Fundamenta Informaticae 146 (2016) 403–430
DOI 10.3233/FI-2016-1393
IOS Press

Designing and Learning Substitutable Plane Graph Grammars

Rémi Eyraud*
Qarma Team
LIF Marseille, France
remi.eyraud@lif.univ-mrs.fr

Jean-Christophe Janodet
IBISC lab, University of Evry,
23 Bd de France, F-91037 Evry, France
janodet@ibisc.univ-evry.fr

Tim Oates
CoRaL lab, CSEE department
University of Maryland
Baltimore County, USA

Frédéric Papadopoulos
IBISC lab, University of Evry,
23 Bd de France, F-91037 Evry, France

Abstract. Though graph grammars have been widely investigated for 40 years, few learning results exist for them. The main reasons come from complexity issues that are inherent when graphs, and *a fortiori* graph grammars, are considered. The picture is however different if one considers *drawings* of graphs, rather than the graphs themselves. For instance, it has recently been proved that the isomorphism and pattern searching problems could be solved in polynomial time for *plane graphs*, that is, planar embeddings of planar graphs. In this paper, we introduce the *Plane Graph Grammars* (PGG) and detail how they differ to usual graph grammar formalism's while at the same time they share important properties with string context-free grammars. In particular, though exponential in the general case, we provide an appropriate restriction on languages that allows the parsing of a graph with a given PGG in polynomial time. We demonstrate that PGG are well-shaped for learning: we show how recent results on string grammars can be extended to PGG by providing a learning algorithm that identifies in the limit the class of *substitutable* plane graph languages. Our algorithm runs in polynomial time assuming the same restriction used for polynomial parsing, and the amount of data needed for convergence is comparable to the one required in the case of strings.

Keywords: Plane graphs, graph grammars, distributional learning, grammatical inference.

*Address for correspondence: 39 rue F. Joliot Curie 13453, Marseille cedex 13, France

1. Introduction

Graph Grammars have been defined and studied for four decades from a language-theoretical standpoint (see [33] for an overview), but the learning of these formalism's is known to be intricate and has hardly been investigated in the literature yet. Most contributions concern heuristics tailored for graphs involved in restricted application domains. This is the case of both most famous algorithms, *Subdue* [6] and *FFSM* [23], and their extensions [31, 25, 28].

On the theoretical side, it appears that learnability results are even rarer and often provide us with preliminary results, rather than effective learning procedures. For instance, E. Jeltsch and H.-K. Kreowski [24] give an algorithm that generates the set of grammars consistent with a given set of graphs. R. Brijder and H. Blockheel [3] investigate the inference of a grammar consisting of a single production rule, given a graph and a distinguished pattern with many occurrences. Few necessary conditions for the learning of graph grammars have also been established under unrestricted Gold's paradigm [7]. Besides, note that new learnability results are anticipated in the framework of recognizable series of (hyper-)graphs [1]. The situation is a bit different in other branches of Machine Learning: several techniques have been proposed to tackle classification problems over graphs in the scope of social network analysis, biological network analysis or image analysis [20]. However, they generally hide the complexity of the graph structures into abstract numerical structures such as graph kernels [26].

There exist many reasons for this, ranging from the profusion of incomparable graph grammar formalism's to the hardness of the model itself. Concerning the latter, many basic problems, such as the search for a subgraph in a graph, and thus the possibility to parse a graph with a grammar, are generally $\mathcal{NP}$-complete [17]. Nevertheless, the main reason for this absence of positive learning results is probably that no kind of graph grammars was designed with the aim of learning. Indeed, two main characteristics have to be shared by the representation formalism if one wants to use it as a model for inference. On the one hand, the graph isomorphism problem needs to be efficiently solvable: the key point to learn graph grammars is to extract knowledge about the structure of the graphs in the learning sample. In addition, for most applications, it suffices to view isomorphic graphs as the same object, so the choice of vertices names is of less importance. If a low priority is given to the understanding of the structure, general machine learning methods can be applied to graph data with great success [36].

The second important characteristic is that the grammar formalism has to capture properties that are observable in a set of data. The most obvious kind of observable properties concerns substructures, *e.g.* the frequency or the relative positions of the subgraphs in the graphs of the sample. But standard graph grammars of different types are not designed for the inference from the observation of properties of sub-structures. For instance, in the framework of Hyperedge-Replacement Grammars (HRG) [10], we can compute from a sample the *set* of external nodes for each sub-hypergraph. However, this set of vertices must be transformed into a *sequence* during the inference stage of a HRG, as this sequence is necessary for the embedding mechanism that is used when a rule is applied to rewrite a hypergraph. In other words, an essential piece of information for the inference of a HRG is not observable in the sample.

From a general standpoint, one way to tackle the difficulties raised by the learning of generative devices (grammars) consists in restricting the class of languages into consideration. That is, the successful approach in Grammatical Inference is often to determine features that are learnable, which usually correspond to observable properties in any set of examples, and then to focus only on the languages that share these characteristics.

Hence, in the case of graph languages, we should first determine which kind of graphs are likely to be learnable, and then choose the kind of grammars to use. For reasons that will be developed in this paper, a promising candidate is the class of *plane graphs*, that is, planar graphs embedded in the plane (see Figure 1 for an example). Note that a *planar* graph has a set X of vertices and a set E of edges as usual, but as soon as this graph is embedded in the plane, it also has a set F of faces, one of them - called the outer face - is different of the others as it corresponds to the part of the plane where the graph is not bounded (it is thus infinite). A planar graph may have several incomparable drawings, so we define a plane graph by fixing the embedding. More formally, a plane graph stands for an isotopy class of planar embeddings for a given planar graph [16]. A plane graph is thus a planar graph that is embedded in the plane without edge-crossing and up to continuous deformations. Given a planar embedding of a planar graph, Fáry [15] proved that it is always possible to move the vertices, within the same isotopy class, so that the edges are drawn with straight-line segments. We shall use such straight-line drawings in the following.

Now, as no common graph grammar formalism captures the specificities of such plane graphs, we choose not to use existing general graph-grammar formalism's, but propose in this paper a new type of grammar, called the *Plane Graph Grammars (PGG)*. These grammars can be seen as face-replacement grammars, thus constitute an interesting alternative to standard node-replacement or hyperedge-replacement grammars. Indeed, their rules replace one face by a new plane graph, which is sewn in the mother graph using a syntactic gluing law. We provide theoretical results about these grammars regarding the possibility to efficiently parse a plane graph, and compare them with other types of graph grammars. More precisely, the restriction which allows to parse a graph in polynomial time forces the number of subgraphs whose outer face contains a given number of nodes to grow polynomially with the size of the graphs in the language represented.

We then investigate the learning of PGG, and prove that one can get formal learnability results in this setting. We believe that this is quite an interesting improvement *w.r.t.* the state of the art. Concerning the difficulties, notice that when one is trying to learn from graphs, negative data are usually not available. We know since the work by E. M. Gold [19] that it is not possible to identify in the limit any superfinite class[1] of languages from positive data, and thus need to restrict ourselves to a subclass of plane graph languages. The recent successes of distributional learning for string grammars [4] and tree grammars [27] motivate us to define an analogue of substitutable context-free languages [5] for plane graph languages.

Notice that a preliminary version of this paper appeared in the Proceedings of ICGI'12 [14], but the present paper is substantially different: in [14] we tackled the problem of learning *Binary* Plane Graph grammars, a type of PGG where the production rules had binary right hand-sides and were thus similar to Chomsky normal forms. Moreover, we omitted the study of their properties. In this paper, we consider general PGG, show that they have the same language expressiveness than binary PGG, improve the definition of the rewriting mechanism and propose new conditions for the parsing problem to be achievable in polynomial time. We also improve the learning algorithm and thus establish a more general learnability result for substitutable plane-graph languages.

Preliminaries about plane graphs are given in Section 2. The definition of Plane-Graph Grammars as well as the rewriting mechanism is detailed in Section 3, where we also compare them with node-replacement grammars and hyperedge-replacement grammars. We prove formal properties of these grammars in Section 4, in the scope of the parsing problem. Next Section 5 is devoted to the

[1] A class is superfinite if it contains all possible finite languages and at least one infinite language.

learning of PGG, and is thus the core of the paper: the substitutability property is first adapted, then the learning procedure is described, and a learnability result is finally proved for substitutable plane graph languages. We conclude the paper with a discussion in Section 6.

2. On plane graphs

We have introduced the plane graphs using the notion of embeddings, *i.e.*, functions that map vertices to points, and edges to curves. However, this mathematical approach is quite unsuitable for designing algorithms. As the set of faces is the corner stone to describe plane graphs, we introduce *plane graph systems* [22] below, which allow us to describe any *connected* plane graph through its faces.

Let $X \subset \mathbb{N}$ be the alphabet of vertices. Let X^* be the set of all *strings* over X, and ϵ the empty string. Given a string $x = a_1 \dots a_n$, we denote $|x| = n$ its length and x^R the reverse string of x, that is to say $x^R = a_n \dots a_1$. We also define $\texttt{first}(x)$ to be a_1. A *circular string* is intuitively a string in which the last symbol is followed by the first; more precisely, there is neither a first nor a last symbol but a mapping associating to each symbol the next one. We denote a circular string by $[u]$, with the convention that if u and v are two strings, then $[uv] = [vu]$. The set of all circular strings over X is denoted by $X^\top$. We set $[x]^R = [x^R]$. Finally, given an alphabet X, we can extend any function $\phi : X \to X$ to strings: $\forall x = a_1 \dots a_n \in X^*, \hat{\phi}(x) = \phi(a_1) \dots \phi(a_n)$, to circular strings: $\hat{\phi}([x]) = [\hat{\phi}(x)]$, to sets of strings: $\hat{\phi}(S) = \{\hat{\phi}(x) : x \in S\}$, and to sets of pairs of strings: $\hat{\phi}(C) = \{\{\hat{\phi}(x), \hat{\phi}(x')\} : \{x, x'\} \in C\}$.

Now consider the plane graph of Figure 1. The outer face is f_1 and bounded (inner) faces are f_2 and f_3. Each face has only one boundary since the graph is connected. Such a boundary can be described by a circular string of vertices in which two consecutive vertices and the last and first vertices are linked by an edge. Conventionally, we follow a boundary by leaving it to the right. In other words, the bounded face is on the left of the walk. Hence, the boundary of face f_3 is $[53634]$, or equivalently $[63453]$, by circular permutation.

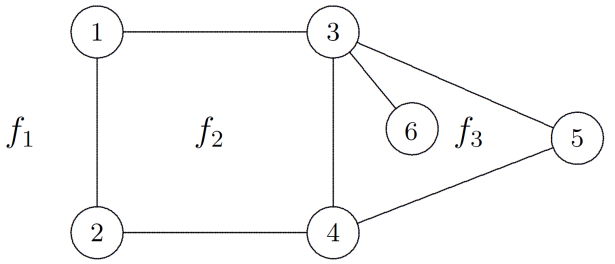

Figure 1. A plane graph with 3 faces.

We now introduce the following description system for connected plane graphs:

Definition 2.1. (Plane Graph System [22])
A *plane graph system* (PGS for short) is a tuple $S = \langle X, E, F, o, \mathcal{D} \rangle$ such that

1. $\langle X, E \rangle$ is a connected simple graph [18],
2. F is a finite nonempty set of symbols called the *faces*,

3. $o \in F$ is a special face called the *outer face* and

4. $\mathcal{D} : F \to X^\top$ is a function, called the *boundary descriptor*, that maps any face to its boundary.

For sake of readability, we shall make no distinction between a face f and the description of its boundary $\mathcal{D}(f)$. In consequence, function $\mathcal{D}$ will be kept implicit most of the time and we simply denote by $\langle X, E, F, o\rangle$ the plane graph system S.

Note that every plane graph can be described with a plane graph system (see below for an example), but the converse does not hold in general. We thus introduce further conditions below:

Definition 2.2. (Valid PGS)
A PGS $S = \langle X, E, F, o, \mathcal{D}\rangle$ is *valid* if:

1. For all $f \in F$ and $x, y \in X$ and $u \in X^*$, if $\mathcal{D}(f) = [xyu]$ then $\{x, y\} \in E$;

2. For all $e = \{x, y\} \in E$, there exist a unique face $f \in F$ such that $\mathcal{D}(f) = [xyu]$, and an unique face $f' \in F$ such that $\mathcal{D}(f') = [yxv]$, for some $u, v \in X^*$;

3. For all $f \in F$ and $x, y, z, z' \in X$ and $u, v \in X^*$, if $\mathcal{D}(f) = [xyzuxyz'v]$ or $\mathcal{D}(f) = [zxyuz'xyv]$ then $z = z'$;

4. Euler's formula holds, that is, $|X| - |E| + |F| = 2$.

Validity allows any PGS to denote a plane graph. Indeed, the two first conditions ensures that the sets of vertices and edges and faces are well-defined; the third condition eliminates all the circular strings that cannot describe the boundary of any face in any planar embedding of any graph; the fourth condition implies that the surface on which the graph is drawn has a null genus, and is thus is a plane (thanks to the infinite face o).

This property is established by the following theorem. As the problem is quite far from the core of this paper, the proof is given in Appendix A. Notice that all the PGS we shall consider in this paper will be valid, so this property will be kept implicit, most of the time:

Theorem 2.3. Any valid PGS describes an unique plane graph (up to continuous deformation).

Let $\mathbb{G}$ be the set of all the (valid) plane graph systems. The *size* of a PGS $G = \langle X, E, F, o\rangle$ is $|G| = \sum_{f\in F} |f|$. Given any edge e, we denote by $\text{faces}(e)$ the set of faces incident to edge e. Notice that $\text{faces}(e)$ can contain either 1 or 2 faces (only one in the case of a *pendant edge*). We use $\texttt{nodes}(f)$ and $\texttt{edges}(f)$ for the set of vertices and edges along the boundary of face f, respectively.

For instance, consider the plane graph of Figure 1. The corresponding PGS is $S = \langle X, E, F, o\rangle$ with $X = \{1, 2, 3, 4, 5, 6\}$, $E = \{\{1, 2\}, \{1, 3\}, \{2, 4\}, \{3, 4\}, \{3, 5\}, \{3, 6\}, \{4, 5\}\}$, $F = \{f_1, f_2, f_3\}$, $o = f_1$ and $f_1 = [13542]$, $f_2 = [1243]$, $f_3 = [34536]$. Moreover, we have $\text{faces}(\{3, 4\}) = \{f_2, f_3\}$, $\text{faces}(\{3, 6\}) = \{f_3\}$, $\texttt{nodes}(f_3) = \{3, 4, 5, 6\}$ and $\texttt{edges}(f_3) = \{\{3, 4\}, \{4, 5\}, \{5, 3\}, \{3, 6\}\}$.

Definition 2.4. (Set of connected faces)
Let $S = \langle X, E, F, o\rangle$ be a PGS. Two distinct faces $f, f' \in F$ are *adjacent* if $\exists e \in E : \text{faces}(e) = \{f, f'\}$. The faces of a subset $K \subseteq F$ are *connected* if $\forall f, f' \in K$, a sequence $f = f_0, f_1, \ldots f_n = f'$ of faces in K exists such that $\forall i \in \{0, 1, \ldots, n-1\}$, f_i and f_{i+1} are adjacent.

Given a subset $K \subseteq F$ of connected inner faces, we denote by $\texttt{outer}(K)$ the (boundary of the) outer face of that set. For instance, on the PGS of Figure 1, $\texttt{outer}(\{f_2, f_3\}) = f_1$ and $\texttt{outer}(\{f_3\}) = [354]$. Notice that $\texttt{outer}(K)$ can be computed in polynomial time using the *normalization* procedure introduced in [22].

Let us finally introduce the notion of subgraph that we will use throughout the rest of this paper:

Definition 2.5. (Pattern)
Given a PGS $G = \langle X, E, F, o\rangle$ and a set $F' \subseteq F \setminus \{o\}$ of connected faces, the PGS $G' = \langle \texttt{nodes}(F'), \texttt{edges}(F'), F' \cup \{\texttt{outer}(F')\}, \texttt{outer}(F')\rangle$ is called a *pattern* of G. By extension, any renaming of the vertices and edges and faces of G' will also be called a *pattern* of G.

For instance, the PGS G of Figure 1 has 3 patterns: $\langle\{1,2,3,4\}$,$\{\{1,2\},\ldots\}$,$\{[1243],[1342]\},[1342]\rangle$, $\langle\{3,4,5,6\},\{\{3,4\},\ldots\},\{[34536],[354]\},[354]\rangle$, and G itself.

More general notions of subgraphs exist (based on subsets of vertices and edges, independently on faces), but they often induce intractable problems. In particular, it was shown in [22] that searching for a pattern in a PGS is tractable in polynomial time, whereas searching for general subgraphs in *planar* graphs is a $\mathcal{NP}$-complete problem. In other words, the *drawing* of a planar graph is much more informative than the planar graph alone. In the following, term *subgraph* will exclusively mean *pattern*.

2.1. Concatenation

The concatenation of two PGS is a basic operation that allows one to glue together two distinct plane graphs using their outer face.

Definition 2.6. Let $G_1 = \langle X_1, E_1, F_1, o_1\rangle$ and $G_2 = \langle X_2, E_2, F_2, o_2\rangle$ be two PGS, and $\phi : \texttt{nodes}(o_1) \to \texttt{nodes}(o_2)$ a partial bijective function. We say that G_1 and G_2 are *concatenable* following ϕ if

- $F_1 \cap F_2 = \emptyset$,
- $(X_1 \setminus \texttt{nodes}(o_1)) \cap (X_2 \setminus \texttt{nodes}(o_2)) = \emptyset$, and
- $\exists k > 1$: $\hat{\phi}(X_1) \cap X_2 = \{\phi(a_1), \ldots, \phi(a_k)\}$ and $o_1 = [a_1 \ldots a_k y]$ and $o_2 = [\phi(a_k) \ldots \phi(a_1) z]$ with $y \in (X_1 \setminus X_2)^*$ and $z \in (X_2 \setminus X_1)^*$ and $|yz| \geq 1$.

Intuitively, two PGS are concatenable following ϕ if they can be glued together by merging pairwise nodes of their outer face following ϕ. This requires that they can only share nodes of their outer face, and that consecutive edges of one outer face correspond to reverse consecutive edges in (the image by ϕ of) the other outer face. In consequence, the gluing stage does not modify the inner faces.

Definition 2.7. (Concatenation)
Let $G_1 = \langle X_1, E_1, F_1, o_1\rangle$ and $G_2 = \langle X_2, E_2, F_2, o_2\rangle$ be two PGS concatenable following function ϕ. The *concatenation* of G_1 and G_2 following ϕ, written $G_1 \diamond_\phi G_2$, is the PGS $G = \langle X_1 \cup X_2 \setminus \{a_1, \ldots, a_k\}, E_1 \cup E_2 \setminus \{\{a_i, a_{i+1}\} : 1 \leq i < k\}, \hat{\phi}(F_1 \setminus \{o_1\}) \cup (F_2 \setminus \{o_2\}) \cup \{o\}, o\rangle$ with $o = [\phi(a_k) y \phi(a_1) z]$.

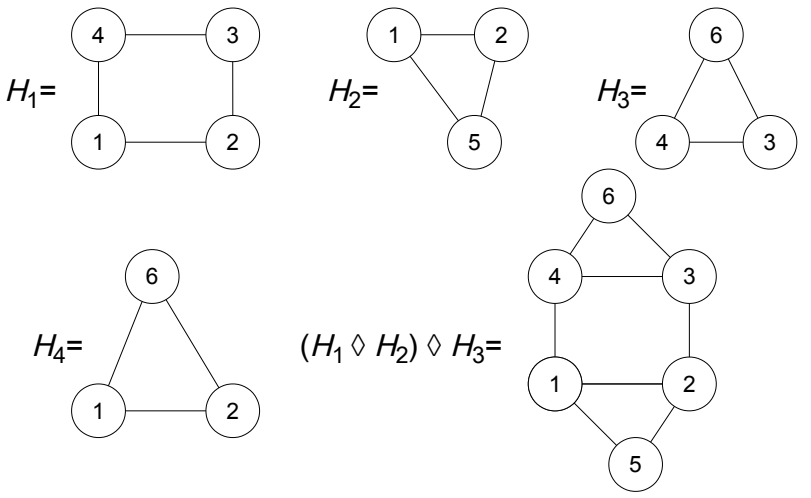

Figure 2. Example of concatenation: H_1 and H_2 are concatenable following the identity function and so is H_2 and H_4. The same occurs for $H_1 \diamond H_2$ and H_3. H_2 and H_3 are not concatenable following the identity function, which is also the case of H_1 and H_4.

If the function ϕ is the identity, we shall write $G_1 \diamond G_2$ instead of $G_1 \diamond_{id.} G_2$.

Concatenation is well-defined, that is, if G_1 and G_2 are valid concatenable PGS, then $G_1 \diamond_\phi G_2$ is necessarily a valid PGS. Indeed, the conditions on the external faces ensure that no new face is created by concatenation, but the outer one which is modified; moreover, the concatenability property ensures us that the technical properties on the boundaries are satisfied; finally, Euler's relation holds, since $(|X_1| + |X_2| - k) - (|E_1| + |E_2| - (k-1)) + (|F_1| - 1 + |F_2| - 1 + 1) = 2$.

Examples of concatenable and non-concatenable PGS are given in Figure 2. For instance H_1 and H_2 are concatenable following the identity function but it is not the case of H_1 and H_4 since the third requirement of Definition 2.6 is not met. These examples also show that the associativity of graph concatenation is not ensured: $(H_1 \diamond H_2)$ and H_3 are concatenable following the identity function, but H_2 and H_3 are not concatenable following the identity and thus $H_1 \diamond (H_2 \diamond H_3)$ is not defined. In the absence of brackets in a sequence of concatenations, we will consider the left to right organization: $H_1 \diamond H_2 \diamond H_3$ is to be read as $(H_1 \diamond H_2) \diamond H_3$.

2.2. Plane isomorphism

We finally need a way to compare two PGS:

Definition 2.8. (Plane isomorphism [22])
Let $G_1 = \langle X_1, E_1, F_1, o_1 \rangle$ and $G_2 = \langle X_2, E_2, F_2, o_2 \rangle$ be two PGS. We say that G_1 and G_2 are *plane-isomorphic*, written $G_1 \cong_p G_2$, if there exist a 1-to-1 mapping $\chi : X_1 \to X_2$ over the vertices and a 1-to-1 mapping $\xi : F_1 \to F_2$ over the faces such that (1) the outer face is preserved: $\xi(o_1) = o_2$ and (2) the boundaries are preserved: $\forall f_1 \in F_1, f_2 \in F_2$, if $\xi(f_1) = f_2$ and $f_1 = [a_1 \dots a_n]$ then $f_2 = [\chi(a_1) \dots \chi(a_n)]$.

Plane-isomorphism is decidable in $\mathcal{O}(|E_1| \cdot |E_2|)$ time [22]. The key property to prove this result is that, given a PGS G and an edge e, we can define an ordering over all the other edges which is

unique and computable in linear time thanks to a traversal of the PGS. So the isomorphism algorithm consists in finding two edges e_1 in G_1 and e_2 in G_2, generating the ordering of all the edges in both PGS and using them to generate possible isomorphism functions. A similar strategy is used to check whether a given PGS is a pattern of any other PGS, that is, searching for patterns is also tractable in polynomial time [22].

We can now define the notion of plane graph language:

Definition 2.9. A set L of PGS is a *plane graph language* if it is closed under plane-isomorphism: for all $G_1, G_2 \in \mathbb{G}$ such that $G_1 \cong_p G_2$, we have $G_1 \in L \iff G_2 \in L$.

The following technical but crucial lemma deals with the link between concatenation and plane-isomorphism. Informally, if two graphs are concatanable then graphs that are plane-isomorphic to them are also concatenable (using a different function). Moreover, the concatenated graphs are plane-isomorphic.

Lemma 2.10. Let G_1, G_1', G_2 and G_2' be four PGS such that $G_1 \cong_p G_1'$ and $G_2 \cong_p G_2'$. Suppose that G_1 and G_2 are concatenable following a function ϕ. Then there exist a PGS G_1'' such that $G_1'' \cong_p G_1'$, and a function ϕ' such that G_1'' and G_2' are concatenable following ϕ' and $G_1 \diamond_\phi G_2 \cong_p G_1'' \diamond_{\phi'} G_2'$.

Proof:
As $G_1 \cong_p G_1'$ (resp. $G_2 \cong_p G_2'$), there exist two 1-to-1 mapping $\chi_1 : X_{G_1} \to X_{G_1'}$ (resp. $\chi_2 : X_{G_2} \to X_{G_2'}$) and $\xi_1 : F_1 \to F_1'$ (resp. $\xi_2 : F_2 \to F_2'$) fulfilling the conditions of plane-isomorphism. As G_1 and G_2 are concatenable following ϕ, there exist vertices $a_1, \ldots, a_k$ and (possibly empty) sequences of nodes y and z such that $o_{G_1} = [a_1 \ldots a_k y]$ and $o_{G_2} = [\phi(a_k) \ldots \phi(a_1) z]$. Moreover we have $o_{G_1'} = [\chi_1(a_1) \ldots \chi_1(a_k)\hat{\chi_1}(y)]$ and $o_{G_2'} = [\chi_2 \circ \phi(a_k) \ldots \chi_2 \circ \phi(a_1)\hat{\chi_2}(z)]$.

Let $\phi' = \chi_2 \circ \phi \circ \chi_1^{-1}$. Let us denote $a_i' = \chi_1(a_i)$ for all $1 \leq i \leq k$. Clearly we have $o_{G_1} = [a_1' \ldots a_k' \hat{\chi_1}(y)]$ and $o_{G_2} = [\phi'(a_k') \ldots \phi'(a_1')\hat{\chi_2}(z)]$. Moreover, $|\hat{\chi_1}(y)\hat{\chi_2}(z)| = |yz| \geq 1$. Finally, if $X_{G_1'} \setminus \texttt{nodes}(o_{G_1'}) \cap X_{G_2'} \setminus \texttt{nodes}(o_{G_2'}) \neq \emptyset$, we can rename the inner nodes of G_1' to create a new PGS G_1'' such that none of its inner nodes share the same name with a node of G_2', and $G_1'' \cong_p G_1'$. Therefore $G_1 \cong_p G_1''$ and G_1'' and G_2' are concatenable following ϕ'.

We need to show that $G_1 \diamond_\phi G_2 \cong_p G_1'' \diamond_{\phi'} G_2'$. Let χ_1' and ξ_1' the functions defining the plane-isomorphism between G_1 and G_1''. Let $\chi : X_{G_1 \diamond_\phi G_2} \to X_{G_1'' \diamond_{\phi'} G_2'}$ be the function such that $\chi(v) = \chi_1'(v)$ if $v \in X_{G_1}$ and $\chi(v) = \chi_2(v)$ if $v \in X_{G_2}$. As χ_1' and χ_2 are 1-to-1 mappings over distinct domains, so is χ. Now let $\xi : F_{G_1 \diamond_\phi G_2} \to F_{G_1' \diamond_{\phi'} G_2'}$ be the function such that $\xi(f) = \xi_1'(f)$ if $f \in F_{G_1}$, and $\xi(f) = \xi_2(f)$ if $f \in F_{G_2}$, and $\xi(o_{G_1 \diamond_\phi G_2}) = o_{G_1'' \diamond_{\phi'} G_2'}$. As ξ_1' and ξ_2 are 1-to-1 mapping and $F_{G_1} \cap F_{G_2} = \emptyset$, we deduce that ξ is a 1-to-1 mapping. In addition, as (χ_1', ξ_1') and (χ_2, ξ_2) preserve the boundaries of the faces, it is also the case of (χ, ξ) by construction. □

3. The grammars for plane graph languages

The new graph grammar formalism which is introduced below is based on the following idea: a grammar consists of rules that explain how to replace some face by a pattern that can be made of several faces but whose outer face is the same than the one being replaced. Two types of rules must be given: the *plane graph productions* which allow one to get infinite and rich languages of plane graphs, and the *lexical* rules which allow one to stop the generation process.

We first need to introduce non-terminal symbols, which are essentially the names that we give to the faces that are replaced by the rules of the grammar. As in the framework of the tree grammars [12], we suppose that we have a ranked alphabet for soundness and type-checking reasons.

Definition 3.1. (Non-terminal symbol)
A *non-terminal symbol* is a couple (N, r) where N is a *name* and $r \geq 2$ is the *rank* of the non-terminal. For sake of simplicity, we assume that for all non-terminal symbols (N_1, r_1) and (N_2, r_2), if $N_1 = N_2$, then $r_1 = r_2$. We shall thus unambiguously write $\texttt{rank}(N)$ for the rank r of non-terminal (N, r).

In a grammar rule, a non-terminal symbol may be attached to some face whose number of nodes is equal to the rank of the non-terminal. In order to describe these faces, we introduce the following non-terminal gadgets:

Definition 3.2. (Non-terminal gadget)

- A *non-terminal gadget* is a couple denoted N_x where $x = a_1 \dots a_r$ is a sequence of r pairwise-distinct vertices and (N, r) is a non-terminal symbol.
- Any non-terminal gadget N_x with $x = a_1 \dots a_r$ implicitly denote a (valid) PGS with a unique inner face which is bounded by string x. This PGS is formally defined as follows: $[\![N_x]\!] = \langle X, E, \{[x], [x]^R\}, [x]^R\rangle$ with $X = \{a_1, \dots a_r\}$ and $E = \{\{a_i, a_{i+1}\} : 1 \leq i < r\} \cup \{(a_r, a_1)\}$.
- We say that two gadgets $N^1_{x_1}$ and $N^2_{x_2}$ are *concatenable* if PGS $[\![N^1_{x_1}]\!]$ and $[\![N^2_{x_2}]\!]$ are concatenable (following the identity). In this case, we denote by $[\![N^1_{x_1} N^2_{x_2}]\!]$ the PGS $[\![N^1_{x_1}]\!] \diamond [\![N^2_{x_2}]\!]$.

An example of non-terminal gadget is given in Figure 3: the left-hand PGS corresponds to gadget A_{1234}, and is more precisely the graphical representation of the PGS $[\![A_{1234}]\!]$. Notice that we attach the non-terminal symbol (here A) with a dashed line to the vertex that appears at the head of the string that defines the non-terminal gadget (here vertex 1, that is, the first vertex in the string of gadget A_{1234}). With this convention, one can equivalently use textual and graphical description of

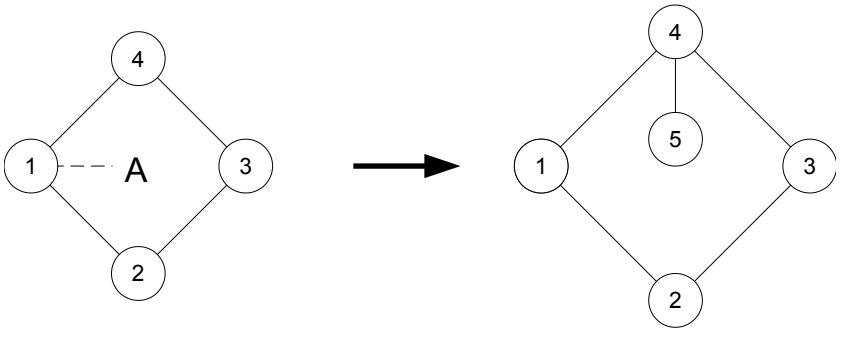

Figure 3. A graphical representation of the plane graph lexical rule $A_{1234} \to \langle\{1, 2, 3, 4, 5\}, \{(1, 2), (2, 3), (3, 4), (4, 5)\}, \{[123454], [1432]\}, [1432]\rangle$. Notice that it is possible to simplify the textual notation of a lexical rule: the PGS on the right handside is fully determined by its inner face. In this example we shall write $A_{1234} \to [123454]$. In order to get a complete drawing (which would allow one to deduce the textual description of the rule), a dashed line attaches the non-terminal symbol to the vertex that appears at the head of the string describing the non-terminal gadget of the left-hand side. Note that in the rest of this manuscript, the textual description of the rule will generally be sufficient, thus no equivalent graphical representation will be given in most cases.

non-terminal gadgets. Many other examples of non-terminal gadgets appear in both the left-hand side and the right-hand side of Figure 4. The reader may check that gadgets A_{2541} and N_{2645} on the one hand, and N_{2645} and B_{2346} on the other hand are respectively concatenable.

We now define the two types of grammar rules used in our grammars. The former denotes the terminal rules while the latter denotes the recursive rules.

Definition 3.3. (Plane graph lexical rule)
A *plane graph lexical rule* is a pair (N_x, G_*), written $N_x \to G*$, where (1) N_x is a non-terminal gadget and (2) $G_* = \langle X_*, E_*, F_*, o_* \rangle$ is a valid PGS such that $|F_*| = 2$ and $o_* = [x^R]$.

A graphical representation of plane graph lexical rule is shown in Figure 3. Notice that in any rule $N_x \to G_*$, the outer face of G_* must be $[x^R]$, which is exactly the same as that of $[\![N_x]\!]$, but the boundary of the (unique) inner face is not necessarily $[x]$: there may be pendant edges.

Definition 3.4. (Plane graph production)
A *plane graph production* is a tuple $(N^0_{x_0}, \dots N^k_{x_k})$, written $N^0_{x_0} \to N^1_{x_1} \dots N^k_{x_k}$, where

1. $N^0_{x_0}, \dots, N^k_{x_k}$ are $k+1$ non-terminal gadgets with $k \geq 2$ and
2. $N^1_{x_1}, \dots, N^k_{x_k}$ are consecutively concatenable, that is, PGS $[\![N^1_{x_1} \dots N^i_{x_i}]\!]$ and $[\![N^{i+1}_{x_{i+1}}]\!]$ are concatenable for all $2 \leq i < k$, so that final PGS $[\![N^1_{x_1} \dots N^k_{x_k}]\!]$ is valid, and
3. the outer face of PGS $[\![N^1_{x_1} \dots N^k_{x_k}]\!]$ is $[x_0]^R$, which is exactly the same as that of $[\![N^0_{x_0}]\!]$.

A production can be seen as the development of a face f made of $\texttt{rank}(N^0)$ vertices into k adjacent faces whose overall shape is the same as that of f. Figure 4 graphically shows the plane graph production $N_{2341} \to A_{2541}\ N_{2645}\ B_{2346}$.

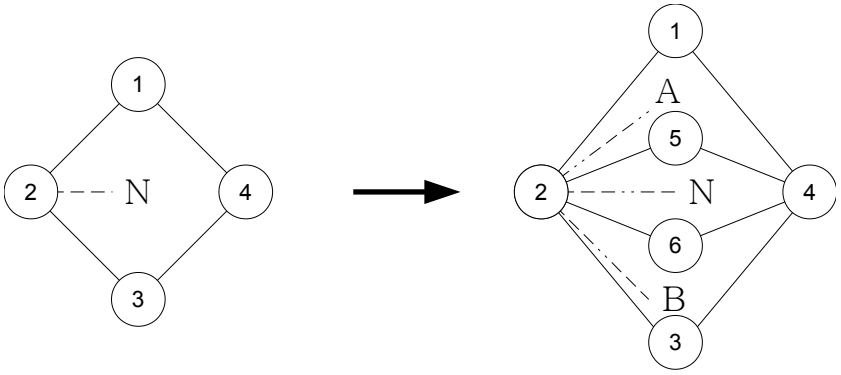

Figure 4. A graphical representation of the (recursive) plane graph production $N_{2341} \to A_{2541}\ N_{2645}\ B_{2346}$. Recall that the dashed lines attach each non-terminal symbol to the vertex appearing at the head of the string corresponding to that face in the non-terminal gadget.

We can now introduce the plane graph grammars that we consider in this paper:

Definition 3.5. (Plane graph grammar)
A *plane graph grammar (PGG)* $\mathcal{G}$ is a tuple $\langle \mathcal{N}, P_L, P, \mathcal{A} \rangle$ such that $\mathcal{N}$ is a set of non-terminal symbols, P_L is a set of plane graph lexical rules, P is a set of plane graph productions and $\mathcal{A} \subseteq \mathcal{N}$ is the set of axioms.

Example 3.6. Let $\mathcal{G}_1$ be the grammar $\langle\{(N,4),(A,4),(B,4)\}, P_L, P, \{(N,4)\}\rangle$, with $P_L = \{A_{1234} \to [123454], B_{1234} \to [15123634], N_{1234} \to [1234]\}$ and $P = \{N_{2341} \to A_{2541}\ N_{2645}\ B_{2346}\}$. The unique production of this grammar is the one represented in Figure 4 while the first of the three lexical rules is depicted in Figure 3. Both latter lexical rules are easy to draw from their definition.

In order to describe the derivation process of a PGS in a PGG, we need to introduce the plane graph analogue of string grammar sentential forms. Contrary to the string case, where such forms are just strings containing both terminal and non-terminal symbols, a plane sentential form consists of a PGS together with a labeling function. The role of this function is to attached non-terminals to faces: if a face is not labeled with a non-terminal, then it cannot be rewritten and can thus be considered as a terminal face; on the other hand, a face that is labeled can be seen as a non-terminal face: all the rules rewriting this non-terminal can be applied.

Definition 3.7. (Plane sentential form)
Let $\mathcal{G} = \langle \mathcal{N}, P_L, P, \mathcal{A}\rangle$ be a plane graph grammar. A *plane sentential form* is a couple $\langle G, \mathcal{L}\rangle$ where $G = \langle X, E, F, o\rangle$ is a valid PGS and $\mathcal{L} : F \to \mathcal{N} \times X$ is a partial function such that if $\mathcal{L}(f) = (N, a)$, then $|\texttt{nodes}(f)| = \texttt{rank}(N)$ and $a \in \texttt{nodes}(f)$.

Function $\mathcal{L}$ labels some faces with non-terminal symbols. It more precisely attaches the label to one distinguished vertex a of the face. This allows us to introduce some control during the application of a rule. Indeed, this trick is used to avoid all possible rotations of the right hand-side of the rule when this right hand-side is glued in the mother graph. We formally detail this below and a sequence of plane sentential forms is given in Figure 5.

3.1. Applying a lexical rule

Let $S = \langle G, \mathcal{L}\rangle$, with $G = \langle X, E, F, o\rangle$, be a sentential form. Consider a variant of lexical rule $R : N_{a_1 \dots a_m} \to G_*$, that is, a version of this rule where the vertices and the faces were consistently replaced with names that do not appear in G. We assume that $G_* = \langle X_*, E_*, \{f_*, o_*\}, o_*\rangle$ with $o_* = [a_1 \dots a_m]^R$.

We say that rule R is *applicable* to the sentential form S if there exists a face $f = [a'_1 \dots a'_m] \in F$ of G such that $\mathcal{L}(f) = (N, a'_1)$. In this case, we denote ϕ the *matching function* $\{a_1 \mapsto a'_1, \dots, a_m \mapsto a'_m\}$, extended to all other vertices of G_* (and more precisely, of f_*) by setting $\phi(a) = a$ if $a \notin \{a_1, \dots a_m\}$.

Applying lexical rule R to sentential form S consists in replacing the face f of S by the PGS G_*, which is sewn to the rest of S using matching function ϕ. The result is a new sentential form where $\mathcal{L}(f)$ is not defined anymore. More formally, *applying R to S following f* consists in creating the sentential form $S' = \langle G', \mathcal{L}'\rangle$ with $G' = \langle X', E', F', o'\rangle$ such that

- $F' = F \setminus \{f\} \cup \hat{\phi}(f_*)$;
- $o' = o$;
- $X' = X \cup \hat{\phi}(X_*)$;
- $E' = E \cup \hat{\phi}(E_*)$;
- $\forall f' \in F, f' \neq f$, if $\mathcal{L}(f')$ is defined then $\mathcal{L}'(f') = \mathcal{L}(f')$.

Plane graph grammars associate no semantics to the names chosen for the nodes and the faces, and generate plane graphs whose nodes and faces all have distinct names. This is explicit when a rule is applied: we use rule variants. This property will be essential to prove context-freeness properties (see next section).

Another interesting consequence is that the PGS G' which is created by applying a lexical rule is valid as soon as the PGS G which is rewritten is valid. Indeed, PGS G and G_* do not share any face nor vertex names, which implies the syntactic conditions for validity. As for Euler's formula, we have $|X_*| = |E_*|$ (since $|F_*| = 2$), so $|X'| - |E'| + |F'| = |X| - |E| + |F| - 1 + 1 = 2$.

Example 3.8. The lexical rule $A_{1234} \to [123454]$ in $\mathcal{G}_1$ of Example 3.6 is applicable to the sentential S_3 of Figure 5. Once applied, the result is the sentential form S_4 of the same figure.

3.2. Applying a production

Let $S = \langle G, \mathcal{L}\rangle$, with $G = \langle X, E, F, o\rangle$, be a sentential form. Consider a variant of a production $R : N^0_{a^0_1 \dots a^0_m} \to N^1_{a^1_1 \dots a^1_{n_1}} \dots N^k_{a^k_1 \dots a^k_{n_k}}$. As above, we say that rule R is *applicable* to sentential form S if there exists a face $f = [a'_1 \dots a'_m] \in F$ such that $\mathcal{L}(f) = (N^0, a'_1)$. We denote ϕ as well the *matching function* $\{a^0_1 \mapsto a'_1, \dots, a^0_m \mapsto a'_m\}$, extended to all other vertices of PGS $[\![N^1_{a^1_1 \dots a^1_{n_1}} \dots N^k_{a^k_1 \dots a^k_{n_k}}]\!]$ by setting $\phi(a) = a$ if $a \notin \{a^0_1, \dots, a^0_m\}$.

Applying R to S following f consists in creating the sentential form $S' = \langle G', \mathcal{L}'\rangle$ with $G' = \langle X', E', F', o'\rangle$ such that

- $F' = (F \setminus \{f\}) \bigcup_{1 \le i \le k} \{\hat{\phi}([a^i_1 \dots a^i_{n_i}])\}$
- $o' = o$;
- $X' = X \bigcup_{1 \le i \le k} \texttt{nodes}(\hat{\phi}([a^i_1 \dots a^i_{n_i}]))$;
- $E' = E \bigcup_{1 \le i \le k} \text{edges}(\hat{\phi}([a^i_1 \dots a^i_{n_i}]))$;
- New labeling function $\mathcal{L}'$ is as follows:
  - $\mathcal{L}'(\hat{\phi}([a^i_1 \dots a^i_{n_i}])) = (N^i, \phi(a^i_1))$ for all $1 \le i \le k$, and
  - $\forall f' \in F, f' \neq f$, if $\mathcal{L}(f')$ is defined, then $\mathcal{L}'(f') = \mathcal{L}(f')$.

Example 3.9. The production $N_{2341} \to A_{2541}\ N_{2645}\ B_{2346}$ in $\mathcal{G}_1$ of Example 3.6 is applicable to the sentential S_1 of Figure 5 (and it is also applicable to S_2). When applying it, the sentential form S_2 in the same figure is created.

Concerning the preservation of validity, note that the syntactic conditions holds again thanks to the renaming of the production before we apply the rule. Checking for Euler's formula is nevertheless a bit more complicated, because the intersections of the sets which are used to define X' and E' are not empty. So let us step back: the application of rule R returns to plug PGS $G_\diamond = [\![N^1_{a^1_1 \dots a^1_{n_1}} \dots N^k_{a^k_1 \dots a^k_{n_k}}]\!]$ in the hole left by face f. Suppose that $G_\diamond = \langle X_\diamond, E_\diamond, F_\diamond, o_\diamond\rangle$. Due to the validity of $G_\diamond$ (justified in Section 2.1), we have $|X_\diamond| - |E_\diamond| + |F_\diamond| = 2$. Moreover, the outer face of $G_\diamond$ matches the string $a^0_1 \dots a^0_m$, so face $o_\diamond$ is described with m vertices and m edges. Finally, the application of a rule

eliminate inner face f in G, and outer face $o_\diamond$ of $G_\diamond$. Therefore, we have $|X'| - |E'| + |F'| = (|X| + |X_\diamond| - m) - (|E| + |E_\diamond| - m) + (|F| - 1 + |F_\diamond| - 1) = 2$, and Euler's formula holds.

Hence, the application of a lexical rule or a production to a PGS replaces a face by a PGS whose outer face is the previous face, while the rest of the graph is unchanged. As we have seen above, these mechanisms preserve the validity of the PGS, and are thus well-defined and consistent.

3.3. Representable languages

Given a plane graph grammar $\mathcal{G} = \langle \mathcal{N}, P_L, P, \mathcal{A} \rangle$, we say that a plane graph $G = \langle X, E, F, o \rangle$ is generated by $\mathcal{G}$, or that $\mathcal{G}$ derives G, if there exists a sequence of sentential forms $S_1, \ldots, S_n$ such that

- $S_1 = \langle G_1, \mathcal{L}_1 \rangle$ is an *initial sentential form*, that is, $G_1 = \langle X_1, E_1, F_1, o_1 \rangle$ with $F_1 = \{o^R, o\}$ and $o_1 = o$ and $\mathcal{L}_1$ is only defined for o^R: $\mathcal{L}_1(o^R) = (N, a)$, with $N \in \mathcal{A}$ and $a \in \text{nodes}(o^R)$,
- $\forall i, 1 \leq i < n$: S_{i+1} is obtained from S_i by applying a rule of $\mathcal{G}$,
- $S_n = \langle G_n, \mathcal{L}_n \rangle$ with $G_n = G$ and $\forall f, \mathcal{L}_n(f)$ is not defined.

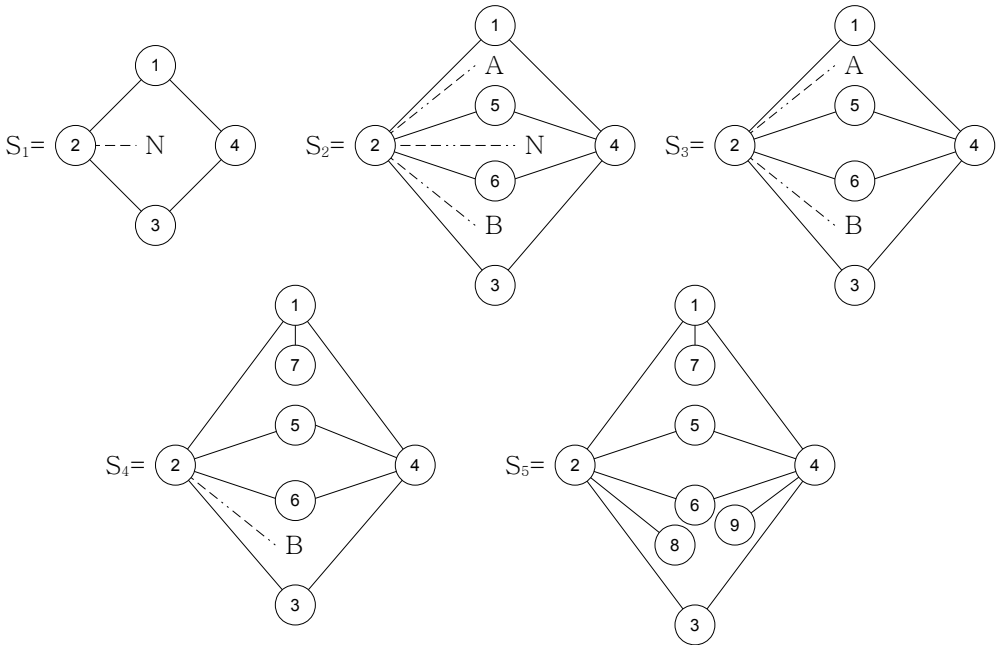

Figure 5. A graphical representation of an example of a derivation in the grammar $\mathcal{G}_1$ in Example 3.6. This derivation corresponds to the sequence of sentential forms S_1, S_2, S_3, S_4, S_5. The functions $\mathcal{L}_i$, $1 \leq i \leq 5$, are represented via dash lines.

The *length of the derivation* is $n - 1$. An example of a derivation of length 4 is given in Figure 5. Notice that every PGS generated by a PGG is necessarily valid because the application of any type of rules (lexical rule or production) preserves validity.

We will write $N \Rightarrow_{\mathcal{G}}^* G$, or simply $N \Rightarrow^* G$ if $\mathcal{G}$ is obvious from the context, when $G = \langle X, E, F, o \rangle$ is derivable with $\mathcal{G}$ using an initial sentential form S_1 where $\mathcal{L}_1$ is defined only on o^R

and $\mathcal{L}_1(o^R) = (N, a)$, for some $a \in \texttt{nodes}(o^R)$. If the length of the derivation is n, we shall write $N \Rightarrow_{\mathcal{G}}^{n} G$ as usual.

The language represented by $\mathcal{G}$ is $L(\mathcal{G}) = \{G : \exists G', \exists N \in \mathcal{A}\ s.t.\ N \Rightarrow_{\mathcal{G}}^{*} G' \wedge G' \cong_p G\}$.

Note that any PGS can be represented by a PGG. Indeed, we simply need an axiom of the grammar to generate the outer face of the PGS and the concatenation of the faces of PGS (without pendant edges), each of them labeled with a different non-terminal. Then we add as many lexical rules as the number of non-terminals to complete each face with pendant edges. Nevertheless, as it is discussed later on, not all then plane graph *languages* can be generated with a PGG.

3.4. Plane graph grammars and related formalism's

Two main types of graph grammars have been investigated in the literature. The first one is the family of node replacement grammars [32] and relies on a mechanism that replaces one given node by a subgraph using gluing conditions; many gluing conditions were studied and yield several subfamilies of node-replacement grammars. This is clearly different from how the generative process occurs with plane graph grammars and thus a comparison between these two formalism's is difficult. However, a dual graph can be built from each planar graph [37], where each node corresponds to a face in the original graph, edges in one graph being edges in the other. From this standpoint, replacing a face in the primal graph (*i.e.* the original PGS) by a pattern corresponds to substituting a node of the dual graph by the corresponding dual subgraph. Embeddings[2] in node replacement grammars differ from embeddings in plane graph grammars in that it relies on node label semantics.

Hyperedge replacement grammars [10] are another type of graph grammar formalism's that seems closer to the one introduced in this paper. Indeed, in these grammars, a hyperedge, *i.e.* a labeled entity that links up several nodes together, is replaced by a subgraph. One can imagine that plane graph grammars are a special kind of hyperedge replacement grammars, seeing a face labeled by a non-terminal as a hyperedge with the same label. However, these two elements are of different nature. For instance, there is no order on the outer nodes of a hyperedge and it is not possible to define a unique one. This implies that if the same non-terminal can be derived following two different ordering of the external nodes, then it needs to corresponds to two different hyperedges.

The main difference between these formalism's and plane graph grammars is the semantics they attach to the node label. Indeed, in both cases, the labels of the nodes of a glued subgraph are the ones the right hand-side of the rule, and if a rule is used several times, it generates the same node labels each time. As a consequence, the set of labels that can be found in a graph generated by such a grammar is bounded. Plane graph grammars are of different nature as each node has a unique label, which allows the closure of graph languages under isomorphism. The embedding mechanism (described in Section 3) does thus not rely on label value. Together with the polynomial testability of sub-isomorphism, this is a remarkable property from a learning standpoint: extracting and comparing patterns from a set of graphs is easy and informative. We shall see in detail how this is useful in Section 5. If one want the nodes to carry semantic information, the formalism is easily modifiable with the adding of a labeling function, without modifying the core of its generative mechanism.

It is worth noticing that other rewriting formalism's have been introduced for graph data in particular practical contexts (see for instance the work on 3Gmap L-systems for fruit modeling [2]). These types of formalism usually use a large number of parameters and constraints on rule application which does not make them good candidates for learning.

[2] An embedding is the information about how to glue the new subgraph within the rest of the graph

4. Properties of plane graph grammars

4.1. Context-freeness property

A class of grammars that has the context-freeness property corresponds to a formalism where parts of a derivation that start from different non-terminals of a sentential form do not interfere with each other [8]. Intuitively, this is not the case of Plane Graph Grammars as the name of a node created during the derivation is linked with the name of pre-existing nodes. Hence, the distinct parts of a derivation cannot be treated in any order since the resulting PGS will not have the same node names.

However, this only affects the names of the nodes, and if one is interested in the structure of the generated PGS then each parts of the derivation can be done independently. In other words, the order in which the rules are applied will generate different PGS, but they will all be plane-isomorphic. We thus will be able to describe a derivation tree and a parsing procedure based on the CYK algorithm for string grammars.

The following theorem states that when a PGS is derived from a non-terminal, either it is given directly by a lexical rule, or it is the concatenation of plane graphs obtained from non-terminals that appear together on the right handside of a production whose left handside is the given non-terminal.

Theorem 4.1. (Context-freeness)
Let $\mathcal{G} = \langle \mathcal{N}, P_L, P, \mathcal{A} \rangle$ be a PGG. Let $(N, r) \in \mathcal{N}$ and H a PGS. $N \Rightarrow^* H$ if and only if

- $N \to H$ in P_L,
- Or
  1. $\exists N_x \to N^1_{x_1} \dots N^m_{x_m}$ in P and
  2. there exist m PGS $H_1, \dots H_m$ such that
     - $\forall i, 1 \leq i \leq m$, $N^i \Rightarrow^* H_i$
     - $H \cong_p H_1 \diamond \dots \diamond H_m$

Proof:
[Sketch] $\Longrightarrow$
Let $N \Rightarrow^k H$. There thus exists a sequence of sentential forms $(G_1, \mathcal{L}_1), \dots, (G_{k+1}, \mathcal{L}_{k+1})$ such that $\forall j \leq k$, G_{j+1} is obtained from G_j by applying a rule of the grammar, $(G_1, \mathcal{L}_1)$ is an initial sentential form with $\mathcal{L}_1(o^R_{G_1}) = (N, a)$, for some $a \in \text{nodes}(o^R_{G_1})$, and $G_{k+1} = H$. If G_2 is obtained from G_1 using a lexical rule, then $k = 1$ and $N \to H$ is in P_L. Otherwise, the first rule is a production $N_x \to N^1_{x_1} \dots N^m_{x_m}$ and $G_2 = [\![N^1_{x_1} \dots N^m_{x_m}]\!] = \langle X_{G_2}, E_{G_2}, \cup_{1 \leq i \leq m} \{[x_i]\} \cup [x^R] \}, [x^R] \rangle$, $\mathcal{L}_2([x_i]) = (N^i, \texttt{first}(x_i))$, $\forall i$. As a rule does not modify the pre-existing nodes and applies only to a specific face, the other steps of the derivation replace one of the faces $[x_i]$ by a PGS: no step replacing $[x_i]$, or the set of faces previously derived from $[x_i]$, interferes with the steps that rewrite $[x_j]$, $1 \leq j, i \leq m$, $i \neq j$. Hence, for all $i \leq m$, the sequence of steps that recursively rewrites $[x_i]$ generates a PGS H_i and we have $H = H_1 \diamond \dots \diamond H_m$ (which is correctly defined since the outer face of H_i is $[x_i]^R$, $\forall i$). As each face $[x_i]$ of G_2 is labeled by the non-terminal N^i, we also have $N^i \Rightarrow^* H_i$.

$\Longleftarrow$
Suppose $\exists N_x \to N^1_{x_1} \dots N^m_{x_m}$ in P and there exist m PGS $H_1, \dots, H_m$ such that $N^i \Rightarrow^* H_i$, for all $1 \leq i \leq m$, and $H \cong_p H_1 \diamond \dots \diamond H_m$. Then the initial sentential form $(G_1, \mathcal{L}_1)$,

with $G_1 = \langle \texttt{nodes}([x]), \text{edges}[x]), \{[x], [x^R]\}, [x^R]\rangle$ and $\mathcal{L}_1([x]) = (N, \texttt{first}(x))$, can be rewritten into the sentential form $(G_2, \mathcal{L}_2)$, with $G_2 = \langle X_{G_2}, E_{G_2}, \cup_{1\leq i\leq m}\{[x_i]\} \cup \{[x^R]\}, [x^R]\rangle$ and $\mathcal{L}_2([x_i]) = (N^i, \texttt{first}(x_i))$, $\forall i$ (using the rule). There then exists a sequence of sentential forms $(G_2, \mathcal{L}_2)$, ..., $(G_k, \mathcal{L}_k)$ such that $G_k = \langle X_{G_k}, E_{G_k}, F_{H_1} \cup_{2\leq i\leq m} \{[x_i]\} \cup \{[x^R]\}, [x^R]\rangle$ and $\mathcal{L}_k$ is defined only on $[x_i]$, with $\mathcal{L}_m([x_i]) = (N^i, \texttt{first}(x_i))$, $\forall i, 2 \leq i \leq m$. As $N^2 \Rightarrow^* H_2$, there exist a sequence of sentential forms $(G_k, \mathcal{L}_k), \ldots, (G_l, \mathcal{L}_l)$ that rewrite the face $[x_2]$ of G_2 into a subgraph H'_2 such that $H'_2 \cong_p H_2$ (the label of some internal nodes of H_2 can already exists in G_k and thus another label has to be chosen). Repeating the same reasoning for the other non-terminals of the rule, we obtain that for all $2 \leq i \leq m$, a PGS $H'_i \cong_p H_i$ exists such that $N \Rightarrow^* H' = H_1 \diamond H'_2 \diamond \ldots H'_m$. We have $H_1 \diamond H'_2 \cong_p H_1 \diamond H_2$, and for all $i < m$ $H_1 \diamond \ldots \diamond H_i \diamond H'_{i+1} \cong_p H_1 \diamond \ldots \diamond H_i \diamond H_{i+1}$. Notice that $H_1 \diamond \ldots \diamond H_i$ and H'_{i+1} are concatenable following the identity function since $\texttt{outer}(H'_{i+1}) = \texttt{outer}(H_{i+1})$ by construction. This implies $H' \cong_p H$. □

One of the consequences of this result is that we can define a normal form for plane graph grammars, where the number of non-terminals on the right hand-side is exactly two. Indeed, given any production rule $N_x \to N^1_{x_1} \ldots N^m_{x_m}$ with $m > 2$, one can replace it by two rules $N_x \to N'_{x'} N^m_{x_m}$ and $N'_{x'} \to N^1_{x_1} \ldots N^{m-1}_{x_{m-1}}$, with $x' = \texttt{outer}(\{[x_i] : 1 \leq i \leq m-1\})$, which are both correctly defined rules. The set of graphs that N can derive is unchanged and the process can be recursively reproduced until only 2 non-terminals appear in each right hand-sides.

We will consider in the rest of this paper that all the PGG are in such a normal form.

4.2. A parsing algorithm

Theorem 4.1 provides a straightforward parsing algorithm, given as pseudo-code in Algorithm 1.

Algorithm 1: Plane graph grammar parsing algorithm

Input: A PGG $\mathcal{G} = \langle \mathcal{N}, P_L, P, \mathcal{A}\rangle$ in normal form and a PGS G
Output: TRUE if $G \in L(\mathcal{G})$, FALSE otherwise

```
foreach N ∈ A do
    if DERIVE(G, N, G) then
        return TRUE
return FALSE
```

Proposition 4.2. For all PGS G, Algorithm 1 terminates and yields TRUE iff $G \in L(\mathcal{G})$.

Proof:
Algorithm 1 is nothing else than an algorithmic version of the context-freeness lemma. Since the size of the patterns H_1 and H_2 is smaller than the size of H, the algorithm must eventually terminate. □

Since there is only a finite set of lexical rules and of productions in $\mathcal{G}$, there are only polynomially many possibilities to consider in steps (1) and (3) of the DERIVE procedure. The plane-isomorphism can also be tested in polynomial time. Hence, there is a single point that may cause an exponential running time of the algorithm: the number of candidates H_1 and H_2 to test in step (4). Therefore we

Algorithm 2: DERIVE Procedure

Input: A PGG $\mathcal{G} = \langle \mathcal{N}, P_L, P, \mathcal{A} \rangle$ in normal form, a non-terminal $N \in \mathcal{N}$ and a PGS G
Output: TRUE if $N \Rightarrow_{\mathcal{G}}^{*} G$, FALSE otherwise

```
if N → H ∈ P_L and H ≅_p G then
    return TRUE
foreach N_x → N_y^1 N_z^2 ∈ P do
    if ∃H_1, H_2 such that DERIVE(N^1, H_1) and DERIVE(N^2, H_2) and G ≅_p H_1 ⋄ H_2 then
        return TRUE
return FALSE
```

aim at finding a condition to impose on $L(\mathcal{G})$ that implies a polynomial upper bound on the number of such candidates. The following restriction is inspired by the k-separability, defined for hyperedge replacement grammars [29].

Definition 4.3. (Rank)
For k in $\mathbb{N}$, the k-rank of a PGS G is the number of patterns of G whose outer face contains k nodes. For every language L of PGS, $\text{rank}_L : \mathbb{N} \to \mathbb{L}$ is defined by

$$\text{rank}_L(n) = max_{1 \leq k \leq order(L,n)} \{k\text{-rank}(G) : G \in L \text{ and } |G| \leq n\}$$

where $order(L, n) = max\{|\texttt{outer}(S)| : S \text{ is a pattern of } G \in L \text{ and } |G| \leq n\}$.

Note that the order of a language L and an integer n is the maximal number of nodes of the outer face of a pattern of a plane graph G in L whose size is at most n.

Informally, the rank for n of a language considers the graphs of the language whose size are less than n and corresponds to the maximum number of patterns having the same number of nodes on the outer face among all of these graphs. In other word, to evaluate $\text{rank}_L(n)$ for a given n, we need to look at all the graphs of size less than n in the language, for each of these graphs to count how many patterns have the same number of nodes on their outer face, and to return the maximum obtained.

The idea behind the rank of a plane graph language is to link the size of the PGS of the languages with the number of their subgraphs that have an outer face of a given size. The aim is to tackle the combinatorial explosion that can occur when one is checking whether a rule can be applied. Indeed, to test if a rule $N_x \to N_y^1 N_z^2$ can be applied to derived a PGS G, Algorithm 2 needs to be recursively called on all decompositions of G into two patterns whose outer faces contain $|y|$ and $|z|$ nodes. In general, the number of such decompositions is exponential in the size of the grammar: this is the case for example of the PGS that correspond to a checkerboard.

Proposition 4.4. Let $\mathcal{G}$ be a PGG. If $\text{rank}_{L(\mathcal{G})}(n)$ is polynomial in n then Algorithm 1 can be implemented in time polynomial in the size of its input.

Proof:
Given a production $N_x \to N_y^1 N_z^2$ only patterns of $|y|$ (resp. $|z|$) nodes on the outer face can be derived from N^1 (resp. N^2). As $\text{rank}_{L(\mathcal{G})}(n)$ is polynomial, there are a polynomial number of patterns with $|y|$ (resp. $|z|$) external nodes on outer face. The number of candidates H_1, H_2 is thus polynomial if $G \in L(\mathcal{G})$. □

5. Learning substitutable plane graph languages

This section shows how plane graph grammars are good candidates for grammatical inference: their nice properties allow to extend works on distributional learning of string grammars. The simplest string class that has been proven learnable in this approach, is the one of substitutable context-free languages [5].

5.1. Substitutable plane graph languages

The core of the learning algorithm for this class is to observe the distribution of substrings into contexts and then to use the simple properties of substitutable languages to infer a correct grammar. For this reason we first need a notion of context to transpose this work to plane graph languages.

Definition 5.1. (Plane context)
A plane context is a tuple $C = \langle X, E, F, h, o\rangle$ such that (1) $\langle X, E, F, o\rangle$ is a plane graph system and (2) $h \in F \setminus \{o\}$ is a distinguished face called the *hole* of context C and (3) h has no pendant edge.

The plane-isomorphism relation is extended to contexts in the obvious way: two contexts $C = \langle X, E, F, h, o\rangle$ and $C' = \langle X', E', F', h', o'\rangle$ are plane-isomorphic if $\langle X, E, F, o\rangle \cong_p \langle X', E', F', o'\rangle$ and the image of h by the bijection on the faces is h', *i.e.* $\xi(h) = h'$.

Let $S = \langle X, E, F, o\rangle$ and $S' = \langle X', E', F', o'\rangle$ be two PGS such that $X \cap X' = \emptyset$. Let $f \in F$ and $f' \in F'$ be two faces. Every 1-to-1 mapping $\phi : \texttt{nodes}(f) \to \texttt{nodes}(f')$ can be extended to the set of all vertices X as follows: $\hat{\phi} : X \to \texttt{nodes}(f') \cup X$ such that $\hat{\phi}(a) = \phi(a)$ if $a \in \texttt{nodes}(f)$ and $\hat{\phi}(a) = a$ otherwise. It can then be extended in the usual way to sets of nodes, faces, sets of faces, to PGS, and to plane contexts.

We can now define the *gluing* or *wrapping* operation.

Definition 5.2. (Gluing)
Let $C = \langle X, E, F, h, o\rangle$ be a context and $S = \langle X', E', F', o'\rangle$ be a PGS such that $X \cap X' = \emptyset$. Let ϕ be a bijective function from $\texttt{nodes}(o')$ to $\texttt{nodes}(h)$. The *gluing* of S into C following gluing function ϕ, denoted $C \odot_\phi S$, is the PGS $G = \langle X_G, E_G, F_G, o_G\rangle$ such that

- $X_G = X \cup X' \setminus \texttt{nodes}(o')$,
- $E_G = E \cup \hat{\phi}(E')$,
- $F_G = (F \setminus \{h\}) \cup \hat{\phi}(F' \setminus \{o'\})$ and
- $o_G = o$.

Notice that S is a pattern of G.

Figure 6 gives an example of a plane context in (a), of a plane graph in (b), of the gluing in (c) of the graph in (b) into the context in (a).

We now need to define the notion of substitutability. Informally, two patterns of a given language are substitutable if the fact that they appear in the same context once, implies they always appear in the same context, glued in a similar way.

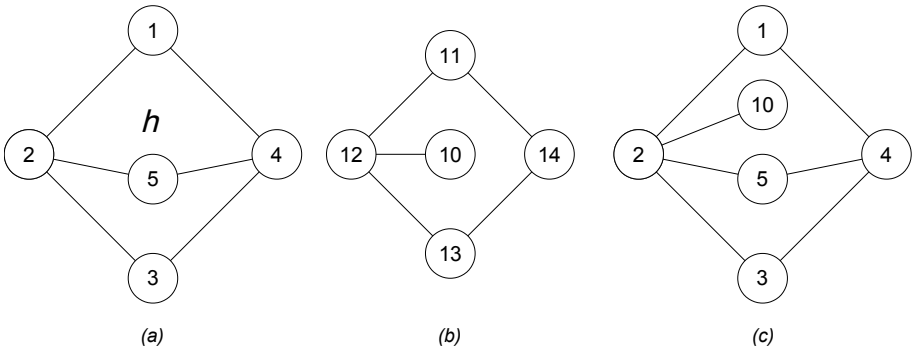

Figure 6. *(a)* A context whose hole is the face h. *(b)* A plane graph. *(c)* The gluing of the graph into the context following the gluing function $\phi(11) = 1, \phi(12) = 2, \phi(13) = 5, \phi(14) = 4$.

Definition 5.3. (Substitutability)
Two PGS $G = \langle X, E, F, o\rangle$ and $G' = \langle X', E', F', o'\rangle$ are *substitutable w.r.t.* a plane graph language L if whenever there exist two contexts C and C', $C \cong_p C'$, and two gluing functions ϕ and ϕ' such that $C \odot_\phi G$ is in L and $C' \odot_{\phi'} G'$ is in L, then for all contexts C'',

$$\left(\exists \phi_1 :\ C'' \odot_{\phi_1} G \in L\right) \iff \left(\exists \phi_2 :\ C'' \odot_{\phi_2} G' \in L\right)$$

where ϕ_1 and ϕ_2 are gluing functions such that, for all $a \in \texttt{nodes}(o)$, for all $b \in \texttt{nodes}(o')$, if $\phi(a) = \phi'(b)$ then $\phi_1(a) = \phi_2(b)$.

If G and G' are substitutable *w.r.t.* a language L, we will note $G \equiv_S^L G'$, or $G \equiv_S G'$ when there is no ambiguity.

The following lemma states that substitutability is not affected by plane-isomorphism.

Lemma 5.4. Let G, G' and G'' be PGS. If $G \cong_p G'$ and $G' \equiv_S G''$ then $G \equiv_S G''$.

Proof:
[Sketch] Let χ be the 1-to-1 function that maps the vertices of G onto those of G' (as in the definition of plane isomorphism). Let C be a context such that there exists a gluing functions ϕ such that $C \odot_\phi G''$ in L. As $G' \equiv_S G''$ there exists ϕ' such that $C \odot_{\phi'} G'$ in L. By construction we have $C \odot_{\phi' o \chi} G \cong_p C \odot_{\phi'} G'$ and thus $C \odot_{\phi' o \chi} G$ is in L. □

We can then define substitutable plane graph languages:

Definition 5.5. (Substitutable languages)
A plane graph language L is substitutable iff all pairs of patterns that share a context are substitutable *w.r.t.* L.

The following technical lemma states that substitutability in a substitutable language is a congruence with respect to concatenation:

Lemma 5.6. Let L be a substitutable plane graph language and G_1, G_2, G_1', G_2' be 4 PGS s.t. both G_1 and G_2, and G_1' and G_2' are concatenable following the identity function. If $G_1 \equiv_S^L G_1'$ and $G_2 \equiv_S^L G_2'$ then $G_1 \diamond G_2 \equiv_S^L G_1' \diamond G_2'$

Proof:
[Sketch] We index each element of the definition of a PGS by its name. For instance X_{G_1} are the vertices of G_1, $F_{G_1 \diamond G_2}$ are the faces of $G_1 \diamond G_2$, $o_{G'_1 \diamond G'_2}$ is the outer face of $G'_1 \diamond G'_2$.

Let C be a plane context such that there exists a gluing function ϕ: $C \odot_\phi G_1 \diamond G_2$ is in L. We suppose *w.l.o.g.* that the set of nodes of C is distinct of the set of nodes of the 4 PGS under consideration. We want to show that there exists a function χ such that $C \odot_\chi G'_1 \diamond G'_2 \in L$.

Let $C_{C \diamond \hat{\phi}(G_1)} = \langle X_C \cup \hat{\phi}(X_{G_1}) \cup \mathtt{nodes}(\hat{\phi}(o_{G_2})), E_C \cup \hat{\phi}(E_{G_1}) \cup \mathtt{edges}(\hat{\phi}(o_{G_2})), (F_C \setminus h) \cup \hat{\phi}(F_{G_1} \setminus o_{G_1}) \cup \hat{\phi}(o^R_{G_2}), \hat{\phi}(o^R_{G_2}), o_C \rangle$. Less formally, $C_{C \diamond \hat{\phi}(G_1)}$ is the context C with G_1 glued in it following ϕ (its hole is thus the face $\hat{\phi}(o^R_{G_2})$). Notice that $C_{C \diamond \hat{\phi}(G_1)}$ is correctly defined as its faces are connected since, by definition of the concatenation, there exists $e \in E_{G_1}$, $e \in \mathtt{edges}(o_{G_1 \diamond G_2})$ and $\phi : \mathtt{nodes}(o_{G_1 \diamond G_2}) \to \mathtt{nodes}(h)$. It is easy to verify by using the definitions that $C_{C \diamond \hat{\phi}(G_1)} \odot_\phi G_2 = C \odot_\phi G_1 \diamond G_2$.

As $G_2 \equiv_S G'_2$ there exists ϕ' such that $C_{C \diamond \hat{\phi}(G_1)} \odot_{\phi'} G'_2 \in L$. But $C_{C \diamond \hat{\phi}(G_1)} \odot_{\phi'} G'_2$ is equal by construction to $C \odot_{id} \hat{\phi}(G_1) \diamond \hat{\phi}'(G'_2)$ where id is the identity function. Using the same kind of construction, we can construct the context $C_{C \diamond \hat{\phi}'(G'_2)}$ such that $C_{C \diamond \hat{\phi}'(G'_2)} \odot_{id} \hat{\phi}(G_1) = C \odot_{id} \hat{\phi}(G_1) \diamond \hat{\phi}'(G'_2) \in L$. As $G_1 \equiv_S G'_1$ and $\hat{\phi}(G_1) \cong_p G_1$, there exists ϕ'' such that $C_{C \diamond \hat{\phi}'(G'_2)} \odot_{\phi''} G'_1 \in L$. Again, this is equivalent to write $C \odot_{id} \hat{\phi}''(G'_1) \diamond \hat{\phi}'(G'_2)$, and as $\hat{\phi}''(G'_1) \diamond \hat{\phi}'(G'_2) \cong_p G'_1 \diamond G'_2$ then by Lemma 5.4 and the closure under plane-isomorphism of L, there exists χ: $C \odot_\chi G'_1 \diamond G'_2 \in L$. □

Finally, we can define a notion of congruence classes:

Definition 5.7. (Congruence classes)
Given a plane graph language L and a pattern G of a graph of the language, the congruence class of G in L, denoted $\lceil G \rceil_L$ (or simply $\lceil G \rceil$), is the set of all patterns substitutable with G in L: $\lceil G \rceil_L = \{G' : G' \equiv_L G\}$.

In a substitutable plane graph language, if two patterns appear once in the same context, they belong to the same congruence class.

5.2. The learner

Our learning algorithm is described in Algorithm 3.

As we are interested in the class of substitutable plane graph languages, the learning algorithm has to deal with the distribution of contexts between subgraphs. To do so, it computes, from a finite set S of PGS of the language, the *observable congruence classes*: two PGS G and G' are in the same observable congruence class if there exist $k \geq 2$ and $G_1, \ldots, G_k$ such that $G = G_1$, $G' = G_k$, and $\forall i < k$, G_i and G_{i+1} appear at least once in the same context in the sample S. As in the case of strings, this computation can be done using a substitution graph [5] or by hashing from subgraphs to list of contexts. Notice that two PGS that are observed in the same congruence class are substitutable but that the converse is not true in general: two PGS that are substitutable may not be observed in the same congruence class in a given sample.

Given a set of connected inner faces F, we define $split(F)$ to be the set of couples (F_1, F_2) such that $F_1 \cup F_2 = F$ and G_{F_1} and G_{F_2} are concatenable following the identity, where for $i \in \{1, 2\}$, $G_{F_i} = \langle \mathtt{nodes}(F_i), \mathtt{edges}(F_i), F_i \cup \{\mathtt{outer}(F_i)\}, \mathtt{outer}(F_i) \rangle$. The function $number\_nodes(C)$ returns the number of nodes of the outer face of the graphs in the observable congruence class C.

Algorithm 3: Learning algorithm for substitutable plane graph languages

Input: A learning set of plane graph systems $LS = \{G_i\}_{i=1}^{n}$
Output: A plane graph grammar $\langle \mathcal{N}, P_L, P, \mathcal{A} \rangle$

$CC \leftarrow compute\_observable\_congruence\_classes(LS)$;
$\mathcal{N} \leftarrow \emptyset$; $P_L \leftarrow \emptyset$; $P \leftarrow \emptyset$; $\mathcal{A} \leftarrow \emptyset$;
foreach *Observable congruence class* C_i *of* CC **do**
  $\mathcal{N} \leftarrow \mathcal{N} \cup \{(N^i, number\_nodes(C_i))\}$;
  foreach G *in* C_i **do**
    $\mathcal{N}^t(G) \leftarrow N^i$;
    if $G \in LS$ **then**
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{N^i\}$
foreach $G = \langle X_G, E_G, F_G, [(a_1 \dots a_n)^R] \rangle$ *in* V **do**
  if $|F_G| = 2$ **then**
    $P_L \leftarrow P_L \cup \{\mathcal{N}^t(G)_{a_1 \dots a_n} \rightarrow G\}$;
  else
    foreach $(F_1, F_2) \in split(F_G \setminus \{o_G\})$ **do**
      $P \leftarrow P \cup \{\mathcal{N}^t(G_*)_{a_1 \dots a_m} \rightarrow \mathcal{N}^t(G_1)_{b_1 \dots b_n}\, \mathcal{N}^t(G_2)_{c_1 \dots c_p}\}$ where
      $[b_1 \dots b_n] = \mathtt{outer}(F_1)$, $[c_1 \dots c_p] = \mathtt{outer}(F_2)$, $b_1 = b$, $c_1 = c$ and
      $\forall i : G_i = \langle X_i, E_i, F_i \cup \{\mathtt{outer}(F_i)\}, \mathtt{outer}(F_i) \rangle$;
return $\langle N, P_L, P, \mathcal{A} \rangle$

5.3. Identification in the limit

We now define our learning criterion. This is set-driven identification from positive text, with a polynomial bound on computation:

Definition 5.8. (Set-driven identification in polynomial time)
A representation class $\mathbb{R}$ is set-driven identifiable from positive data in polynomial time iff there exist a polynomial p and an algorithm $\mathcal{A}$ such that:

1. Given a positive sample LS of size m, $\mathcal{A}$ returns a representation $R \in \mathbb{R}$ in time $p(m)$;
2. For each representation $R \in \mathbb{R}$ there exists a characteristic set CS such that if $CS \subseteq LS$, $\mathcal{A}$ returns a representation R' such that $L(R) = L(R')$.

Note that the size of a set of plane graphs LS is defined as $|LS| = \sum_{G \in LS} |G|$.

The initial definition of this learning paradigm requires the size of the characteristic sample to be polynomial in the size of the target representation [21]. However, this definition, initially designed for the learning of regular string languages, is already unsuitable as a model for context free string grammars [38]: classes of languages made of one element are not learnable in this paradigm if the unique element of the language is exponentially larger than the corresponding grammar. The same argument holds for graph grammars so one cannot expect this requirements to be fulfilled. As it is beyond the scope of this paper to attempt to resolve this difficulty, we shall thus adopt this approach in this paper: for a complete discussion, the reader is referred to devoted chapter in [13].

5.3.1. Time complexity

The number of patterns (and thus of contexts) that can be generated from a given PGS can be exponential in the size of that PGS (it is the case for instance of the plane graph corresponding to a grid, like a chest board). So the size of observable congruence classes is in general exponential in the size of the learning sample. This is a well-known problem while using graph grammar formalism's as it is related to the one of having an efficient parsing algorithm. However, the requirement of having a language of polynomial rank, needed for efficient parsing (see Section 4.2) implies that the number of patterns to considerate is polynomial in the size of the learning sample. Therefore the number of elements that have to be taken into account to compute the congruence classes is polynomial.

To compute these observable congruence classes, we also need to compare all pairs of contexts to decide if they are plane isomorphic. This can be done in polynomial time in the size of the contexts [22]. For the same reason, testing if two PGS are plane isomorphic can be done in polynomial time and thus so is the construction of the congruence classes.

All other steps of Algorithm 3 are polynomial in the size of the observable congruence classes.

5.3.2. Proof the hypothesis is not too large

The following lemma states that patterns in the sample can be generated by the output grammar.

Lemma 5.9. If $G = \langle X, E, F, o\rangle$ is a subgraph of a sample LS, then there exists a plane graph G' such that $\mathcal{N}^t(G) \Rightarrow^* G'$ and $G \cong_p G'$.

Proof:
[Sketch] The proof can be done by induction on the number of faces of the graph. if $|F| = 2$, then by the construction of the grammar there is a lexical rule $\mathcal{N}^t(G)_{a_n \dots a_1} \to G$ with $[a_1 \dots a_n] = o$. Suppose the property holds for graphs with $|F| = k \geq 2$ faces. Let F_1 and F_2 be two sets of connected faces such that $F_1 \cap F_2 = \emptyset$ and $F_1 \cup F_2 = F_G \setminus \{o_G\}$. Let G_1 (resp. G_2) be the PGS whose inner faces are F_1 (resp. F_2). We have $G_1 \diamond G_2 = G$. G_1 and G_2 are also subgraphs of LS by definition and, by construction of the grammar, there exists a rule $\mathcal{N}^t(G)_{a_1 \dots a_m} \to \mathcal{N}^t(G_1)_{b_1 \dots b_n} \mathcal{N}^t(G_2)_{c_1 \dots c_n}$ with $[a_1 \dots a_m] = o$, $[b_1 \dots b_n] = \texttt{outer}(F_1) (= o_{G_1})$ and $[c_1 \dots c_p] = \texttt{outer}(F_2) (= o_{G_2})$.

This rule can be applied to the sentential form $\langle G', \mathcal{L}'\rangle$, with $G' = \langle X', E', \{o^R, o\}, o\rangle$, $\mathcal{L}'(o^R) = (\mathcal{N}^t(G), a_1)$. It gives the sentential form $\langle G'', \mathcal{L}''\rangle$, with $G'' = \langle X'', E'', \{\hat{\phi}(\texttt{outer}(F_1)), \hat{\phi}(\texttt{outer}(F_2)), o\}, o\rangle$, $\mathcal{L}''(\texttt{outer}(F_1)) = (\mathcal{N}^t(G_1), b_1)$ and $\mathcal{L}''(\texttt{outer}(F_2)) = (\mathcal{N}^t(G_2), c_1)$ By the inductive hypothesis there exist G'_1 and G'_2 such that $\mathcal{N}^t(G_1) \Rightarrow^* G'_1$, $\mathcal{N}^t(G_2) \Rightarrow^* G'_2$, $G_1 \cong_p G'_1$ and $G_2 \cong_p G'_2$. G'_1 and G'_2 might not be concatenable, as they can have inner nodes that share the same label. However, w.l.o.g. one can change the labels of one of the graph, for instance G'_1, in order to obtain a PGS G''_1 that is plane-isomorphic to G_1 and concatenable to G'_2. Thus we have $\mathcal{N}^t(G) \Rightarrow^* G''_1 \diamond G'_2$ and, by Lemma 2.10, $G \cong_p G'_1 \diamond G'_2$. □

Lemma 5.10. For all subgraphs G of a learning sample LS, for all PGS G', if $\mathcal{N}^t(G) \Rightarrow^* G'$ then G and G' are substitutable.

Proof:
[Sketch] Let $G = \langle X, E, F, o\rangle$. As the lemma holds for $G' \cong_p G$, we restrict ourselves to the case $G' \ncong_p G$. By induction on the length of the derivation k. If $k = 1$, then it means that a lexical

production $\mathcal{N}^t(G)_{a_1 \dots a_n} \to G''$ is applied and that $G'' \cong_p G'$. By the construction of the lexical rules, it means that G'' is a subgraph of LS that appears in the congruence class than G and thus G and G'' are substitutable. Lemma 5.4 implies that $G' \equiv_S G$.

Suppose this is true for all derivations of length strictly less than k and let G' be a PGS obtained from $\mathcal{N}^t(G)$ using k derivation steps. It means that there exists a sequence of sentential form $S_1, \dots, S_k$, such that $\forall i$, S_i is derived from S_{i-1}, $S_i = \langle G_i, \mathcal{L}_i \rangle$ with $G_1 = \langle X_1, E_1, \{o^R, o\}, o \rangle$, $\mathcal{L}_1(o^R) = (\mathcal{N}^t(G), a)$ for some $a \in \texttt{nodes}(o)$, and $G_k = G'$, $\mathcal{L}_k$ being undefined for all faces of G_k. S_2 is obtained from S_1 applying a rule $\mathcal{N}^t(G)_{a_1 \dots a_m} \to \mathcal{N}^t(G_{F_1})_{b_1 \dots b_n} \mathcal{N}^t(G_{F_2})_{c_1 \dots b_p}$, where $\texttt{outer}(F_1) = [b_1 \dots b_n]$ and $\texttt{outer}(F_2) = [c_1 \dots c_p]$, $G_{F_i} = \langle X_{F_i}, E_{F_i}, F_i \cup \texttt{outer}(F_i), \texttt{outer}(F_i) \rangle$, for $i \in \{1, 2\}$. By construction, there exists G_* in the same observable congruence class of G such that $G_* = G_{F_1} \diamond G_{F_2}$ and thus $G_{F_1} \diamond G_{F_2} \equiv_S G$. There exist G'_{F_1} and G'_{F_2} such that $\mathcal{N}^t(G_{F_1}) \Rightarrow^* G'_{F_1}$, $\mathcal{N}^t(G_{F_2}) \Rightarrow^* G'_{F_2}$ and $G_k = G'_{F_1} \diamond G'_{F_2}$. By recursion, $G'_{F_1} \equiv_S G_{F_1}$ and $G'_{F_2} \equiv_S G_{F_2}$. As Lemma 5.6 holds, we have $G'_{F_1} \diamond G'_{F_2} \equiv_S G_{F_1} \diamond G_{F_2}$ and thus $G_k \equiv_S G$. □

Theorem 5.11. For all samples of a language L, the output $\mathcal{G}$ of Algorithm 3 is such that $L(\mathcal{G}) \subseteq L$.

Proof:
Let $G \in L(\mathcal{G})$. Then there exists a plane graph G' in the learning sample and a plane graph G'' such that $\mathcal{N}^t(G') \in \mathcal{A}, \mathcal{N}^t(G') \Rightarrow^* G''$ and $G'' \cong_p G$. Lemma 5.10 states that G'' and G' are substitutable and thus $G \equiv_S^L G'$. As G' is an element of L, $G \in L$. □

5.3.3. Proof the hypothesis is large enough

To prove that the hypothesis is large enough, we need to define a characteristic set, *i.e.* a subset of the target language L_* which ensures that the output $\mathcal{G}$ of the algorithm is such that $L(\mathcal{G}) = L_*$.

Construction of a characteristic sample. Let $\mathcal{G}_* = \langle \mathcal{N}_*, P_{L*}, P_*, \mathcal{A}_* \rangle$ be a target grammar. We will assume without loss of generality, that $\mathcal{G}_*$ is reduced, that is to say for every non-terminal N, (1) there exists a derivation that starts from an axiom and labels at least one face with N, and (2) a PGS without any non-terminal labeling a face can be derived from a sentential form where one face is labeled by N. We are going to define a set $CS(\mathcal{G}_*)$ of plane graphs of L_*, such that Algorithm 3 will identify L_* from any superset of $CS(\mathcal{G}_*)$.

Given a non-terminal N^k, we define $C(N^k)$ to be one of the smallest context $\langle X_{G_k}, E_{G_k}, F_{G_k}, h_k, o_{G_k} \rangle$ such that there exists a sequence of sentential forms $\langle G_1, \mathcal{L}_1 \rangle, \dots, \langle G_k, \mathcal{L}_k \rangle$ with $\langle G_1, \mathcal{L}_1 \rangle$ being an initial sentential form such that $F_{G_1} = [o_{G_k}, o^R_{G_k}]$, $\mathcal{L}_1(o^R_{G_k}) = (N^i, a_1)$, $N^i \in \mathcal{A}_*$, and $\forall i, 1 \leq i < k$, $\langle G_{i+1}, \mathcal{L}_{i+1} \rangle$ is obtained from $\langle G_i, \mathcal{L}_i \rangle$ by applying a rule of $\mathcal{G}_*$, $\mathcal{L}_k(h_k) = (N^k, a_k)$ for some $a_k \in \texttt{nodes}(h_k)$, $\mathcal{L}_k$ is undefined on other faces.

We also define $G(N^k)$ to be one of the smallest PGS such that $N^k \Rightarrow^*_{\mathcal{G}_*} G(N^k)$.

We can now define the characteristic set $CS(\mathcal{G}_*)$. For each production $N^i_x \to N^j_y N^k_z$ in P_*, we add to $CS(\mathcal{G}_*)$ the PGS $C \odot_\phi \hat{\chi}(G_1) \diamond \hat{\chi}(G_2)$ where $\phi : \texttt{nodes}([x]) \to \texttt{nodes}(h)$ is a bijective function, $C = C(N^i), G_1 = G(N^j), G_2 = G(N^k)$ and $\chi : \texttt{nodes}(o_{G_1}) \cup \texttt{nodes}(o_{G_2}) \to \texttt{nodes}([y]) \cup \texttt{nodes}([z])$ is a bijective function such that $\hat{\chi}(o_{G_1}) = [y]$ and $\hat{\chi}(o_{G_2}) = [z]$. For each lexical rule $N^i_x \to G$ in P_{L*} we add to $CS(\mathcal{G}_*)$ the PGS $C \odot_\phi G$ where $\phi : \texttt{nodes}([x]) \to \texttt{nodes}(h)$ is a bijective function and $C = C(N^i)$.

The cardinality of this set is at most $|P_*| + |P_{L*}|$ which is clearly polynomially bounded. In general the cardinality of the set will not polynomially bound the size of the sample, as it is already

the case for string context-free grammars (see [5] for a detailed discussion). However, notice that if there exists a polynomial-sized structurally complete sample – that is to say a sample where for each production rule there is at least one plane graph that can be generated by using it [11] – then the size of our characteristic set is polynomial. In addition, one can show that the size of this characteristic set is polynomial in the size of the target grammar and of its *thickness*[3], following the refinement of the learning paradigm suggested by Ryo Yoshinaka [38].

Convergence. We must show that for any substitutable plane graph grammar $\mathcal{G}_*$, if the sample LS contains the characteristic sample $CS(\mathcal{G}_*)$, then $L(\mathcal{G}) = L(\mathcal{G}_*)$ where $\mathcal{G} = \langle \mathcal{N}, P_L, P, \mathcal{A} \rangle$ is the inferred grammar.

Lemma 5.12. If $N \Rightarrow^*_{\mathcal{G}_*} G$ then there exists a subgraph G' of the learning sample and a plane graph G'' such that $N \Rightarrow^*_{\mathcal{G}_*} G', \mathcal{N}^t(G') \Rightarrow^*_{\mathcal{G}} G''$ and $G'' \cong_p G$.

Proof:
[Sketch] By recursion on the number of derivation steps k in $\mathcal{G}_*$. If $k = 1$ then there exists $N \to G'$ in P_{L*}, $G' \cong_p G$. By construction of the characteristic sample, G' is a subgraph of LS and thus $\mathcal{N}^t(G') \to G'$ is in P_L.

Suppose it is true for all derivations of size less than $k > 1$. There exists a sequence of sentential forms $\langle G_1, \mathcal{L}_1 \rangle, \ldots, \langle G_k, \mathcal{L}_k \rangle$ such that $\langle G_1, \mathcal{L}_1 \rangle$ is an initial sentential form with $\mathcal{L}(f_1) = (N, a)$, S_{i+1} is obtained from S_i by using a rule of $\mathcal{G}_*$, $G_k = G$ and $\mathcal{L}_k$ is not defined for any face. Let $N_x \to N^i_y N^j_z$ be the rule applied to S_1 to obtain S_2. By construction, there exist G_1 and G_2, $N^i \Rightarrow^*_{\mathcal{G}_*} G_1$, $N^j \Rightarrow^*_{\mathcal{G}_*} G_2$, and $G_1 \diamond G_2 = G$.

By recursion, there exist two subgraphs of LS, G'_1 and G'_2, and two PGS G''_1 and G''_2 such that $N^i \Rightarrow^*_{\mathcal{G}_*} G'_1$, $N^j \Rightarrow^*_{\mathcal{G}_*} G'_2, \mathcal{N}^t(G'_1) \Rightarrow^*_{\mathcal{G}} G''_1, \mathcal{N}^t(G'_2) \Rightarrow^*_{\mathcal{G}} G''_2$ and $G''_1 \cong_p G_1$, $G''_2 \cong_p G_2$. Notice that this implies there exists a renaming function ϕ on the vertices of the external faces of G''_1 and G''_2 such that $\hat{\phi}(G''_1)$ and $\hat{\phi}(G''_2)$ are concatenable and $\hat{\phi}(G''_1) \diamond \hat{\phi}(G''_2) \cong_p G$ (Lemma 2.10).

By construction of the characteristic sample, there exist two subgraphs G'''_1 and G'''_2 of LS such that $G'''_1 \diamond G'''_2$ is a subgraph of LS, $G'''_1 \cong_p G(N^i)$ and $G'''_2 \cong_p G(N^j)$. As $L(\mathcal{G}_*)$ is a substitutable language, we have $G'''_1 \equiv_S G'_1$ and $G'''_2 \equiv_S G'_2$. Thus G'_1 and G'''_1 appear in the same component and thus correspond to the same non-terminal (and similarly for G'''_2 and G'_2). As there is a rule $\mathcal{N}^t(G'''_1 \diamond G'''_2)_x \to \mathcal{N}^t(G'''_1)_y \mathcal{N}^t(G'''_2)_z$ in P, we have $\mathcal{N}^t(G'''_1 \diamond G'''_2) \Rightarrow^*_{\mathcal{G}} \hat{\phi}(G''_1) \diamond \hat{\phi}(G''_2)$. □

Theorem 5.13. Let $\mathcal{G}_*$ be the target plane graph grammar corresponding to a substitutable plane graph language. Algorithm 3 returns a grammar $\mathcal{G}$ from any sample containing $CS(\mathcal{G}_*)$ such that $L(\mathcal{G}) = L(\mathcal{G}_*)$.

Proof:
If $G \in L(\mathcal{G}_*)$ then there exists $N \in \mathcal{A}_*$ such that $N \Rightarrow^*_{\mathcal{G}_*} G$. By Lemma 5.12, it implies that there exists a subgraph G' of the learning sample and a plane graph G'' such that $G' \in L(\mathcal{G}_*), \mathcal{N}^t(G') \Rightarrow^*_{\mathcal{G}} G''$ and $G'' \cong_p G$. By construction of the grammar, $\mathcal{N}^t(G') \in \mathcal{A}$ and thus $G \in L(\mathcal{G})$. Therefore $L(\mathcal{G}_*) \subseteq L(\mathcal{G})$. Due to Theorem 5.11, we have $L(\mathcal{G}) = L(\mathcal{G}_*)$. □

[3]The thickness of a grammar is the size of the largest element in the set of the smallest elements that can be generated by each non-terminal.

6. Discussion

In addition to substitutability, other restrictions on the learned class have been done, explicitly or not. First, the grammar formalism implies that the number of nodes of the outer face of any generated PGS has to be bounded: otherwise an infinite number of axioms would be needed. Then, the requirement of having a polynomial rank, that is used both for efficient parsing and for the polynomial computation time of the learning algorithm, is clearly restrictive.

Despite all these issues, this paper describes, to our knowledge, one of the first positive formal learning result for a non-trivial class of graph grammars. The work on substitutable string languages [5] has been the starting point of several positive learning results on more complex classes, and similar developments are likely to be tractable for plane graph languages. It seems to be the case for instance of the extension to contexts with several holes [39] using multiple context-free grammars (that are a context-sensitive formalism with a polynomial time parsing algorithm). It might also be possible to adapt the learning algorithm in a way that allows a learning result in the PAC paradigm [35]. Finally, due to the interest of planar graphs in image processing [34], it is likely that the learning of plane graph grammars, and more generally grammatical inference techniques, could be used to tackle image classification tasks.

References

[1] Bailly R, Denis F, Rabusseau G. Recognizable Series on Hypergraphs, Proc. 9th Intern. Conf. on Language and Automata Theory and Applications (LATA'15), LNCS 8977, 2015, pp. 639–651. doi:10.1007/978-3-319-15579-1_50.

[2] Bohl E, Terraz O, Ghazanfarpour D. Modeling fruits and their internal structure using parametric 3Gmap L-systems, The Visual Computer, 2015;31(6-8):819–829. doi:10.1007/s00371-015-1108-9.

[3] Brijder R, Blockeel H. On the inference of non-confluent NLC graph grammars, Journal of Logic and Computation, 2011;23(4):799–814. doi: 10.1093/logcom/exr046.

[4] Clark A. Towards General Algorithms for Grammatical Inference, in: 21st International Conference, ALT 2010, Canberra, Australia, October 6-8, 2010. Proceedings, LNCS 6331, 2010, pp. 11–30. Invited Paper. doi:10.1007/978-3-642-16108-7_2.

[5] Clark A, Eyraud R. Polynomial Identification in the Limit of Substitutable Context-free Languages, Journal of Machine Learning Research, 2007(8):1725–1745. Available from: `http://www.jmlr.org/papers/v8/clark07a.html`.

[6] Cook DJ, Holder LB. Graph-Based Data Mining, IEEE Intelligent Systems, 2000;15(2):32–41. doi: 10.1109/5254.850825.

[7] Costa Florêncio C. Identification of Hyperedge-Replacement Graph Grammars, in: Proc. 7th Intern. Workshop on Mining and Learning with Graphs (MLG'09), Leuven, Belgium, 2009, 19–21.

[8] Courcelle B. An axiomatic definition of context-free rewriting and its application to NLC graph grammars, Theoretical Computer Science, 1987;55(2-3):141–181. doi:10.1016/0304-3975(87)90102-2.

[9] Damiand G, Lienhardt P. Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing, A K Peters/CRC Press, 2014. ISBN-1482206528, 9781482206524.

[10] Drewes F, Kreowski HJ, Habel A. Hyperedge Replacement Graph Grammars, in: Handbook of Graph Grammars, World Scientific Publishing Co., Inc., [33], 1997, pp. 95–162. ISBN: 9810228848.

[11] Dupont P, Miclet L, Vidal E. What Is the Search Space of the Regular Inference?, in: Proc. 2nd Inter. Colloquium in Grammatical Inference (ICGI'94), LNCS 862, 1994, 25–37. doi: 10.1007/3-540-58473-0_134.

[12] Engelfriet J. Tree automata and tree grammars, 1975, DAIMI FN-10 (Lecture Notes), Aarhus University.

[13] Eyraud R, Heinz J, Yoshinaka R. Efficiency in the Identification in the Limit Learning Paradigm, in: Topics in Grammatical Inference (J. Heinz, J. M. Sempere, Eds.), Springer-Verlag, 2016, pp. 25-46. doi: 10.1007/978-3-662-48395-4_2.

[14] Eyraud R, Janodet JC, Oates T. Learning Substitutable Binary Plane Graph Grammars, in: Proc. 11th Intern. Colloquium in Grammatical Inference (ICGI'12) vol. 21 of JMLR Worshops and Conference Proceedings, 2012, pp. 114–128. Available from: http://dblp.uni-trier.de/db/journals/jmlr/jmlrp21.html#EyraudJO12

[15] Fáry I. On straight line representation of planar graphs, Acta Univ Szeged. Sect. Sci. Math, 1948;11:229–233.

[16] Fusy E. Combinatoire des graphes planaires et applications algorithmiques (in English), Ph.D. Thesis, Ecole Polytechnique - ParisTech, 2007.

[17] Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979. ISBN: 0716710447.

[18] Gibbons A. Algorithmic graph theory, Cambridge University Press, 1985. ISBN: 9780521288811.

[19] Gold E. Language Identification in the Limit, Information and Control, 1967;10(5):447–474. doi: 10.1016/S0019-9958(67)91165-5.

[20] Harchaoui Z, Bach FR. Image Classification with Segmentation Graph Kernels, in: Proc. 13th Conference on Computer Vision and Pattern Recognition (CVPR'07), IEEE Computer Society, 2007. doi: 10.1109/CVPR.2007.383049.

[21] de la Higuera C. Characteristic Sets for Polynomial Grammatical Inference, Machine Learning Journal, 1997;27(2):125–138. doi: 10.1023/A:1007353007695.

[22] de la Higuera C, Janodet JC, Samuel E, Damiand G, Solnon C. Polynomial Algorithms for Open Plane Graph and Subgraph Isomorphisms, Theoretical Computer Science, 2013;498:76–99. Available from: http://dx.doi.org/10.1016/j.tcs.2013.05.026.

[23] Huan J, Wang W, Prins J. Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism, in: Proc. 3rd Intern. Conf. on Data Mining (ICDM'03), IEEE Computer Society, 2003, pp. 549–552. ISBN: 0-7695-1978-4.

[24] Jeltsch E, Kreowski HK. Grammatical Inference Based on Hyperedge Replacement, in: Proc. 4th International Workshop on Graph Grammars and their Application to Computer Science, LNCS 532, 1991, pp. 461–474. doi: 10.1007/BFb0017406.

[25] Jonyer I, Holder LB, Cook DJ. MDL-Based Context-Free Graph Grammar Induction, International Journal of Artificial Intelligence Tools, 2004;13:65–79. Available from: http://dx.doi.org/10.1142/S0218213004001429.

[26] Kadri H, Ghavamzadeh M, Preux P. A Generalized Kernel Approach to Structured Output Learning, in: Proc. 30th Intern. Conf. in Machine Learning (ICML'13) vol. 28 of JMLR Workshops and Conference Proceedings, 2013, pp. 471–479. Available from: https://hal.inria.fr/hal-00695631.

[27] Kasprzik A, Yoshinaka R. Distributional Learning of Simple Context-Free Tree Grammars, in: Proc. 22nd Intern. Conf. on Algorithmic Learning Theory (ALT'11), LNAI 6925, 2011, pp. 398–412. doi: 10.1007/978-3-642-24412-4_31.

[28] Kukluk J, Holder L, Cook D. Inference of Edge Replacement Graph Grammars, International Journal on Artificial Intelligence Tools, 2008;17(3):539–554. Available from: http://dx.doi.org/10.1142/S0218213008004047.

[29] Lautemann C. The Complexity of Graph Languages Generated by Hyperedge Replacement, Acta Informatica, 1989;27(5):399–421. doi: 10.1007/BF00289017.

[30] Lienhardt P. N-Dimensional Generalized Combinatorial Maps and Cellular Quasi-Manifolds, International Journal of Computational Geometry and Applications, 1994;4(3):275–324. Available from: http://dx.doi.org/10.1142/S0218195994000173.

[31] Matsuda T, Motoda H, Washio T. Graph-based induction and its applications, Advanced Engineering Informatics, 2002;16(2):135–143. doi: 10.1016/S1474-0346(02)00005-8.

[32] Nagl M. Formal languages of labelled graphs, Computing, 1976;16(1-2):113–137. doi: 10.1007/BF02241984.

[33] Rozenberg G, Ehrig H. Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1–3, World Scientific, 1997. ISBN: 98-102288-48.

[34] Samuel E, de la Higuera C, Janodet JC. Extracting Plane Graphs from Images, in: Proc. of 13th Intern. Workshop on Structural and Syntactic Pattern Recognition (SSPR'10), LNCS 6218, 2010, pp. 233–243. doi: 10.1007/978-3-642-14980-1_22.

[35] Shibata C, Yoshinaka R. PAC Learning of Some Subclasses of Context-Free Grammars with Basic Distributional Properties, in: Proc. 24th Intern. Conf. in Algorithmic Learning Theory (ALT'13), LNCS 8139, 2013, pp. 143–157. doi: 10.1007/978-3-642-40935-6_11.

[36] Vishwanathan S, Schraudolph N, Kondor RI, Borgwardt K. Graph Kernels, Journal of Machine Learning Research, 2010;11:1201–1242. Available from: http://dl.acm.org/citation.cfm?id=1756006.1859891.

[37] Whitney H. Non-separable and planar graphs, Proc. of the National Academy of Science, U.S.A., 1931;17(2):125–127. doi: 10.1090/S0002-9947-1932-1501641-2.

[38] Yoshinaka R. Identification in the Limit of (k,l)-Substitutable Context-Free Languages, in: Proc. 9th Intern. Colloquium in Grammatical Inference (ICGI'08), LNAI 5278, 2008, pp. 266–279. doi: 10.1007/978-3-540-88009-7_21.

[39] Yoshinaka R. Efficient learning of multiple context-free languages with multidimensional substitutability from positive data., Theoretical Computer Science, 2011;412(19):1821–1831. doi:10.1016/j.tcs.2010.12.058.

A. Proof of Theorem 2.3

In this appendix, we prove that if a PGS $S = \langle X, E, F, o, \mathcal{D}\rangle$ is valid, then it denotes a plane graph. Let us remark that $\langle X, E\rangle$ is a connected simple graph by assumption. Moreover, Condition (1) of Def. 2.2 ensures that the boundary of every face is made of well-defined edges. Typically, no face can be described with a boundary like $[xx\ldots]$, and if a boundary like $[xy\ldots]$ appears, then $\{x, y\}$ is an edge. Notice that sets X and E are redundant in a valid PGS, since we can compute them from the boundaries of the faces.

Now, in order to prove Theorem 2.3, we are going to show that a valid PGS actually defines a 2D-combinatorial map $M = \langle D, \alpha, \beta\rangle$, thus a structure such that D is a finite set of *darts*, $\alpha : D \to D$ is a permutation over D (that is, a one-to-one mapping) and $\beta : D \to D$ is an involution over D (that is,

a one-to-one mapping such that $\beta \circ \beta(d) = d$ for all $d \in D$). The semantics of combinatorial maps and equivalent formalism's such as generalized maps has been pioneered by Lienhardt [30].

Let us first define the set of darts as follows: $D = \{\vec{xy} : \exists f \in F, \exists u \in X^*, \mathcal{D}(f) = [xyu]\}$. We now define mapping $\beta : D \to D$ by declaring $\beta(\vec{xy}) = \vec{yx}$. We claim that β is well-defined and obviously an involution over D. Indeed, if $\vec{xy} \in D$, then there exists a face $f \in F$ such that $\mathcal{D}(f) = [xyu]$ for some $u \in X^*$. By Condition (1) of Def.2.2, we deduce that pair $\{x, y\}$ is an edge. So by Condition (2), there exists a face $f' \in F$ such that $\mathcal{D}(f') = [yxv]$ for some $v \in X^*$, thus $\vec{yx} \in D$.

The definition of $\alpha : D \to D$ is a bit more intricate. Consider a dart $\vec{xy} \in D$. Then there exists an unique face $f \in F$ such that $\mathcal{D}(f) = [xyu]$ for some $u \in X^*$. We can assume without loss of generality that $|u| > 0$. Otherwise, face f is bounded by a single edge, and as the graph $\langle X, E\rangle$ is connected, this graph is actually reduced to a single edge, thus the property straightforwardly holds. Hence, we assume that there exists $z \in X$ such that $\mathcal{D}(f) = [xyzu]$ for some $u \in X^*$, and we set $\alpha(\vec{xy}) = \vec{yz}$.

We claim that α is a permutation over D. Firstly, any dart $\vec{xy}$ has a unique image by α. Indeed, suppose that $\alpha(\vec{xy}) = \vec{yz}$. By Condition (2) of Def.2.2, there exists an unique face $f \in F$ such that $\mathcal{D}(f) = [xyzu]$ for some $u \in X^*$. So if $\vec{xy}$ was α-sewn with another dart $\vec{yz'}$, then we would have $\mathcal{D}(f) = [xyzuxyz'v]$ for some $u, v \in X^*$. By Condition (3), we have $z = z'$ in this case.

Secondly, α is injective: suppose that $\alpha(\vec{zx}) = \vec{xy}$ and $\alpha(\vec{z'x}) = \vec{xy}$. By Condition (2) of Def. 2.2, there exists an unique face $f \in F$ such that $\mathcal{D}(f) = [xyzu]$ for some $u \in X^*$. If both $\vec{zx}$ and $\vec{z'x}$ are α-sewn with dart $\vec{xy}$, then we have $\mathcal{D}(f) = [zxyuz'xyv]$ for some $u, v \in X^*$. By Condition (3), we also have $z = z'$ in this case.

Finally, thanks to Condition (4) of Def. 2.2, the genus [9] of map M is null, so it can be embedded on a sphere with no crossing edges. Moreover, the fact that we distinguish face o as the external face in the PGS allows us to (1) eliminate this face from the map and (2) continuously deform the sphere into a plane so that we finally get a plane graph (remember that a plane is isomorphic to a sphere minus a point). Note that if Condition (4) does not hold, then a PGS may denote a drawing with no crossing edges on a surface whose genus is > 0, thus a torus with $k > 0$ holes.