consider models which do not use auxiliary elements at all. Typical representatives of this category are *contextual grammars* in [23], *pure context-free grammars* in [24], and *locally testable languages* in [25, 32]. In this paper, we use the so-called *context rewriting systems* (CRS), which are similar to *string-rewriting systems* (SRS) except that the rewriting rules of CRS are extended by *contexts* that limit their application. In the literature SRS are also known as *semi-Thue systems* as they were invented in 1914 by Axel Thue. Since their inception a lot of attention has been paid to the theory of SRS [5] and a lot of models used today in formal language theory are based on some kind of string-rewriting. For instance, SRS play a central role in the definition of the so called *Church-Rosser languages* (CRL) [26]. A language is called a *Church-Rosser language*, if it consists of those strings that, placed within the context of certain auxiliary strings t_1 and t_2, reduce to a certain "accepting" symbol with respect to a finite, length-reducing and confluent string-rewriting system. Apart from the final symbol and the symbols occurring in the contexts t_1 and t_2, also other non-terminal symbols are allowed in this definition. Although the rewriting process with respect to the string-rewriting system considered is inherently non-deterministic, the confluence of the system ensures that each reduction sequence will lead to the same result. The natural generalization of this definition led to the development of the broad concept of the so-called *McNaughton families* [4]. By placing various restrictions on the finite string-rewriting systems used we will obtain different families of languages. We refer the interested reader to the paper [4] where these families are studied in detail. Our approach is reminiscent of the *delimited string-rewriting systems* (DSRS) introduced in [17], which are expressive enough to define a nontrivial class of languages containing all regular languages and some context-free languages. In [17] there was presented a novel algorithm *LARS* (Learning Algorithm for Rewriting Systems) which identifies a large

subclass of these languages in polynomial time. In fact, a simplified version of LARS [16] identifies any delimited string-rewriting system in the limit. The main difference between delimited string-rewriting systems and context rewriting systems is that delimited string-rewriting systems use a specific order relation over the set of all terms and rules in order to make always only one single rule eligible for application for any given input string. This makes them an efficient (often linear) parsing device for strings with the membership problem decidable in polynomial time. Context rewriting systems, on the other hand, are nondeterministic and do not use any ordering. To test whether a word w belongs to the language $L(M)$ accepted by a given CRS M, one has to check whether w can be reduced to the empty word λ by a sequence of applications of the instructions of M. If we assume that the instructions are length-reducing, then every such sequence has at most $|w|$ steps. But as there could be several instructions that are applicable to the same word, or there could be several places at which a given instruction can be applied, all such sequences must be checked. It would be much better if we could assume that each and every sequence of applications of instructions of M reduces w to λ, if $w \in L(M)$. In this case we could restrict ourselves only to leftmost sequences of reductions, and accordingly, membership in $L(M)$ would be decidable deterministically in time $O(|w|)$. We call such context rewriting systems *λ-confluent* and show that the proposed learning algorithm can be used to identify λ-confluent context rewriting system in the limit from informant. Although λ-confluence can be useful in practical applications, in [29] it has been shown that λ-confluence is not even recursively enumerable for very restricted context rewriting systems. Nevertheless, we will show that the proposed learning algorithm works in the limit even if we do not check the λ-confluence of the inferred model.

The proposed inference algorithm can work with many different types of restricted context-rewriting systems (including λ-confluent systems). One special type of CRS that we consider in this paper is the so-called *clearing restarting automaton*, which, based on a limited context, can only delete a substring of the current content of its tape. We use clearing restarting automata to prove some lower bounds on the complexity of learning. Clearing restarting automata and other related models are studied intensively in [9]. As their name suggests, clearing restarting automata have roots in the broad family of *restarting automata* [21], which were introduced as a tool for modeling some techniques used for natural language processing. The interested reader is referred to an excellent survey [27].

The paper has the following structure. In Section 2 we fix the notation and introduce *context rewriting systems* as our main framework for all models considered in this paper. In Section 3 we discuss the paradigm of polynomial identification, as defined in [19, 15]. In Section 4 we introduce our learning algorithm built on this learning paradigm. In Section 5 we relate clearing restarting automata to the model of deterministic finite automata. Conclusions are presented in Section 6.

This paper is an extended version of the paper [8]. In this paper we have improved the notation and added several sections concerning primarily the complexity of learning.

2. Theoretical Background

We assume that the reader is familiar with the basic concepts of formal language and automata theory. As our reference concerning this field we use the monograph [20]. An *alphabet* is a finite nonempty set. The elements of an alphabet Σ are called *letters* or *symbols*. A *word* or *string* over an alphabet Σ is a finite sequence consisting of zero or more letters of Σ, whereby the same letter may occur several times. The sequence of zero letters is called the *empty word*, written λ. The set of all words (all nonempty

words, respectively) over an alphabet Σ is denoted by Σ^* (Σ^+, respectively). If x and y are words over Σ, then so is their *catenation* (or *concatenation*) xy (or $x \cdot y$), obtained by juxtaposition, that is, writing x and y one after another. Catenation is an associative operation and the empty word λ acts as an identity: $w\lambda = \lambda w = w$ holds for all words w. Because of the associativity, we may use the notation w^i in the usual way. By definition, $w^0 = \lambda$.

Let u be a word in Σ^*, say $u = a_1 \dots a_n$ with $a_i \in \Sigma$. We say that n is the *length* of u and we write $|u| = n$. The sets of all words over Σ of length k, or at most k, are denoted by Σ^k and $\Sigma^{\leq k}$, respectively. By $|u|_a$, for $a \in \Sigma$, we denote the total number of occurrences of the letter a in u. The *reversal* (*mirror image*) of u, denoted u^R, is the word $a_n \dots a_1$. Finally a *factorization* of u is any sequence u_1, ..., u_t of words such that $u = u_1 \cdots u_t$.

For a pair u, v of words we define the following relations: u is a *prefix* of v, if there exists a word z such that $v = uz$; u is a *suffix* of v, if there exists a word z such that $v = zu$; and u is a *factor* (or *subword*) of v, if there exist words z and z' such that $v = zuz'$. Observe that u itself and λ are subwords, prefixes and suffixes of u. Other subwords, prefixes and suffixes are called *proper*.

Subsets, finite or infinite, of Σ^* are referred to as (*formal*) *languages* over Σ.

In formal language theory in general, there are two major types of mechanisms for defining languages: *acceptors* and *generators*. Acceptors are usually defined in terms of *automata*, which work as follows: they are given an input word and after some processing they either accept or reject this input word. For instance, the so-called *finite automata* consist of a finite set of internal states and a set of rules that govern the change of the current state when reading a given input symbol. The finite automaton reads a given input word from left to right starting in a specific *starting state*. After reading the input word it accepts only if it ends in one of its *accepting states*, otherwise it rejects. If the next state is always uniquely determined by the current state and the current input symbol, we say that the automaton is deterministic. More formally, we define a deterministic finite automaton as follows: A *deterministic finite automaton* (DFA) A is a quintuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of *states*, Σ is an *input alphabet*, $\delta : Q \times \Sigma \to Q$ is a *state transition function*, $q_0 \in Q$ is a *starting state*, and $F \subseteq Q$ is a set of *final states*. The language recognized by A is $L(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$, where δ^* is defined recursively as: $\delta^*(q, \lambda) = q$ and $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$ for all $q \in Q$, $w \in \Sigma^*$ and $a \in \Sigma$. Finite automata recognize the family of *regular languages*, which plays a central role in the whole formal language theory.

Generators, on the other hand, usually generate the language using some finite set of rules. Typically they are defined in terms of grammars. One of the most famous is the classical *Chomsky hierarchy* of grammars (and corresponding languages), which consists of *phrase-structure*, *context-sensitive*, *context-free*, and *regular* grammars (they are also called type 0, type 1, type 2, and type 3 grammars, respectively). A *context-free grammar* (CFG) is a quadruple $G = (\Sigma, V, P, S)$ 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 $\Rightarrow$. We denote by $L(G)$ the language $\{w \in \Sigma^* \mid S \Rightarrow^* w\}$.

In this paper we use the so-called *context rewriting systems* as a framework for all considered models.

Definition 2.1. (Context rewriting systems [9])
A *context rewriting system* (CRS for short) is a system $M = (\Sigma, \Gamma, \Phi)$, where Σ is an *input alphabet*, $\Gamma \supseteq \Sigma$ is a *working alphabet* not containing the special symbols ¢ and \$, called *sentinels*, and Φ is a finite set of *instructions* of the form:

$$(x, z \to t, y) ,$$

where x is called the *left context*, $x \in \{\lambda, ¢\} \cdot \Gamma^*$, y is called the *right context*, $y \in \Gamma^* \cdot \{\lambda, \$\}$, and $z \to t$ is called the *instruction-rule*, $z,t \in \Gamma^*$. The *width* of the instruction $\phi = (x, z \to t, y)$ is $|\phi| = |xzty|$. The *width* of the context rewriting system M is $|M| = \max_{\phi \in \Phi} |\phi|$ and the *size* of the context rewriting system M is $\mathsf{size}(M) = \sum_{\phi \in \Phi} |\phi|$. If the input alphabet and the working alphabet of M are known from the context, we use M and Φ interchangeably. We also use a shorter notation (Σ, Φ) to denote a CRS (Σ, Σ, Φ) without auxiliary symbols.

For arbitrary words $u,v,z,t \in \Gamma^*$, a word $w = uzv$ *can be rewritten* into utv (denoted as $uzv \vdash_M utv$ or $uzv \vdash_\Phi utv$) if and only if there exists an instruction $\phi = (x, z \to t, y) \in \Phi$ such that x is a suffix of $¢ \cdot u$ and y is a prefix of $v \cdot \$$. We often underline the rewritten part of the word w, and if the instruction ϕ is known we use $\vdash_M^{(\phi)}$ instead of $\vdash_M$, i.e., $u\underline{z}v \vdash_M^{(\phi)} utv$.

The relation $\vdash_M \subseteq \Gamma^* \times \Gamma^*$ is called the *rewriting relation*.

Let $l \in \{\lambda, ¢\} \cdot \Gamma^*$, and $r \in \Gamma^* \cdot \{\lambda, \$\}$. A word $w = uzv$ *can be rewritten in the context* (l,r) into utv (denoted as $uzv \vdash_M utv$ *in the context* (l,r)) if and only if there exists an instruction $\phi = (x, z \to t, y) \in \Phi$, such that x is a suffix of $l \cdot u$ and y is a prefix of $v \cdot r$. Each definition that uses the rewriting relation $\vdash_M$ can be *relativized* to any context (l,r). Unless told otherwise, we will use the *standard context* $(l,r) = (¢, \$)$.

A context rewriting system $M = (\Sigma, \Gamma, \Phi)$ is called *simplified* if for every $\phi = (x, z \to t, y) \in \Phi$: $z \not\vdash^*_{\Phi - \{\phi\}} t$ in the context (x,y).

The *language* associated with M is defined as $L(M) = \{w \in \Sigma^* \mid w \vdash_M^* \lambda\}$, where $\vdash_M^*$ is the reflexive and transitive closure of $\vdash_M$. Note that, by definition, $\lambda \in L(M)$.

Example 2.2. ([8])
Let $M = (\Sigma, \Phi)$ be a CRS with $\Sigma = \{a,b\}$ and Φ consisting of the following two instructions:

$$(1) \quad (a, ab \to \lambda, b),$$
$$(2) \quad (¢, ab \to \lambda, \$).$$

Then we have $aaa\underline{ab}bbb \vdash_M^{(1)} aa\underline{ab}bb \vdash_M^{(1)} a\underline{ab}b \vdash_M^{(1)} \underline{ab} \vdash_M^{(2)} \lambda$, which means that $aaaabbbb \vdash_M^* \lambda$. So the word $aaaabbbb$ is accepted by M. It is easy to see that M recognizes the language $L(M) = \{a^n b^n \mid n \geq 0\}$.

Example 2.3. Let $M = (\Sigma, \Phi)$ be a CRS with $\Sigma = \{a,b\}$ and Φ consisting of only one instruction $(\lambda, ab \to \lambda, \lambda)$. Then we have $aba\underline{ab}bab \vdash_M ab\underline{ab}ab \vdash_M \underline{ab}ab \vdash_M \underline{ab} \vdash_M \lambda$, which means that $abaabbab \vdash_M^* \lambda$. So the word $abaabbab$ is accepted by M. It is easy to see that M recognizes the *Dyck language* of correct parentheses over Σ, i.e., the language generated by S in the grammar: $S \to TS \mid \lambda$; $T \to aSb$.

Definition 2.4. Let $M = (\Sigma, \Gamma, \Phi)$ be a context rewriting system. We say that M is:

1. *length-reducing*, if for each instruction $\phi = (x, z \to t, y) \in \Phi : |z| > |t|$.
2. *confluent* if, for all $u,v,w \in \Gamma^*$, $u \vdash_M^* v$ and $u \vdash_M^* w$ imply that there exists some $z \in \Gamma^*$ such that $v \vdash_M^* z$ and $w \vdash_M^* z$ hold.
3. *λ-confluent* if, for all $u,v \in \Gamma^*$, $u \vdash_M^* \lambda$ and $u \vdash_M^* v$ imply that $v \vdash_M^* \lambda$.

All context rewriting systems have the following basic property.

Lemma 2.5. (Error Preserving Property, [27])
Let $M = (\Sigma, \Gamma, \Phi)$ be a CRS and u,v be two words over Γ. If $u \vdash_M^* v$ and $u \notin L(M)$, then $v \notin L(M)$.

All λ-confluent context rewriting systems can be characterized in the following way.

Lemma 2.6. (Correctness Preserving Property, [27])
Let $M = (\Sigma, \Gamma, \Phi)$ be a CRS. M is λ-confluent if and only if for all $u, v \in \Gamma^*$ the following property holds: if $u \vdash_M^* v$ and $u \in L(M)$, then $v \in L(M)$.

It is easy to see that general CRS can simulate any type 0 grammar (according to the Chomsky hierarchy [20]). Hence we will not study CRS in their general form, since they are too powerful (they can represent all recursively enumerable languages). Instead, we will always put some restrictions on the instructions and then study such restricted models.

Definition 2.7. (Restrictions [8])
In this paper we consider only length-reducing context rewriting systems without auxiliary symbols. In addition, we consider two types of restrictions: *local* and *global* restrictions. *Local restrictions* (1, 2, 3) restrict each instruction individually (in other words, the decision whether the instruction satisfies a local restriction does not depend on other instructions). *Global restrictions* (4) restrict the whole set of instructions.

1. We can restrict the *length of contexts* to a positive integer constant k. More precisely, we can restrict each instruction $(x, z \to t, y)$ of a CRS $M = (\Sigma, \Gamma, \Phi)$ to satisfy the following constraints: $x \in LC_k := \Gamma^k \cup \{¢\} \cdot \Gamma^{\leq k-1}$ and $y \in RC_k := \Gamma^k \cup \Gamma^{\leq k-1} \cdot \{\$\}$.

   We also include a special case $k = 0$. In this case we require that each instruction $(x, z \to t, y)$ of a CRS $M = (\Sigma, \Gamma, \Phi)$ satisfies: $x = y = \lambda$.

   In addition, we use a special notation $k = \cdot$, if the length of contexts k is not specified. In this case we define $LC_k = \{\lambda, ¢\} \cdot \Gamma^*$ and $RC_k = \Gamma^* \cdot \{\lambda, \$\}$.

   If a context rewriting system $M = (\Sigma, \Gamma, \Phi)$ satisfies the above restrictions, then we call such system M a *k-context rewriting system* (k-CRS for short). We extend this notation to all classes derived from context rewriting systems: If $\mathscr{M}$ is a class of context rewriting systems, then k-$\mathscr{M}$ denotes the class of all context rewriting systems $M = (\Sigma, \Gamma, \Phi) \in \mathscr{M}$ such that, for every instruction $(x, z \to t, y) \in \Phi$: $x \in LC_k$ and $y \in RC_k$.

   Naturally, if we increase the length of contexts used in instructions of a context rewriting system, we do not decrease their expressiveness.

2. We can restrict the *width* of a context rewriting system $M = (\Sigma, \Gamma, \Phi)$ to be bounded from above by a positive integer constant l. In that case every instruction $\phi = (x, z \to t, y) \in \Phi$ satisfies the following constraint: $|\phi| = |xzty| \leq l$.

   We call such system M a context rewriting system *with maximal width* l.

   If $\mathscr{M}$ is a class of context rewriting systems, then $(\cdot, l)$-$\mathscr{M}$ denotes the class of all context rewriting systems $M \in \mathscr{M}$ such that, for every instruction $\phi \in \Phi$: $|\phi| \leq l$. Similarly, (k, l)-$\mathscr{M}$ denotes the class of all context rewriting systems $M \in k$-$\mathscr{M}$ such that, for every instruction $\phi \in \Phi$: $|\phi| \leq l$.

3. We can restrict the *instruction-rules* of a context rewriting system $M = (\Sigma, \Gamma, \Phi)$. There are too many possibilities how to restrict instruction-rules, so we list only few examples. We can restrict each instruction $\phi = (x, z \to t, y)$ of a CRS M to satisfy:

(a) $t = \lambda$.

(b) t is a subword of z.

(c) t is at most one letter, i.e., $|t| \leq 1$.

4. Finally, we can restrict the context rewriting system to be λ-confluent. Notice that this is a global restriction affecting the whole *set of instructions*.

For each combination of the above restrictions we get a different class of context rewriting systems $\mathscr{M}$ with possibly different properties and expressiveness. By $\mathscr{L}(\mathscr{M})$ we denote the corresponding class of languages, i.e., $\mathscr{L}(\mathscr{M}) = \{L(M) \mid M \in \mathscr{M}\}$.

Definition 2.8. ([8])
Let $\mathscr{M}$ be a class of CRS restricted according to Definition 2.7. We say that $M = (\Sigma, \Phi) \in \mathscr{M}$ has:

1. *Minimal set of instructions*, if for every $N = (\Sigma, \Psi) \in \mathscr{M}$, $\Psi \subset \Phi \Rightarrow L(N) \neq L(M)$.

2. *Minimal width*, if for every $N = (\Sigma, \Psi) \in \mathscr{M}$, $|N| < |M| \Rightarrow L(N) \neq L(M)$.

By using the framework of context rewriting systems we can define many interesting classes that have been studied intensively in several papers. In the following, we discuss *clearing*, Δ-*clearing*, *subword-clearing* and *limited-context restarting automata*.

Definition 2.9. (Derived Classes)

1. A *clearing restarting automaton* [9] (cl-RA for short) is a CRS $M = (\Sigma, \Phi)$, where for each instruction $\phi = (x, z \to t, y) \in \Phi$: $z \in \Sigma^+$ and $t = \lambda$. Since t is always the empty word, we use the short notation $\phi = (x, z, y)$.

2. A Δ-*clearing restarting automaton* [9] (Δcl-RA for short) is a CRS $M = (\Sigma, \Gamma, \Phi)$, where $\Gamma = \Sigma \cup \{\Delta\}$, $\Delta \notin \Sigma$, and for each instruction $\phi = (x, z \to t, y) \in \Phi$: $z \in \Gamma^+$ and $t \in \{\lambda, \Delta\}$.

3. A *subword-clearing restarting automaton* [7] (scl-RA for short) is a CRS $M = (\Sigma, \Phi)$, where for each instruction $\phi = (x, z \to t, y) \in \Phi$: $z \in \Sigma^+$ and t is a subword of z, such that $|t| < |z|$.

4. A *limited-context restarting automaton* [2, 3, 28] (lc-RA for short) of type $\mathscr{R}_0$ is a CRS $M = (\Sigma, \Gamma, \Phi)$, where each instruction $\phi = (x, z \to t, y) \in \Phi$ is length-reducing. For lc-RA of type $\mathscr{R}_1$ we require, in addition, that $|t| \leq 1$, and for lc-RA of type $\mathscr{R}_2$ we require, in addition, that $|t| \leq 1$, $x \in \{¢, \lambda\}$, and $y \in \{\$, \lambda\}$.

Remark 2.10. Speaking about a cl-RA M (Δcl-RA, scl-RA, lc-RA M, respectively) we use "automata terminology," e.g., we say that M *accepts* a word w if $w \in L(M)$. By definition, each cl-RA (Δcl-RA, scl-RA, lc-RA) accepts λ. If we say that a cl-RA (Δcl-RA, scl-RA, lc-RA) M *recognizes* (or *accepts*) a language L, we always mean that $L(M) = L \cup \{\lambda\}$. This implicit acceptance of the empty word can be avoided by a slight modification of context rewriting systems, but in principle, we would not get a more powerful model. We simply ignore the empty word in this setting.

Clearing restarting automata are studied in [9]. In this paper they play a role of the most restricted model and we will use them to prove some lower bounds on the complexity of learning. Although clearing restarting automata are very restricted, they can recognize all regular languages, some context-free languages and even some non-context-free languages. However, there are some context-free languages that are outside the class of languages accepted by clearing restarting automata. For instance, the language $L = \{a^n cb^n \mid n \geq 0\}$ is not recognized by any clearing restarting automaton. On the other hand, it can be easily shown that this language is recognized by the subword-clearing restarting automaton $M = (\{a,b,c\}, \Phi)$ with $\Phi = \{(a, acb \to c, b), (¢, acb \to \lambda, \$)\}$. In general, however, not all context-free languages can be recognized by subword-clearing restarting automata [8]. Interestingly, in [10] it has been shown that $\mathsf{CFL} \subseteq \mathscr{L}(\Delta\mathsf{cl\text{-}RA})$, thus only one auxiliary symbol Δ is needed to recognize all context-free languages.

Obviously, limited-context restarting automaton (lc-RA) of type $\mathscr{R}_1$ is a proper extension of the clearing restarting automaton, while lc-RA of type $\mathscr{R}_2$ is incomparable to the clearing restarting automaton. In [28] limited context restarting automata and their confluent versions are put into the context of *McNaughton families of languages* [4], relating the classes of languages accepted by these automata in particular to the class GCSL of *growing context-sensitive languages* [6, 14] and to the class CRL of *Church-Rosser languages* [26].

The decidability of λ-confluence for clearing restarting automata and limited-context restarting automata is studied in [29]. It turns out that λ-confluence is not even recursively enumerable for clearing restarting automata and that it is decidable in double exponential time for limited-context restarting automata of type $\mathscr{R}_2$. Although the undecidability of λ-confluence for clearing restarting automata might seem prohibitive for learning λ-confluent CRS, we will see that we do not need to verify λ-confluence at all. Interestingly, a stronger notion of *confluence* is decidable for all limited-context restarting automata of type $\mathscr{R}_0$.

3. Learning

We now turn to our learning problem. We follow the approach used in [17].

Definition 3.1. ([17])
Let $\mathscr{L}$ be a class of languages represented by some class $\mathscr{M}$ of models.

1. A *sample* S for a language $L \in \mathscr{L}$ is a pair (S^+, S^-) 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, denoted as $\mathsf{size}(S)$, is the sum of the lengths of all the strings in S^+, S^-. Formally, $\mathsf{size}(S) = \mathsf{size}(S^+) + \mathsf{size}(S^-)$, where $\mathsf{size}(T) = \sum_{w \in T} |w|$.
2. An $(\mathscr{L}, \mathscr{M})$-learning algorithm $\mathscr{A}$ is a program that takes as input a sample and outputs a representation from $\mathscr{M}$.

We use the paradigm of polynomial identification, as defined in [19, 15]. In this paradigm we require 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 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 by using the size of a minimal *characteristic* sample, whose presence should ensure identification.

Definition 3.2. (Polynomial Identification [17])
A class $\mathscr{L}$ of languages is *identifiable in polynomial time and data* for a class $\mathscr{M}$ of models if and only if there exists an algorithm $\mathscr{A}$ and two polynomials $\alpha(\cdot)$ and $\beta(\cdot)$ such that:

1. Given a sample $S = (S^+, S^-)$ for $L \in \mathscr{L}$ of size m, $\mathscr{A}$ returns a model (hypothesis) $M \in \mathscr{M}$ in $O(\alpha(m))$ time and M is *consistent with* S, i.e. $S^+ \subseteq L(M)$ and $S^- \cap L(M) = \emptyset$.

2. For each model $M \in \mathscr{M}$ representing the language $L \in \mathscr{L}$, there exists a finite *characteristic sample* $S_0 = (S_0^+, S_0^-)$ of size at most $O(\beta(\mathsf{size}(M)))$ such that, on all samples $S = (S^+, S^-)$ for L that verify $S_0^+ \subseteq S^+$ and $S_0^- \subseteq S^-$, $\mathscr{A}$ returns a model $N \in \mathscr{L}$ which is equivalent to M.

4. Learning Algorithm

In this section we propose a general learning algorithm for inferring various restricted context rewriting systems from *informant* and show that, under certain conditions, it is possible to identify any hidden target (ordinary or λ-confluent) context rewriting system in the limit by using this algorithm.

In the following, the term *model* refers to any context rewriting system $M \in \mathscr{M}$, where $\mathscr{M}$ is a fixed class of context rewriting systems restricted according to Definition 2.7 (our hypothesis space). Our focus will be on the *polynomial identification* [17] of the corresponding class of languages $\mathscr{L} = \{L(M) \mid M \in \mathscr{M}\}$ as defined in Section 3. In other words, we would like to design a $(\mathscr{L}, \mathscr{M})$-learning algorithm $\mathscr{A}$ such that both conditions of Definition 3.2 are satisfied. The Condition 1 of Definition 3.2 is easily satisfied by returning a model $M = (\Sigma, \Phi)$ with $\Phi = \{(¢, w \to \lambda, \$) \mid w \in S^+, w \neq \lambda\}$, when given a sample $S = (S^+, S^-)$. In this case $L(M) = S^+$ and, in addition, M is confluent, and thus also λ-confluent. Unfortunately, as the language $L(M)$ is always finite, only finite languages can be learned in this way. We need a more sophisticated algorithm that is able to deal also with infinite languages. To this end we first present in Section 4.1 an auxiliary inference procedure (Algorithm 1) that tries to infer a model consistent with the given sample such that the width of the model is restricted from above by some constant. This restriction is useful because it leads to models that can generalize over the presented sample. Then, in Section 4.2 we use this auxiliary inference procedure as a component in our final learning algorithm.

There is one additional subtlety involved in our learning schema that we need to discuss – one of the (optional) input parameters of our learning algorithm is the length of contexts k. This parameter, if specified, restricts the hypothesis space $\mathscr{M}$ to the class k-$\mathscr{M}$ (see Definition 2.7 for details). The class k-$\mathscr{M}$ allows only instructions, where the length of contexts is constrained to be *exactly* k letters long. The only exception is, of course, the case when the contexts contain a sentinel. In that case they can be shorter than k letters. An informal intuition behind constraining the contexts is as follows. The *context* (x, y) used in the instruction $\phi = (x, z \to t, y)$ limits the applicability of the *instruction-rule* $z \to t$, so that we can rewrite the word z to t only if z is placed inside the context (x, y). The longer the contexts x and y are the smaller is the "chance" that the instruction-rule $z \to t$ will be applied in a "wrong" context. The main motivation for constraining the contexts arises from the famous n-gram model, which is often used, e.g., in *language modeling* or *statistical machine translation* [22]. By changing the length of contexts k we can influence the sparsity of the resulting model just as we can influence the sparsity of the n-gram model by specifying n. If we overestimate the length of contexts k, the resulting model may become too sparse and may not generalize well. On the other hand, underestimating k can lead to overgeneralization and divergence, as our target language may not be in the class $\mathscr{L}(k\text{-}\mathscr{M})$. It can be

an interesting research direction to investigate some heuristics for choosing the right length of contexts. But from the perspective of the identification in the limit we only need to choose long enough contexts, because it can be easily shown that for any class $\mathscr{M}$ of context rewriting systems restricted according to Definition 2.7 the following inclusions hold: $\mathscr{L}(0\text{-}\mathscr{M}) \subseteq \mathscr{L}(1\text{-}\mathscr{M}) \subseteq \mathscr{L}(2\text{-}\mathscr{M}) \subseteq \ldots$. Moreover, if we do not specify the length of contexts k, the proposed learning algorithm will figure it out in the limit.

4.1. Learning With Restricted Width

The problem we are interested in here can be best described as follows. Given a sample $S = (S^+, S^-)$ for some language $L \in \mathscr{L}$, we would like to find a model (hypothesis) $M \in (k,l)\text{-}\mathscr{M}$ consistent with S (i.e., $S^+ \subseteq L(M)$ and $S^- \cap L(M) = \emptyset$), where k is the *optional* length of contexts and $l \geq 1$ is the finite prescribed maximal width of instructions. We use the notation $k = \cdot$ if k is not specified, and we assume that $S^+ \cap S^- = \emptyset$ and $\lambda \in S^+$. As $(k,l)\text{-}\mathscr{M}$ is a finite class, we could enumerate all models $M \in (k,l)\text{-}\mathscr{M}$ and return the first model M consistent with S. There are, however, two obstacles in doing the inference in this way. First, there are, in general, double exponentially many models $M \in (k,l)\text{-}\mathscr{M}$ with respect to the width l. This follows easily from the fact that there are, in general, exponentially many instructions that have the width bounded from above by l and each model $M \in (k,l)\text{-}\mathscr{M}$ can be specified as a subset of these instructions. Second, λ-confluence is often undecidable. Since we are interested in polynomial identification as described in Section 3, we need to use a different apporach. The following Algorithm 1 is an auxiliary inference procedure that will be used as a core component in our learning algorithm.

Algorithm 1: Auxiliary inference procedure $\mathsf{Infer}_{\mathscr{M}}(S,k,l)$

Input : Sample $S = (S^+, S^-)$ over Σ, $S^+ \cap S^- = \emptyset, \lambda \in S^+$.
Length of contexts $k \geq 0$, or $k = \cdot$, if not specified.
Maximal width of instructions $l \geq 1$.
Output: A model $M \in (k,l)\text{-}\mathscr{M}$ consistent with S, or **Fail**.

1 $\Phi \leftarrow \mathsf{Assumptions}(S^+,k,l)$;
2 **while** $\exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash^{(\phi)} w_+$ **do**
3 $\Phi \leftarrow \Phi \setminus \{\phi\}$;
4 **if** $\mathscr{M}$ *is a class of* λ*-confluent models* **then**
5 **while** $\exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash^{(\phi)} w_-$ **do**
6 $\Phi \leftarrow \Phi \setminus \{\phi\}$;
7 $\Phi \leftarrow \mathsf{Simplify}(\Phi)$;
8 **if** $\mathsf{Consistent}(\Phi, S)$ **then**
9 **return** *Model M with the set of instructions* Φ;
10 **return Fail**;

Algorithm 1 deserves an explanation. First, in Step 1 the function $\mathsf{Assumptions}(S^+,k,l)$ returns some set of *instruction candidates*. We may assume that every returned instruction $\phi = (x, z \to t, y) \in \Phi$ is a *legal* instruction of the class $(k,l)\text{-}\mathscr{M}$ (i.e., ϕ satisfies all *local* restrictions of the class $(k,l)\text{-}\mathscr{M}$). Let us assume, for a moment, that Φ already contains all instructions of a hidden target model H. Then in Cycle 2–3 we gradually remove all instructions that allow a reduction from a negative sample to a positive

sample, i.e., instructions that violate the *error preserving property* (Lemma 2.5). If $\mathscr{M}$ is a class of λ-confluent models (i.e., if we want to infer a λ-confluent model) then in Cycle 5–6 we gradually remove all instructions that allow a reduction from a positive sample to a negative sample, i.e., instructions that violate the *correctness preserving property* (Lemma 2.6). These filtered instructions are definitely not in the set of instructions of the target model H, therefore their removal will not cause any problem. In Step 7 we remove redundant instructions in order to get a *simplified* model. In Step 8 we check whether the remaining set of instructions Φ is consistent with the given sample S, i.e.: (1) for all $w_+ \in S^+ : w_+ \vdash_\Phi^* \lambda$ and (2) for all $w_- \in S^- : w_- \nvdash_\Phi^* \lambda$. The condition (1) obviously holds if we obtained all instructions of the target model H in Step 1. However, the condition (2) is not guaranteed after Cycle 2–3. Similarly, if $\mathscr{M}$ is a class of λ-confluent models, it is not guaranteed that after Cycle 5–6 the resulting model will be λ-confluent. The success of the above algorithm, therefore, depends both on the initial instruction candidates obtained in Step 1 and on the given sample S. Nevertheless, we will show that if we have a "reasonable" function Assumptions, then there always exists a finite *characteristic sample* $S_0 = (S_0^+, S_0^-)$ corresponding to the target model H.

The time complexity of Algorithm 1 depends both on the time complexity of the function Assumptions in Step 1 and on the time complexity of the simplification function and the consistency check in Steps 7 and 8. As we will see below, there exist "correct" functions Assumptions (for any class $\mathscr{M}$ of CRS restricted according to Definition 2.7) that run in polynomial time. If the function Assumptions runs in polynomial time, then also the size of the set Φ is polynomial (with respect to the size of the input) and therefore also Cycles 2–3 and 5–6 run in polynomial time. If $\mathscr{M}$ is a class of λ-confluent models then the Steps 7 and 8 can be done in polynomial time, because we can restrict ourselves to the leftmost sequences of reductions. In most cases, however, we are not able to verify the λ-confluence of the inferred model, because λ-confluence is not even recursively enumerable for clearing restarting automata [29]. In addition, if $\mathscr{M}$ is a class of ordinary context rewriting systems then it is an open problem whether we can do Steps 7 and 8 in polynomial time. Nevertheless, in the case of λ-confluent models we do not need to verify the λ-confluence at all, as we will converge to the λ-confluent model in the limit.

In the following Definition 4.1 we define what we mean by the term *correct* function Assumptions.

Definition 4.1. (Correct assumptions [8])
We call the function Assumptions *correct* with respect to a class $\mathscr{M}$ of context rewriting systems restricted according to Definition 2.7, if:

1. For every set $S^+ \subseteq \Sigma^*$, the set of instructions $\Phi = \mathsf{Assumptions}(S^+, k, l)$ is finite. Additionally, every instruction $\phi = (x, z \to t, y) \in \Phi$ is a *legal* instruction of the class (k,l)-$\mathscr{M}$ (i.e., ϕ satisfies all local restrictions of the class (k,l)-$\mathscr{M}$).

2. For every $M = (\Sigma, \Phi) \in (k,l)$-$\mathscr{M}$ with *minimal set of instructions*, there exists a finite set $S_0^+ \subseteq L(M)$ such that for every $S^+ \supseteq S_0^+ : \Phi \subseteq \mathsf{Assumptions}(S^+, k, l)$.

Definition 4.2. (Monotone assumptions [8])
We call the function Assumptions *monotone* if:

1. For every $S_1^+ \subseteq S_2^+ \subseteq \Sigma^* : \mathsf{Assumptions}(S_1^+, k, l) \subseteq \mathsf{Assumptions}(S_2^+, k, l)$, and

2. For every $l_1 \leq l_2 : \mathsf{Assumptions}(S^+, k, l_1) \subseteq \mathsf{Assumptions}(S^+, k, l_2)$.

Example 4.3. ([8])
The most trivial example of a correct function Assumptions is to return all possible legal instructions ϕ of the class (k,l)-$\mathscr{M}$. The correctness and monotonicity follow easily. However, the number of returned instructions is, in general, exponentially large with respect to l, therefore such function would be of little interest in real applications.

Example 4.4. ([8])
In this example we define two very natural examples of *monotone* functions Assumptions that are correct with respect to any particular class $\mathscr{M}$ of context rewriting systems restricted according to Definition 2.7.

1. $\mathsf{WeakAssumptions}_{\mathscr{M}}(S^+,k,l) := \{\phi = (x,z \to t,y) \mid \phi$ is a legal instruction of the class (k,l)-$\mathscr{M}$ and $\exists w_1,w_2 \in S^+ : xzy$ is a subword of $¢w_1\$$ and xty is a subword of $¢w_2\$\}$.

   The basic intuition behind this function is the assumption that if both patterns xzy and xty occur in the set of positive samples as subwords, then it is justified to replace the word z with t in the context (x,y). Note that if k is specified then the more we increase the length of contexts k the smaller (or equal) the number of such patterns we will find in the set of positive samples. The contexts serve here as a *safety cushion* against the inference of incorrect instructions.

2. $\mathsf{StrongAssumptions}_{\mathscr{M}}(S^+,k,l) := \{\phi = (x,z \to t,y) \mid \phi$ is a legal instruction of the class (k,l)-$\mathscr{M}$ and $\exists w_1,w_2 \in S^+ : w_1 \vdash^{(\phi)} w_2\}$.

   This condition is even more restrictive than the previous one. It basically says that the instruction $\phi = (x,z \to t,y)$ is justified only in the case when there are positive samples $w_1,w_2 \in S^+$ such that we can obtain w_2 from w_1 by using this instruction.

Note that WeakAssumptions and StrongAssumptions are analogous to respectively k,l *substitutability* used in [31] and k,l *local substitutability* introduced in [13].

Lemma 4.5. ([8])
Both functions Assumptions from Example 4.4 are *monotone* and *correct* with respect to any fixed class $\mathscr{M}$ of context rewriting systems restricted according to Definition 2.7.

Proof:
It is easy to see that both functions from Example 4.4 are monotone and, in addition, for every set $S^+ \subseteq \Sigma^*$: $\mathsf{StrongAssumptions}_{\mathscr{M}}(S^+,k,l) \subseteq \mathsf{WeakAssumptions}_{\mathscr{M}}(S^+,k,l)$. Therefore, we only need to prove the correctness for the more restrictive function $\mathsf{StrongAssumptions}_{\mathscr{M}}$. The correctness of the function $\mathsf{WeakAssumptions}_{\mathscr{M}}$ will follow immediately. Let $M = (\Sigma,\Phi)$ be any CRS from (k,l)-$\mathscr{M}$ with minimal set of instructions. The minimality of M implies that for every instruction $\phi \in \Phi$ there exists a word $w_\phi \in L(M)$ such that the instruction ϕ is used in every accepting computation $w_\phi \vdash_M^* \lambda$ (otherwise we could remove the instruction ϕ from M). Without loss of generality we may assume that the instruction ϕ must be used in the very first step of every accepting computation $w_\phi \vdash_M^* \lambda$. (It does not mean that ϕ is the only applicable instruction. There may also be some other instructions applicable to w_ϕ, but they will definitely not lead to any accepting computation). Let us fix, for every $\phi \in \Phi$, some accepting computation $w_\phi \vdash^{(\phi)} w'_\phi \vdash_M^* \lambda$. Now define $S_0^+ := \bigcup_{\phi\in\Phi}\{w_\phi, w'_\phi\}$. Apparently $S_0^+ \subseteq L(M)$. Moreover, $\Phi \subseteq \mathsf{StrongAssumptions}_{\mathscr{M}}(S_0^+,k,l)$. The correctness follows easily from the monotonicity of the function $\mathsf{StrongAssumptions}_{\mathscr{M}}$. □

Algorithm 2 shows a possible implementation of the function $\mathsf{WeakAssumptions}_{\mathscr{M}}$ and Algorithm 3 shows a possible implementation of the function $\mathsf{StrongAssumptions}_{\mathscr{M}}$. Both of these algorithms have polynomial time complexity with respect to size of the input S^+, because there are at most quadratically many subwords in the set of positive samples (with respect to the $\mathsf{size}(S^+)$). In both of these algorithms we use the variable Map as a dictionary-like data structure (e.g., hash table).

Algorithm 4 shows a possible implementation of the function Simplify.

Algorithm 2: Implementation of $\mathsf{WeakAssumptions}_{\mathscr{M}}(S^+,k,l)$

Input : Set of positive samples S^+ over Σ, $\lambda \in S^+$.
Length of contexts $k \geq 0$, or $k = \cdot$, if not specified.
Maximal width of instructions $l \geq 1$.
Output: A set of legal instructions Φ of the class (k,l)-$\mathscr{M}$.

```
Φ ← ∅;
Map ← ∅;
foreach w+ ∈ S+ and each αxzyβ = ¢w+$ such that x ∈ LCk, y ∈ RCk, |xzy| ≤ l do
    Map[(x,y)] ← Map[(x,y)] ∪ {z};
foreach x,y,z,t, such that z ≠ t and z,t ∈ Map[(x,y)] do
    if φ = (x,z → t,y) is a legal instruction of (k,l)-M then
        Φ ← Φ ∪ {(x,z → t,y)};
return Φ;
```

Algorithm 3: Implementation of $\mathsf{StrongAssumptions}_{\mathscr{M}}(S^+,k,l)$

Input : Set of positive samples S^+ over Σ, $\lambda \in S^+$.
Length of contexts $k \geq 0$, or $k = \cdot$, if not specified.
Maximal width of instructions $l \geq 1$.
Output: A set of legal instructions Φ of the class (k,l)-$\mathscr{M}$.

```
Φ ← ∅;
Map ← ∅;
foreach w+ ∈ S+ and each αxtyβ = ¢w+$ such that x ∈ LCk, y ∈ RCk, |xty| ≤ l do
    Map[(x,y)] ← Map[(x,y)] ∪ {t};
DS+ ← ¢ · S+ · $ = {¢w+$ | w+ ∈ S+};
foreach w+ ∈ S+ and each αxzyβ = ¢w+$ such that x ∈ LCk, y ∈ RCk, |xzy| ≤ l do
    foreach t ∈ Map[(x,y)], t ≠ z do
        if φ = (x,z → t,y) is a legal instruction of (k,l)-M and αxtyβ ∈ DS+ then
            Φ ← Φ ∪ {(x,z → t,y)};
return Φ;
```

In the following Example 4.6 we illustrate how Algorithm 1 behaves when used on clearing restarting automata as the underlying class of models. We use the parameters $k = 1$ and $l = 6$.

Algorithm 4: Implementation of Simplify(Φ)

Input : Set of instructions Φ.
Output: Simplified set of instructions Φ.

```
while ∃φ = (x, z → t, y) ∈ Φ : z ⊢*_{Φ−{φ}} t in the context (x, y) do
    Φ = Φ − {φ}
return Φ;
```

Example 4.6. ([7])
Consider the class of clearing restarting automata (cl-RA) and the function $\mathsf{WeakAssumptions}_{\mathsf{cl\text{-}RA}}$ (from Example 4.4). Imagine that our goal is to infer a model for the language $L = \{a^n b^n \mid n \geq 0\}$. Let us try $S^+ = \{\lambda, ab, aabb\}$. First, we would like to estimate the set $\Phi = \mathsf{WeakAssumptions}_{\mathsf{cl\text{-}RA}}(S^+, l, k)$ for $k = 1$ and $l = 6$. The set of all subwords of the delimited positive samples ¢S^+\$ is: $SW^+ = \{\lambda$, ¢, a, b, \$, ¢\$, ¢a, aa, ab, bb, b\$, ¢aa, ¢ab, aab, abb, ab\$, bb\$, ¢ab\$, ¢aab, $aabb$, abb\$, ¢$aabb$, $aabb$\$, ¢$aabb$\$\}. An instruction (x,z,y), where $x \in LC_1 = \{a,b,¢\}$, $y \in RC_1 = \{a,b,\$\}$, $|z| > 0$ and $|xzy| \leq l$, is justified, according to the definition of $\mathsf{WeakAssumptions}_{\mathsf{cl\text{-}RA}}$, if and only if both $xzy, xy \in SW^+$. Thus, only the following reductions are justified: ¢$\underline{aa}$ ⊢ ¢a, $a\underline{ab} \vdash ab$, $a\underline{bb} \vdash ab$, $b\underline{b}\$ \vdash b\$$, ¢$\underline{ab}\$ \vdash$ ¢\$, $a\underline{abb} \vdash ab$, ¢$\underline{aabb}\$ \vdash$ ¢\$. Therefore, $\mathsf{WeakAssumptions}_{\mathsf{cl\text{-}RA}}(S^+, l, k) = \{(¢, \underline{a}, a), (a, \underline{a}, b), (a, \underline{b}, b), (b, \underline{b}, \$), (¢, \underline{ab}, \$), (a, \underline{ab}, b), (¢, \underline{aabb}, \$)\}$. Apparently, the following instructions are bad: $(¢, \underline{a}, a)$, $(a, \underline{a}, b)$, $(a, \underline{b}, b)$, $(b, \underline{b}, \$)$. We can remove them easily by taking $S^- = \{aab, abb\}$. We do not need to add anything else to S^+. The inference procedure (Algorithm 1) $\mathsf{Infer}_{\mathsf{cl\text{-}RA}}((S^+, S^-), l, k)$ will correctly output the model $N = (\{a,b\}, \{(¢, \underline{ab}, \$), (a, \underline{ab}, b)\})$ (after simplification). The function Simplify removes the instruction $(¢, \underline{aabb}, \$)$ from the inferred automaton as it can be simulated by the remaining instructions.

In the following Theorem 4.7 we state our first positive result concerning the grammatical inference of restricted (ordinary or λ-confluent) context rewriting systems.

Theorem 4.7. ([8])
Let $\mathscr{M}$ be a class of context rewriting systems restricted according to Definition 2.7 and let function Assumptions be correct with respect to $\mathscr{M}$. Then, for every model $M \in (k,l)\text{-}\mathscr{M}$ there exists a finite *characteristic sample* $S_0 = (S_0^+, S_0^-)$ such that, on all samples $S = (S^+, S^-)$ for $L(M)$ that verify $S_0^+ \subseteq S^+$ and $S_0^- \subseteq S^-$, Algorithm 1 $\mathsf{Infer}_{\mathscr{M}}(S,k,l)$ returns a model $N \in (k,l)\text{-}\mathscr{M}$ equivalent to M.

Proof:
Let $M = (\Sigma, \Phi) \in (k,l)\text{-}\mathscr{M}$. Without loss of generality we may assume that M has a minimal set of instructions. According to Definition 4.1, there exist $S_0^+ \subseteq L(M)$ such that for every $S^+ \supseteq S_0^+ : \Phi \subseteq \mathsf{Assumptions}(S^+, k, l)$. Let us initialize the set of negative samples S_0^- to the empty set. Let Θ denote the set of *all legal instructions* $\phi = (x, z \to t, y)$ of the class $(k,l)\text{-}\mathscr{M}$. (There are only finitely many such instructions, as $|\phi| \leq l$). There are two cases we need to consider.

1. The class $\mathscr{M}$ is not restricted to be λ-confluent.
2. The class $\mathscr{M}$ is restricted to be λ-confluent.

In the first case 1 we say that the instruction $\phi \in \Theta$ is *bad* if there exist $w_- \notin L(M), w_+ \in L(M) : w_- \vdash^{(\phi)} w_+$. In the second case 2 we say that the instruction $\phi \in \Theta$ is *bad* if there exist $w_- \notin L(M), w_+ \in L(M) :$

$(w_- \vdash^{(\phi)} w_+) \vee (w_+ \vdash^{(\phi)} w_-)$. In both cases, we call the pair of words (w_-, w_+) the *witness* for the bad instruction ϕ. Similarly, in the first case 1 we say that the instruction ϕ is *disabled* by (S_0^+, S_0^-) if there exist $w_- \in S_0^-, w_+ \in S_0^+ : w_- \vdash^{(\phi)} w_+$. In the second case 2 we say that the instruction ϕ is *disabled* by (S_0^+, S_0^-) if there exist $w_- \in S_0^-, w_+ \in S_0^+ : (w_- \vdash^{(\phi)} w_+) \vee (w_+ \vdash^{(\phi)} w_-)$. Now consider the following Algorithm 5:

Algorithm 5: Extension of Sample $S_0 = (S_0^+, S_0^-)$

Input : Sample $S_0 = (S_0^+, S_0^-)$.
Output: Extended sample $S_0 = (S_0^+, S_0^-)$.

1 **while** $\exists$ bad *instruction* $\phi \in \Theta$ *such that* ϕ *is not* disabled *by* (S_0^+, S_0^-) **do**
2 Let $w_- \notin L(M), w_+ \in L(M)$ be a *witness* for ϕ;
3 $S_0^+ \leftarrow S_0^+ \cup \{w_+\}$;
4 $S_0^- \leftarrow S_0^- \cup \{w_-\}$;

Every added pair w_+, w_- effectively disables at least one instruction from Θ, so the whole procedure is definitely finite. Now consider any finite set of positive samples $S^+ \supseteq S_0^+$ and any finite set of negative samples $S^- \supseteq S_0^-$ consistent with M. If we run the learning Algorithm 1 $\mathsf{Infer}_{\mathscr{M}}(S^+, S^-, k, l)$, then in Step 1 we obtain some set of instructions including all instructions from Φ. (This is guaranteed by the correctness of the function Assumptions). Note that no instruction from Φ is bad. In Cycle 2–3 (and also in Cycle 5–6, if $\mathscr{M}$ is a class of λ-confluent models) the Algorithm 1 gradually removes all bad instructions. (This is because the sets S_0^+ and S_0^- are constructed in such a way, that all bad instructions are disabled by (S_0^+, S_0^-).) After these cycles we are left only with correct instructions including all instructions from Φ, so the resulting model is apparently equivalent to the model M. In addition, if $\mathscr{M}$ is a class of λ-confluent models, the resulting model is also λ-confluent. This is because if we take any $w \in L(M)$, then for every instruction ϕ of the resulting model and for every w' such that $w \vdash^{(\phi)} w'$ we have $w' \in L(M)$. Otherwise the instruction ϕ would be bad. This means that the resulting model is *correctness-preserving*, and therefore also λ-confluent. □

Example 4.8. ([7])
Consider the class of clearing restarting automata (cl-RA) and the function $\mathsf{WeakAssumptions}_{\mathsf{cl\text{-}RA}}$ as in Example 4.6. If we use $M = (\{a,b\}, \{(¢, \underline{ab}, \$), (a, \underline{ab}, b)\})$ as our target model recognizing the language $L(M) = \{a^n b^n \mid n \geq 0\}$ and the parameters $k = 1$, $l = 6$, then it can be shown that the following sets of positive and negative samples: $S_0^+ = \{a^n b^n \mid 0 \leq n \leq 6\}$ and $S_0^- = \{aab, abb, aaab, abbb, aaaab, aaabb, aabbb, abbbb, aaaaab, aaaabb, aabbbb, abbbbb, aaaaabb, aabbbbb, aaaaaabb, aabbbbbb\}$ represent the corresponding characteristic sample from Theorem 4.7.

Theorem 4.7 only guarantees the existence of the characteristic sample $S_0 = (S_0^+, S_0^-)$. It does not provide any bounds on the size of the sample S_0. The size of the resulting characteristic sample S_0 depends not only on the target model $M \in (k,l)$-$\mathscr{M}$, but also on the used function Assumptions. In the following Example 4.9 we will see that, in general, the size of the smallest characteristic sample $S_0 = (S_0^+, S_0^-)$ can be exponentially large with respect to the size of the target model. We use clearing restarting automata and the function StrongAssumptions (from Example 4.4) to prove this fact.

Example 4.9. In this example we will construct a sequence of 2-clearing restarting automata: $M_0 = (\Sigma_0, \Phi_0), M_1 = (\Sigma_1, \Phi_1), \ldots$, such that for all $i \in \{0,1,2,\ldots\} : \Sigma_i = \{a_0, a_1, \ldots, a_i\}$, and $\Phi_i \subseteq \Phi_{i+1}$. We will prove that for each $i \in \{0,1,2,\ldots\}$ the size of the automaton M_i is polynomial with respect to i, and that there exists an instruction $\phi_i \in \Phi_i$ such that the smallest word $w_i \in L(M_i)$, for which the instruction ϕ_i is applicable, has an exponential length with respect to i, and thus also with respect to the size of the automaton M_i. In constructing these automata we will use a technique of sending signals from one sentinel to the other (and vice versa), which was widely applied in [9].

The automaton $M_0 = (\Sigma_0, \Phi_0)$ accepts only one word: $a_0a_0a_0a_0$, where $\Sigma_0 = \{a_0\}$ and Φ_0 contains only one instruction $(¢, a_0a_0a_0a_0, \$)$.

The best way, how to describe other automata in the sequence is to show how they work in the reverse direction (i.e. how they generate words starting from the empty word by using the inverse of the rewriting relation $\vdash$ defined as $\vdash^{-1}$). The automaton M_1 just sends a signal a_1 from the left sentinel ¢ to the right sentinel \$ starting from the word $a_0a_0a_0a_0$, as follows:

$$¢\lambda\$ \vdash ¢\underline{a_0}a_0a_0a_0\$ \vdash ¢\mathbf{\underline{a_1}}a_0a_0a_0a_0\$ \vdash ¢\mathbf{a_1}a_0\mathbf{\underline{a_1}}a_0a_0a_0\$ \vdash ¢\mathbf{a_1}a_0\mathbf{a_1}a_0\mathbf{\underline{a_1}}a_0a_0\$ \vdash$$
$$¢\mathbf{a_1}a_0\mathbf{a_1}a_0\mathbf{a_1}a_0\mathbf{\underline{a_1}}a_0\$ \vdash ¢\mathbf{a_1}a_0\mathbf{a_1}a_0\mathbf{a_1}a_0\mathbf{a_1}a_0\mathbf{\underline{a_1}}\$.$$

This can be easily achieved by the following set of instructions: $\Phi_1 = \{(¢, a_0a_0a_0a_0, \$), (¢, \underline{a_1}, a_0a_0), (a_1a_0, \underline{a_1}, a_0a_0), (a_1a_0, \underline{a_1}, a_0\$), (a_1a_0, \underline{a_1}, \$)\}$. It can be easily verified that the only words accepted by the automaton M_1 are the words shown in the above accepting computation. This is because if you proceed in the reverse direction, starting from the empty word, then you basically cannot get anything else than what we have in the above computation. This property also holds for all subsequent automata in the sequence.

The automaton $M_2 = (\Sigma_2, \Phi_2)$ is similar to M_1, except that this time it will send a signal a_2 from the right sentinel \$ to the left sentinel ¢. The reason why we want to send a signal in a reverse direction is that we want to preserve the nice property of having only one possible accepting computation. The automaton M_2 works as follows (it starts exactly where the previous automaton M_1 has ended):

$$¢a_1a_0a_1a_0a_1a_0a_1a_0a_1\$ \vdash ¢a_1a_0a_1a_0a_1a_0a_1a_0a_1\mathbf{\underline{a_2}}\$ \vdash$$
$$¢a_1a_0a_1a_0a_1a_0a_1a_0\mathbf{\underline{a_2}}a_1\mathbf{a_2}\$ \vdash ¢a_1a_0a_1a_0a_1a_0a_1\mathbf{\underline{a_2}}a_0\mathbf{a_2}a_1\mathbf{a_2}\$ \vdash$$
$$¢a_1a_0a_1a_0a_1a_0\mathbf{\underline{a_2}}a_1\mathbf{a_2}a_0\mathbf{a_2}a_1\mathbf{a_2}\$ \vdash ¢a_1a_0a_1a_0a_1\mathbf{\underline{a_2}}a_0\mathbf{a_2}a_1\mathbf{a_2}a_0\mathbf{a_2}a_1\mathbf{a_2}\$ \vdash$$
$$\ldots$$
$$¢\mathbf{a_2}a_1\mathbf{a_2}a_0\mathbf{a_2}a_1\mathbf{a_2}a_0\mathbf{a_2}a_1\mathbf{a_2}a_0\mathbf{a_2}a_1\mathbf{a_2}a_0\mathbf{a_2}a_1\mathbf{a_2}\$$$

To enable this kind of computation we only need to add the following instructions: $(\circ\circ, \underline{a_2}, \$)$, $(\circ\circ, \underline{a_2}, \circ a_2)$, $(¢\circ, \underline{a_2}, \circ a_2)$, and $(¢, \underline{a_2}, \circ a_2)$, where $\circ$ is a placeholder for any of the symbols from $\{a_0, a_1\}$ (of course, different occurrences of the placeholder $\circ$ can be substituted by different symbols). As in the previous case, we have only one possible computation.

Now we can generalize the above construction also to other automata in the sequence. The automaton M_i, for $i > 0$, is obtained from the automaton M_{i-1} as follows:

1. If i is odd then the automaton M_i will send a signal from the left sentinel ¢ to the right sentinel \$, thus, in order to obtain Φ_i, we only need to add the following instructions to Φ_{i-1}: $(¢, \underline{a_i}, \circ\circ)$, $(a_i\circ, \underline{a_i}, \circ\circ)$, $(a_i\circ, \underline{a_i}, \circ\$)$, and $(a_i\circ, \underline{a_i}, \$)$, where $\circ$ is a placeholder for any of the symbols from

$\{a_0, a_1, \ldots, a_{i-1}\}$.

2. If i is even then the automaton M_i will send a signal from the right sentinel \$ to the left sentinel ¢, thus, in order to obtain Φ_i, we only need to add the following instructions to Φ_{i-1}: $(\circ\circ, \underline{a_i}, \$)$, $(\circ\circ, \underline{a_i}, \circ a_i)$, $(¢\circ, \underline{a_i}, \circ a_i)$, and $(¢, \underline{a_i}, \circ a_i)$, where $\circ$ is a placeholder for any of the symbols from $\{a_0, a_1, \ldots, a_{i-1}\}$.

First observe that the size of the automaton M_i is polynomial with respect to i. This can be easily proved inductively by using a simple observation that we add only $O(i^3)$ new instructions to Φ_{i-1} when constructing Φ_i.

Now consider any $i > 0$. If i is even then let us take the instruction $\phi_i = (¢, \underline{a_i}, a_{i-1}a_{i-2})$. This instruction can be applied only after the previous signal a_{i-1} has arrived to the left sentinel ¢. In other words, it can be applied only to the longest word in $L(M_{i-1})$. However, the length of the longest word in $L(M_j)$ is exponential with respect to j, for all $j \geq 0$, because every time the signal traverses from one end to the other, the length of the resulting word more than doubles.

In the case when i is odd we can take the instruction $\phi_i = (a_{i-2}a_{i-1}, \underline{a_i}, \$)$, which can be applied only after the previous signal a_{i-1} has arrived to the right sentinel \$.

The above example clearly shows that sometimes we need to consider an exponentially large set (i.e. a set that has an exponentially large binary representation) of positive samples S^+ in order to obtain all instructions of the hidden target model. The argument is based on the use of the function StrongAssumptions (from Example 4.4) combined with the class of clearing restarting automata. This naturally rises an open question, whether this phenomenon also applies to other functions and other classes of restricted context rewriting systems.

As you can see, our auxiliary inference procedure (Algorithm 1) is relatively simple and straightforward, and, in addition, it sometimes fails. This naturally rises a question whether one can design a more sophisticated algorithm that can, for instance, always return a model with the restricted maximal width that is consistent with the given set of positive and negative samples. It turns out that there is little hope in finding such algorithm, because the task of finding a clearing restarting automaton consistent with a given set of positive and negative samples is NP-hard, provided there is an upper bound on the width of its instructions.

Theorem 4.10. ([7])
Let $l \geq 2$ be a fixed integer. It is NP-complete to decide whether there exists a $(0,l)$-cl-RA $M = (\Sigma, \Phi)$ consistent with a given sample $S = (S^+, S^-)$, $S^+ \cap S^- = \emptyset$, $\lambda \in S^+$.

In this paper we prove a more general Theorem 4.11.

Theorem 4.11. Let $k \geq 1$ and $l \geq 4k+4$ be fixed integers. It is NP-hard to decide whether there exists a (k,l)-cl-RA $M = (\Sigma, \Phi)$ consistent with a given sample $S = (S^+, S^-)$, $S^+ \cap S^- = \emptyset$, $\lambda \in S^+$.

Proof:
Consider a 3-SAT formula $\psi = \bigwedge_{i=1}^{n} C_i$, where clause $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$, and $\ell_{i,1}$, $\ell_{i,2}$, $\ell_{i,3}$ are literals having pairwise different variables, for all $i \in \{1, 2, \ldots, n\}$. Let $\Omega = \{a_1, a_2, \ldots, a_m\}$ be the set of all variables occurring in ψ. In the following, we will (effectively) construct a finite sample $S = (S^+, S^-)$, $S^+ \cap S^- = \emptyset$, $\lambda \in S^+$, such that the following holds: the formula ψ is satisfiable if and only if there exists

a (k,l)-cl-RA $M = (\Sigma, \Phi)$ consistent with $S = (S^+, S^-)$. Our alphabet Σ will contain all symbols from $\Omega \cup \overline{\Omega}$, where $\overline{\Omega} = \{\overline{a_i} \mid a_i \in \Omega\}$, and $\Omega \cap \overline{\Omega} = \emptyset$. We define $\overline{\overline{a}} = a$ for all $a \in \Omega$. In addition, Σ will contain also some other special symbols. First set $S^+ := \{\lambda\}, S^- := \emptyset$. For each clause $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ add the negative sample $w^-_{C_i} = \Box^k \overline{\ell_{i,1}} \Box^k \overline{\ell_{i,2}} \Box^k \overline{\ell_{i,3}} \Box^k$ to the set of negative samples S^-, where $\Box$ is a special dummy symbol that will later match the contexts of the instructions. For each variable $a \in \Omega$, add the following positive samples: $w^+_0 = \Box^k a \Box^t \overline{a} \Box^k$, $w^+_1 = \Box^k a \Box^{k+t}$, $w^+_2 = \Box^{k+t} \overline{a} \Box^k$, and $w^+_3 = \Box^{4k}$ to the set of positive samples S^+, and also add the negative sample $w^-_4 = \Box^k a \Box^{2k} \overline{a} \Box^k$ to the set of negative samples S^-, where $t = l - 2k - 3$. Since $l \geq 4k+4$, it follows that $t \geq 2k+1$, and thus $w^+_0 \neq w^-_4$. Our next goal is to allow only the following types of instructions, where $a \in \Omega$:

(a) $(\Box^k, a, \Box^k)$.

(b) $(\Box^k, \overline{a}, \Box^k)$.

(c) $(¢, \Box^k a \Box^{k+t}, \$)$.

(d) $(¢, \Box^{k+t} \overline{a} \Box^k, \$)$.

(e) $(¢, \Box^{4k}, \$)$.

All of these instructions have the width at most l. (The longest are the instructions of the type (c) and (d), having the width $2k+t+3 = l$). On the other hand, it is not possible to store the whole word $w^+_0 = \Box^k a \Box^t \overline{a} \Box^k$ in one instruction containing both sentinels (such us, for instance, $(¢, \Box^k a \Box^t \overline{a} \Box^k, \$)$), since its width would be $2k+t+4 = l+1 > l$. Also note, that the instructions (c) and (d) will never interfere with any of the negative samples $w^-_{C_i} = \Box^k \overline{\ell_{i,1}} \Box^k \overline{\ell_{i,2}} \Box^k \overline{\ell_{i,3}} \Box^k$, because: $|\Box^k \overline{\ell_{i,1}} \Box^k \Box^k \Box^k| = |\Box^k \Box^k \Box^k \overline{\ell_{i,3}} \Box^k| = 4k+1$, while $|\Box^k a \Box^{k+t}| = |\Box^{k+t} \overline{a} \Box^k| = 2k+t+1 = l-2 \geq 4k+2$. The same holds for the negative sample $w^-_4 = \Box^k a \Box^{2k} \overline{a} \Box^k$, because $|\Box^k a \Box^{2k} \Box^k| = |\Box^k \Box^{2k} \overline{a} \Box^k| = 4k+1$.

In the following we introduce a general technique, how to prohibit the inference of any specific undesirable instruction. Suppose that we want to prohibit the instruction $\phi = (x,z,y)$, where $|xzy| \leq l$. Let x' (y', respectively) be the largest possible subword of x (y, respectively) not containing the sentinels (¢, \$); thus either $x = x'$, or $x = ¢x'$ (either $y = y'$, or $y = y'\$$, respectively). There are only the following four possible cases:

1. $x = ¢x'$ and $y = y'\$$
2. $x = ¢x'$ and $y = y'$
3. $x = x'$ and $y = y'\$$
4. $x = x'$ and $y = y'$

In the first case we only need to add the word $x'zy'$ to the set of negative samples S^- and the word $x'y'$ to the set of positive samples S^+ in order to prohibit the instruction $\phi = (x,z,y) = (¢x', z, y'\$)$. In the other cases let us first introduce a new symbol $\Box_\phi$, which we also add to our alphabet Σ. This is what we do in the particular cases:

1. $S^- := S^- \cup \{x'zy'\}$, $S^+ := S^+ \cup \{x'y'\}$.
2. $S^- := S^- \cup \{x'zy'\Box_\phi\}$, $S^+ := S^+ \cup \{x'y'\Box_\phi\}$.

3. $S^- := S^- \cup \{\Box_\phi x'zy'\}$, $S^+ := S^+ \cup \{\Box_\phi x'y'\}$.
4. $S^- := S^- \cup \{\Box_\phi x'zy'\Box_\phi\}$, $S^+ := S^+ \cup \{\Box_\phi x'y'\Box_\phi\}$.

For every prohibited instruction ϕ we add two new samples: a positive sample w_ϕ^+ to the set S^+ and a negative sample w_ϕ^- to the set S^-. It is easy to see, that in each case we have effectively prohibited the instruction $\phi = (x,z,y)$. Note that this is the only instruction not containing the symbol $\Box_\phi$ (and having $x \in LC_k, y \in RC_k$) that reduces w_ϕ^+ to w_ϕ^-. Later, we will have to verify the consistency of the constructed (k,l)-cl-RA M also with these newly added samples, i.e. that for every prohibited instruction ϕ the following holds: $w_\phi^+ \vdash_M^* \lambda$ and $w_\phi^- \nvdash_M^* \lambda$. In all cases, the newly added positive sample w_ϕ^+ can always be reduced to the empty word in one step by using the instruction $(¢, w_\phi^+, \$)$. This is because the width of this instruction is $2 + |w_\phi^+| \leq 2 + 2 + |x'y'| \leq 4 + 2k \leq l$. If we use a new symbol $\Box_\phi$ in w_ϕ^+, then the instruction $(¢, w_\phi^+, \$)$ will be applicable only to this specific sample $w_\phi^+ \in S^+$, and thus will not interfere with other samples. On the other hand, the verification of the second condition ($w_\phi^- \nvdash_M^* \lambda$) is more difficult. Fortunately, we will always use the symbol $\Box_\phi$ in w_ϕ^-, i.e. the first case ($x = ¢x'$ and $y = y'\$$) will never occur in our proof.

Now we have all necessary ingredients to finish the proof. For every $a \in \Omega$ consider the following positive sample: $w_0^+ = \Box^k a \Box^t \overline{a} \Box^k$. By using the above technique, we disable all instructions applicable to this word having the width at most l except for the instructions of the form (a) – (e). Observe, that we will never attempt to disable any instruction of the form $\phi = (¢x', z, y'\$)$. This is because the word $w_0^+ = \Box^k a \Box^t \overline{a} \Box^k$ (as we have already mentioned above) is too long. Moreover, there is only polynomially many disabled instructions, since there is only polynomially many subwords of the above word. Now we have completely specified the sets of positive and negative samples S^+, S^-, $S^+ \cap S^- = \emptyset$, $\lambda \in S^+$, and thus we can proceed with the proof.

($\Rightarrow$) Suppose that ψ is satisfiable, i.e. there exists an assignment $v : \Omega \to \{0,1\}$ such that $v^*(C_i) = 1$ for all $i \in \{1,2,\ldots,n\}$, where v^* denotes the natural extension of v to formulas. We will show that there exists a (k,l)-cl-RA $M = (\Sigma, \Phi)$ consistent with $S = (S^+, S^-)$. Consider the following (k,l)-cl-RA $M = (\Sigma, \Phi)$: First, add to Φ the following set of instructions: $\Phi_1 = \{(\Box^k, a, \Box^k) \mid a \in \Omega : v(a) = 1\} \cup \{(\Box^k, \overline{a}, \Box^k) \mid a \in \Omega : v(a) = 0\}$. It can be easily observed that, for all literals $l \in \Omega \cup \overline{\Omega}$: $\Box^k l \Box^k \vdash_M \Box^k \Box^k \Leftrightarrow v(l) = 1$, or equivalently: $\Box^k \overline{l} \Box^k \vdash_M \Box^k \Box^k \Leftrightarrow v(l) = 0$. Therefore no negative sample $w_{C_i}^- = \Box^k \overline{\ell_{i,1}} \Box^k \overline{\ell_{i,2}} \Box^k \overline{\ell_{i,3}} \Box^k$, where $i \in \{1,2,\ldots,n\}$, can be reduced to the positive sample $\Box^{4k}$ by using the instructions from Φ_1, because otherwise it would mean that $v(\ell_{i,1}) = v(\ell_{i,2}) = v(\ell_{i,3}) = 0$. Next, add to Φ the following set of instructions $\Phi_2 = \{(¢, \Box^k a \Box^{k+t}, \$), (¢, \Box^{k+t} \overline{a} \Box^k, \$) \mid a \in \Omega\} \cup \{(¢, \Box^{4k}, \$)\}$. As we have already stated above, no instruction from Φ_2 will ever interfere with any negative sample $w_{C_i}^- = \Box^k \overline{\ell_{i,1}} \Box^k \overline{\ell_{i,2}} \Box^k \overline{\ell_{i,3}} \Box^k$, or $w_4^- = \Box^k a \Box^{2k} \overline{a} \Box^k$. Finally, add to Φ all instructions $(¢, w_\phi^+, \$)$, where w_ϕ^+ was added to the set of positive samples S^+ during the process of disabling the undesirable instruction ϕ. These instructions are applicable only to words containing these special symbols $\Box_\phi$, so we do not have to care about them at all. Now we will show that the constructed (k,l)-cl-RA $M = (\Sigma, \Phi)$ is consistent with the sample $S = (S^+, S^-)$. (The width of instructions of M is apparently bounded from above by l). First, it is easy to see, that for each variable $a \in \Omega$ the following positive samples: $w_0^+ = \Box^k a \Box^t \overline{a} \Box^k$, $w_1^+ = \Box^k a \Box^{k+t}$, $w_2^+ = \Box^{k+t} \overline{a} \Box^k$, $w_3^+ = \Box^{4k}$ are all reducible to the empty word λ. The positive sample w_0^+ is always reducible to either w_1^+, or w_2^+, depending on whether $(\Box^k, a, \Box^k) \in \Phi_1$, or $(\Box^k, \overline{a}, \Box^k) \in \Phi_1$. The other positive samples: w_1^+, w_2^+, w_3^+ can be reduced to the empty word λ in one single step by using the corresponding instruction from Φ_2. The negative sample $w_4^- = \Box^k a \Box^{2k} \overline{a} \Box^k$ clearly cannot be

reduced to the empty word λ. We can clear either the letter a, or $\overline{a}$ by using the corresponding instruction from Φ_1, but then we get the irreducible word $\square^k\square^{2k}\overline{a}\square^k$, or $\square^k a\square^{2k}\square^k$. None of the instructions from Φ_2 can be applied to such a word, since the length $|\square^k a\square^{2k}\square^k| = |\square^k\square^{2k}\overline{a}\square^k| = 4k+1$, while $|\square^k a\square^{k+t}| = |\square^{k+t}\overline{a}\square^k| = 2k+t+1 = l-2 \geq 4k+2$. It remains to be shown, that no negative sample w^-_ϕ, which was added to the set of negative samples S^- during the process of disabling some instruction ϕ, can be reduced to the empty word λ. Both the negative sample w^-_ϕ and the corresponding positive sample w^+_ϕ contain the special symbol $\square_\phi$. The negative sample w^-_ϕ without this special symbol $\square_\phi$ is basically a subword of some word $\square^k a\square^t\overline{a}\square^k$. First observe, that the only instruction from Φ, that could be possibly applied to the negative sample w^-_ϕ, is some instruction from Φ_1. Without loss of generality assume that we can apply the instruction $\rho = (\square^k, a, \square^k) \in \Phi_1$ to the word w^-_ϕ, i.e. $w^-_\phi \vdash^{(\rho)} w'$. This also implies that $(\square^k, \overline{a}, \square^k) \notin \Phi_1$. It is easy to see, that no other instruction from Φ can be applied to the resulting word w', except possibly the instruction $(¢, w^+_\phi, \$)$. But if $(¢, w^+_\phi, \$)$ was applicable to w', it would have implied that we wanted to disable the instruction ρ itself, which is not possible, since ρ is of the form (a). Thus, we have shown that the constructed (k,l)-cl-RA $M = (\Sigma, \Phi)$ is consistent with $S = (S^+, S^-)$. The size of the constructed set of positive and negative samples is linear with respect to the size of the formula ψ.

($\Leftarrow$) Now suppose that there exists a (k,l)-cl-RA $M = (\Sigma, \Phi)$ consistent with $S = (S^+, S^-)$. We will show that ψ is satisfiable, i.e. we will construct an assignment $v : \Omega \to \{0,1\}$ such that $v^*(C_i) = 1$ for all $i \in \{1, 2, \ldots, n\}$. First observe, that for each $a \in \Omega$: either $(\square^k, a, \square^k) \in \Phi$, or $(\square^k, \overline{a}, \square^k) \in \Phi$. Consider the positive sample $w^+_0 = \square^k a\square^t\overline{a}\square^k$. We know that $\square^k a\square^t\overline{a}\square^k \vdash^*_M \lambda$. Let $\phi \in \Phi$ be the first instruction used in any such accepting computation. The instruction ϕ is either of the form $(\square^k, a, \square^k)$, or $(\square^k, \overline{a}, \square^k)$, because all other instructions are disabled. Moreover, it cannot happen that both instructions $(\square^k, a, \square^k)$, $(\square^k, \overline{a}, \square^k)$ are in Φ, because it would mean that $\square^k a\square^{2k}\overline{a}\square^k \vdash^*_M \square^{4k}$, where $\square^k a\square^{2k}\overline{a}\square^k \in S^-$ and $\square^{4k} \in S^+$. Now let us define the assignment $v : \Omega \to \{0,1\}$ as follows: for each $a \in \Omega$: $v(a) = 1$ if $(\lambda, a, \lambda) \in \Phi$, and $v(a) = 0$ if $(\lambda, \overline{a}, \lambda) \in \Phi$. For each clause $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ we have a negative sample $w^-_{C_i} = \square^k\overline{\ell_{i,1}}\square^k\overline{\ell_{i,2}}\square^k\overline{\ell_{i,3}}\square^k \in S^-$. Therefore, $(\square^k, \overline{\ell_{i,1}}, \square^k) \notin \Phi$ or $(\square^k, \overline{\ell_{i,2}}, \square^k) \notin \Phi$ or $(\square^k, \overline{\ell_{i,3}}, \square^k) \notin \Phi$, which is equivalent to $v(\ell_{i,1}) = 1$ or $v(\ell_{i,2}) = 1$ or $v(\ell_{i,3}) = 1$. This means that ψ is satisfiable. □

Note that if the consistency check for cl-RA was in NP, then also deciding the existence of a (k,l)-cl-RA $M = (\Sigma, \Phi)$ consistent with a given sample $S = (S^+, S^-)$ would be in NP. This follows from the fact that the number of instructions that we need to guess is bounded from above by $\mathsf{size}(S^+)$, because for every positive sample $w_+ \in S^+$ the accepting computation $w_+ \vdash^*_M \lambda$ uses at most $|w_+|$ many instructions.

4.2. Learning Without Restrictions

Now we turn to learning without the restricted width of instructions. As we have already stated we will use the auxiliary inference procedure (Algorithm 1) from Section 4.1 as a component in our learning algorithm. The inference procedure itself requires the specification of the maximal width of instructions $l \geq 1$, so it is only natural to try all widths $l = 1, 2, \ldots$, until the inference procedure succeeds. The resulting learning algorithm is shown in Algorithm 6.

In turns out that the learning Algorithm 6 is not only able to identify any hidden target model in the limit, it also infers a model with *minimal width*.

Algorithm 6: Learning algorithm $\mathsf{UnconstrainedInfer}_{\mathscr{M}}(S,k)$

Input : Sample $S=(S^+,S^-)$ over Σ, $S^+\cap S^-=\emptyset, \lambda\in S^+$.
Length of contexts $k\geq 1$, or $k=\cdot$, if not specified.
Output: A model $M\in k$-$\mathscr{M}$ consistent with S

```
1 l_max ← 2 + max{|w| | w ∈ S⁺};
2 for l = 1 ... l_max do
3 |   M ← Infer_M(S,k,l);
4 |   if M ≠ Fail then
5 |   |   return M;
6 return M with instructions Φ = {(¢, w → λ, $) | w ∈ S⁺};
```

Theorem 4.12. Let $\mathscr{M}$ be a class of ordinary k-CRS restricted according to Definition 2.7 (where $k\geq 1$ or $k=\cdot$ if not used) and $\mathscr{L}$ be the corresponding class of languages. Let function Assumptions be monotone and correct with respect to $\mathscr{M}$. Then:

1. Given a sample $S=(S^+,S^-)$ for $L\in\mathscr{L}$, $\mathsf{UnconstrainedInfer}_{\mathscr{M}}(S,k)$ returns a model (hypothesis) $N\in\mathscr{M}$ *consistent with* S.

2. For each model $M\in\mathscr{M}$ representing the language $L\in\mathscr{L}$, there exists a finite *characteristic sample* $S_0=(S_0^+,S_0^-)$ such that, on all samples $S=(S^+,S^-)$ for L that verify $S_0^+\subseteq S^+$ and $S_0^-\subseteq S^-$, $\mathsf{UnconstrainedInfer}_{\mathscr{M}}(S,k)$ returns an equivalent model $N\in\mathscr{L}$ with *minimal width*.

Proof:
(1) Cycle 2 – 5 is finite and in both Steps 5 and 6 we return a model from $\mathscr{M}$ consistent with S. (2) Without loss of generality we may assume that the the model $M=(\Sigma,\Phi)\in\mathscr{M}$ representing the target language L has minimal width $|M|=l_0$ and a minimal set of instructions. Similarly as in the proof of Theorem 4.7 we can define the term *bad* instruction with respect to the target language $L(M)$. Let $\Theta\supseteq\Phi$ denote the set of all legal instruction of the class (k,l_0)-$\mathscr{M}$ that are not bad w.r.t. $L(M)$. Then apparently $\bar{M}:=(\Sigma,\Theta)\in(k,l_0)$-$\mathscr{M}$ and $L(\bar{M})=L(M)$. In addition, there exists a word $w_0\in L(\bar{M})$ such that every accepting computation $w_0\vdash_{\bar{M}}^{*}\lambda$ must use some instruction $\phi\in\Theta$ with $|\phi|=l_0$ (otherwise the width $|M|=l_0$ would not be minimal). According to Theorem 4.7 there exists a characteristic sample $S_0=(S_0^+,S_0^-)$ such that, on all samples $S=(S^+,S^-)$ for $L(M)$ that verify $S_0^+\subseteq S^+$ and $S_0^-\subseteq S^-$, the inference procedure $\mathsf{Infer}_{\mathscr{M}}(S,k,l_0)$ will return a model $N\in(k,l_0)$-$\mathscr{M}$ equivalent to M. We may assume that $w_0\in S_0^+$. Our goal is now to show that for every $l<l_0$ the inference procedure $\mathsf{Infer}_{\mathscr{M}}(S,k,l)$ will **Fail**. For every $l<l_0$: $\mathsf{Assumptions}(S^+,k,l)\subset\mathsf{Assumptions}(S^+,k,l_0)$. This is because the set $\mathsf{Assumptions}(S^+,k,l_0)$ contains all instructions of M (this follows from the minimality of Φ), and some of these instructions have the width equal to l_0. Since all bad instructions from $\mathsf{Assumptions}(S^+,k,l_0)$ are disabled by (S^+,S^-), the inference procedure $\mathsf{Infer}_{\mathscr{M}}(S,k,l)$ will filter all bad instructions returned by $\mathsf{Assumptions}(S^+,k,l)$. The resulting set of instructions Φ' will contain only the correct instructions, i.e., $\Phi'\subseteq\Theta$. However, the instructions of M that have the width equal to l_0 will be missing in this set Φ'. This implies that the word w_0 cannot be reduced to λ by using only the instructions from Φ'. Therefore, the inference procedure $\mathsf{Infer}_{\mathscr{M}}(S,k,l)$ will **Fail**. □

If $\mathscr{M}$ is a class of λ-confluent models, then both the simplification (Step 7) and the consistency check (Step 8) in the inference procedure $\mathsf{Infer}_{\mathscr{M}}$ (Algorithm 1) can be done in polynomial time (provided that we restrict ourselves to the left-most reductions). The only problem is that the λ-confluence itself is undecidable in most cases. Fortunately, according to Theorem 4.7, if we use a sample S that contains the characteristic sample S_0, then we do not need to check the λ-confluence at all as the inferred model will be λ-confluent. Additionally, according to proof of Theorem 4.12, if we use a width l that is smaller than the minimal width l_0 of the target model then the inference procedure $\mathsf{Infer}_{\mathscr{M}}(S,k,l)$ will **Fail**, because it will not be able to reduce the word $w_0 \in S_0^+$ to the empty word (by using any kind of reductions, not only the left-most reductions). This gives us the following Corollary 4.13.

Corollary 4.13. Let $\mathscr{M}$ be a class of λ-confluent k-CRS restricted according to Definition 2.7 (where $k \geq 1$ or $k = \cdot$ if not used) and $\mathscr{L}$ be the corresponding class of languages. Let function Assumptions be monotone and correct with respect to $\mathscr{M}$. Then for each model $M \in \mathscr{M}$ representing the language $L \in \mathscr{L}$, there exists a finite *characteristic sample* $S_0 = (S_0^+, S_0^-)$ such that, on all samples $S = (S^+, S^-)$ for L that verify $S_0^+ \subseteq S^+$ and $S_0^- \subseteq S^-$, $\mathsf{UnconstrainedInfer}_{\mathscr{M}}(S,k)$ returns an equivalent model $N \in \mathscr{L}$ with *minimal width* in *polynomial time* (provided that we restrict ourselves only to the left-most reductions).

Note that this result deviates from our original goal of the polynomial identification as defined in Definition 3.2. First, we are not able to satisfy the condition 1 of Definition 3.2, because we are not able to verify the λ-confluence of the inferred model. Second, we are not able to guarantee that the function $\beta(\cdot)$ in the condition 2 of Definition 3.2 is polynomially bounded.

5. Clearing Restarting Automata and Finite Automata

The NP-hardness results from Section 4.1 resemble the famous result of Gold [19] who showed that the construction of a deterministic finite state automaton consistent with the given data is, in general, computationally difficult, if the number of states is bounded from above by n.

Theorem 5.1. For every n-state deterministic finite automaton A there exists an equivalent clearing restarting automaton M such that $|M| = n+1$.

Proof:
Let $A = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite state automaton, where Q is a set of n states, Σ is an alphabet, $\delta : Q \times \Sigma \to Q$ is a transition function, $q_0 \in Q$ is a start state and $F \subseteq Q$ is a set of accept states. Let $\delta^* : Q \times \Sigma^* \to Q$ denote the transition function extended to words, i.e. for every $q \in Q$: $\delta^*(q, \lambda) := q$ and for every $w \in \Sigma^*$ and $a \in \Sigma$: $\delta^*(q, wa) := \delta(\delta^*(q, w), a)$. For every word $w = w_1 \dots w_n \in \Sigma^n$ there exist words $x_w, z_w, y_w \in \Sigma^*$ such that $w = x_w z_w y_w$, $0 < |z_w| \leq n$, and $\delta^*(q_0, x_w) = \delta^*(q_0, x_w z_w)$. This follows from the so-called *pigeon hole principle*: There are only n states and the following sequence: $\delta^*(q_0, \lambda)$, $\delta^*(q_0, w_1)$, $\delta^*(q_0, w_1 w_2)$, …, $\delta^*(q_0, w_1 w_2 \dots w_n)$ contains exactly $n+1$ elements. Therefore, there exist $0 \leq i < j \leq n$ such that: $\delta^*(q_0, w_1 \dots w_i) = \delta^*(q_0, w_1 \dots w_j)$. If you set $x_w = w_1 \dots w_i$, $z_w = w_{i+1} \dots w_j$ and $y_w = w_{j+1} \dots w_n$, then $w = x_w z_w y_w$, $0 < |z_w| \leq n$, and $\delta^*(q_0, x_w) = \delta^*(q_0, x_w z_w)$. Now consider a cl-RA $M = (\Sigma, \Phi)$, where $\Phi = \{(¢, w, \$) \mid w \in L(A) \text{ and } |w| < n\} \cup \{(¢ x_w, z_w, y_w) \mid w \in \Sigma^n\}$. Apparently, $|M| = n+1$. Consider any word $w \in \Sigma^*$. If $0 < |w| < n$, then $w \in L(M) \Leftrightarrow w \in L(A)$. Suppose that $|w| \geq n$, i.e. $w = x_u z_u y_u v$, where $u \in \Sigma^n$ and $v \in \Sigma^*$. Apparently, $x_u z_u y_u v \in L(A) \Leftrightarrow x_u y_u v \in L(A)$. The

application of the instruction $(¢x_u, z_u, y_u) \in \Phi$ on the word w does not change the acceptance of this word by the automaton A. It only shortens the length of this word, so after a finite many steps we get a nonempty word that is shorter than n letters. □

On the other hand, the number of instructions of a clearing restarting automaton is not, in general, polynomially bounded with respect to the number of states of an equivalent deterministic finite state automaton. We illustrate this in the following Example 5.2.

Example 5.2. For every positive integer $k > 1$ consider the finite language $L_k = \{w \in \{a,b\}^* \mid |w|_a = |w|_b = k\}$. It is easy to find a deterministic finite state automaton A_k with polynomially many states with respect to k that recognizes the language L_k. The automaton A_k only needs to store in its state how many as and bs it has read so far, i.e. it needs only $(k+1)^2$ states. On the other hand, if a cl-RA $M = (\Sigma, \Phi)$ recognizes the language L_k then for every word $w \in L_k : (¢, w, \$) \in \Phi$. Proof (by contradiction): Suppose that there exists a word $w \in L_k : (¢, w, \$) \notin \Phi$. No instruction of M can rewrite the whole word w to the empty word in one single step. Therefore, any accepting computation starting from the input word w consists of at least two steps, e.g. $w \vdash_M w' \vdash_M^* \lambda$. But the word $w' \notin L_k$, because if you clear any proper subword of the word w you get a word outside the language L_k. This is a contradiction, because $w' \vdash_M^* \lambda \Rightarrow w' \in L(M)$. We have shown that $|\Phi| \geq |L_k| = \binom{2k}{k}$ which is exponential with respect to k.

Nevertheless, there exist regular languages that have a compact representation in the class of clearing restarting automata, but that cannot be compactly represented by using deterministic finite state automata. Consider, for instance, the sequence of finite languages $L(M_0), L(M_1), L(M_2), \ldots$ from Example 4.9. This sequence of languages has a compact representation in the class of clearing restarting automata because the size of every automaton M_i is polynomial with respect to i. On the other hand, for every i the length of the longest word $w_i \in L(M_i)$ is exponential with respect to i. Therefore, the smallest deterministic finite state automaton A_i recognizing the language $L(M_i)$ must have at least $|w_i|$ states, i.e. exponentially many states. Otherwise, we could apply the pigeon hole principle on the word w_i which would imply that the language $L(A_i)$ is an infinite language.

6. Conclusions

We have introduced a general learning algorithm for inferring various restricted types of context rewriting systems. We have shown that, under certain conditions, it is possible to identify in the limit any target context rewriting system with minimal width of instructions from informant. In addition, we have shown that the learning algorithm works in polynomial time with respect to the size of the given sample when used on λ-confluent context rewriting systems. For ordinary context rewriting systems the time complexity of the consistency check is an open problem.

Acknowledgements

I would like to thank Friedrich Otto and František Mráz for their support and careful proofreading of almost all revisions of this paper and for all suggestions and advices that significantly contributed to the quality of this paper.

References

[1] Angluin, D.: Queries and concept learning, *Machine Learning*, **2**(4), 1988, 319–342.

[2] Basovník, S.: *Master's Thesis: Learning restricted restarting automata using genetic algorithm*, Charles University, Faculty of Mathematics and Physics, Prague, Prague, 2010.

[3] Basovník, S.: Learning limited context restarting automata by genetic algorithms, *Theorietag* (J. Dassow, B. Truthe, Eds.), Otto-von-Guericke-Universität, Magdeburg, 2011.

[4] Beaudry, M., Holzer, M., Niemann, G., Otto, F.: McNaughton families of languages, *Theoretical Computer Science*, **290**(3), 2003, 1581–1628.

[5] Book, R. V., Otto, F.: *String-rewriting systems*, Springer-Verlag, New York, NY, USA, 1993.

[6] Buntrock, G., Otto, F.: Growing context-sensitive languages and Church-Rosser languages, *ICOM*, **141**, 1998, 1–36.

[7] Černo, P.: Clearing Restarting Automata and Grammatical Inference, *Proceedings of the Eleventh International Conference on Grammatical Inference* (J. Heinz, C. de la Higuera, T. Oates, Eds.), 21, 2012.

[8] Černo, P.: Grammatical Inference of Lambda-Confluent Context Rewriting Systems, *Workshop on Non-Classical Models of Automata and Applications (NCMA)* (S. Bensch, F. Drewes, R. Freund, F. Otto, Eds.), 294, Österreichisches Computer Gesellschaft, 2013.

[9] Černo, P., Mráz, F.: Clearing Restarting Automata, *Fundamenta Informaticae*, **104**(1), 2010, 17–54.

[10] Černo, P., Mráz, F.: Delta-Clearing Restarting Automata and CFL, in: *Developments in Language Theory* (G. Mauri, A. Leporati, Eds.), vol. 6795 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2011, 153–164.

[11] Cicchello, O., Kremer, S. C.: Inducing grammars from sparse data sets: a survey of algorithms and results, *Journal of Machine Learning Research*, **4**, December 2003, 603–632.

[12] Clark, A.: Three Learnable Models for the Description of Language, in: *Language and Automata Theory and Applications* (A.-H. Dediu, H. Fernau, C. Martin-Vide, Eds.), vol. 6031 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2010, 16–31.

[13] Coste, F., Garet, G., Nicolas, J.: Local Substitutability for Sequence Generalization, *ICGI 2012* (J. Heinz, C. de la Higuera, T. Oates, Eds.), 21, University of Maryland, MIT Press, Sep 2012.

[14] Dahlhaus, E., Warmuth, M.: Membership for growing context sensitive grammars is polynomial, in: *CAAP '86* (P. Franchi-Zannettacci, Ed.), vol. 214 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 1986, 85–99.

[15] De La Higuera, C.: Characteristic Sets for Polynomial Grammatical Inference, *Machine Learning*, **27**(2), May 1997, 125–138.

[16] De La Higuera, C.: *Grammatical Inference: Learning Automata and Grammars*, Cambridge University Press, New York, NY, USA, 2010.

[17] Eyraud, R., de la Higuera, C., Janodet, J.-C.: LARS: A learning algorithm for rewriting systems, *Machine Learning*, **66**, 2007, 7–31.

[18] Gold, E. M.: Language identification in the limit, *Information and Control*, **10**(5), 1967, 447–474.

[19] Gold, E. M.: Complexity of automaton identification from given data, *Information and Control*, **37**(3), 1978, 302–320.

[20] Hopcroft, J. E., Motwani, R., Ullman, J. D.: *Introduction to Automata Theory, Languages and Computation*, 3. edition, Pearson Addison-Wesley, Upper Saddle River, NJ, 2007, ISBN 978-0-321-51448-6.

[21] Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting Automata, *FCT'95* (H. Reichel, Ed.), 965, Springer, Dresden, Germany, August 1995.

[22] Jurafsky, D., Martin, J. H.: *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*, 2 edition, Prentice Hall, 2008, ISBN 0131873210.

[23] Marcus, S.: Contextual Grammars, *Revue Roum. Mathy. Pures Appl.*, **14**(10), 1969, 1525–1534.

[24] Maurer, H., Salomaa, A., Wood, D.: Pure grammars, *Information and Control*, **44**(1), 1980, 47–72.

[25] McNaughton, R.: Algebraic decision procedures for local testability, *Theory of Computing Systems*, **8**, 1974, 60–76.

[26] McNaughton, R., Narendran, P., Otto, F.: Church-Rosser Thue systems and formal languages, *JACM*, **35**, 1988, 324–344.

[27] Otto, F.: Restarting automata, in: *Recent Advances in Formal Languages and Applications* (Z. Ésik, C. Martín-Vide, V. Mitrana, Eds.), vol. 25 of *Studies in Computational Intelligence*, Springer, Berlin, 2006, 269–303.

[28] Otto, F., Černo, P., Mráz, F.: Limited Context Restarting Automata and McNaughton Families of Languages, *Workshop on Non-Classical Models for Automata and Applications (NCMA)* (R. Freund, M. Holzer, B. Truthe, U. Ultes-Nitsche, Eds.), books@ocg.at, Österreichisches Computer Gesellschaft, 2012.

[29] Otto, F., Mráz, F.: Lambda-Confluence is Undecidable for Clearing Restarting Automata, *CIAA 2013, Proceedings*, 7982, Berlin, 2013.

[30] Pitt, L.: Inductive Inference, DFAs and Computational Complexity, *2nd Int. Workshop on Analogical and Inductive Inference (AII*, 1989.

[31] Yoshinaka, R.: Identification in the Limit of k,l-Substitutable Context-Free Languages, in: *Grammatical Inference: Algorithms and Applications* (A. Clark, F. Coste, L. Miclet, Eds.), vol. 5278 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2008, ISBN 978-3-540-88008-0, 266–279.

[32] Zalcstein, Y.: Locally Testable Languages, *J. Comput. Syst. Sci*, **6**(2), 1972, 151–167.