Noname manuscript No.
(will be inserted by the editor)

Learning Context Free Grammars on Proteins by Local Substitutability

François Coste · **Gaëlle Garet** · **Jacques Nicolas**

Received: date / Accepted: date

Abstract The grammatical inference field has long been driven by language theoretic considerations and the interest of better understanding natural language learning. With its ability to produce large sets of sequences and to check predictions on new sequence membership, genomics has become an appealing application domain. This paper studies a task of importance in bioinformatics, the annotation of new proteins with respect to banks of already annotated protein sequences. Proteins present several difficulties with respect to grammatical inference, mainly their size and the long range interactions that share its elements through folding in space. Our contribution relates both to formal aspects (generalization criterion and class of languages) and to practical aspects (algorithms and experiments) of the inference process. Starting from the work of Clark and Eyraud [2007] and Yoshinaka [2008] on inference of substitutable and k,l-substitutable languages respectively, we introduce new classes of substitutable languages based on local rather than global substitutability, a reasonable assumption with respect to protein structures. The paper provides a practical algorithm supporting the determination of string substitutability by fixed-size left and right contexts. The interest of this approach is shown on the challenging sets of proteins proposed in Dyrka and Nebel [2009] and results are compared with stochastic context free grammars and simple regular expressions learning. With a preliminary coding step of conserved substrings in the protein learning set, it is possible to learn grammars with high specificity and good sensitivity.

Keywords Local substitutable languages · grammatical inference · bioinformatics · proteins

François Coste
Inria Rennes - Bretagne Atlantique, Campus universitaire de Beaulieu, Rennes, France
E-mail: francois.coste@inria.fr
Gaëlle Garet
E-mail: gaelle.garet@inria.fr
Jacques Nicolas
E-mail: jacques.nicolas@inria.fr

1 Introduction

Grammatical inference aims at producing machines or rewriting systems that recognize a language from a sample of sequences. The field has long been driven by language theoretic considerations and the interest of better understanding natural language learning. With the availability of large training sets of sequences and the opportunity to check predictions on new sequence membership through new experiments, genomics has progressively become an appealing application domain for sequence model learning. Technological improvements now allow many biological laboratories to obtain new genomic sequences of good quality for an affordable price and the cumulated volume of these data has greatly increased. Genomic banks are fed continuously by large sets of DNA or RNA sequences coming from high throughput machines and to date, almost 2000 species have their genome completely sequenced. The new generation of sequencing technologies has introduced the study of sequence variations within populations and it becomes even more important to be able to identify which of the variants are causal with respect to a particular disease state or phenotypic trait (Zhang et al. [2011]).

Protein annotation is a task of first importance in this context. It involves the search of protein coding genes within the sequences and the prediction of the function of these new proteins in the cell by comparison of their sequences with known families. Indeed, proteins are generally highly conserved through species and this allows to delineate families and super-families of related elements and predict some functional properties with good accuracy on the basis of common shared motifs. There exist several databases (Hunter et al. [2012]) and a broad range of methods and softwares for the discovery of characteristic protein motifs from sets of sequences. As stated in Galperin and Koonin [2010], the annotation of protein families is far more realistic than the annotation of individual proteins since it allows to clearly focus on a particular functional aspect (individual proteins have generally multiple functional effects leading to different phenotypic traits, a phenomenon referred as 'pleiotropy'). Many popular machine learning methods have been successful in Bioinformatics, especially the *discriminative* ones such as the neural networks and support vector machines which have been shown to be very efficient even for sequence classification tasks (see for instance Baldi and Brunak [2001]).

As a matter of fact, it is not so natural to use discriminative methods on natural sequences. This has been quite evident in natural language studies. The notion of counter-example is generally fuzzy and there exists a significant imbalance between the cardinality of positive and negative classes. Moreover, predictive accuracy is not the sole objective of machine learning. Of utter importance is the understanding of possible generative mechanisms at work for the production of observed sequences. Roughly, there exists two possible modeling approaches when considering analyzing and learning natural languages from positive examples of correct sentences. The first one looks for a hierarchy of concepts as in the phrase structures introduced by Chomsky [1957] to study syntax acquisition by children. The second one is word-centered and looks for dependencies between the different properties of neighbour words as in Lexicalized Tree Adjoining Grammars (Schabes [1990]) in order to better account for local constraints and to obtain simpler grammars. Most of results in grammatical inference are obtained in the framework of Chomsky's hierarchy, but the second framework has the advantage of stressing the importance of lexicalization and analysis of local dependencies. In fact, an old idea at the basis of practical inference algorithms and recently formalized in Clark and Eyraud [2007] relies on the notion of common contexts: substitutability. Two sequences are substitutable if they occur in the same context in

the pairs of sentences where they appear. The idea is then to generalize the observation of a pair of sentences that only differ by two included sequences by inferring the substitutability of these sequences. This is safe (the class of languages is learnable in the limit) if every pair of sequences appearing in a common context is substitutable, a criterion defining the class of substitutable languages. The k,l-substitutable languages (Yoshinaka [2008]) is a natural extension of this concept requiring a minimal fixed length on the considered contexts for its applicability. In this paper, we further explore the concept of substitutability for genomic sequences.

The linguistic metaphor has been used indeed for a long time in molecular biology, and applying computational linguistics tools to represent, understand and handle biological sequences is a natural continuation of this metaphor. Using formal grammars has been advocated in particular by Searls [2002] and Chiang et al. [2006] whose articles provide a good introduction to the different levels of expressiveness required to model macromolecular sequences and to related grammatical formalisms. A legitimate question is to ask to what extend could natural language learning models contribute to learning on biological macromolecules.

A striking characteristics of biological sequences with respect to induction relies on the concept of *conservation* and it is another particularity of the sequences we consider in this paper. In biology, genetic variation (by mutations, recombination of chromosomes, crossing-over and other sources of sequence variation) is a fundamental source of diversity which is opposed to natural selection. A conserved feature among a set of sequences is an evidence of selection through evolution and is thus likely to be important for the family. Sequence conservation can be detected by pairwise (Altschul et al. [1990]), or by multiple sequences alignments (Thompson et al. [1994]). They can act globally on the whole length of the sequences or look for local similarities. In this work, we use partial local multiple alignments (PLMA, Kerbellec [2008]), that are able to find all the conservation blocks in a learning set of proteins.

The most successful learning approach to date on protein sequence models is likely to be the parameter estimation approach. Models can be deduced from alignments by computing a score estimating the likelihood of the presence of some letter at a given position in the model. Applied to grammatical models, the principle is to choose a generic simple grammar topology, for instance profile Hidden Markov Models (pHMM) and to fit the parameter probabilities (or weights) to maximize the likelihood of the available sample of sequences from the family (Durbin et al. [1998]; Bailey et al. [2009]). However, since the function of proteins have close relation with their spatial structure, it is far from optimal to analyze them with a fixed simple left-to-right grammar structure.

Of course, learning the grammar is a more demanding task and progresses are slower in this domain, but some steps have already been taken. A subset of regular expressions have been used in the database Prosite (Hulo et al. [2006]), some expressions being inferred by the Pratt motif discovery method (Jonassen et al. [1995]), and we have proposed a method and a tool (Coste and Kerbellec [2005]) to model families of protein sequences with finite automata. Regular grammar topologies or even less expressive formalisms can be sufficient to characterize protein families in many cases, but they cannot model (potentially nested or crossing) long-term dependencies such as contacts of amino-acids that are far in the sequence but close in the 3D folding of the protein. Bryant et al. [2006] uses Inductive Logic Programming and is thus closely related to grammatical inference since it aims at producing Definite Clause Grammars. Authors have implemented a variety of preprocessing steps including a Pratt motif search in order to get a rich background knowledge and have tested

their approach on families of G-protein coupled receptors, restricting learning to subsequences corresponding to known membrane spanning regions of the proteins. The focus of the paper is more on the interest of background knowledge than on the class of languages that are the target of learning and it is thus difficult to assess the contribution of the learned grammar structure. Computational complexity seems also a major concern of this method. In Dyrka and Nebel [2009], the authors propose to learn by genetic algorithms a combination of stochastic context free grammars related to different amino acid physico-chemical properties, which are shown to produce relevant protein binding site descriptors. They have obtained interesting results by carefully splitting sequences in separated domains, as in the previous study. Since it is one of the most advanced application of grammatical inference on proteins to date, we decided to start our own experiments on the same data set.

We propose in this paper to investigate how inference based on substitutability can be applied for modeling families of protein sequences by context-free grammars. In preliminary experiments, we have remarked that almost no generalization was brought by these approaches because the condition to enable one word y_1 to be replaced by another word y_2 was almost never satisfied. As a matter of fact y_1 and y_2 need to be surrounded by the same context w in two sequences and y_1 has to exist as a substring of yet another sequence to imply the existence of a fourth sequence containing y_2. If the sequences are long, observing a double occurrence of the common context w and a double occurrence of y_1, given that at least one of these substrings has to be long, has a low likelihood in practice. Moreover, in this reversible language type of approach, heads (and tails) have to be completely conserved from the beginning (to the end) of the sequences, *i.e.* the context has to be the same around y_1 and y_2 on the full length of the two sequences. In our test sets, this did not occur, except for long y_1 and y_2 that were almost never repeated in other sequences. It seemed clear that more local characterizations are needed in practice, in the spirit of locally testable languages, and this paper introduces a variant of the substitutability criterion for this purpose.

The next section introduces the class of locally substitutable languages, relates it to other classes of languages and give some of its basic properties. Section 3 proposes a new algorithm that builds a grammar from a sample according to k,l-local substitutability. Some emphasis is put on practical issues, focusing especially on a reduced grammar form that is more easy to interpret and more efficient to build. Finally, Section 4 describes an experimentation on the protein data sets issued from the Dyrka & Nebel benchmark.

2 Substitutable Languages

2.1 Definitions and Notations

Before defining the new class of local substitutable languages, we briefly recall the definition of substitutable and k,l-substitutable properties. We use standard definitions and notations for languages and grammars.

An *alphabet* Σ is a finite nonempty set of symbols called *letters*. A *string* w over Σ is a finite sequence $w = a_1 a_2 ...$ of letters. The term $|w|$ denotes the length of w and the empty string of length 0 will be indicated by λ. Let Σ^* be the set of all strings. A *grammar* is a quadruple $G =< V, \Sigma, P, S >$ where Σ is a finite alphabet of terminal symbols, V is a finite alphabet of variables or non-terminals, P is a finite set of production rules, and $S \in V$ is the start symbol. We denote by $L(G) = \{w \in \Sigma^* : S \Rightarrow_G^* w\}$ the language defined by the grammar.

Substitutable and k,l-substitutable languages are defined by:

Definition 1 [Substitutable language (Clark and Eyraud [2007])] A language L is substitutable iff for any $x_1, y_1, z_1, x_2, y_2, z_2 \in \Sigma^*$ such that $y_1, y_2 \neq \lambda$,

$$x_1 y_1 z_1 \in L \wedge x_1 y_2 z_1 \in L \Rightarrow (x_2 y_1 z_2 \in L \Leftrightarrow x_2 y_2 z_2 \in L).$$

Definition 2 [k,l-substitutable language (Yoshinaka [2008])] A language L is k,l-substitutable iff for any $x_1, y_1, z_1, x_2, y_2, z_2 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l$, such that $uy_1v, uy_2v \neq \lambda$,

$$x_1 u y_1 v z_1 \in L \wedge x_1 u y_2 v z_1 \in L \Rightarrow (x_2 u y_1 v z_2 \in L \Leftrightarrow x_2 u y_2 v z_2 \in L).$$

The class of substitutable context free languages is the class of substitutable languages that are context free. k,l-substitutable context free languages are defined similarly.

As pointed out by Yoshinaka [2008], the class of substitutable context-free languages introduced in Clark and Eyraud [2007] are the analogue of zero-reversible regular languages. Like zero-reversible languages have been extended to the hierarchy of k-reversible regular languages, Yoshinaka [2008] defines the hierarchy of k,l-substitutable context-free languages, where substitutable context-free languages are the $0,0$-substitutable context-free languages.

2.2 Locally Substitutable Languages

As stated in the introduction section, the substitutability property is hard to achieve on long sequences (let us note here that this requirement also holds for k,l-substitutability), more local characterizations are needed in practice.

We propose here to introduce new language classes by considering only local contexts around words rather than the global ones required in substitutable languages. A consequence of this relaxation is to introduce the need for an additional parameter, the size of the context used. To allow an asymmetric left or right bias, we introduce two parameters k and l and define the class of k,l-local substitutable languages by:

Definition 3 [k,l-local substitutable language] A language L is k,l-local substitutable if for any $x_1, y_1, z_1, x_2, y_2, z_2, x_3, z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l$, such that $uy_1v, uy_2v \neq \lambda$,

$$x_1 u y_1 v z_1 \in L \wedge x_3 u y_2 v z_3 \in L \Rightarrow (x_2 y_1 z_2 \in L \Leftrightarrow x_2 y_2 z_2 \in L).$$

Then in a k,l-local substitutable language, all substrings y_1 can be substituted by substrings y_2 as soon as there exist sequences in this language in which they share a common (local) context u, v of size $|u| = k$ and $|v| = l$. If a language is k,l-local substitutable then it is m,n-local substitutable for any $m \geq k$ and $n \geq l$ and the hierarchy is strict.

To simplify the proofs and the definitions, we will assume that contexts are always defined: the alphabet Σ can be extended to $\Sigma' = \Sigma \cup \{\$\}$ where \$ is a new symbol not in Σ and sequences are padded at their extremities with this new symbol (each sequence is added k symbols \$ before its beginning and l symbols \$ after its end). Using this convention, the class of substitutable languages (Clark and Eyraud [2007]) can be stated as being the class of ∞,∞ -local substitutable.

It is also possible to define the local counterpart of k,l-substitutable languages:

Definition 4 [k,l-local context substitutable language] A language L is k,l-local context substitutable if for any $x_1, y_1, z_1, x_2, y_2, z_2, x_3, z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l$, such that $uy_1v, uy_2v \neq \lambda$,

$$x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_2uy_1vz_2 \in L \Leftrightarrow x_2uy_2vz_2 \in L).$$

Again, if a language is k,l-local context substitutable then it is m,n-local context substitutable for any $m \geq k$ and $n \geq l$ and the hierarchy is strict.

Compared to k,l-local substitutable languages, the difference is that y_1 can be substituted by y_2 only in the contexts they share. This raises an important distinction between the considered contexts: the contexts used *to define* the equivalence classes (the set of substitutable y_i) and the contexts required *for the application* of these equivalence classes. In the definition above, both contexts are of the same length. To mark the difference between the two kinds of contexts and generalize the class of languages, it is possible to introduce the class of i,j-local k,l-context substitutable language, where (i,j) constrains the context size used in the local way to define the equivalence classes and (k,l) sets the minimal length condition on contexts where these substitutability classes apply.

Formally, i,j-local k,l-context substitutable languages are defined with respect to the relative lengths of the two kinds of contexts [1] by:

1. $k \leq i \wedge l \leq j$ (brave substitutability)
2. for any $x_1,y_1,z_1,x_2,y_2,z_2,x_3,z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l, a \in \Sigma^{i-k}, b \in \Sigma^{j-l}$ such that $uy_1v, uy_2v \neq \lambda$,

$$x_1auy_1vbz_1 \in L \wedge x_3auy_2vbz_3 \in L \Rightarrow (x_2uy_1vz_2 \in L \Leftrightarrow x_2uy_2vz_2 \in L)$$

3. $k \geq i \wedge l \geq j$ (cautious substitutability)
4. for any $x_1,y_1,z_1,x_2,y_2,z_2,x_3,z_3 \in \Sigma^*, c \in \Sigma^{k-i}, d \in \Sigma^{l-i}, r \in \Sigma^i, s \in \Sigma^j$ such that $ry_1s, ry_2s \neq \lambda$,

$$x_1cry_1sdz_1 \in L \wedge x_3ry_2sz_3 \in L \Rightarrow (x_2cry_1sdz_2 \in L \Leftrightarrow x_2cry_2sdz_2 \in L)$$

Let us remark that if a language is i,j-local k,l-context substitutable, then it is i,j-local a,b-context substitutable with $a \geq k$ and $b \geq l$ and that this hierarchy with respect to context size is strict. Similarly, it is m,n-local k,l-context substitutable with $m \geq i$ and $n \geq j$ and this hierarchy with respect to locality size is strict. Obviously, it is also m,n-local a,b-context substitutable for $a \geq k$ and $b \geq l$ and $m \geq i$ and $n \geq j$.

The figure 1 shows the inclusion hierarchy of i,j-local k,l-context substitutable languages with respect to the different parameters, the upper bound being the class of context-free languages.

The following table sums up the different classes of substitutable languages seen so far in the general framework of the i,j-local k,l-context substitutable language:

Language	local definition	contextual application
substitutable (Clark and Eyraud [2007])	(∞,∞)	$(0,0)$
k,l-substitutable[2] (Yoshinaka [2008])	(∞,∞)	(k,l)
k,l-local substitutable	(k,l)	$(0,0)$
k,l-local context substitutable	(k,l)	(k,l)
i,j-local k,l-context substitutable	(i,j)	(k,l)

[1] Only the two main cases are detailed here, the two other cases could be defined in a similar way

[2] Could also be named k,l-context substitutable

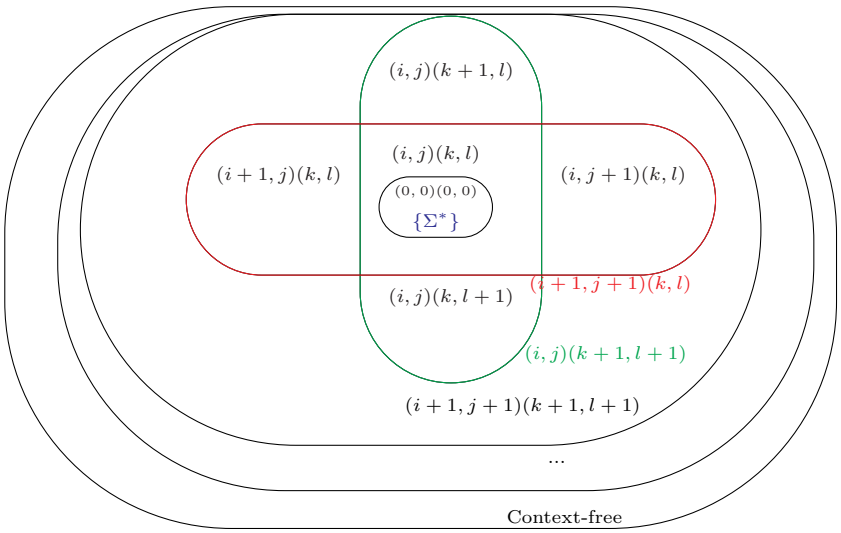


Fig. 1 Hierarchy of (i,j)-Local(k,l)-Context Substitutable Languages

Link with k-testable languages

The set of k,l-local context substitutable languages form an appealing class mixing local and contextual substitution in a symmetrical way with few parameters. Moreover, we can establish a link between this class of languages and the family of locally testable languages, an interesting subclass of regular languages learnable from positive examples only, according to theoretical and practical points of view (Garcia et al. [1990]; Garcia and Vidal [1990]; Yokomori et al. [1994]).

Definition 5 (strictly k-testable language) Let $L_k(w)$ and $R_k(w)$ be the prefix and the suffix of w of length k, respectively. Further, let $I_k(w)$ be the set of interior solid substrings of w of length k.
A language L over S is strictly k-testable if and only if there exist finite sets A,B,C such that $A,B,C \subseteq S^k$, and for all w with $|w| \geq k$, $w \in L$ if and only if $L_k(w) \in A, R_k(w) \in B, I_k(w) \subseteq C$.

In any k-testable language, the following property clearly holds:

$$\forall x_1,y_1,x_2,y_2 \in \Sigma^*, u \in \Sigma^k,\ x_1uy_1 \in L \wedge x_2uy_2 \in L \Rightarrow x_1uy_2 \in L \wedge x_2uy_1 \in L$$

The same way, the following property holds:

$$\forall x_2,y_1,x_3,y_2 \in \Sigma^*, u \in \Sigma^k, x_2uy_1 \in L \wedge x_3uy_2 \in L \Rightarrow x_2uy_2 \in L$$

The combination of these two properties gives:

$$\forall x_1,y_1,x_2,y_2,x_3 \in \Sigma^*, u \in \Sigma^k, x_1uy_1 \in L \wedge x_3uy_2 \in L \Rightarrow (x_2uy_1 \in L \Leftrightarrow x_2uy_2 \in L)$$

In terms of definition 4, the latter property is exactly the definition of $k,0$-local context substitutable languages. The k-testable languages are thus $k,0$-local context substitutable. One can proceed symmetrically and deduce the inclusion of l-testable languages in $0,l$-local context substitutable.

The class of k,l-local context substitutable languages can thus be seen as a bidirectional extension of k-testable languages, in the same way that k,l-substitutable languages are the counter part of k-reversible languages. The figure 2 displays an overview of the different classes of languages discussed so far and their inclusions.

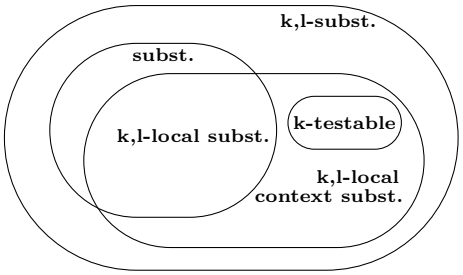


Fig. 2 Inclusion of the different substitutable language classes and k-testable languages

Before studying the inference of local substitutable languages, we present some of their properties in the next section.

2.3 Closure properties

Yoshinaka [2008] has demonstrated some properties verified by k,l-substitutable languages. We present here similar results on the locally substitutable languages. Proofs are available in Coste et al. [2012].

Proposition 1 *Locally substitutable languages are not closed under intersection with regular sets, concatenation, complement, union, Kleene closure, λ-free homomorphism, inverse homomorphism, reversal and closed under intersection and λ-free inverse homomorphism.*

While the main results are negative, except for the closure under intersection, this last proposition suggests that, as for local languages, morphic generator grammatical inference methodologies embedding expert knowledge in the sequences by renaming symbols (Garcia et al. [1987]) could be developed for learning locally substitutable languages.

3 Generalization Algorithm

3.1 Generalization by Substitutability

In Coste et al. [2012], we straightforwardly adapted to locally substitutable languages the simple learning algorithm presented in Yoshinaka [2008]. From a given sample set K and

the pair of parameters k and l, a grammar is built according to k,l-local substitutability constraints defined in table **Characterization** 1 or k,l-local context substitutability constraints defined in table **Characterization** 2. It can be noticed here that like in Yoshinaka [2008], the considered grammars have at most two nonterminals in the right-hand-side and that these algorithms do not necessarily return a grammar of the right class of languages, even if they are expected to converge towards the target language when enough examples are available.

Characterization 1 $\hat{G}_{LS}$ (k,l-local substitutability)

Data : Set of sequences K, parameters k and l
Result : Grammar $\hat{G}$
/* Nonterminal definition */
$V = \{[y] \mid xyz \in K, y \neq \lambda\} \cup \{S\}$
/* Induction of rules */
/* Initial rule */
$P_K = \{S \to [w] \mid w \in K\}$
/* Terminal rules */
$\cup \{[a] \to a \mid a \in \Sigma\}$
/* Branching rules */
$\cup \{[xy] \to [x][y] \mid [xy],[x],[y] \in V_K\}$
/* Substitutability rules */
$\cup \{[y_1] \to [y_2] \mid x_1uy_1vz_1 \in K, x_2uy_2vz_2 \in K, \mid u \mid = k, \mid v \mid = l\}$

Characterization 2 $\hat{G}_{LCS}$ (k,l-local context substitutability)

Same as **Characterization** 1 except for the last line:
/* Substitutability rules */
$\cup \{[uy_1v] \to [uy_2v] \mid x_1uy_1vz_1 \in K, x_2uy_2vz_2 \in K, \mid u \mid = k, \mid v \mid = l\}$

Given that the k,l-local substitutable languages and the k,l-local context substitutable languages are included in the class of the k,l-substitutable languages, they are learnable with the algorithm presented in Yoshinaka [2008]. It has still to be investigated, but the similarity of the representations and algorithms should allow to obtain learnability results similar to those obtained by Yoshinaka [2008] or Luque and López [2010] in our setting. We limit ourselves here to a more pragmatic perspective and focus on the difference that these new classes of languages introduce in learning by least generalization approaches.

First let us illustrate by some examples the different languages that can be learned by minimal generalization from the set of positive data $K = \{abcde, abfde, yzcji, vzmjk\}$ with respect to the chosen class of languages for small values of k and l.

- $0,0$-substitutability (Clark and Eyraud [2007])
  $N \to abXde|yzXji|vzmjkX \to c|f$
  $L = \{abcde, abfde, yzcji, vzmjk, yzfji\}$
- $1,1$-substitutability (Yoshinaka [2008])
  $N \to aXe|yzcji|vzmjkX \to bcd|bfd$
  $L = \{abcde, abfde, yzcji, vzmjk\}$

- 1,1-local substitutability
  $N \rightarrow abXde|yzXji|vzXjkX \rightarrow c|f|m$
  $L = \{abcde, abfde, yzcji, vzmjk, yzmji, vzcjk, yzfji, vzfjk, abmde\}$
- 1,1-local context substitutability
  $N \rightarrow aXe|yX_2i|vX_2kX \rightarrow bcd|bfdX_2 \rightarrow zmj|zcj$
  $L = \{abcde, abfde, yzcji, vzmjk, yzmji, vzcjk\}$

More generally, given a learning sample set K, one can establish an inclusion hierarchy between the least general generalizations produced for the different kinds of language constraints. If $L_X(K)$ denotes the least general language of class X including K, the following inclusions hold:

Proposition 2 $L_{k,l\text{-substitutable}}(K) \subseteq L_{k,l\text{-local context substitutable}}(K)$

Proof $x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_2uy_1vz_2 \in L \Leftrightarrow x_2uy_2vz_2 \in L)$
If $x_1 = x_3 \wedge z_1 = z_3$:
$x_1uy_1vz_1 \in L \wedge x_1uy_2vz_1 \in L \Rightarrow (x_2uy_1vz_2 \in L \Leftrightarrow x_2uy_2vz_2 \in L)$
So, all the words that are added to satisfy k,l-substitutability are also added for k,l-local context substitutability.

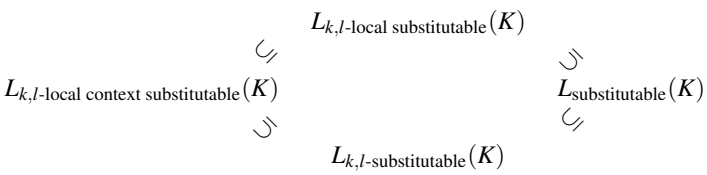
Proposition 3 $L_{substitutable}(K) \subseteq L_{k,l\text{-local substitutable}}(K)$

Proof $x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_2y_1z_2 \in L \Leftrightarrow x_2y_2z_2 \in L)$
If $x_1u = x_3u(= x_4) \wedge vz_1 = vz_3(= z_4)$:
$x_4y_1z_4 \in L \wedge x_4y_2z_4 \in L \Rightarrow (x_2y_1z_2 \in L \Leftrightarrow x_2y_2z_2 \in L)$

Proposition 4 $L_{k,l\text{-local context substitutable}}(K) \subseteq L_{k,l\text{-local substitutable}}(K)$

Proof $x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_2y_1z_2 \in L \Leftrightarrow x_2y_2z_2 \in L)$
If $x_2 = x_4u \wedge z_2 = vz_4$:
$x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_4uy_1vz_4 \in L \Leftrightarrow x_4uy_2vz_4 \in L)$

To sum up these propositions, we get the following inclusions between the different substitutable language closures of a given set of sequences K:



The defined characterizations are useful from the theoretical point of view but have little practical value. These grammars suffer from embedding too many ambiguities and redundancies. For instance, the grammars presented in the previous examples have been obtained through a tough selection of meaningful rules. From a parsing perspective or even with the simple goal of manually checking and interpreting the rules, the grammars are too large and the generation process has to be refined. The next section proposes a characterization of a more practical reduced form, avoiding unnecessary nonterminals and production rules in the learned grammar.

3.2 A Reduced Grammar

We keep on the main notations used in Yoshinaka [2012] but introduce everywhere the new parameters k and l bounding context sizes. We recall that the alphabet Σ is assumed to be extended by a new symbol and that the sequences are padded at their extremities with this new symbol, so as contexts are always defined.

An element of $\Sigma^k \times \Sigma^l$ is called a context. The empty context is $\langle \lambda, \lambda \rangle$, or equivalently $\langle \$^k, \$^l \rangle$. The composition of a string $y \in \Sigma^*$ and a context $\langle u, v \rangle \in \Sigma^k \times \Sigma^l$ is $\langle u, v \rangle \odot y = uyv \in \Sigma^*$. This notation can be naturally extended to sets.

For a language $L \subseteq \Sigma^*$, $Sub(L)$ denotes the set of strings $\{y \in \Sigma^* \mid w \odot y \in L \text{ for some context } w\}$, and $Con_{k,l}(L)$ denotes the set of contexts $\{w \in \Sigma^k \times \Sigma^l \mid w \odot y \in Sub(L) \text{ for some } y \in \Sigma^*\}$,

Clark and Eyraud [2007] have introduced for two strings y_1 and y_2 the concepts of strong substitutability (for every context w, $w \odot y_1 \in L$ iff $w \odot y_2 \in L$) and weak substitutability (for some context w, $w \odot y_1 \in L$ and $w \odot y_2 \in L$). In these definitions, k and l are assumed to be infinite.

The new classes of languages that we have introduced in section 2 are based on a simple extension of these concepts:

Definition 6 [Strong (k,l)-local substitutability] For a language L, two strings y_1 and $y_2 \in L$ are (k,l)-local substitutable iff for any $x_1, x_2, z_1, z_2 \in \Sigma^*$, $u \in \Sigma^k, v \in \Sigma^l$,

$$x_1 u y_1 v z_1 \in L \Longleftrightarrow x_2 u y_2 v z_2 \in L.$$

Definition 7 [Weak (k,l)-local substitutability] For a language L, two strings y_1 and $y_2 \in L$ are weakly (k,l)-local substitutable iff there exists $x_1, x_2, z_1, z_2 \in \Sigma^*$, $u \in \Sigma^k, v \in \Sigma^l$,

$$x_1 u y_1 v z_1 \in L \wedge x_2 u y_2 v z_2 \in L.$$

Our goal is then to improve the legibility of grammars and derivation trees with respect to substitutability. This includes the selection of meaningful nonterminals with respect to the substitutability relation and the production of derivation trees that reflect the hierarchical structure of substitutable elements. A direct consequence of reducing the number of nonterminals is to reduce grammar complexity. Overall our approach will reduce ambiguities and redundancies in grammars and in turn it will improve the efficiency of parsing. The next two subsections describe in detail the elaboration of a grammar from a weak substitutability relation computed on a learning set.

3.2.1 Nonterminals and substitutability classes

The first cause of redundancy in Characterization 1 is due to the broad definition of nonterminals:

$$V = \{[y] \mid xyz \in K, y \neq \lambda\}$$

This definition implies that all subsequences of the learning sample K are associated with different nonterminals. The number of elements in V is quadratic with respect to the size of the sequences in K. Then, the number of substitutability rules, which depends on

the number of terminal pairs linked by substitutability, grows in $O(n^4)$ if n is the size of the sequences. Thus, the reduction of the number of nonterminals is a genuine challenge.

A closer inspection of the way grammars are built easily shows that the important nonterminals are those that correspond to substitutable elements. Generalization of a sample K starts from its weakly (local) substitutable substrings and generalize K to a language that ensures strong (local) substitutability for all these substrings: each time a substitutable pair is found, its effect is propagated in a transitive way by creating the corresponding substitutability rule. A *class of substitutability* is a set of elements closed under the transitive closure of the weak substitutability relation.

A simple idea is to use a unique nonterminal for each class of substitutability and we illustrate this on a small example.

Consider the learning sample $K = \{bc, ac\}$. Words a and b are substitutable in this example. The characterization 1 results in a grammar including the following rules:

$$X_1 \to a$$

$$X_2 \to b$$

$$X_3 \to c$$

$$X_1 \to X_2$$

$$X_2 \to X_1$$

$$S \to X_1 X_3$$

In such a type of grammar, it is difficult to extract the information on substitutable structures that are coded through chains of substitutability rules. Directly using substitutable classes for nonterminals leads to more compact grammars that can be easily interpreted in terms of substitutability and improve parsing efficiency by eliminating sources of redundancy. The previous set of rules will become:

$$X \to a|b$$

$$S \to Xc$$

For instance the derivation of bc is $S \to Xc \to bc$ with this grammar instead of $S \to X1c \to X2c \to bc$ in the previous grammar.

To continue to reduce the number of nonterminals, a few notations need to be introduced. Let $\mathscr{S}$ denote the set of substitutability classes on K and $[x]$ the class of substitutability of substring x. It is possible to apply standard concatenation on substitutability classes, i.e. $[x][y] = \{uv \mid u \in [x], v \in [y]\}$. A direct consequence of the definition of a substitutability class is that

$$\forall x, y \in \Sigma^+, [x][y] \subseteq [xy]. \tag{1}$$

Reducing the number of necessary nonterminal, can be done by considering some substitutability classes (we call them composite) that can be rewritten with the help of others and can thus be safely discarded.

Composite and prime substitutability class A substitutability class $[x] \in \mathscr{S}$ is *composite* if:

$$\exists y_1 \dots y_n \in \Sigma^+,\ [x] = [y_1] \dots [y_n], n > 1. \quad (2)$$

A substitutability class $[x] \in \mathscr{S}$ is *prime* if it is not composite.

A simpler characterization of composite classes is possible, due to structuration of substitutability classes on K:

Lemma: A substitutability class $[x] \in \mathscr{S}$ is composite iff

$$\exists y, z \in \Sigma^+,\ [x] \subseteq [y][z]$$

The lemma is a simple consequence of the fact that for any decomposition of a substring $x = yz$ in K using two substitutability classes $[y]$ and $[z]$, the substitutability class $[yz]$ exists. Thus any decomposition of a string in more than two substitutability classes may be reduced to a decomposition in strictly two classes. Moreover, equation 1 ensures that it is sufficient to test an inclusion for testing the equality in equation 2. The equation 1 also entails that testing the inclusion on a decomposition into two classes is more stringent than testing the decomposition into more than two classes (ambiguity is preserved).

Our algorithm is based on the selection of nonterminals that correspond to prime substitutability classes. This is achieved in two steps for efficiency, by first working on decomposition tests where one of the elements is a string and then working iteratively on the general case where the decomposition is made of two substitutability classes.

Note that in case of local context substitutable languages the induction of substitutability classes needs to take into account the context: starting from the contexts of substrings of K, the classes will be based on the substrings within these contexts.

So far, we have presented how substitutability rules can be discarded and used to reduce the set of nonterminals. We are now considering the branching rules. Using a complete set of nonterminals in Characterization 1 had the benefit of allowing a Chomsky normal form for these branching rules. By contrast, the choice of prime substitutable classes for nonterminals requires a maximal decomposition of strings. The next subsection details the general form of such decomposition rules.

3.2.2 Branching rules

The set of nonterminals being set to the set of prime substitutability classes, we focus now on reducing the number of branching rules. Rather than having a grammar in Chomsky normal form, that is anyway not the most a parsimonious representation, we propose to build a grammar recognizing the same language as in Characterization 1 and 2 by searching for all the decompositions of the sequences in constituents of prime classes and replacing each constituents of prime substitutability class by their class (ensuring then the generalization).

The idea is to keep only all the more general rules defined on substitutability classes and to discard the ones that are subsumed by others. From a given sample set K and the pair of parameters k and l, we characterize the reduced form of grammars built according to k,l-local substitutability constraints by **Characterization** 3. The reduced form of k,l-local context substitutability grammar is defined likewise except that V_K is the set of prime local context induced substitutability classes.

An example of the gain obtained by this reduced form is presented in appendix A. The next section presents an algorithm inferring directly this reduced form, trying to avoid useless computation for practical efficiency.

Characterization 3 Reduced $\hat{G}_{LS}$ (k,l-local substitutability)

Data : Set of sequences K, parameters k and l
Result : Grammar $\hat{G}_{k,l-LS}(K) = \langle \Sigma_K, V_K, P_K, S \rangle$
/* Nonterminal and substitutability rules definition */
Σ_K =Set of symbols used in K
V_K =Set of prime local substitutability classes of K
$C = \{y | [y] \in V_K\}$(all substrings in K generated by prime classes)
/* Induction of rules */
/* Branching rules */

$$P_K = \{[y] \longrightarrow [y_1]...[y_i]...[y_{i+s}]...[y_n] \quad | \quad y_1...y_i...y_{i+s}...y_n = y \in C,\ y_i \in C,\ \nexists z \in V_K, y_i...y_{i+s} \in [z]\}$$
$$\cup \{[a] \longrightarrow a \mid a \in \Sigma_K\}$$
$$S = \{[w] \mid w \in K\}$$

3.3 ReGLiS : a practical algorithm for learning $RG_{L(C)S}$

We present in this subsection a new algorithm (see algorithm 1) that builds efficiently the reduced grammar, thus providing a first practical implementation of the generalization scheme presented in Coste et al. [2012]. It consists of four phases: detecting substitutability classes, filtering composite classes, building a canonical grammar based on the remaining classes and generalizing the grammar by a direct optimization of introduced branching rules.

We detail more the algorithm here. First, the substitutability classes are identified. This is done classically by searching for connected components (denoted $cc(SG)$ in algorithm) in a substitutability graph SG (see algorithm 2). Composite classes can then be filtered out by checking that it cannot be right or left factorized by another substitutability class in SG. Let us note here that to reduce even more the number of nonterminals to handle in the program, we don't introduce a new nonterminal for substitutability classes containing only one element since they will be useless in the following generalization step.

In second step, the canonical grammar is built. It contains one nonterminal by substitutability class kept and for each element of the substitutability class (a constituent of Σ^*) a rule is introduced allowing to rewrite the nonterminal of the class (left hand side of the rule) into the constituent (right-hand-side of the rule). At this step, one nonterminal can produce only the elements of his class. The language is composed only of the sequences from the training set.

The core of the algorithm consists in the induction of optimal branching rules. The goal is to find all the possible non redundant decompositions of the right hand sides into subwords to be replaced by their substitutability class nonterminal, performing hence the generalization.

For that, a parsing graph for each right-hand-side is built (see figure 3): nodes are positions in the right-hand-side sequence, an edge exist between two nodes if it exists a non terminal or a letter producing the substring between the two nodes, each edge is labeled by the corresponding non terminal or letter. The problem is to find all the paths, i.e sets of nodes, from the beginning of sequence to the end and find a minimum subset of these, a path (set of nodes) being said minimal if it does not exist another path strictly including in it.

Intuitively, the graph represents all the possible replacement by substitutability class nonterminals in the right-hand-side and we search for the paths of most general substitutability classes (minimal number of nodes) covering the right-hand-side, discarding thus the redundant ones that are subsumed by others, allowing to reduce the right-hand-side into smaller, more general ones.

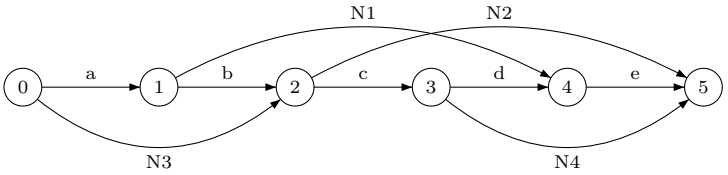


Fig. 3 find all minimum decompositions of abcde : aN1e and N3N2

Since language recognized by nonterminal is increasing, it can be required to perform again the optimization with new edges corresponding to new possible parsing of right-hand-sides. To be sure not to miss such possible optimization, the optimization is repeated until convergence but we expect that this could also be solved by a cautious ordering from smallest to biggest right-hand-sides.

Building the parsing graph, finding its minimal cover are detailed respectively in algorithm 3 and algorithm 4.

3.4 Grammars obtained on simple examples

The algorithm uses the evidence of substitutability in the learning sample as a useful induction criterion but it is not constrained to learn substitutable languages. This subsection aims at illustrating on a few standard examples its behaviour.

Example 1 Let $K = \{a, ab, abbc\}$.
This example has been first introduced by [Yoshinaka, 2008]. It shows that even if a characteristic sample (in Angluin's sense) of the language including K were given, the actual substitutable learning algorithms do not necessarily converge to this language. In the sample K, a and ab are substitutable strings. It follows that ac and abc must be in any substitutable language including K. Actually the least $(0,0)$-substitutable, or $(1,1)$-local substitutable language including K is $a\{b,c\}^*$. The grammar learned using the criteria of substitutability or $(1,1)$-local substitutability generalization is the following :

$$S \to a \mid S\ X_2 \mid S\ X_2\ c$$
$$X_2 \to b \mid X_2\ X_2\ c$$

It does not accept ac and does not generate in consequence a substitutable language.
It appears that using the $(1,0)$-local substitutability criterion the algorithm produces the correct language (production rules become $\{\ S \to a \mid S\ X_2,\ \ X_2 \to b \mid c\}$), but there is no guarantee in the general case to find parameters allowing to stay in the substitutable language classes.

When our algorithm is applied with k and l values greater than the maximal length of strings in K, it simulates the behavior of the algorithms relying on global contexts which find (k,l)-substitutable and substitutable languages. We give two examples from which context free languages may be learned.

Example 2 $K = \{ab, aabb, aaabbb, aaaabbbb, aaaaabbbbb\}$

Algorithm 1 Build reduced $\hat{G}_{LS}$ (ReGLiS) (k,l-local substitutability)

Input *Set of sequences K, Alphabet* Σ*, int k, int l*
Output *Grammar* $G = \langle \Sigma, V_K, P_K, S \rangle$
/ Nonterminal and substitutability rules definition */*
$V_K \longleftarrow \emptyset$
$P \longleftarrow \emptyset$
$SG =$ `Build_substitutability_graph`(K,k,l)
for $X_i \in cc(SG)$ **do**
  if $X_i \cap K \neq \emptyset$ **then**
    $V_K \longleftarrow V_K \cup [X_i]$
    $S \longleftarrow [X_i]$
    for $node \in X_i$ **do**
      $P \longleftarrow P \cup ([X_i] \rightarrow node)$
  / Composite classes filter */*
  else if $|X_i| > 1$ **and**
  $\nexists X \in cc(SG), \forall u \in X_i, \exists v,w \in \Sigma^+ \mid u = vw, v \in X$ **and**
  $\nexists X \in cc(SG), \forall u \in X_i, \exists v,w \in \Sigma^+ \mid u = vw, w \in X$ **then**
    $V_K \longleftarrow V_K \cup [X_i]$
    for $node \in X_i$ **do**
      $P \longleftarrow P \cup ([X_i] \rightarrow node)$
repeat
  $P_K \longleftarrow P$
  $P \longleftarrow \emptyset$
  / Induction of branching rules */*
  for $([X_i] \rightarrow u) \in P_K$ **do**
    if $|u| = 1$ **then**
      $P \longleftarrow P \cup ([X_i] \rightarrow u)$
    else
      $PG =$ `Build_parsing_graph`(u, P_K)
      for $v \in$ `Min_rhs_covers`$(PG, |u|)$ **do**
        $P \longleftarrow P \cup ([X_i] \rightarrow v)$
until $P_K = P$;
return $\langle \Sigma, V_K, P_K, S \rangle$

Algorithm 2 *Build_substitutability_graph*(Set of sequences K, int k, int l)

$V = \{u \in \Sigma^+ \mid u \in Sub(K)\}$
$E = \{(u,v) \in V \times V \mid u \neq v, \exists l \in \Sigma^k, r \in \Sigma^l, lur \in Sub(K), lvr \in Sub(K)\}$
return *Graph* $\langle V, E \rangle$

Example 3 $K = \{c, acb, aacbb, aaacbbb\}$

With k=l=100 the algorithm learn the $a^n b^n$, Dyck(a,b) and $a^n c b^n$ languages. The corresponding grammars are given in the table below. Note that these languages are context free but not local (context) substitutable unless k and l are considered infinite.

criterion	local substitutability	local context substitutability
Example 2	$S \rightarrow X_1\ X_2$ $X_1 \rightarrow a \mid X_1\ S$ $X_2 \rightarrow b \mid S\ X_2$	$S \rightarrow a\ S\ b \mid ab$
Example 3	$S \rightarrow a\ S\ b \mid c$	

Algorithm 3 *Build_parsing_graph*(Sequence u, set of rules P)

$V = \{i \in [0, |u|]\}$
$E = \emptyset$
$L = \emptyset$
for $i \in V$ **do**
  for $j \in V$ **and** $i < j$ **and** $u_{ij} \neq u$ **do**
    if $([X] \rightarrow u_{ij}) \in P$ **then**
      $E \longleftarrow E \cup (i, j)$
      $L \longleftarrow L \cup (i, j, [X])$
    else if $u_{ij} \in \Sigma$ **then**
      $E \longleftarrow E \cup (i, j)$
      $L \longleftarrow L \cup (i, j, u_{ij})$
return $Graph\langle V, E, L\rangle$

Algorithm 4 *Min_rhs_covers*(Graph $\langle V, E, L\rangle$, int n)

Table paths$[n] \longleftarrow \emptyset$
$R \longleftarrow \emptyset$
paths$[0] \longleftarrow \{(0)\}$
for $i = 1 \rightarrow n$ **do**
  / memorize minimal paths arriving in i wrt inclusion */*
  paths$[i] \longleftarrow$ *subsetmin*$(\bigcup$ *paths*$[j] \cup \{i\} \mid (j, i) \in E)$
for *path* $\in$ *paths*$[n]$ **do**
  rhs =""
  for $elt = 0 \rightarrow |path| - 1$ **do**
    $rhs = rhs.u$ with $u : (path[elt], path[elt + 1], u) \in L$
  $R = R \cup rhs$
return R

4 Application to Protein Families

4.1 Experimental Setting and Learning Process

When dealing with protein sequences, it is important to take into account the similarities between the 20 amino-acids (the alphabet Σ_a of proteins) arising from shared physico-chemical properties: some amino-acid replacements have no impact while others have an effect on the function or the structure of the protein. To take into account this knowledge directly in the learning sequences, a standard approach consists in recoding the proteins on a smaller property-based alphabet, such as the hydropathy index or the Dayhoff encoding (Yokomori et al. [1994], Peris et al. [2006] and Peris et al. [2008]). Instead of making use of these static coding schemes, this work has adopted a more specific data-driven approach. It relies on the detection of local similarities in the training set by building a partial local multiple alignment (PLMA) of the sequences. Each short strongly conserved region in the PLMA (also called block) will form one of the characters for recoding sequences.The computation of PLMA is also the first step performed in Protomata-Learner (Kerbellec [2008]). But whereas the choice of the alignment parameters is important in Protomata-Learner to tune the desired level of generalization, we as used only a basic set of parameters in this study [3].

[3] in practice, we have used the same following command line for all PLMA: paloma -i foo.fasta -o foo.plma –block-mode maxWeightFragments –transClosure -t 5 -M 7 -q 2

Figure 4 provides an overview of the automated pre-processing and post-processing steps necessary to apply the main generalization algorithm ReGLiS on protein sequences. Apart from blocks, it is necessary to take into account the existence of subsequences that appear in no block and this explains the rather complicated procedure to get a practical grammar directly able to parse protein sequences from the learned grammar. The reader not interested by these practical considerations on protein sequence parsing can safely ignore the following paragraph detailing the procedure steps.

PLMA are generated (step *a*) using Protomata-Learner, version 2.0[4], which works on a set of protein sequences S. PLMA form a set of, non-overlapping and non-crossing [5], alignment snippets on S named PLMA blocks. Each PLMA block is itself made up of a set of conserved substrings of same length from a subset of the training sequences (with only one substring per sequence involved in the block). A block implicitly aligns its substrings by their relative position, *i.e.* each position in a block corresponds to aligned characters for all its sequences.

Sequences are recoded according to the PLMA blocks that are crossed by the sequence (step *b*.1) and in parallel the information of the amino-acids composition of each block in retained in a grammar G_a (step *b*.2). This is achieved by adding for each block B of length l, a rule:

$$B \rightarrow P_1 ... P_l,$$

and for each amino-acid A at position p in the block, a rule :

$$P_p \rightarrow N_A.$$

It is in fact a little more complex since some a priori knolwledge on proteins can be introduced in this grammar. Specifically, we added a simple error model on allowed sequences, based on the standard amino acid substitution matrix Blosum62, which scores the degree to which a given amino-acid can be substituted by another one without functional loss. Thus for each pair of amino-acid pair (A,C) for which the score reflects a mutation that is more frequent than just by chance, the following rule is added:

$$N_A \rightarrow C.$$

This is a neat way to artificially enlarge the size of the training sample that is usually too small to provide this fine-level variation information on protein sequences

The rationale behind block recoding is that non conserved substrings are likely to be uninteresting for the characterization of the protein family. Then, like in Kerbellec [2008], one has to fix the treatment of substrings that belong to a sequence but not to a block. If a substring occurs between two blocks and no block is found between these two blocks in other sequences, it is considered as a gap and a rule is dedicated to its recognition:

$$Gap \rightarrow \Sigma_a Gap | \lambda.$$

Other substrings that perform a shortcut of one or several blocks are called "exceptions". In order to avoid overgeneralization, we have kept each exception $E_0 \dots E_n$ as a new block and added a dedicated rule in the grammar:

[4] available at `http://tools.genouest.org/tools/protomata/`

[5] *i.e.* consistent in the sense of Abdeddaïm and Morgenstern [2001]

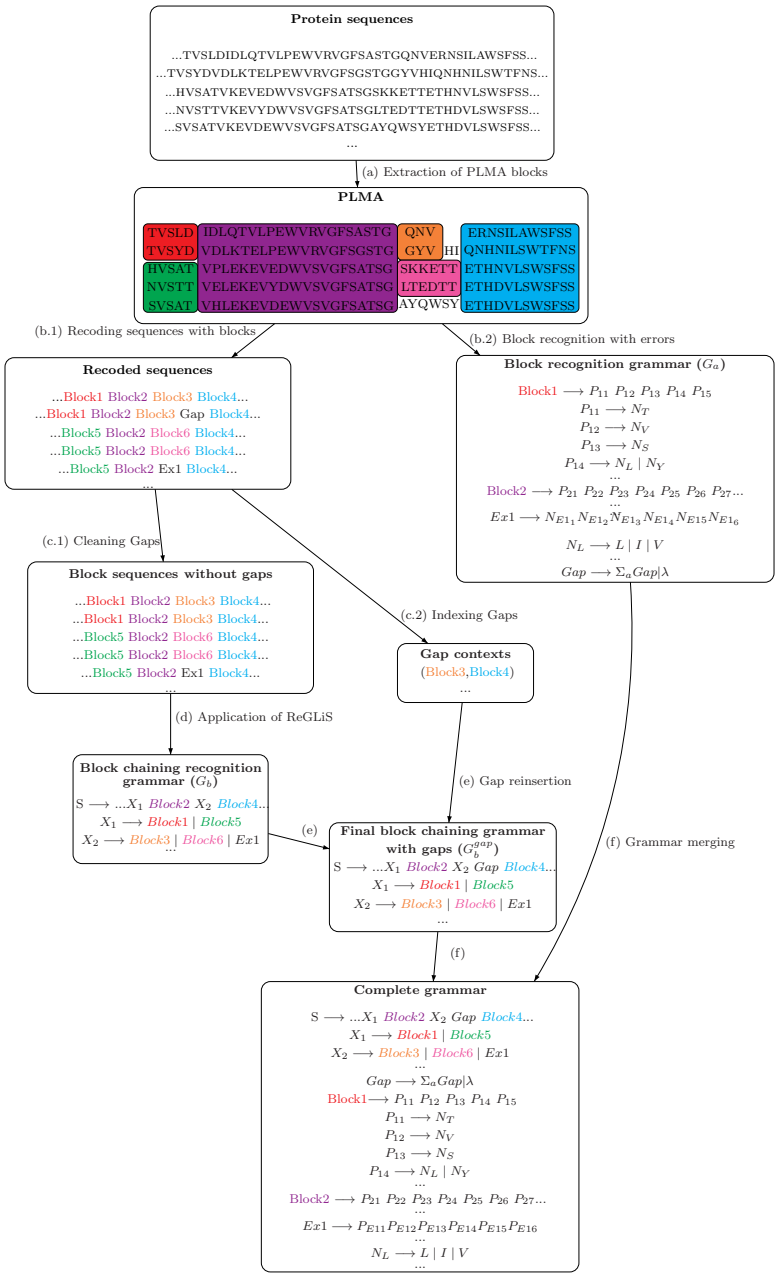


Fig. 4 Overall Diagram of CFG Learning Steps on Proteins

$$E \rightarrow N_{E_0} ... N_{E_n}.$$

Before applying ReGLiS on the recoded sequences, gaps are removed (step $c.1$) to avoid a meaningless generalization (a context with only gaps is not significant). The immediate context (left and right block) of each gap is stored (step $c.2$) for further reinsertion.

The main step d applies the algorithm on the PLMA block sequences without gaps and produces a core grammar G_b.

Gaps are then reinserted in the grammar during step e by replacing each rule $L \rightarrow ...\alpha_i\alpha_{i+1}...$ in G_b where the last character of α_i and the first character of α_{i+1} form the context of a gap, by the rule:

$$L \rightarrow ...\alpha_i \; Gap \; \alpha_{i+1}...$$

This leads to a grammar G_b^{gap} that is then merged to G_a by changing terminals blocks into non terminals (step f).

4.2 Experiments

Our method has been evaluated on a set of sequences which belong to the legume lectins protein family -Prosite entry PS00307[6](Hulo et al. [2006])-. This protein family is used as a benchmark in one of the rare study in the literature applying grammar learning on protein sequences with higher order dependencies (Dyrka and Nebel [2009]). Prosite is a database collecting protein domains and families with an associated signature that is either a regular expression or a HMM profile matching a characteristic region of the protein sequence. Prosite provides for each family the set of known true positive, false positive and false negative hits with respect to the proposed signature. Lectins are proteins that are generally found in plant seeds and play a role in binding calcium and manganese ions.

This section presents the results obtained for the various generalization criteria defined so far: substitutable, local substitutable and local context substitutable.

First tests

We started with the experimental setting presented in Coste et al. [2012]. Three sets of sequences are built from Prosite data:

- the training set already used in Dyrka and Nebel [2009], except that the entire protein sequences are processed rather than the subsequences of length 50 around the active site. This makes the issue harder but is more realistic on protein families. This set contains 22 sequences.
- the negative test set, made of the ten first sequences in the list of false positive hits provided by Prosite.
- the positive test set is also made of ten sequences from Prosite data, six in the true positiveand four in the false negative hit list.

Once a grammar has been learned using the new algorithm ReGLiS, all sequences of the test sets have been parsed using a tailored deductive parser starting from the work of Shieber et al. [1995].

[6] `http://prosite.expasy.org/PS00307`

Table 1 provides a summary of results in terms of Precision, Recall and F-measure (Precision is the ratio of true positive over all predicted positive, Recall is the ratio of true positive over all positive in the test set and F-measure is the harmonic mean of precision and recall). The last lines of the table give an overview of the behavior of the grammar learned in Dyrka and Nebel [2009] on our test sets. As stated before, these results are not fully comparable to ours since the grammar is learned on carefully chosen substrings of the sequences. In their approach each parsed sequence obtains a score, and thus precision and recall depend on a threshold value for this score. We have provided results for three characteristic values of the threshold: a maximal precision, a maximal recall, and a maximal F-measure. Of course it is hard in practice to fix the threshold and the true score-based result would be an intermediary point somewhere on the ROC curve.

The grammars learned by substitutability have all a high specificity. However, whereas global substitutability leads to a weak recall, locality clearly improves the applicability of learned grammars. These first results are consistent with our previous results on this test set.

	Precision	Recall	F-measure
Substitutable	1	0.2	0.33
Local context substitutable	1	0.6	0.75
Local substitutable	1	0.7	0.82
Stochastic CFG	1	0.1	0.18
Dyrka and Nebel [2009]	0.3	1	0.46
	0.8	0.9	0.85

Table 1 Sequence annotation by grammars obtained for the PS00307 family

Extended Results

With the new version of our learning algorithm, it becomes possible to run more demanding tests. ReGLiS took by comparison around one day for one experiment using the old version while it takes around two minutes using the new version.

We thus decided to consider the whole family PS00307 (75 sequences) and used a 10-fold cross-validation for parameter tuning. Table 2 provides a summary of results in terms of Precision, Recall and F-measure that shows the previous level of performance can be achieved when using the whole family.

We also compare our results with Prosite regular expressions built on the basis of expert-provided multiple alignments of subsequences of these families. The PS00307 family is described by the following pattern, where brackets indicate alternative letters at some positions and positions are separated by dashes:

$$[LIV]-[STAG]-V-[DEQV]-[FLI]-D-[ST].$$

The pattern spans 7 positions whereas the whole protein is about 300 letters long. In consequence, such a pattern has a good recall but weak precision (0.4) due to its generality.

In contrast, our method takes into account whole sequences. The corresponding grammar has thus a high specificity, something we expected in order to get safe predictions on new sequences. The interesting point is that the level of recall remains relatively high with respect to such a specific grammar.

For illustration purpose the grammar produced by our algorithms for 5,5-local substitutability has 161 nonterminals and 749 production rules. Obviously, there remains some range of improvement to further reduce such a grammar and generalize its rules. With respect to the original work of Yoshinaka on substitutable languages or our own work on local substitutable languages, it is however already a great improvement with respect to the size of the grammar and the efficiency of the associated parsing.

	Precision	Recall	F-measure
Substitutable	1	0.2	0.33
Local context substitutable$(4,4)$	1	0.6	0.75
Local substitutable$(5,5)$	1	0.7	0.82
Prosite	0.4	0.7	0.5

Table 2 Sequence annotation by grammars obtained for the PS00307 family with 10-fold cross-validation

Figure 5 presents the variation of recall with respect to the size of parameters k and l. It clearly shows the importance of tuning k and l values. Note that these parameters are expressed in terms of number of blocks since each learning sequence is coded at the level of blocks.

In all experiments, parameters k and l have be tuned by cross validation on the training set and part of the testing set. A careful experimentation on a larger set of sequences should consider only the training set for this purpose. Another reasonable assumption is that some a priori knowledge may exist on the length of relevant contexts in the target application. In the case of proteins, small contexts are expected since interactions concerns few amino-acids. Practical values range from 3 to 7. In all cases, one should avoid higher values since it comes down to using (global) substitutability. We have not made extensive tests on this issue and it is certainly a valuable research track.

It can be noticed that LCS produces less general grammars than LS. This is essentially due to more or less restrictive applicability of substitutability: it occurs only within a given context for LCS whereas no context is required for LS. It is thus a general behaviour that leads to a better recall for LS.

Results on the complete Dirka& Nebel's benchmark

Comparative results for the others studied families in Dyrka and Nebel [2009] are presented in table 3. The training sets for the four families are those defined and used in Dyrka and Nebel [2009]. For the substitutable and local substitutable results, we use entire sequences to learn. As in the previous experiment, k and l like have been tuned using 10-cross validation and retaining the bests values We remind that stochastic CFG in Dyrka and Nebel [2009] are learned only on part of the sequences and precision and recall depends on the chosen threshold. Again, we indicate the results for a maximum precision, a maximum threshold and a maximum F-measure. For families PS00219 and PS00063, a Prosite motif is available and we added the comparison with the Prosite scores.

Overall, it can be observed that local substitutability, using well chosen k and l, significantly improves the recall without losing accuracy unlike global substitutability. Stochastic grammars are better in terms of F-measure, but if precision is fixed to a high level, the recall

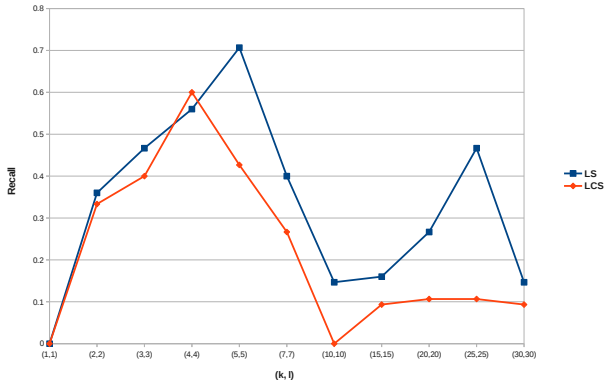


Fig. 5 Recall with respect to different values of (k,l)

obtain by ReGLiS is usually better. Moreover, our best results are comparable to Prosite, which is considered as an expert predictor.

Hence, although the method has still to be refined in order to get slightly more general grammars, local substitutability may already considered as a new promising and effective criterion for sequence generalization in all cases where an excellent precision is mandatory.

	Zinc finger			MPI phos.		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Substitutable	1	0.1	0.36	1	0.15	0.26
(3,3)-Local substitutable	1	0.2	0.33	1	0.5	0.67
(4,4)-Local substitutable	1	0.25	0.4	1	0.6	0.75
(5,5)-Local substitutable	1	0.33	0.5	**1**	**0.67**	**0.8**
(6,6)-Local substitutable	1	0.5	0.67	1	0.62	0.77
(7,7)-Local substitutable	**1**	**0.55**	**0.7**	1	0.53	0.69
Stochastic CFG	1	0.1	0.18	1	0.3	0.46
Dyrka and Nebel [2009]	0.15	1	0.26	0.5	1	0.67
	0.75	0.87	0.85	0.98	0.89	0.93
	PS00219			PS00063		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Substitutable	1	0.2	0.33	1	0.23	0.37
(3,3)-Local substitutable	**1**	**0.72**	**0.84**	1	0.58	0.73
(4,4)-Local substitutable	1	0.7	0.82	1	0.6	0.75
(5,5)-Local substitutable	1	0.68	0.8	1	0.66	0.8
(6,6)-Local substitutable	1	0.6	0.75	**1**	**0.7**	**0.82**
(7,7)-Local substitutable	1	0.5	0.67	1	0.65	0.79
Prosite	1	0.6	0.75	1	0.8	0.89
Stochastic CFG	-	-	-	1	0.05	0.1
Dyrka and Nebel [2009]	-	-	-	0.1	1	0.18
	1	1	1	0.79	0.65	0.71

Table 3 Sequence annotation by grammars obtained for the PS00307 family with 10-fold cross-validation

Full details of experiments and results are available at the url `http://www.irisa.fr/dyliss/reglis`.

5 Conclusion

This paper has been motivated by turning into practice a general and powerful concept in grammatical inference, substitutability. We have introduced for this purpose new refinements taking into account the presence of fixed size repeated contexts either during the matching phase (selecting the repeated patterns in the instances) or during the generalization phase of learning. This has led to the introduction of the new classes of locally substitutable languages extending the notion of k-testability beyond regular languages like substitutable and (k,l)-substitutable languages extends k-reversibility.

We have proposed a generic practical algorithm, ReGLiS, that is based on a simplification of the generalization scheme proposed in Coste et al. [2012] and leads for the first time to a program that can be applied in bioinformatics for the characterization of a whole protein family (e.g. more than a hundred sequences of size 1000). Although it remains necessary to test more sequences to study the robustness of our method, our results already show that it is possible to achieve a sensitivity comparable to the best methods in this domain while keeping a very high level of specificity, a major concern with respect to the costly experimental validation of predictions. Another important aspect is that we have used limited knowledge specific of the domain during learning: contrary to the work described in Dyrka and Nebel [2009], we started from the whole sequences and do not make use of any knowledge on long range dependencies. The substitutability criterion is at the core of the observed performances and appears thus to have a general interest for other application domains.

In order to gain confidence in the results, it is necessary to address traditional issues on the support of each induction step. The evidence for the chosen substitution classes should be asserted by carefully designed statistics. Our intent is to study in depth some enzyme families and to try to capture enzyme substrate specificity within the learned models. This will probably require the introduction of other types of data such as partial spatial structures. On the theoretical side, learnability of k,l-local substitutable languages or k,l-local context substitutable languages results from the polynomial identification in the limit result of Yoshinaka [2008] since the class of the k,l-substitutable languages includes the two others. However, we would like to investigate if the restriction of the class of languages and the weaker substitutability operator combined with the reduced form of the grammar proposed here would not allow us get rid of the thickness of the language parameter in the polynomial. This would led to a stronger learnability result and will contribute to establish this class as a fruitful candidate for grammatical inference from positive examples.

Acknowledgements The authors wish to acknowledge the reviewers for their insightful comments to the manuscript. This work has been partially funded by French ANR under project Idealg.

References

Abdeddaïm S, Morgenstern B (2001) Speeding up the dialign multiple alignment program by using the ‘greedy alignment of biological sequences library’ (gabios-lib). In: Gascuel O, Sagot MF (eds) Computational Biology, Lecture Notes in Computer Science, vol 2066,

Springer Berlin Heidelberg, pp 1–11, DOI 10.1007/3-540-45727-5_1, URL http://dx.doi.org/10.1007/3-540-45727-5_1

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. Journal of molecular biology pp 403–410

Bailey TL, Boden M, Buske FA, Frith M, Grant CE, Clementi L, Ren J, Li WW, Noble WS (2009) Meme suite: tools for motif discovery and searching. Nucleic Acids Research 37(suppl 2):W202–W208

Baldi P, Brunak S (2001) Bioinformatics: The Machine Learning Approach, 2nd edn. Cambridge: MIT Press

Bryant C, Fredouille D, Wilson A, Jayawickreme C, Jupe S, Topp S (2006) Pertinent background knowledge for learning protein grammars. In: Furnkranz J, Scheffer T, Spiliopoulou M (eds) Proceedings of the 17th European Conference on Machine Learning, Springer-Verlag, Berlin, no. 4212 in Lecture Notes in Artificial Intelligence, pp 54–65, URL http://www.comp.rgu.ac.uk/staff/chb/research/papers/bryant_ecml06.pdf

Chiang D, Joshi AK, Searls DB (2006) Grammatical representations of macromolecular structure. Journal of Computational Biology 13(5):1077–1100

Chomsky N (1957) Syntactic Structures. Mouton & Co.

Clark A, Eyraud R (2007) Polynomial identification in the limit of substitutable context-free languages. Journal of Machine Learning Research 8:1725–1745

Coste F, Kerbellec G (2005) A similar fragments merging approach to learn automata on proteins. In: Gama J, Camacho R, Brazdil P, Jorge A, Torgo L (eds) ECML, Springer, Lecture Notes in Computer Science, vol 3720, pp 522–529

Coste F, Garet G, Nicolas J (2012) Local Substitutability for Sequence Generalization. In: Heinz J, de la Higuera C, Oates T (eds) ICGI 2012, MIT Press, Washington, États-Unis, JMLR Workshop and Conference Proceedings, vol 21, pp 97–111

Durbin R, Eddy S, Krogh A, Mitchison G (1998) Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press

Dyrka W, Nebel JC (2009) A stochastic context free grammar based framework for analysis of protein sequences. BMC Bioinformatics 10(1):323+

Galperin MY, Koonin EV (2010) From complete genome sequence to 'complete' understanding? Trends in Biotechnology 28(8):398 – 406, DOI 10.1016/j.tibtech.2010.05.006

Garcia P, Vidal E (1990) Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. IEEE Trans Pattern Anal Mach Intell 12(9):920–925

Garcia P, Vidal E, Casacuberta F (1987) Local languages, the successor method, and a step towards a general methodology for the inference of regular grammars. Pattern Analysis and Machine Intelligence, IEEE Transactions on PAMI-9(6):841 –845, DOI 10.1109/TPAMI.1987.4767991

Garcia P, Vidal E, Oncina J (1990) Learning locally testable languages in the strict sense. In: First int. workshop on Algorithmic Learning theory, ALT'90, pp 325–338

Hulo N, Bairoch A, Bulliard V, Cerutti L, Castro ED, Langendijk-genevaux PS, Pagni M, Sigrist CJA (2006) The prosite database. Nucleic Acids Res 34:227–230

Hunter S, Jones P, Mitchell A, Apweiler R, Attwood TK, Bateman A, Bernard T, Binns D, Bork P, Burge S, de Castro E, Coggill P, Corbett M, Das U, Daugherty L, Duquenne L, Finn RD, Fraser M, Gough J, Haft D, Hulo N, Kahn D, Kelly E, Letunic I, Lonsdale D, Lopez R, Madera M, Maslen J, McAnulla C, McDowall J, McMenamin C, Mi H, Mutowo-Muellenet P, Mulder N, Natale D, Orengo C, Pesseat S, Punta M, Quinn AF, Rivoire C, Sangrador-Vegas A, Selengut JD, Sigrist CJA, Scheremetjew M, Tate J, Thimmajanarthanan M, Thomas PD, Wu CH, Yeats C, Yong S (2012) Interpro in 2011:

new developments in the family and domain prediction database. Nucleic Acids Research 40(D1):D306–D312

Jonassen I, Collins J, Higgins D (1995) Finding flexible patterns in unaligned protein sequences. Protein Science 4(8):1587–1595

Kerbellec G (2008) Apprentissage d'automates modélisant des familles de séquences protéiques. PhD thesis, Université Rennes 1

Luque FM, López GGI (2010) Pac-learning unambiguous k, l-nts $^{<=}$ languages. In: Sempere JM, García P (eds) ICGI, Springer, Lecture Notes in Computer Science, vol 6339, pp 122–134

Peris P, López D, Campos M, Sempere JM (2006) Protein motif prediction by grammatical inference. In: Sakakibara Y, Kobayashi S, Sato K, Nishino T, Tomita E (eds) ICGI, Springer, Lecture Notes in Computer Science, vol 4201, pp 175–187

Peris P, López D, Campos M (2008) Igtm: An algorithm to predict transmembrane domains and topology in proteins. BMC Bioinformatics 9

Schabes Y (1990) Mathematical and computational aspects of lexicalized grammars. PhD thesis, Philadelphia, PA, USA, aAI9101213

Searls DB (2002) The language of genes. Nature 420(6912):211–217, DOI 10.1038/nature01255, URL `http://dx.doi.org/10.1038/nature01255`

Shieber SM, Schabes Y, Pereira FCN (1995) Principles and implementation of deductive parsing. Journal of Logic Programming 24(1–2):3–36

Thompson JD, Higgins DG, Gibson TJ (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Research 22:4673–4680

Yokomori T, Ishida N, Kobayashi S (1994) Learning local languages and its application to protein α-chain identification. In: HICSS (5), pp 113–122

Yoshinaka R (2008) Identification in the limit of (k,l)-substitutable context-free languages. In: Proceedings of the 9th international colloquium conference on Grammatical inference: theoretical results and applications, ICGI'09, pp 266–279

Yoshinaka R (2012) Integration of the dual approaches in the distributional learning of context-free grammars. In: Proceedings of the 6th international conference on Language and Automata Theory and Applications, Springer-Verlag, LATA'12, pp 538–550

Zhang J, Chiodini R, Badr A, Zhang G (2011) The impact of next-generation sequencing on genomics. Journal of Genetics and Genomics 38(3):95 – 109, DOI 10.1016/j.jgg.2011.02.003

A Example of reduction of grammar on natural language

In this appendix, we show the interest of prime classes and reduce grammar form. Let a training set K as following :

K ={"Major General was here yesterday morning.",
"Major General went here yesterday morning.",
"Major General will be there tomorrow morning.",
"He will be gone tomorrow evening."}

The following grammar show the grammar obtain without chosen prime classes as non terminals and keeping the Chomsky normal form to decompose rules :

$X_{47} \rightarrow' yesterday'$
$X_{46} \rightarrow X_3X_{43}|X_{11}X_{19}|X_{23}X_{13}$
$X_{45} \rightarrow X_{20}X_2$
$X_{44} \rightarrow X_{20}X_1|X_{45}X_{29}|X_{34}X_{16}|X_9X_{15}$
$X_{43} \rightarrow X_{20}X_{19}|X_{45}X_{13}$
$X_{42} \rightarrow' tomorrow'$
$X_{41} \rightarrow X_{42}X_{15}$
$X_{40} \rightarrow X_{30}X_{46}|X_{21}X_{43}|X_{39}X_{19}|X_{25}X_{13}|X_{21}X_{34}|X_8X_{13}$
$N_0 \rightarrow X_{30}X_{24}|X_{21}X_{31}|X_{10}X_{32}|X_{36}X_{17}|X_{26}X_{15}|X_{39}X_1$
$|X_{25}X_{29}|X_{40}X_{41}|X_{21}X_{44}|X_8X_{29}|X_{40}X_{16}|X_{33}X_{15}$
$X_{29} \rightarrow X_{13}X_{16}|X_4X_{15}|X_{13}X_{41}|X_{38}X_{15}$
$X_{28} \rightarrow X_{27}X_{47}$
$X_{25} \rightarrow X_{30}X_{23}|X_{21}X_{45}|X_{39}X_2$
$X_{24} \rightarrow X_3X_{31}|X_{18}X_{32}|X_{37}X_{17}|X_{14}X_{15}|X_{11}X_1|X_{23}X_{29}|X_{46}X_{41}$
$X_{27} \rightarrow' here'$
$X_{26} \rightarrow X_{30}X_{14}|X_{21}X_{12}|X_{10}X_{28}|X_{36}X_{47}|X_{39}X_{35}|X_{25}X_{38}|X_{40}X_{42}$
$X_{21} \rightarrow' He'|X_{30}X_3$
$X_{20} \rightarrow' will'$
$X_{23} \rightarrow X_3X_{45}|X_{11}X_2$
$X_{22} \rightarrow X_6X_{27}$
$X_8 \rightarrow X_{21}X_{45}|X_{39}X_2$
$X_9 \rightarrow X_{20}X_5|X_{45}X_4|X_{34}X_{42}$
$X_2 \rightarrow' be'$
$X_3 \rightarrow' Major'$
$X_1 \rightarrow X_2X_{29}|X_{19}X_{16}|X_5X_{15}|X_{19}X_{41}|X_{35}X_{15}$
$X_6 \rightarrow' was'|'went'$
$X_4 \rightarrow X_{13}X_{42}$
$X_5 \rightarrow X_2X_4|X_{19}X_{42}$
$X_{32} \rightarrow X_{27}X_{17}|X_{28}X_{15}$
$X_{33} \rightarrow X_{21}X_9|X_{39}X_5|X_8X_4|X_{40}X_{42}$
$X_{30} \rightarrow General$
$X_{31} \rightarrow X_6X_{32}|X_{22}X_{17}|X_{12}X_{15}|X_{20}X_1|X_{45}X_{29}|X_{43}X_{41}$
$X_{36} \rightarrow X_{30}X_{37}|X_{21}X_{22}|X_{10}X_{27}$
$X_{37} \rightarrow X_3X_{22}|X_{18}X_{27}$
$X_{34} \rightarrow X_{20}X_{19}|X_{45}X_{13}$
$X_{35} \rightarrow X_2X_{38}|X_{19}X_{42}$
$X_{38} \rightarrow X_{13}X_{42}$
$X_{39} \rightarrow X_{30}X_{11}|X_{21}X_{20}$
$X_{18} \rightarrow X_3X_6$
$X_{19} \rightarrow X_2X_{13}$
$X_{10} \rightarrow X_{30}X_{18}|X_{21}X_6$
$X_{11} \rightarrow X_3X_{20}$
$X_{12} \rightarrow X_6X_{28}|X_{22}X_{47}|X_{20}X_{35}|X_{45}X_{38}|X_{43}X_{42}$
$X_{13} \rightarrow' there'|'gone'$
$X_{14} \rightarrow X_3X_{12}|X_{18}X_{28}|X_{37}X_{47}|X_{11}X_{35}|X_{23}X_{38}|X_{46}X_{42}$
$X_{15} \rightarrow' morning'|'evening'$
$X_{16} \rightarrow X_{42}X_{15}$
$X_{17} \rightarrow X_{47}X_{15}$

It is easy to see there is a more simple way to write the grammar, exhibiting substitutable classes and keep the same language. With our algorithm (extraction of prime classes and construction of reduced grammar) we obtain the following grammar :

Reduced grammar :

$S \rightarrow X_3X_4X_2$
$X_1 \rightarrow was \mid went$
$X_2 \rightarrow morning \mid evening$
$X_3 \rightarrow He \mid Major\ General$
$X_4 \rightarrow will\ be\ X_5\ tomorrow \mid X_1\ here\ yesterday$
$X_5 \rightarrow there \mid gone$