Proceedings of Machine Learning Research 93:155–168, 2019 International Conference on Grammatical Inference

Learning local substitutable context-free languages from positive examples in polynomial time and data by reduction

François Coste FRANCOIS.COSTE@INRIA.FR
Jacques Nicolas JACQUES.NICOLAS@INRIA.FR
Univ Rennes, Inria, CNRS, IRISA
F-35000 Rennes

Editors: Olgierd Unold, Witold Dyrka, and Wojciech Wieczorek

Abstract

To study more formally the approach by reduction initiated by ReGLiS, we propose a formal characterization of the grammars in reduced normal form (RNF) which can be learned by this approach. A modification of the core of ReGLiS is then proposed to ensure returning RNF grammars in polynomial time. This enables us to show that local substitutable languages represented by RNF context-free grammars are identifiable in polynomial time and thick data (IPTtD) from positive examples.

Keywords: Learnability, substitutable languages, context-free grammar, normal form.

1. Introduction

The idea that strings occurring in the same contexts are substitutable, an early intuition from linguistics (Harris, 1954), has been particularly fruitful in grammatical inference for learning context-free or mildly context-sensitive languages from positive examples. It has been the root of unsupervised empirical grammatical inference approaches targeting natural languages: seminal algorithms of the field such as EMILE (Adriaans, 1992; Adriaans and Vervoort, 2002), ABL (van Zaanen, 2000) or ADIOS (Solan et al., 2005) basically detect strings sharing some contexts (either on the basis of clustering, alignment or fixed-length window along paths in a pseudograph) and group them in substitutability classes to build accordingly a context-free (or context-sensitive) grammar. More fundamental studies have been later initiated by Clark and Eyraud (2007) to better understand these approaches. They focused on the simplest case, when it is sufficient for two strings to share one context to be substitutable everywhere, thereby defining the class of *substitutable languages*. They proposed SGL, a simple polynomial learning algorithm enabling to prove a first polynomial identification in the limit result for substitutable context-free languages.

While substitutability paved the way to learnability results on more and more expressive languages (Yoshinaka, 2008; Clark, 2010; Yoshinaka, 2011; Clark and Yoshinaka, 2014), Clark and Eyraud (2007) stated themselves that their polynomial identification in the limit result was unsatisfactory since it was obtained "by relaxing the constraint for polynomial size of the characteristic set, to merely requiring polynomial cardinality", and would "allow algorithms to use exponential amounts of computation, by specifying an exponentially long string in the characteristic set". The problem was left open for further study, asking for a consensual theoretical setting able to deal with context-free grammars.

The learnability result targeted by Clark and Eyraud (2007) is the polynomial time and data identification in the limit, defined by de la Higuera (1997), which provides a realistic learnability criterion for tractable grammatical inference from given data by bounding polynomially 1) the running time of the learning algorithm and also 2) the size of the data required in the training sample to ensure identification:

Definition 1 (Polynomial time and data identification in the limit) *A representation class $\mathcal{R}$ is identifiable in the limit from polynomial time and data (IPTD) iff there exist two polynomials $p()$ and $q()$ and an algorithm $\mathcal{A}$ such that:*

- *Given a training sample S of size $\|S\|$, $\mathcal{A}$ returns a representation R in $\mathcal{R}$ consistent with S in $\mathcal{O}(p(\|S\|))$;*
- *For each representation R in $\mathcal{R}$ of size $\|R\|$, there exists a sample CS (named characteristic sample) of size at most $\mathcal{O}(q(\|R\|))$, such that $\mathcal{A}$ returns a representation R' equivalent with R for any sample S containing CS.*

The first results by de la Higuera (1997) show the difficulty of learning languages in this setting: context-free grammars, linear grammars, simple deterministic grammars, and non-deterministic finite automata are not IPTD while, in contrast, deterministic finite automata are IPTD. This later result illustrates the importance of the choice of the representation for being IPTD. Deterministic automata are IPTD while non-deterministic automata are not IPTD, although they represent the same class of languages. The point is that non-deterministic finite automata can be exponentially smaller than the deterministic ones for the same language. A characteristic sample of polynomial size with respect to the deterministic target can then be too big with respect to an exponentially smaller non-deterministic automaton representing the same language, while no smaller training set may ensure the convergence towards an equivalent automaton. Although one may argue about the importance of the representation choice (see for instance Eyraud et al., 2016), IPTD has remained the classical learnability criterion to meet for practical and non-statistical grammatical inference from given data. The most expressive grammars that have been shown to be IPTD so far are essentially subclasses of linear grammars: the even linear grammars and hierarchies built upon them, as a consequence of the positive result for deterministic automata (Takada, 1988; Sempere and García, 1994; Takada, 1994), and more recently deterministic linear grammars (de la Higuera and Oncina, 2002; Calera-Rubio and Oncina, 2004).

These results have been obtained for complete samples, *i.e.* from positive and negative examples, but IPTD criterion is also interesting for learning from text, *i.e.* from positive examples only (see Yoshinaka, 2007, for a discussion). In that setting, the substitutability-based learning approach seems well suited to get IPTD results for expressive classes of grammars. The polynomial *time* part of IPTD has already been proven for the inference of substitutable and k,l-substitutable context-free languages (Clark and Eyraud, 2007; Yoshinaka, 2008). The problem is that the characteristic sample *size* cannot be bounded polynomially with respect to the size of any context-free grammar representing substitutable languages. Indeed, as pointed out by (Clark and Eyraud, 2007), if we consider the languages consisting of the single string a^{2^n} (the symbol a repeated 2^n times) represented by the context-free grammars (see notations in section 2) $G_n = \langle \{a\}, \{N_i : 1 \leq i \leq n\}, \{N_i \to N_{i-1}N_{i-1} : 1 < i \leq n\} \cup \{N_1 \to aa\}, \{N_n\} \rangle$, the size of the unique training sample that can be used is always exponentially bigger than the grammar's size.

A solution proposed by Yoshinaka (2008) is to change the theoretical setting to consider not only the representation's size, but also its thickness τ_R which is the maximum length of the smallest word generated from each rule (see Yoshinaka, 2009). Requiring the size of the characteristic sample to be at most $\mathcal{O}(q(\|R\|\,\tau_R))$ instead of $\mathcal{O}(q(\|R\|))$ defines the *identification in polynomial time and thick data* (IPTtD) learnability criterion (Yoshinaka, 2009; Eyraud et al., 2016).

The work presented here is based on previous research motivated by the practical application of substitutability-based learning framework to proteins. It led us to introduce a hierarchy of subclasses of substitutable languages, based on local rather than global contexts substitutability criterion, and to propose ReGLiS, an efficient algorithm learning context-free grammars in *reduced normal form* for these classes directly(Coste et al., 2012a,b, 2014). Designed to make the most from incomplete training sets, ReGLiS is not guaranteed to run in polynomial time when the training set is not characteristic. Moving from a practical to a theoretical perspective, we focus here on the core of the approach by reduction of ReGLiS. Providing a new formal characterization of grammars in *reduced normal form* (RNF), we propose a modification of the reduction algorithm, limiting its generalization capacities to ensure returning a RNF grammar in polynomial time, enabling us to show that *local substitutable languages represented by context-free grammars in reduced normal form are identifiable in polynomial time and thick data (IPTtD) from text.*

2. Substitutable languages

First, we present the definitions and notations that will be used.

Formal languages. Let Σ be a non-empty finite set of atomic symbols. Σ^* is the set of all finite strings over the alphabet Σ. We denote the length of a string x by $|x|$, the empty string by ϵ, the set of non-empty strings $\Sigma^* \setminus \{\epsilon\}$ by Σ^+, the set of strings of length k $\{x\colon |x| = k\}$ by Σ^k and the set of strings of length smaller than or equal to k $\{x\colon 0 \leq |x| \leq k\}$ by $\Sigma^{\leq k}$. We will assume an order $\prec$ on Σ and its standard order extension on strings of Σ^* (comparing the first different symbols with $\prec$ if they are of same length, or their lengths with $<$ otherwise). The concatenation of strings is denoted $x_1 \cdot x_2$ or simply x_1x_2. A string or language x concatenated k times is denoted x^k. Given a string x, we denote by $x[i]$, with $1 \leq i \leq |x|$, the symbol at the position i in x and by $x[i,j]$, with $1 \leq i \leq j \leq |x|$, the substring $x[i] \dots x[j]$ of x from positions i to j.

For a language $L \subseteq \Sigma^*$, its set of substrings is $\mathrm{Sub}(L) = \{y \in \Sigma^* : x, z \in \Sigma^*, xyz \in L\}$ and its set of contexts is $\mathrm{Con}(L) = \{\langle x, z\rangle \in \Sigma^* \times \Sigma^* : y \in \Sigma^*, xyz \in L\}$. The empty context is $\langle \epsilon, \epsilon\rangle$. The distribution $D_L(y)$ of a string $y \in \Sigma^*$ for a language L is the set of contexts surrounding it in L: $D_L(y) = \{\langle x, z\rangle \in \Sigma^* \times \Sigma^* \colon xyz \in L\}$. Two strings y_1 and y_2 in Σ^* are syntactically congruent for a language L, denoted $y_1 \equiv_L y_2$, iff $D_L(y_1) = D_L(y_2)$. The equivalence relation $\equiv_L$ defines a congruence on the monoid Σ^* since $y_1 \equiv_L y_2$ implies $\forall x, z$ in $\Sigma^*, xy_1z \equiv_L xy_2z$. We denote the *congruence class* of y by $[y]_L = \{y' \in \Sigma^* \colon y \equiv_L y'\}$ and, by extension, of a set of congruent strings Y by $[Y]_L$. The congruence class $[\epsilon]_L$ is called the *unit congruence class*. The set $\{y\colon D_L(y) = \emptyset\} = \Sigma^* \setminus \mathrm{Sub}(L)$, when non-empty, is called the *zero congruence class*. A congruence class is non-zero if it is a subset of $\mathrm{Sub}(L)$. Defining classically the concatenation of languages L_1

and L_2 by $L_1L_2 = \{y_1y_2 \colon y_1 \in L_1, y_2 \in L_2\}$, let us remark that for any strings y_1, y_2 in Σ^* we have $[y_1y_2]_L \supseteq [y_1]_L[y_2]_L$.

We will also consider the local distribution $D_L^{k,l}(y)$ of a string y in a language L, where k and l define respectively the size of left and right contexts. To handle short strings and their borders in a smooth way as in (Luque and López, 2010), we will consider in the sequel that the strings w of a language are virtually extended to $\triangleleft^k w \triangleright^l$, where $\triangleleft$ and $\triangleright$ are specific start and end terminal symbols (not in Σ), and that local contexts can start and end with these symbols, *i.e.* that left and right contexts can be taken respectively in $\Sigma^{\triangleleft k} = (\Sigma \cup \triangleleft)^k \cap (\triangleleft^*.\Sigma^*)$ and $\Sigma^{l\triangleright} = (\Sigma \cup \triangleright)^l \cap (\Sigma^*.\triangleright^*)$. The set of k,l-local extended contexts of y in L is then $E_L^{k,l}(y) = \{\langle u', v' \rangle \in \Sigma^{\triangleleft k} \times \Sigma^{l\triangleright} \colon u'yv' \in \mathrm{Sub}(\triangleleft^k L \triangleright^l)\}$ and the corresponding set of k,l-local contexts is $D_L^{k,l}(y) = \{\langle u, v \rangle \in \Sigma^{\leq k} \times \Sigma^{\leq l} \colon \exists \langle su, ve \rangle \in E_L^{k,l}(y)$ with $s \in \triangleleft^*, e \in \triangleright^*\}$.

Classes of substitutable languages. Interested by studying learnability of natural languages in the substitutability framework, Clark and Eyraud (2007) introduced the class of substitutable languages on the basis of the very simple criterion that strings sharing one common context are substitutable. More formally, we define two non-empty strings y_1 and y_2 to be *weakly substitutable* in a language L, denoted $y_1 \doteq_L y_2$, if $D_L(y_1) \cap D_L(y_2) \neq \emptyset$. A language L is *substitutable* (Clark and Eyraud, 2007) iff for any $y_1, y_2 \in \Sigma^+ \colon y_1 \doteq_L y_2 \implies y_1 \equiv_L y_2$. The preceding definition of weak substitutability is based on *global* contexts. Motivated by the application to proteins, we came up in (Coste et al., 2012a) with a less demanding criterion based on *local* contexts. Introducing parameters k and l specifying the sufficient left and right lengths of the local contexts used as evidence for strings substitutability, we define two non-empty strings to be weakly k,l-local substitutable, denoted by $\boldsymbol{y_1 \doteq_L^{k,l} y_2}$ if $D_L^{k,l}(y_1) \cap D_L^{k,l}(y_2) \neq \emptyset$. A language L is said k,l-*local substitutable* (Coste et al., 2012a) iff for any $y_1, y_2 \in \Sigma^+ \colon \boldsymbol{y_1 \doteq_L^{k,l} y_2 \implies y_1 \equiv_L y_2}$.

We denote by $\mathcal{L}_{\mathrm{S}}$ the class of substitutable languages, by $\mathcal{L}_{k,l\text{-LS}}$ the class of k,l-local substitutable languages for given k and l and by $\mathcal{L}_{\text{-LS}}$ the whole class of local substitutable languages (defined as the union of the languages $\mathcal{L}_{k,l\text{-LS}}$ for finite k and l). From the definition, any language in $\mathcal{L}_{k,l\text{-LS}}$ is also in $\mathcal{L}_{k',l'\text{-LS}}$ for any $k' \geq k$ and $l' \geq l$ (since $y_1 \doteq_L^{k,l} y_2 \implies y_1 \equiv_L y_2 \implies y_1 \doteq_L^{k',l'} y_2$). Each class of $\mathcal{L}_{k,l\text{-LS}}$ is strictly included in the class $\mathcal{L}_{k+1,l\text{-LS}}$ and $\mathcal{L}_{k,l+1\text{-LS}}$, forming a double hierarchy of included classes that tends in the limit towards the class of substitutable languages $\mathcal{L}_{\mathrm{S}}$ that can be considered as the $\mathcal{L}_{\infty,\infty\text{-LS}}$ class of languages.

Context-free grammars. In next sections, we consider the problem of learning substitutable languages represented by *context-free grammars*. A context-free grammar (CFG) is defined as a 4-tuple $G = \langle \Sigma, N, P, N_0 \rangle$ where Σ is a finite alphabet of terminal symbols, N is a finite alphabet of non-terminal symbols, $N_0 \in N$ is the start symbol (also named axiom) and P is a finite set of production rules of the form $A \to \alpha$ where the left-hand side (lhs) A belongs to N and the right-hand side (rhs) α belongs to $(N \cup \Sigma)^*$. We say that a sentential form $\delta A \gamma$ from $(N \cup \Sigma)^+$ can be derived into $\delta\alpha\gamma$, denoted $\delta A\gamma \Rightarrow_G \delta\alpha\gamma$, if there exists a production rule $A \to \alpha$ in P. The transitive closure of $\Rightarrow_G$ is denoted $\Rightarrow_G^+$ and its reflexive transitive closure $\Rightarrow_G^*$. Given a sentential form α in $(N \cup \Sigma)^+$, we denote by $\hat{L}(\alpha)$ the set of sentential forms $\{\beta \in (N \cup \Sigma)^* : \alpha \Rightarrow_G^* \beta\}$ and $L(\alpha)$ the set of strings $\{w \in \Sigma^* : \alpha \Rightarrow_G^* w\}$ that can be derived from α. The context-free language defined by a

context-free grammar G is $L(G) = L(N_0)$. We will assume that target grammars contain no useless symbols or rules (Hopcroft et al., 2003).

3. Learning grammars in reduced normal form

To study the approach by reduction from a theoretical point of view, we introduce here a simplified version of ReGLiS (Reduced Grammar inference by Local Substitutability), modified to ensure returning a RNF grammar in polynomial time. The pseudo-code of this new algorithm, named ReGLiS$_{\text{core}}$, is given in Algorithms 1 and 2, using $[x]$ to denote the class in $\mathcal{C}_S$ of a substring or set of substrings x. In subsection 3.1, we present first the approach by reduction introduced by ReGLiS from a general point of view and we introduce a formal characterization of the RNF grammars which are targeted by the learning algorithm. The details of the pseudo-code for an efficient reduction based on parsing graphs are then presented in subsection 3.2.

Algorithm 1: ReGLiS$_{\text{core}}$.

Input: Set of strings S on alphabet Σ, int k, int l
Output: Reduced grammar consistent with S

`/* Partition substrings into substitutability classes wrt evidence in` S `*/`
$\mathcal{C}_S \leftarrow \{\{y\} : y \in \text{Sub}(S) \setminus \{\epsilon\}\}$ `/* Partition substrings into different classes */`
foreach $y_1, y_2 : y_1 \doteq_S^{k,l} y_2$ **do** `/* Merge classes of substitutable substrings */`
  $\mathcal{C}_S \leftarrow (\mathcal{C}_S \setminus \{[y_1], [y_2]\}) \cup \{[y_1] \cup [y_2]\}$
`/* Keep only` $\mathcal{C}_S$`-prime substitutability classes */`
$\mathcal{P}_S \leftarrow \{C \in \mathcal{C}_S : \forall C_1, C_2 \in \mathcal{C}_S, C \not\subseteq C_1 C_2\}$
`/* Build bottom grammar */`
$N \leftarrow \emptyset, P \leftarrow \emptyset$
foreach $C \in (\mathcal{P}_S \cup \{[S]\})$ **do**
  $N \leftarrow N \cup \{N_C\}$
  foreach $y \in C$ **do**
    $P \leftarrow P \cup \{N_C \to y\}$
`/* Generalize grammar by reduction */`
try
  $R \leftarrow \emptyset$;
  foreach $(N_C \to \alpha) \in P$, ordered by increasing $|\alpha|$ **do**
    `/* Reduce rule (see Algorithm 2) */`
    foreach $\beta \in$ `reduce_rhs`(α, R) **do**
      if $\beta \notin N$ **then**
        $R \leftarrow R \cup \{N_C \to \beta\}$
  `/* Check RNF condition 2 */`
  if $\exists N_C \neq N_{[S]} : |\{(N_C \to \beta) \in R\}| = 1$ **then**
    raise exception FAIL (Non-terminal for composite class)
catch exception (FAIL)
  return $\langle \Sigma, \{N_0\}, \{N_0 \to w : w \in S\}, N_0 \rangle$ `/* Trivial RNF grammar of` S `*/`
`/* Success */`
return $\langle \Sigma, N, R, N_{[S]} \rangle$

3.1. Learning by reduction

The general sketch of the approach is to start from a bottom grammar, representing the training sample S and the classes of substitutable substrings from S that are implied by weak k, l-local substitutability evidence in S.

More precisely, the bottom grammar for S contains a non-terminal N_C for each class C of substitutable substrings from S and a rule $N_C \to y$ to derive directly from N_C each substring y in C. The language generated from each non-terminal N_C is then the set of strings from C. Its start symbol being the non-terminal for the class including S ($N_{[S]}$ in the pseudo-code), the bottom grammar recognizes only the training set S.

Generalization is then achieved by reduction of the right-hand sides. The idea is to replace rules of the form $B \to \delta\alpha\gamma$ such that α can be derived from non-terminal A, by rules $B \to \delta A\gamma$. Such a reduction shortens the right-hand side of the rule and generalizes the derived language, from $L(\delta)L(\alpha)L(\gamma)$ to $L(\delta)L(A)L(\gamma)$ with $L(\alpha) \subseteq L(A)$. Remark that if $L(A) = L(\alpha)$ this reduction, as well as the non-terminal A, is useless from the generalization point of view and introduces unnecessary additional derivation steps for parsing. Otherwise, the generalization ensures the required substitutability of strings derived from this occurrence of α by all the strings in their substitutability class, which will be represented by A. Also note that reductions can compete on the right-hand side of a rule, resulting in potentially incomparable generalizations. These will require the replacement of one rule by several reduced alternatives to ensure required substitutability.

Combined with the removal of the non-axiom non-terminals that are head of exactly one derivation rule (replacing their occurrences by the right-hand side of the rule) and with the unification of non-terminals that can be derived into the same right-hand side, reductions of bottom's grammar rules will converge towards a grammar with fully reduced rules, said in reduced normal form (RNF). Instead of simply defining RNF as the result of the reduction, we propose here a formal characterization of RNF which does not rely on any predefined class of languages:

Definition 2 (RNF) *A context-free grammar $\langle \Sigma, N, P, N_0 \rangle$ is in reduced normal form if:*

1. $\forall A \in N, L(A) = [L(A)]_L$ *(a non-terminal represents exactly one congruence class)*
2. $\forall A \in N, A \neq N_0 \colon |\{(A \to \alpha) \in P\}| > 1$ *(each non-terminal, other than the axiom, has alternative derivations)*
3. $\forall A, B \in N \colon (B \Rightarrow_G^* \delta\alpha\gamma \wedge A \Rightarrow_G^* \alpha) \implies (\exists (B \to \delta' A \gamma') \in P \colon \delta' \Rightarrow_G^* \delta \wedge \gamma' \Rightarrow_G^* \gamma)$ *(existence of reduced rules)*
4. $\forall B \in N, \forall (B \to \beta_1), (B \to \beta_2) \in P \colon L(\beta_1) \subseteq L(\beta_2) \implies \beta_1 = \beta_2$ *(only fully reduced rules)*

RNF is a canonical form for local substitutable languages (Coste et al., 2014). The first condition implies that the class of RNF-grammars is a subclass of strongly congruential context-free grammars (SC-CFG) (Scicluna, 2014). We conjecture that RNF could also be a canonical form for the class of languages represented by SC-CFG.

The second condition discards non-terminals giving rise to vacuous local derivation trees (Clark, 2011). The corresponding classes of congruence are said to be *composite* since they can be factorized into the concatenation of several congruence classes. Intuitively, if $A \to \alpha_1\alpha_2 \dots \alpha_{|\alpha|}$ is the unique derivation of A, then $L(A) = [L(A)] = [L(\alpha_1)][L(\alpha_2)] \dots [L(\alpha_{|\alpha|})] = [L(\alpha_1)L(\alpha_2) \dots L(\alpha_{|\alpha|}]$. Formally:

Definition 3 (Composite class) *Let L be a language whose set of non-zero and non-unit congruence classes is $\mathcal{C}_L$. A class $C \in \mathcal{C}_L$ is* composite *for L iff: $\exists C_1, C_2 \in \mathcal{C}_L, C = C_1C_2$*

We say that a congruence class is *prime* if it is not composite. From the two first conditions, RNF grammars have non-terminals only for prime congruence classes. Non-terminal alphabet being finite by definition, reduced normal form can represent only languages with a finite set of primes, excluding thus substitutable context-free languages with an infinite number of primes (Clark, 2013). In contrast, this does not restrict the class of local substitutable languages that can be represented, since their number of congruence classes, and thus of prime congruence classes, is always finite (the number of contexts of fixed size k, l on a finite alphabet characterizing them being finite). Our intuition is that the class of substitutable context-free languages with a finite number of primes actually corresponds to the class of local substitutable languages.

The last two conditions ensure to have only fully reduced rules. The premise of the third condition is the same as in the definition of Non-Terminal Separable (NTS) languages (Boasson and Sénizergues, 1985). This condition implies that RNF grammars are NTS, but the goal is here to ensure the existence of rules in (fully) reduced form. Introducing such rules does not change the language represented by the grammar because of the first condition. Keeping only right-hand sides generating most general languages, as implied by the fourth condition, will ensure to keep only fully reduced rules since they subsume the other rules.

RNF grammars are compact NTS and SC-CFG representations of languages without useless non-terminals and with smallest non-redundant right-hand-sides. As for deterministic versus non-deterministic representations of automata, RNF grammars can yet be exponentially bigger than non-RNF context-free grammars: in the example of languages $\{a^{2^n}\}$, RNF grammars are $\langle\{a\}, \{N_0\}, \{N_0 \to a^{2^n}\}, N_0\rangle$ (with only one non-terminal since congruence classes are unique) and are exponentially bigger than the context-free grammars G_n presented in the introduction. Note that for any other finite language F, the RNF grammar of F looks alike and is: $\langle\Sigma, \{N_0\}, \{N_0 \to w \colon w \in F\}, N_0\rangle$ (but represents a substitutable language only if F is already susbstitutable).

3.2. Efficient reduction

We detail here how k, l-local substitutable languages are learnt efficiently by reduction from a training sample S in Algorithm 1.

Bottom grammar on prime substitutability classes. The set $\mathcal{C}_S$, partitioning substrings from S into classes of substitutable substrings, is built classically by initializing the partition to singleton sets and merging the classes of each pair of weakly k, l-local substitutable substrings in S.

Algorithm 2: reduce_rhs

Input: String $\alpha \in (\Sigma \cup N)^+$, Set R of rewriting rules from N to $(\Sigma \cup N)^+$
Output: Set of irreducible reductions of α or FAIL exception

```
   /* Initialization */
1  V ← {i ∈ [1, |α| + 1]}  /* set of vertices (parsing positions) */
2  E ← ∅  /* set of labeled directed edges in parsing graph */
3  for i ← 1 to |α| do
       /* Labeled edge to next position */
4      if ∃A: (A → α[i]) ∈ R then
5          E ← E ∪ {(i, i + 1, A)}
6      else
7          E ← E ∪ {(i, i + 1, α[i])}
       /* Trivial irreducible path to next position */
8      IPaths[i, i + 1] ← {(i, i + 1)}
   /* Incremental bottom up computation by dynamic programming */
9  for j ← 3 to |α| + 1 do
10     for span ← 2 to j − 1 do
11         i ← j − span
12         Paths ← ∅  /* set of paths from i to j */
13         for mid ← i + 1 to j − 1 do
14             {π_l} ← IPaths[i, mid] ; β_l ← labeling(π_l, E)
15             {π_r} ← IPaths[mid, j] ; β_r ← labeling(π_r, E)
16             if ∃A: (A → β_l β_r) ∈ R then
17                 if ∃(i, j, B) ∈ E, B ≠ A then
18                     raise exception FAIL (Two non-terminals for same class)
                   /* New edge in parsing graph */
19                 E ← E ∪ {(i, j, A)}
                   /* There is thus a direct path from i to j */
20                 Paths ← {(i, j)}
21             else if Paths ≠ {(i, j)} then
                       /* Path from i to j through mid is π_l chained with π_r */
22                 Paths ← Paths ∪ {π_l + π_r}
           /* Keep only irreducible paths */
23         IPaths[i, j] ← {π ∈ Paths: ∀π' ∈ Paths, π ≻_r π' ⟹ π = π', }
24         if (i ≠ 1 and j ≠ |α| + 1) and |IPaths[i, j]| > 1 then
25             raise exception FAIL (Two irreducible paths from/to internal position)
   /* Return labeling of irreducible paths from first to last positions */
26 return {labeling(π, E) : π ∈ IPaths[1, |α| + 1]}
```

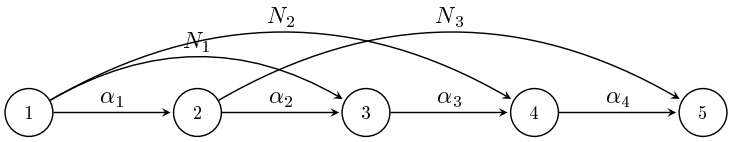

Figure 1: Parsing graph of $\alpha = \alpha_1\alpha_2\alpha_3\alpha_4$ with $N_1 \to \alpha_1\alpha_2$, $N_2 \to N_1\alpha_3$, $N_3 \to \alpha_2\alpha_3\alpha_4$. Full reductions of α are then $N_2\alpha_4$ and $\alpha_1 N_3$, corresponding respectively to irreducible paths $(1,4,5)$ and $(1,2,5)$, while $N_1\alpha_3\alpha_4$ is not fully reduced as witnessed by $(1,3,4,5) \succ_r (1,4,5)$.

Each singleton class C in $\mathcal{C}_S$ contains a unique string y that has no evidence in S for being substitutable with any other string. It would introduce a unique and useless derivation rule of N_C in the bottom grammar, in contradiction with RNF condition 2. Such classes, as well as other classes without evidence in S for being prime, are discarded by ReGLiS and ReGLiS$_{\text{core}}$. To carry out this early detection in ReGLiS$_{\text{core}}$, a class C in $\mathcal{C}_S$ is said *$\mathcal{C}_S$-prime* when it is not included in the concatenation of any two other classes of $\mathcal{C}_S$, *i.e.* $\forall C_1, C_2 \in \mathcal{C}_S, C \nsubseteq C_1C_2$. A $\mathcal{C}_S$-prime class C contains thus at least two sequences y_1, y_2 in C, such that for any of their decomposition u_1v_1 and u_2v_2, we have $[u_1] \neq [u_2]$ and $[v_1] \neq [v_2]$. Assuming that the congruence partition of the substrings of S remains unchanged during generalization, C will require at least two rules to produce sequences y_1 and y_2. It satisfies then RNF condition 2 and is kept by ReGLiS$_{\text{core}}$[1] to build the bottom grammar.

Full reduction with parsing graph. Grammar generalization in ReGLiS$_{\text{core}}$ is a simple pass full reduction of the rules ordered by their length. This ordering enables us to use the result of the reduction of each substring of the rule's right-hand side, to reduce it. As introduced by ReGLiS, full reduction of the rule is performed efficiently by dynamic programming on the parsing graph of the right-hand side of the rule. An example of parsing graph is shown in Figure 1. The vertices are the positions (numbered from 1 to $|\alpha| + 1$) between the symbols of the string α to be reduced. Edges (i, j, l) indicate that $\alpha[i, j-1]$ can be derived from symbol l. To have at most one edge between two positions, edges between consecutive positions $i, i+1$ are labeled by the non-terminal generating $\alpha[i]$ if it exists and by $\alpha[i]$ otherwise. A path π in the parsing graph can then be represented as a sequence of positions, which defines implicitely a reduction β of α (returned by `labeling(`π, E`)`, with E being the set of labeled edges) and can easily be compared to other reductions of α. A path π_1 is *reducible* in another path π_2, denoted $\pi_1 \succ_r \pi_2$, if π_2 is a subsequence of π_1 with same starts and ends. The property $\pi_1 \succ_r \pi_2$ holds iff there exists a series of reductions of α from the one defined by π_1 to the one defined by π_2. A path π is *irreducible* if it is minimal with respect to partial order $\succ_r$. The set of irreducible paths in the parsing graph represent then all full reductions of α.

1. $\mathcal{C}_S$-prime test is a simplified version of ReGLiS' S-Prime test, which can be written $\forall D \in \mathcal{C}_S \colon C \nsubseteq D\Sigma^+ \wedge C \nsubseteq \Sigma^+ D$. In $\mathcal{C}_S$-prime test, C_1 and C_2 need to be in $\mathcal{C}_S$. In S-Prime test, only D needs to be in $\mathcal{C}_S$: the other class E, such that $C = DE$ or $C = ED$, is assumed to exist from substitutable languages properties, even if it is not in $\mathcal{C}_S$. This enables ReGLiS to better discard composite classes in the absence of characteristic sample, when ReGLiS$_{\text{core}}$ will stick to RNF condition 2.

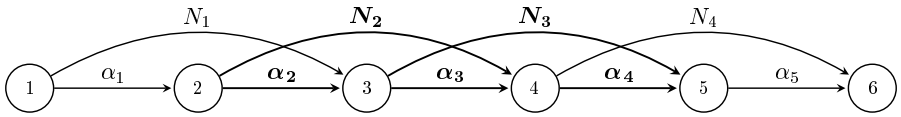

Figure 2: Parsing graphs with $F_{|\alpha|-1}$ irreducible paths. The number of irreducible reductions at vertex i is the sum of the number of irreducible paths reaching $i-1$ and $i-2$, and is thus given by Fibonacci number F_{i-1} with $F_0 = 0, F_1 = 1$.

Finding all irreducible paths can be achieved efficiently by dynamic programming. Pseudo-code of function `reduce_rhs` (Algorithm 2), shows how to do it *simultaneously* with a bottom-up chart parsing construction of the parsing graph. Irreducible paths between pairs of positions are stored in a triangular matrix IPaths. It can be initialized trivially, as well as the parsing graph, for successive positions. Parsing graph and IPaths are then filled incrementally up to increasing positions j, by detecting new edges from positions $i < j$ to j (in increasing span order to ensure bottom up completion of parsing) and by computing IPaths$[i, j]$ accordingly. This requires an additional loop to test for each position *mid* between i and j, if the concatenation of the labels along the irreducible path traversing *mid* matches a rule's right-hand side. In such a case, an edge from i to j labelled by the left-hand side of the rule is added to the parsing graph. The set IPaths$[i, j]$ is computed during the same loop. If an edge is detected, it is (i, j). Otherwise it is the set of irreducible paths among the paths chaining paths of IPaths$[i, mid]$ and IPaths$[mid, j]$. Chaining paths π_l and π_r without duplicating position *mid* is written $\pi_l + \pi_r$ in pseudo-code

4. Analysis of learning complexity

Polynomial time. A key to control the complexity of the algorithm is to limit the number of irreducible paths. While running in polynomial-time when a characteristic sample is included in the training set, ReGLiS can nevertheless generate a non-polynomial number of irreducible paths otherwise. An example of parsing graph with a non-polynomial number of irreducible paths is given in Figure 2. Analyzing this example, it can be noted however that some edges are missing. For instance there should be an edge from position 2 to position 5 if N_2 and N_3 represent prime congruence classes. Indeed, the class represented by the reduction of $\alpha_2\alpha_3\alpha_4$ is represented by two irreducible right-hand-side $N_2\alpha_4$ and $\alpha_2 N_3$ and is thus prime. If there was a non-terminal N_5 for that class and the corresponding edge $(2, 5)$, the substring $\alpha_2\alpha_3\alpha_4$ could be fully reduced in a unique right-hand side N_5. On the other hand, if N_2 (or N_3) does not represent a prime congruence class, the edge between 2 and 4 (or 3 and 5) should be removed, and $\alpha_2\alpha_3\alpha_4$ would be reduced in a unique $\alpha_2 N_3$ (or $N_2\alpha_4$). By generalizing this reasoning, we say that a *parsing graph is nice* (for learning local substitutable languages) if there is *only one irreducible path between its pairs of positions, except between start and end positions* (the "missing" edge would then be the edge for the class itself). This assertion is violated by pairs $(1, 4), (2, 5), (3, 6), (1, 5), (2, 6)$ in Figure 2 and is satisfied in Figure 1.

When a graph is not nice, one can try to fix it by adding the missing edges, and eventually the missing classes. The solution adopted in $\text{ReGLiS}_{\text{core}}$ is simply to raise an error and wait/ask for a more complete training sample, when the test at the end of the `reduce_rhs()` iterations, except the last one, fails. It ensures also the uniqueness of irreducible path between pairs of positions before last iteration. Parsing is then simplified, since $\pi_l, \pi_r, \beta_l, \beta_r$ are unique, and `reduce_rhs()` is then guaranteed to return at most $|\alpha| - 2$ reduced rules from α, ensuring so $\text{ReGLiS}_{\text{core}}$ to run in *polynomial time*.

To establish more precisely time complexity, we recall that the maximal number of substrings of a sequence of length m is $\frac{m(m+1)}{2}$. For a training sample S with n sequences of maximal length m, the total number of substring occurrences (and thus of classes) is $\mathcal{O}(nm^2)$. Complexities for retrieving substitutability classes and building the bottom grammar are thus also $\mathcal{O}(nm^2)$, while complexity for keeping only $\mathcal{C}_S$-prime classes remains the same as for S-Prime test: $\mathcal{O}(nm^3)$ (Coste et al., 2014). The main generalization loop over right-hand sides $|\alpha|$ is done $\mathcal{O}(nm^2)$ times. The complexity of one iteration is that of `reduce_rhs()`: $\mathcal{O}(m^3)$, thanks to the integration of irreducible paths computation in the three nested loops of bottom-up parsing, dominating $\mathcal{O}(m^2)$ generation of reduced right-hand sides. Overall, the *complexity of ReGLiS$_{core}$* is $\mathcal{O}(nm^5)$.

The strategy of raising an error is also used to cope with the detection of two different non-terminals for the same class of substitutability (due to the presence in R of two S-Prime classes for one class) in contradiction with RNF condition 1 and the detection of non-terminals with only one production rule (due to a S-Prime class that is not prime) in contradiction with RNF condition 2. These guards, combined with the full reduction by `reduce_rhs()`, ensuring RNF conditions 3 and 4, enable $\text{ReGLiS}_{\text{core}}$ to return a RNF grammar in the absence of exception. When an error is raised, $\text{ReGLiS}_{\text{core}}$ returns simply the RNF grammar of S. The first condition of IPTtD is then satisfied: $\text{ReGLiS}_{\text{core}}$ returns a RNF grammar consistent with S, in polynomial time with respect to the size of S.

Polynomial data. Clark and Eyraud (2007) defines the characteristic sample of a grammar $G = \langle \Sigma, N, P, N_0 \rangle$ by $CS(G) = \{uwv \in \Sigma^* \colon (A \to \alpha) \in P, (u, v) = c(A), w = w(\alpha)\}$ where G is not assumed to be in any special form, with $w(\alpha)$ and $c(N)$ denoting respectively the smallest string generated from a sentential form α and the smallest context enabling to reach a non-terminal N from N_0. Since each local substitutable language L can be represented by its canonical grammar in RNF $G_R(L)$, we can be more precise here and define the characteristic sample of a language L in $\mathcal{L}_{k,l\text{-LS}}$ by $CS(L) = CS(G_R(L))$. This characteristic set ensures the existence of a non-terminal for each prime class and the possibility of building by reduction the target grammar from the smallest substitutable strings of each class. Other sequences in the training set are then correctly handled (parsing and irreducible paths) by this grammar, allowing to discard eventual redundant classes in $\mathcal{P}_S$ by the test $\beta \notin N$ of $\text{ReGLiS}_{\text{core}}$. The number of strings is linear in the number of rules by definition. From the definition of the thickness of a grammar G ($\tau_G = \max\{|w(\alpha)| \colon (A \to \alpha) \in P\}$), the length of the string generated for a rule is trivially linearly bounded by the thickness. The second condition of IPTtD is then satisfied: for each k, l-local substitutable language L, there is a characteristic sample of polynomial size with respect to the size of its canonical grammar in RNF $G_R(L)$ and its thickness, ensuring $\text{ReGLiS}_{\text{core}}$ to return $G_R(L)$ when it is included in the training set and k, l parameters are given. This shows that k, l-local substi-

tutable languages represented by RNF grammars are IPTtD. Remark that convergence will also be ensured if bigger values are chosen for k and l, from the inclusion of $\mathcal{L}_{k,l\text{-LS}}$ classes of languages. This allows a certain independence with regard to the choice of parameters. At the extreme, since training samples are always finite in practice, choosing for k and l the maximum length of words in S would emulate contexts of infinite lengths for learning any local substitutable language.

5. Discussion and perspectives

We proposed a formal characterization of RNF grammars, which are the grammars obtained by the reduction approach introduced by ReGLiS. This characterization is independent from the class of language represented and we think that RNF could also be useful for the inference of languages represented by strongly congruential grammars. We presented also a simplified version of ReGLiS enabling easier study of the core of the approach by reduction. Introducing the requirement of only one irreducible path between pairs of positions of the parsing graph, except between the first and last positions, provides a way to detect insufficient training samples. An efficient algorithm performing this detection during construction of parsing graph was presented. In practice, this detection could be exploited to add new substitutability classes or initiate interactions with the user. It is also a first step to test if a grammar represents a substitutable language. In this work, it ensures to return a consistent RNF grammar in polynomial time. Combined with the polynomial size of the characteristic sample with respect to the size of the representation and its thickness, this enables to show that *k, l local substitutable languages represented by RNF context-free grammars are IPDtT* with this algorithm.

Acknowledgments

FC and JN wish to acknowledge the reviewers for their detailed comments and suggestions to improve the clarity of the manuscript. FC is also grateful to Rémi Eyraud and Ryo Yoshinaka for helpful comments and discussion on an early version of this work in Kyoto.

References

Pieter W. Adriaans. *Language Learning from a Categorial Perspective*. PhD thesis, Universiteit van Amsterdam, 1992.

Pieter W. Adriaans and Marco Vervoort. The EMILE 4.1 grammar induction toolbox. In *ICGI*, volume 2484 of *Lecture Notes in Computer Science*, pages 293–295. Springer, 2002.

Luc Boasson and Géraud Sénizergues. NTS languages are deterministic and congruential. *J. Comput. Syst. Sci.*, 31(3):332–342, 1985.

Jorge Calera-Rubio and José Oncina. Identifying left-right deterministic linear languages. In *ICGI*, volume 3264 of *Lecture Notes in Computer Science*, pages 283–284. Springer, 2004.

Alexander Clark. Distributional learning of some context-free languages with a minimally adequate teacher. In *ICGI*, volume 6339 of *Lecture Notes in Computer Science*, pages 24–37. Springer, 2010.

Alexander Clark. A language theoretic approach to syntactic structure. In *MOL*, volume 6878 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2011.

Alexander Clark. Learning trees from strings: a strong learning algorithm for some context-free grammars. *Journal of Machine Learning Research*, 14(1):3537–3559, 2013.

Alexander Clark and Rémi Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007.

Alexander Clark and Ryo Yoshinaka. Distributional learning of parallel multiple context-free grammars. *Machine Learning*, 96(1-2):5–31, 2014.

François Coste, Gaëlle Garet, and Jacques Nicolas. Locally substitutable languages for enhanced inductive leaps. In *ICGI*, volume 21 of *JMLR Proceedings*, pages 97–111. JMLR.org, 2012a.

François Coste, Gaëlle Garet, and Jacques Nicolas. Learning context free grammars on proteins by local substitutability. unpublished first submission of ReGLiS, 2012b.

François Coste, Gaëlle Garet, and Jacques Nicolas. A bottom-up efficient algorithm learning substitutable languages from positive examples. In *ICGI*, volume 34 of *JMLR Proceedings*, pages 49–63. JMLR.org, 2014.

Colin de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27(2):125–138, 1997.

Colin de la Higuera and José Oncina. Inferring deterministic linear languages. In *COLT*, volume 2375 of *Lecture Notes in Computer Science*, pages 185–200. Springer, 2002.

Rémi Eyraud, Jeffrey Heinz, and Ryo Yoshinaka. Efficiency in the identification in the limit learning paradigm. In Jeffrey Heinz and José M. Sempere, editors, *Topics in Grammatical Inference*. Springer-Verlag Berlin Heidelberg, 2016.

Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation - international edition (2. ed)*. Addison-Wesley, 2003. ISBN 978-0-321-21029-6.

Franco M. Luque and Gabriel G. Infante López. Pac-learning unambiguous k,l-NTS $^{<=}$ languages. In *ICGI*, volume 6339 of *Lecture Notes in Computer Science*, pages 122–134. Springer, 2010.

James Scicluna. *Grammatical inference of probalistic context-free grammars*. PhD thesis, Nantes University, December 2014.

José M. Sempere and Pedro García. A characterization of even linear languages and its application to the learning problem. In *ICGI*, volume 862 of *Lecture Notes in Computer Science*, pages 38–44. Springer, 1994.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11629–11634, 2005.

Yuji Takada. Grammatical interface for even linear languages based on control sets. *Inf. Process. Lett.*, 28(4):193–199, 1988.

Yuji Takada. A hierarchy of language families learnable by regular language learners. In *ICGI*, volume 862 of *Lecture Notes in Computer Science*, pages 16–24. Springer, 1994.

Menno van Zaanen. ABL: alignment-based learning. In *COLING*, pages 961–967. Morgan Kaufmann, 2000.

Ryo Yoshinaka. Learning efficiency of very simple grammars from positive data. In *ALT*, volume 4754 of *Lecture Notes in Computer Science*, pages 227–241. Springer, 2007.

Ryo Yoshinaka. Identification in the limit of k, l-substitutable context-free languages. In *ICGI*, volume 5278 of *Lecture Notes in Computer Science*, pages 266–279. Springer, 2008.

Ryo Yoshinaka. Learning mildly context-sensitive languages with multidimensional substitutability from positive data. In *ALT*, volume 5809 of *Lecture Notes in Computer Science*, pages 278–292. Springer, 2009.

Ryo Yoshinaka. Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. *Theor. Comput. Sci.*, 412(19):1821–1831, 2011.