Université d'Aix-Marseille

Ecole Doctorale de Mathématiques et Informatique

Habilitation à Diriger des Recherches

Spécialité : Informatique

par

Rémi Eyraud

Grammatical Inference:

Learning Computational Models from Various Types of Structured Data

Préparée au
Laboratoire d'Informatique et Systèmes

Equipe Qarma

soutenue le 16 juillet 2019

Jury:

Dr.	Cécile Capponi	Aix-Marseille University	Member
Pr.	François Denis	Aix-Marseille University	President
Pr.	Makoto Kanasawa	Hosei University	Member
Pr.	Jacques Nicolas	INRIA-IRISA	Reviewer
Pr.	Marc Sebban	University of Saint-Etienne	Reviewer
Pr.	Marc Tommasi	University of Lille	Reviewer

Acknowledgments

Thanks everyone :-)

Contents

Part I

Introductions

Chapter 1

On Grammatical Inference and Machine Learning

Contents

In this chapter, I briefly describe the context of the work presented in this manuscript. I start by interrogating the notion of learning (Section 1.1), in particular the links between machine learning and human learning. I then introduce the field at the core of this document: grammatical inference (Section 1.2). Section 1.3 discusses the notion of identification and its importance in contrast with approximation. Finally, I give an overview of the different works regrouped in this document in Section 1.4.

1.1 Machine learning and human learning

The question of the comparison of machine and human intelligence is a topical issue: generalist press, for instance, is nowadays full of articles wondering when machines will outperform humans, some even pretending it already happened.

It is important to notice that most of the examples given to support such a claim come from machine learning: a particular software being better at detecting tumors than clinicians [Esteva et al., 2017], an "artificial intelligence" beating the world champion of the go game [Silver et al., 2016], a self-driving car having less accident [Sparrow and Howard, 2017], etc.

However, these articles usually keep quiet the inherent difference between humans and machines: they do not share the same model of computation. For instance, the success of tumor detection software is due to the capability of machines to access more medical data in few hours that a radiologist or a dermatologist can do in his lifetime. This is just a recent example of the distinction between cognitive science and cybernetic.

Once this distinction is clearly established, one can follow three different paths:

1. Use machines unique abilities to provide new tools to tackle some tasks,
2. Use machines for comparative purpose in order to better understand the way the human brain works,
3. Use the machine to model the computation of human brain.

Even if some terms, like artificial neural networks [McCulloch and Pitts, 1943], could create confusion, statistical machine learning usually relies on the first approach. Indeed, all recent successes depend of the particularities of computers and have few to do with how humans handle a particular task. For instance, the three examples given above all succeed thanks to the machines' capacity of dealing with huge data sets.

The second approach is a more rarely followed path. An example of that kind of work is the recent rebuttal of the 'poverty of stimulus' [Clark and Lappin, 2011]. This idea was introduced during the Seventies and was used to support the claim that the ability of young children to handle their native language is innate. It asserts that the difficulty of learning a natural language cannot be overcome given the relatively limited data available in the linguistic environment of a toddler. In particular, several grammatical structures have been shown to be absent of the linguistic experience of an infant, though they are quickly recognized as correct by young children. However, we showed that a simple algorithmic idea allows the learning of these grammatical structures despite their absence in the learning sample [Clark and Eyraud, 2007]. By proving false a previously admitted claim, this work in computer science participate to the debate on how human learn.

Few has been done concerning the third approach: it is a scientific topical issue, at the interface of many fields, from cognitive science to neurology, including computer science and mathematics. Important investments have been secured recently [Amunts et al., 2016] but it is just the beginning of what is likely to be a long term research path.

1.2 What is called grammatical inference

Grammatical Inference (GI) is the field that studies the learnability of computational models [de la Higuera, 2010]. The wide varieties of domains that are interested by this thematic implies that it is known under many different names:

- control theorists refer to it as *system identification* [Ljung, 1998],
- machine learners would rather use *grammar learning* or *grammar induction* [Głowacka et al., 2011] ,

- security researchers are familiar with the term *protocol state fuzzing* [De Ruiter and Poll, 2015],
- model checkers use *model learning* [Vaandrager, 2017]
- computational linguists prefer *grammar induction* [Bisk and Hockenmaier, 2015]
- some papers use the term regular inference [Berg et al., 2005] or regular extrapolation [Hagerer et al., 2002] to denote part of the field
- some theoretical works are regroup under the term *active learning* [Angluin, 1987]

Obviously, these terms regrouped at least partially different contexts: the applications are not the same, the type of data considered may vary, even the global goals can be of different nature. However, they all share the same core motivation: to build a grammatical model from some kind of sequential/structured data.

We consider the term *Grammatical Inference* as the one encompassing all the others, and will thus use it in the course of this document.

1.3 Approximation versus Identification

In Statistical Machine Learning (SML) the central goal is to be able to infer a good approximation of the eventual function that generates the data. In other words, we supposed there exists a probabilistic target function that perfectly explains the data and we try to learn a (simpler) function that is close enough to this target. This is the core of most theoretical works in SML: it is explicit in the bias-variance dilemma, we found it behind the PAC learning paradigm and the studies of generalization error bounds, etc. Even when theory is not involved, we focus on showing how good the learned model is by applying protocols like cross-validation to evaluate the quality of the approximation.

However, this widely accepted goal suffers drawbacks that have rarely been discussed until recently. It is the tremendous successes of machine learning applications that revealed them. To take a practical example, imagine you have developed a software using a ML algorithm that is able to automatically drive a car. You have proven, or convincingly evaluated, that the learned model is (very) good: your accuracy is 99% (or 99.9%, or even 99.99999%). The exact percentage does not matter, the important point is that your model is an approximation. This rate does not mean that in front of a given context (datum), the model will fail 1% of the time. It states that, if this given data is part of the 1% on which the model does not work, it will fail each time it encountered it. What the result says is that this data is (very) unlikely.

Concretely, if the model of the self driving cars fails for instance on a rare combination of bad weather and a particular type of asphalt, all cars encountering this condition

will crash. If this happens on a summer Saturday on the A7 highway in Valley of Rhone, or on the Interstate 95 in Eastern USA the day before Thanksgiving, the consequences would be dramatic.

This topical issue is of great importance since it can dampen the enthusiasm of the industry toward ML by, for instance, slowing down the commercialization of self-driving cars.

There exist few known ways to tackle the problem. A possibility is to characterize the data on which the model fails and to use another one when such data has to be faced. However this is difficult, in particular for models for which few theoretical results exist, like deep artificial neural networks.

A more promising path is to change the goal of learning from the one of approximation to the one of identification. Historically, the idea of exactly identifying the target function has been mined since the early times of machine learning: the work of E. M. Gold in the sixties is a good witness [Gold, 1967]. This approach has been followed mainly by researchers from the theoretical computer science community: its targeted models are the one at the core of this science, from finite state automata (FSA) to Chomsky grammars, including transducers and string rewriting systems, among others.

The (mostly practical) successes of statistical machine learning, headed by researchers whose background is thus in statistics, recently eclipsed the works on identification. For instance, very few papers on identification, to not say none, have been published in major international machine learning conferences the past ten years. A majority of today machine learners enjoy advanced knowledge in statistics but know few about computer science models.

However, despite the founding and publishing difficulties of these works, identification results are still obtained and important breakthrough have been obtained during the past decade. Indeed, complex classes of models have been proven identifiable under various framework [Clark et al., 2010, Freivalds and Zeugmann, 2014, Yoshinaka and Clark, 2010, Shibata and Yoshinaka, 2016, Balle and Mohri, 2018], refinements of the learning paradigms have been studied to make them more practical [Yoshinaka, 2008, Eyraud et al., 2016] (see also Chapter 2), and efficient implementations of major algorithms have been made available [Akram et al., 2010, Isberner et al., 2015, Arrivault et al., 2017]. All of these allowed state of the art results, for instance in software engineering [Meijer and van de Pol, 2018], security [van der Lee and Verwer, 2018], and natural language processing [Quattoni et al., 2017].

1.4 Overview of this document

This document regroups most of the research in grammatical inference, in particular in the context of identification, I had the chance to conduct during the last ten years. These works have always been collaborative, as it is often the case for prospective

scientific research: I detail at the beginning of each chapter the context of the work and describe the collaboration that allows its achievements.

The document is structured as follow:

- This introductory part contains a second chapter (Chapter 2) where the formalization of the notion of *identification* is discussed.
- The second part deals with *distributional learning*, a recently explored path in grammatical inference, and is composed of two chapters: the first one (Chapter 3) focuses on learning from sequences while the second one (Chapter 4) tackles the challenge of learning from graphs.
- The third part concentrates on learning string to string functions. Two paths have been explored: when the domain is known (Chapter 5) and when additional locality constraints are available (Chapter 6). These two works are motivated by linguistic considerations.
- A general conclusion completes the scientific part of this document while annexes provide a curriculum vitae and additional information.

Bibliography

H. Ibne Akram, C. de la Higuera, H. Xiao, and C. Eckert. Grammatical inference algorithms in MATLAB. In *Proc. of the International Conference in Grammatical Inference*, pages 262–266, 2010. 6

K. Amunts, C. Ebell, J. Muller, M. Telefont, A. Knoll, and T. Lippert. The human brain project: Creating a european research infrastructure to decode the human brain. *Neuron*, 92(3):574 – 581, 2016. 4

D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987. 5

D. Arrivault, D. Benielli, F. Denis, and R. Eyraud. Scikit-SpLearn: a toolbox for the spectral learning of weighted automata compatible with scikit-learn. In *Conférence francophone en Apprentissage*, 2017. 6

B. Balle and M. Mohri. Generalization bounds for learning weighted automata. *Theoretical Computer Science*, 716:89–106, 2018. 6

T. Berg, O. Grinchtein, B. Jonsson, M. Leucker, H. Raffelt, and B. Steffen. On the correspondence between conformance testing and regular inference. In *Proc. of the Joint European Conference on Theory and Practice of Software Conference on Fundamental Approaches to Software Engineering*, pages 175–189, 2005. 5

Y. Bisk and J. Hockenmaier. Probing the linguistic strengths and limitations of unsupervised grammar induction. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1395–1404, 2015. 5

A. Clark and R. Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007. 4

A. Clark and S. Lappin. *Linguistic Nativism and the Poverty of the Stimulus*. Wiley-Blackwell, Malden, MA, 2011. URL `http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1405187840.html`. 4

Alexander Clark, Rémi Eyraud, and Amaury Habrard. Using contextual representations to efficiently learn context-free languages. *Journal of Machine Learning Research*, 11: 2707–2744, 2010. 6

Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, New York, NY, USA, 2010. 4

J. De Ruiter and E. Poll. Protocol state fuzzing of tls implementations. In *USENIX Security Symposium*, pages 193–206, 2015. 5

A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017. ISSN 0028-0836. 3

R. Eyraud, J. Heinz, and R. Yoshinaka. *Efficiency in the Identification in the Limit Learning Paradigm*, pages 25–46. Springer Berlin Heidelberg, 2016. ISBN 978-3-662-48395-4. doi: 10.1007/978-3-662-48395-4_2. 6

R. Freivalds and T. Zeugmann. Active learning of recursive functions by ultrametric algorithms. In *Proc. of SOFSEM: Theory and Practice of Computer Science*, pages 246–257, 2014. 6

D. Głowacka, J. Shawe-Taylor, A. Clark, C. de la Higuera, and M. Johnson. Introduction to the special topic on grammar induction, representation of language and language learning. *Journal of Machine Learning Research*, 12:1425–1428, 2011. 4

E. M. Gold. Language identification in the limit. *Information and Control*, 10(5): 447–474, 1967. 6

A. Hagerer, H. Hungar, O. Niese, and B. Steffen. *Model Generation by Moderated Regular Extrapolation*, pages 80–95. Springer Berlin Heidelberg, 2002. 5

M. Isberner, F. Howar, and B. Steffen. The open-source learnlib - A framework for active automata learning. In *Proc. of Computer Aided Verification*, pages 487–495, 2015. 6

L. Ljung. *System Identification*, pages 163–173. Birkhäuser Boston, 1998. 4

W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. ISSN 1522-9602. doi: 10.1007/BF02478259. URL https://doi.org/10.1007/BF02478259. 4

J. Meijer and J. van de Pol. Sound black-box checking in the learnlib. In *Proc. of NASA Formal Methods*, pages 349–366, 2018. 6

A. Quattoni, X. Carreras, and M. Gallé. A maximum matching algorithm for basis selection in spectral learning. In *Proc. of the International Conference on Artificial Intelligence and Statistics*, pages 1477–1485, 2017. 6

C. Shibata and R. Yoshinaka. Probabilistic learnability of context-free grammars with basic distributional properties from positive examples. *Theoretical Computer Science*, 620:46–72, 2016. 6

D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 3

R. Sparrow and M. Howard. When human beings are like drunk robots: Driverless vehicles, ethics, and the future of transport. *Transportation Research Part C: Emerging Technologies*, 80:206 – 215, 2017. 3

F. Vaandrager. Model learning. *Commun. ACM*, 60(2):86–95, 2017. ISSN 0001-0782. 5

W. van der Lee and S. Verwer. Vulnerability detection on mobile applications using state machine inference. In *Proc. of the IEEE European Symposium on Security and Privacy Workshops*, pages 1–10, 2018. 6

R. Yoshinaka. Identification in the limit of k, l-substitutable context-free languages. In *Proc. of the International Conference on Grammatical Inference*, pages 266–279, 2008. 6

R. Yoshinaka and A. Clark. Polynomial time learning of some multiple context-free languages with a minimally adequate teacher. In *Proc. of Formal Grammar*, pages 192–207, 2010. 6

Chapter 2

On the Formalization of Learning

Contents

Context of this work

The first version of this chapter was written few years ago with this *Habilitation à Diriger des Recherches* (HDR) in mind. At the time a global draft was achieved, Jeffrey Heinz suggested to use it in the book he was working on with José M. Sempere.

Given the short deadline, it was decided that Jeffrey himself would assist polishing the document while we asked Ryo Yoshinaka to help, since some of the results presented here are based on his work. This rushy effort leaded to the second version of this chapter, titled *Efficiency in the identification in the limit paradigm* [Eyraud et al., 2016] and published in the book *Topics in Grammatical Inference* [Heinz and Sempere, 2016].

This third version is close to the one already published. However, an effort has been made to smoothly insert this chapter into this HDR: other paradigms are given longer description, non-central proofs are omitted, the bibliography has been extended, etc.

2.1 Introduction

2.1.1 On learning paradigms

Computer Science is a science that can be both fundamental and experimental. Machine Learning does not escape this duality and thus two different paths exist when one is interested in validating an algorithmic idea for a learning algorithm.

On the one hand, the practical approach consists in using the available benchmark data to test the quality of the learning algorithm (employing for instance the well-known cross-validation technique).

On the other hand, a theoretical approach is possible by using a learning paradigm, which is an attempt to formalize what learning means. Such a framework provide a setting to study the behavior of learning algorithms and to formally establish their soundness.

We address here this second path by surveying different learning paradigms in the context of grammatical inference. We first introduce preliminary mathematical definitions that are needed for an adequate formalization and that we will try to use throughout the whole document.

We then discuss the relevance of several paradigms toward the goal of grammatical inference, including the well-known *PAC learning* framework [Valiant, 1984] and the one historically called *Active Learning* [Angluin, 1987] (Section 2.2). The rest of the chapter focuses on the most popular paradigm in GI, the one introduced by Gold [1967] and named the *identification in the limit*, whose initial formalization requires the learner to be fed with an infinite sequence of data and the exact convergence to be achieved in a finite time:

- Section 2.3 studies the limitations of its initial definition when efficiency is considered and first attempts to overcome them. These include requirements based on the running time of the studied algorithm.

- Efficiency requirements depending on the incremental behavior of algorithms in the Gold's paradigm are detailed in Section 2.4. Importantly, a set-driven refine-

ment of the paradigm that allows the emancipation from its asymptotic essence is also given in this section.

- Section 2.5 introduces two recent reformulations of the paradigm.

2.1.2 Preliminary definitions

An *alphabet* Σ is a finite non-empty set of symbols called *letters*. A *string* w over Σ (sometime called a *word*) is a finite sequence $w = a_1 a_2 \dots a_n$ of letters. Let $|w|$ denote the length of w, that is, the number of letters in w. Given a set of strings S, we denote $|S|$ its cardinality and $||S||$ its size, i.e. the sum of $|S|$ with the lengths of the strings S contains[1]. In the following, letters will be indicated by $a, b, c, \dots$, strings by $u, v, \dots, z$, and the empty string, that is the string of length 0, by λ. The set of all strings is denoted Σ^*, corresponding to the free monoid generated by Σ. We denote Σ^+ the set $\Sigma^* \setminus \{\lambda\}$.

We assume a fixed but arbitrary total order $<$ on the letters of Σ. As usual, we extend $<$ to Σ^* by defining the *hierarchical order* [Oncina and García, 1992], denoted by $\lhd$, as follows:

$$\forall w_1, w_2 \in \Sigma^*, w_1 \lhd w_2 \text{ iff } \begin{cases} |w_1| < |w_2| \text{ or} \\ |w_1| = |w_2| \text{ and } \exists u, v_1, v_2 \in \Sigma^*, \exists a_1, a_2 \in \Sigma \\ s.t.\ w_1 = u a_1 v_1, w_2 = u a_2 v_2 \text{ and } a_1 < a_2. \end{cases}$$

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

We extend this order to non-empty finite sets of strings: $S_1 \lhd S_2$ iff $||S_1|| < ||S_2||$ or $||S_1|| = ||S_2||$ and $\exists w \in S_1 \setminus S_2$ such that $\forall w' \in S_2 \setminus S_1$, $w \lhd w'$. For instance $\{a\} \lhd \{\lambda, a\}$ and $\{a, b\} \lhd \{aaa\}$.

By a language L over Σ we mean any set $L \subseteq \Sigma^*$. Many classes of languages were investigated in the literature. In general, the definition of a class $\mathbb{L}$ relies on a class $\mathbb{R}$ of abstract machines[2], called *representations* or *computational models*, that characterize all and only the languages of $\mathbb{L}$. The relationship is given by the *naming function* $\mathcal{L} : \mathbb{R} \to \mathbb{L}$ such that: (1) $\forall R \in \mathbb{R}, \mathcal{L}(R) \in \mathbb{L}$ and (2) $\forall L \in \mathbb{L}, \exists R \in \mathbb{R}$ such that $\mathcal{L}(R) = L$. Two representations R_1 and R_2 are *equivalent iff* $\mathcal{L}(R_1) = \mathcal{L}(R_2)$.

Many different classes of representations have been studied in the literature. It is beyond the scope of this chapter to exhaustively list them. However, we introduce the following definition, which is a generalization of some well-known classes of grammars. We will mainly focus on the classes of representations whose characterization can be done in this context.

[1] We define $||S|| = |S| + \sum_{w \in S} |w|$ so that $||\{a\}|| < ||\{\lambda, a\}||$.

[2] This is not strictly necessary: for instance, the substitutable languages [Clark and Eyraud, 2007] have no grammatical characterization.

Definition 2.1 (Generative grammar)**.** $G = \langle \Sigma, N, P, I \rangle$ where Σ is the alphabet of the language, N is a set of variables usually called non-terminals, $P \subset (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$ is the set of generative (production) rules, I is the finite set of axioms, which are elements of $(\Sigma \cup N)^*$.

A generative rule (α, β) is usually denoted $\alpha \to \beta$. It allows the rewritten of elements of $(\Sigma \cup N)^*$ into elements of $(\Sigma \cup N)^*$. Given $\gamma \in (\Sigma \cup N)^*$ we say that a production rule $\alpha \to \beta$ applied to γ if it exists $\eta, \delta \in (\Sigma \cup N)^*$ such that $\gamma = \eta\alpha\delta$. The result of applying this rule on γ is $\eta\beta\delta$. We write $\gamma \Rightarrow \eta\beta\delta$. $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$, and $\Rightarrow_P^*$ is the reflexive and transitive closure of $\Rightarrow$ restricted to the production rules in a given set P.

We define the size of a generative grammar to be the size of the set of its rules, plus the size of its set of axioms: $||G|| = ||I|| + |P| + \sum_{\alpha \to \beta \in P} (|\alpha\beta| + 1)$.

Definition 2.2 (Generated language)**.** Let $G = \langle \Sigma, N, P, I \rangle$ be a generative grammar. $\mathcal{L}(G) = \{w \in \Sigma^* : \exists \alpha \in I \ s.t. \ \alpha \Rightarrow_P^* w\}$.

Example 1. The usual classes of the Chomsky hierarchy are classes of generative grammars. Regular grammars correspond to the restriction $P \subset N \times (\Sigma N \cup \{\lambda\})$, or $P \subset N \times (N\Sigma \cup \{\lambda\})$ by symmetry. The context-free grammars are the ones where $P \subset N \times (\Sigma \cup N)^*$ while the context-sensitive grammars are the ones such that if $\alpha \to \beta \in P$ then $\exists (\gamma, \delta, \eta) \in (\Sigma \cup N)^*$, $A \in N$: $\alpha = \delta A \eta$ and $\beta = \delta\gamma\eta$. If no restrictions are imposed on the rules of the grammar, then the resulting class of representations corresponds to that of the unrestricted grammars. All of these classes were formerly defined with a set of axioms reduced to one element of N [Chomsky, 1956].

Example 2. String Rewriting Systems (SRS) [Book and Otto, 1993] are generative devices where $N = \emptyset$. A rule corresponds to an element of Σ^* rewritten into an element of Σ^* and the set of axioms is made of elements of Σ^*. The language represented by an SRS is the set of strings that can be rewritten using the rules from an element of I.

Some classes of representations that have been studied in grammatical inference are not covered by Definition 2.1. This is the case for instance for multiple context-free grammars [Seki et al., 1991], patterns [Angluin, 1980b], tree [Comon et al., 2007] and graph [Rozenberg, 1997] grammars, etc. While it is not difficult to generalize the definition in order to cover these classes, we conduct the discussion of this chapter in the context of the above definition for concreteness and due to its familiarity.

2.2 PAC learning and other learning paradigms

2.2.1 PAC paradigm

The most known paradigm in machine learning is certainly the Probably Approximately Correct (PAC) criterium [Valiant, 1984] and its refinements [Li and Vitanyi, 1991, Lang-

ford, 2005]. Unlike the identification in the limit paradigm, the PAC framework comes with built-in efficiency requirements so PAC-learners are efficient in important senses. A natural question then is Why work on refinements of the identification in the limit paradigm to add efficiency constraints when the PAC framework can be utilized instead? Evidently, part of the answer is that the PAC learning paradigm concerns approximation: we already discussed in Chapter 1 the limitations of such a goal.

In addition, we argue here that this paradigm is not well-adapted to learning formal languages, as even very simple and well characterized classes of languages are not PAC-learnable [Angluin et al., 2012]. Several theoretical reasons explain this inadequacy, and each of them relates to aspects of the computational models used to represent formal languages.

One of the main reasons is that the VC-dimension of even the simplest models of language representations, namely the finite state automata, is not bounded: the VC-dim of the class restricted to automata of n states is n [Ishigami and Tani, 1997] which makes the whole class not learnable in the PAC sense [Blumer et al., 1989]. Indeed, not even the class of finite languages has finite VC-dimension. This is closely related to the fact that the learning principle of empirical risk minimization [Vapnik, 1995], inherent in most approaches studied under the PAC framework (and more generally in SML), is of little use when formal languages are considered. Indeed, the number of representations *consistent* to a given set of data of a target language, that is to say representations that correctly explain all the data, is often infinite. It is then useless to reduce the hypothesis space to the ones that minimize the error on a given set of data.

Similarly, consider the fact that the PAC paradigm does not suffer from the main drawback of identification in the limit of being asymptotic. Unlike PAC learning, in identification in the limit, there is no guarantee provided about the quality of the hypothesis before the (exact) convergence happens. However this apparent drawback seems to be inherent to the kind of representations of the learning targets considered. Even if two generative grammars have all but one of their rules in common, the languages of these two grammars can be as far apart as one wishes. This problem is inherent to the nature of formal languages and their grammatical representations. This '*Gestalt*-like' property is unavoidable in the formalization of learning: the whole grammar is more than the sum of its rules. In our view, this mainly justifies the use of identification in the limit in the context of grammar learning.

Another reason is that a representation of a formal language is not only a classifier, that is to say a device that defines what is in the language and what is not, but it also gives *structural* information about the elements of the language.

Also, there are concerns that are more independent of the representations. Another particularity of language learning is that a lot of algorithms use only positive examples of a target concept, while the statistical machine learning framework relies on labeled

data[3].

Finally, PAC learning is particularly pertinent in the case of statistical models, where the probability of making a mistake can be evaluated using the hypothesis. This particular attribute of the PAC paradigm is of less value when non-stochastic model learning is of interest. But even while grammatical inference is concerned with learning probability distributions over strings, the power of the considered models makes the paradigm be inadequate: there are for instance infinitely many structurally different probabilistic context-free grammars that define the same set of distributions [Hsu et al., 2013].

To be complete, some positive learning results exist in restrictive versions of the PAC-paradigm, mostly in the case where the target distribution is known to be drawn using a given class of stochastic grammars [Denis and Gilleron, 2001], and with additional restrictions that allow to distinguish the different parts of the target from any sample (see the work of Ron et al. [1995], Clark and Thollard [2004], Shibata and Yoshinaka [2016] for examples).

2.2.2 Active Learning

In the *Active Learning* paradigm, also known as *Query Learning*, *Exact Learning*, or *interactive learning*, the learner interacts with an oracle. The aim is to identify a computational model by querying the oracle in order to get data corresponding to this target.

A wide range of types of queries has been investigated, the two most known being

- the Membership Queries (MQ) [Moore, 1956] where the oracle answers whether a given string belongs to the language or not.
- the Equivalence Queries (EQ) [Angluin, 1987] that allows the learner to know if its current hypothesis is the target one.

In this framework, efficiency is measured by the number of queries the algorithm needs to converge to a hypothesis equivalent to the target: the aim is to design an algorithm that requires a number of queries polynomial in the size of the target model. de la Higuera [2016] offers a recent and complete overview of this paradigm.

Numerous positive learning results have been provided in this context, among which we can cite for instance: Angluin [1987] proved that regular languages are learnable under this paradigm from a *Minimum Adequate Teacher*, that is to say an oracle that answers membership and equivalence queries; The learning of Mealy machines and even register machines have been successfully investigated [Isberner et al., 2014] in the context of model checking; Subclasses of context-free languages [Clark, 2010a] and even beyond context-free [Yoshinaka and Clark, 2010] have been proven learnable.

[3] In SML terms, this framework corresponds to one-class unsupervised learning.

Although this paradigm can be of practical interest as the works on model checking and software specifications have shown, and though they can also be motivated by the study of first language acquisition [Chater et al., 2015], the requirement of an oracle clearly reduces the practical interest of an algorithm investigated in this framework. Indeed, the usual machine learning framework requires the existence of a learning sample but does not provide an oracle[4].

2.2.3 Other learning paradigms

There exist other less studied learning frameworks which eschew identifiability in the limit in order to incorporate notions of computational efficiency. The aim here is not to give an exhaustive list of such paradigms: we just want to give pointers to the main ones.

The first one we would like to describe is derived from the active learning one: the learning algorithm have access to an oracle that answers membership queries together with a finite sample of examples of the language to identify. Pao and Carr III [1978] introduced it in the Seventies, but its success is recent. Indeed, several algorithms have been proposed in this framework during the last decade, mostly in what is called *distributional learning* [Clark, 2010b, Yoshinaka and Kanazawa, 2011, Clark and Yoshinaka, 2014a,b, Kanazawa and Yoshinaka, 2017]. This is also the paradigm used for the work presented in Chapter 3 of this document.

Another learning paradigm that can be used to study GI algorithms is the one of *stochastic finite learning* [Zeugmann, 2003]. In this framework, an algorithm is said to have learned a language if, from any infinite sequence of data of this language drawn from a probability distribution, it stops after having seen a finite number of elements and its hypothesis at that point is correct with high probability. The expected number of examples that the learner needs before convergence forms a measure of the algorithm's efficiency. This approach is similar in its aims to identification in the limit, but it can also be seen as a probably exactly correct paradigm. It is thus a tempting way to fill the gap between PAC-learning and identification in the limit. However, results in this paradigm are hard to obtain and even simple classes of languages are known to be not learnable. In addition, many of the arguments of the previous section on the PAC-paradigm work can be used for this formalization. On the other hand, there are positive results for subclasses of pattern languages [Zeugmann, 2006].

We believe the reasons above, or some combination thereof, have led many scholars to seek a way to incorporate efficiency into the identification in the limit paradigm (as opposed to abandoning the paradigm altogether).

[4]Notice nevertheless that the initial formulation of the PAC-learning criterium required a membership oracle [Valiant, 1984].

2.3 The limits of Gold's paradigm

2.3.1 The importance of efficiency in learning

Gold [1967] introduced in the 1960s a definition of learning called *identification in the limit*, which works as follows. An algorithm is fed with an infinite sequence of data exemplifying a target language. When a new element is given to the algorithm, it may output a hypothesis. The algorithm identifies the language in the limit if for any possible sequence of data for this language, there exists a moment from when the algorithm does not change its hypothesis, and this hypothesis is a correct representation of the target language. When a whole class of languages is considered, the algorithm identifies the class in the limit if it can identify all languages of the class.

The fact that the convergence is required to hold whatever the sequence of data is what makes this paradigm be adversarial [Clark and Lappin, 2011]. This worst-case scenario principle strengthens the value of any algorithmic idea that yields an identification in the limit result for a class of languages [Heinz, 2014].

However, Gold's formulation can be of little help for practical purposes, when one wants to study a learning idea with the aim of applying it to real-world data. This is mainly due to the fact that no efficiency property is required and thus one can assume infinite time and space. This is the reason why several refinements of Gold's model which add polynomial bounds to the requirements of the paradigm have been developed. The purpose of this chapter is to comprehensively review the proposed refinements and the challenges they face. Main results of each approach, along with comparisons, are provided.

2.3.2 Identification in the limit

We now provide a detailed formalization of the identification in the limit paradigm.

A presentation P of a language L is an infinite sequence of data corresponding to L. We note $P[i]$ the i^{th} element of P and P_i the set of the i^{th} first elements of P. If the data contains only elements of L then the presentation is called a *text* of language L. A text T is a complete presentation of L iff for all $w \in L$ there exists $n \in \mathbb{N}$ such that $T[n] = w$. If data in the presentation are instead pairs (w, l), such that $w \in \Sigma^*$ and l is a Boolean valued TRUE if $w \in L$ and FALSE otherwise, then the presentation is called an *informant*. An informant I is a complete presentation of L iff for all $w \in \Sigma^*$ there exists $n \in \mathbb{N}$ such that $I[n] = (w, l)$. In the rest of the chapter, we will only consider complete presentations.

A learning algorithm in this context, sometimes called an *inductive inference machine*, is an algorithm that takes as input larger and larger initial segments of a presentation and outputs, after each input, a hypothesis from a pre-specified hypothesis space.

Definition 2.3 (Identification in the limit [Gold, 1967])**.** A class $\mathbb{L}$ of languages is *identifiable in the limit (IIL)* from text (resp. from informant) if and only if there exists a learning algorithm $\mathfrak{A}$ such that for all language $L \in \mathbb{L}$, for all text T (resp. informant I) of L,

- there exists an index N such that $\forall n \geq N$, $\mathfrak{A}(T_n) = \mathfrak{A}(T_N)$ [resp. $\mathfrak{A}(I_n) = \mathfrak{A}(I_N)$]
- $\mathcal{L}(\mathfrak{A}(T_N)) = L$ [resp. $\mathcal{L}(\mathfrak{A}(I_N)) = L$]

Angluin [1980a] characterizes exactly those classes of languages that are identifiable in the limit from text. The central theorem in this work refers to the presence of 'telltale' finite subsets for each language in the class. Later, in section 2.4.2, we will see an efficiency bound in terms of 'characteristic' finite subsets of languages (these are not exactly the same as Angluin's telltale subsets).

Gold [1967] established three important results in this paradigm. The first is that the class of all finite languages is identifiable in the limit from text. The second is that no superfinite class of languages can be identified in the limit from text. Despite what the name may evoke, a class of languages is *superfinite* if it contains all finite languages and at least one infinite language (the class contains thus an infinite number of languages). The third is that any computably enumerable class whose uniform membership problem is decidable[5] is identifiable in the limit from informant.

The proof of the second result relies on the fact that given a presentation of an infinite language L, there does not exist any index N from which a learner can distinguish the finite language made of the strings seen so far and the infinite language. If the algorithm converges to L on a complete text T for L at N then there is a text for the finite language containing all and only the strings in T_N for which the algorithm will *also* converge to L. Hence the algorithm fails to identify this finite language in the limit.

On the other hand, the learning algorithm for the third result (learning any computably enumerable class with informant) is really naive: it enumerates the elements of the class until it finds the first one consistent with the information so far. In other word, the algorithm always conjectures the first language in the enumeration that accepts all positive examples (labeled TRUE) and rejects all negative ones (labeled FALSE). If it is the correct hypothesis, the algorithm has converged. If not, then there will be an example later in the presentation that will be inconsistent with the current hypothesis and consequently the algorithm will move along down the enumeration to the next consistent language.

This third result, though of positive nature, is one of the main reasons that the identification in the limit paradigm needs to be refined to include a notion of tractability.

[5]The uniform membership problem is the one where given a string and a representation one needs to determine whether the string belongs to the represented language.

'Learning by enumeration' is clearly not tractable and thus is of little use. While it meets the letter of the definition of learning, it violates our intuitions of what learning should be like. At first glance, a natural way to exclude such learning 'solutions' is to add a tractability requirement to the definition in some way. However, as we now discuss, this is more difficult than it may initially appear.

For more on variations of Gold's original paradigm see the chapter *General Highlights since Gold* by Case [2016] in the book *Topics in Grammatical Inference* [Heinz and Sempere, 2016].

2.3.3 Polynomial time

Given the limitations of IIL shown in the previous section, designing requirements to add to the paradigm is needed to strengthen the validity of learning ideas. An intuitive way to deal with that is to constrain the time allowed for the algorithm to make its computations.

Limiting the overall running time appears inappropriate since languages may have infinite cardinality and concomitantly there is no bound on the length of the strings. Thus for any polynomial function p, infinite language L, and number n, there is a presentation P for L such that the first element of P is larger than $p(n)$. Stochastic finite learning [Zeugmann, 2003] would be of great interest to readers concerned with this problem since it replaces this worst-case scenario with a learning framework that focuses on expected convergence (where presentations are drawn according to probability distributions).

A more consensual requirement is *update-time efficiency*. An algorithm is update-time efficient if it outputs a new hypothesis in time polynomial in the size of the data seen so far. This is reasonable as far it goes. Unfortunately, this requirement turns out to be no real restriction at all.

In a seminal paper Pitt [1989] shows that update-efficiency is not sufficient to prove the validity of a learning approach. Indeed, using a method now known as Pitt's trick, he proves that any algorithm that can identify a class in the limit can be transformed into an algorithm that identifies the class in the limit and is update-time efficient.

Informally the proof relies on the fact that, given a presentation P, if a learner converges to a correct hypothesis on the initial sequence P_i, a variant can delay the computation of any interesting hypothesis until having seen P_j $(j > i)$ such that the computation time of the initial learner on P_i is polynomial in $||P_j||$. This variant of the learning algorithm then has an efficient update-time while also fulfilling the conditions of identification in the limit. Pitt's trick essentially trades time for data so that enforcing tractability in this way has no impact. The set of classes of languages identifiable in the limit without the update-time requirement is exactly the same as the set of classes of languages identifiable in the limit with it.

Pitt's trick reveals that algorithms may be able to efficiently output hypotheses, but convergence can only occur after non-reasonable amounts of *data* have been provided. This lessens the practical utility of the theoretical results provide when real data is taken into account.

One may wonder if one can prohibit Pitt's trick, which ignores the great part of the given data, by forcing a learner to respect all the given data. Case and Kötzing [2009] show that apparently reasonable properties to force a learner to take all the examples into account are not strong enough to prevent Pitt's trick actually when learning from text.

2.3.4 Identification of a language and the size of a target representation

Despite the problem described in the previous section, the requirement of polynomial update-time is still desirable. Efforts have been made to enrich the paradigm further such that Pitt-style delaying tricks are not possible.

Most additional requirements are based on the same method: they link the behavior of the algorithm to the size of a representation of the target language. Indeed, though the identification of the target language is required, a polynomial bound cannot be established with respect to the size of the language since non-trivial classes of languages often contain an infinite number of infinite languages. A representation of finite size of the target language is thus needed. Choosing a target representation also focuses the attention on the hypothesis space of the algorithm, which is relevant from a machine learning standpoint.

However, the choice of representations is not central at all in Gold's learning paradigm as a learner's hypotheses can converge to an arbitrary one among equivalent representations for the correct language. The apparent consequence is that the choice of a representation class for a target language class does matter when taking the representation size into account.

But this duality between the identification of a language and an efficiency bound on the size of a target representation has consequences that need to be handled carefully. For example, it is well known that a non-deterministic finite automaton can be exponentially smaller than the smallest deterministic finite automaton accepting the same language. A learning algorithm that behaves efficiently with respect to the size of deterministic finite automata may not be admitted as an efficient algorithm in terms of the size of non-deterministic finite automata. The reader is referred to the chapter *On the inference of finite state automata from positive and negative data* [López and García, 2016] for details on this question.

In general, an inefficient learner can be seen as an efficient learner by choosing a class of redundant representations. Therefore, it is important to make it clear under

which class of representations the efficiency of a learner is discussed.

In principle, the choice of a representation class is arbitrary and seems hard to justify, but in practice there exist orthodox or natural representations for target language classes. For example, minimal deterministic (canonical) finite state automata are widely used to represent regular languages. Since they are uniquely determined based on an algebraic property of regular languages, there is no room to inflate the representation size.

An intuitive way to deal with the duality exposed above would be to define a paradigm where identification is on a target representation and not on a language. The formalization of this idea is known as *strong identification* [Clark, 2014]. However, this approach only makes sense for classes of representations where each language admits a unique representative: otherwise, it is impossible for any algorithm to distinguish between the different grammars generating the same language, and thus the identification cannot succeed. The use of canonical finite-state automata in the work on regular languages [Oncina and García, 1992] is an example of such an approach.

2.4 First refinements

2.4.1 Mind changes and implicit errors of prediction

One way to formalize the notion of convergence with a reasonable amount of data with respect to the size of the representation is to measure the number of *mind changes* [Blum and Blum, 1975, Ambainis et al., 1999]. Another way is to measure the number of *implicit prediction errors* [Pitt, 1989].

A *mind change* occurs when a learning algorithm replaces its current hypothesis with another. Then one adds to the identification in the limit paradigm the requirement that the number of mind changes made before convergence must be bounded by a polynomial function in the size of the representation.

However, Pitt [1989] presents another trick where the algorithm postpones changing its mind solely to meet the requirements of the mind change bound. Consequently, the algorithm maintains untenable hypotheses (ones inconsistent with the data) until a sufficient amount data is seen so that a mind change can occur without violating the polynomial bound on the number of mind changes.

Measuring implicit predictions errors can get around this trick when learning from an informant. When the learner's current hypothesis is inconsistent with a new datum, it is called *an implicit error of prediction*. Then one adds to the identification in the limit paradigm the requirement that the number of times the current hypothesis is in contradiction with the new example has to be polynomial in the size of the target representation. More formally:

Definition 2.4 (Identification in Polynomial Number of Implicit Errors)**.**

- Given a presentation P, an algorithm $\mathfrak{A}$ makes an implicit error of prediction at step n if the language of the hypothesized target $\mathfrak{A}(P_n)$ is in contradiction with $P[n+1]$.
- A class $\mathbb{G}$ of representations is polynomial-time identifiable in the limit in Pitt's sense if $\mathbb{G}$ admits a polynomial time learning algorithm $\mathfrak{A}$ such that for any presentation of $\mathcal{L}(G)$ for $G \in \mathbb{G}$, $\mathfrak{A}$ makes implicit errors of prediction at most polynomial in $||G||$ [Pitt, 1989].
- A class $\mathbb{G}$ of representations is polynomial-time identifiable in the limit in Yokomori's sense if $\mathbb{G}$ admits a polynomial time learning algorithm $\mathfrak{A}$ such that for any presentation P of $\mathcal{L}(G)$ for $G \in \mathbb{G}$, for any natural number n, the number of implicit errors of prediction made by $\mathfrak{A}$ on the n^{th} first examples is bounded by a polynomial in $m \cdot ||G||$, where $m = \max\{|P[1]|, \ldots, |P[n]|\}$ [Yokomori, 1995].

Notice that Yokomori's formulation is a relaxed version of that of Pitt's.

However, if the presentation is a text, there is yet another unwanted Pitt-style delaying trick: the algorithm can output a representation for Σ^*, which will never be in contradiction with the data. It can then wait to see enough examples before returning a pertinent hypothesis.

On the other hand, if the presentation is an informant, then the additional requirement limiting the number of implicit prediction errors is significant because there is no language like Σ^* which is consistent with both the positive and negative examples. Consequently, it can be shown that not all classes of languages identifiable in the limit in polynomial update time are identifiable in the limit in Pitt's sense or in Yokomori's sense: in the former paradigm, an algorithm working in polynomial time can change its hypothesis an exponential number of times before convergence, while in the latter paradigms this is not allowed and cannot be circumvented as in the case of texts. Note this is different from the mind-change requirement, where the delaying trick there works in both kinds of presentations: in that case, the algorithm can choose to not update its hypothesis when a new example contradicts it.

Another property of these requirements is that they are mainly designed for incremental algorithms. Indeed, these paradigms give a lot of importance to the sequence of data, in particular as the parts of two sequences that contains the same elements in a different order might not correspond to the same number of implicit errors (or mind changes). This forces the complexity analysis to consider particularly malevolent sequences of data. However, in many practical frameworks, for instance in Natural Language Processing or Bio-Informatics, we are interested by algorithms that work from a finite *set* of data, where the order of presentation is irrelevant. From this perspective, the (inadvertent) focus on an incremental process appears to be a drawback.

The main positive learning results using this approach concerns the class of very simple languages [Yokomori, 2003, Yoshinaka, 2009]: an algorithm has been designed

that fulfills the requirements of Yokomori's formulation of the paradigm. This class of languages is incomparable with the class of regular languages and contains context-free languages.

2.4.2 Characteristic Sample

The most widely used definition of data efficiency relies on the notion of *characteristic sample*. The characteristic sample is a finite set of data from a language L that ensures the correct convergence of the algorithm on any presentation of L as soon as it is included in the data seen so far. For some, these characteristic samples evoke Angluin's telltale subsets [Angluin, 1980a], also of finite size, which were central to characterizing the nature of classes of formal languages identifiable in the limit from text.

In this learning paradigm [de la Higuera, 1997], it is required that the algorithm need a characteristic sample whose size[6] is polynomial in the size of the target representation. Formally:

Definition 2.5 (Identification in the limit in polynomial time and data)**.** A class of languages $\mathbb{L}$ is *identifiable in the limit in polynomial time and data* from a class $\mathbb{R}$ of representations iff there exist a learning algorithm $\mathfrak{A}$ and a polynomial $p()$ such that for any language $L \in \mathbb{L}$, for any representation $R \in \mathbb{R}$ of L:

- $\mathfrak{A}$ has a polynomial update-time,
- there exists a set of data CS, called a characteristic sample, of size at most $p(||R||)$ such that for any presentation P of L, if $CS \subseteq P_n$ then $\mathfrak{A}(P_n)$ is equivalent to R, and for all $N > n$, $\mathfrak{A}(P_N) = \mathfrak{A}(P_n)$

The idea underlying the paradigm is that if the data available to the algorithm so far does not contain enough information to distinguish the target from other potential targets then it is impossible to learn. This complexity requirement diverges from update-time requirements above in that incremental learning algorithms no longer sit at the core of the paradigm. Indeed, limiting the complexity in terms of the characteristic sample makes possible the set-driven definition that we are developing below.

Definition 2.6 (Set-driven framework)**.** Let $\mathbb{L}$ be a class of languages represented by some class $\mathbb{R}$ of representations.

1. A *sample* S for a language $L \in \mathbb{L}$ is a finite set of data consistent with L. A *positive sample* for L is made only of elements of L. A positive and negative

[6] The size of a sample is the sum of the length of its elements: it has been shown [Pitt, 1989] that its cardinality is not a relevant feature when efficiency is considered, as it creates a risk of collusion: one can delay an exponential computation on a given sample of data and wait for a sufficient number of examples to run the computation on the former sample in polynomial time in the size of the latter.

sample for L is made of pairs (w, l), where l is a boolean such that $l = \text{TRUE}$ if $w \in L$ and $l = \text{FALSE}$ otherwise. The *size* of a sample S is the sum of the size of all its elements plus $|S|$.

2. An $(\mathbb{L}, \mathbb{R})$-learning algorithm $\mathfrak{A}$ is a program that takes as input a sample for a language $L \in \mathbb{L}$ and outputs a representation from $\mathbb{R}$.

We can now formalize the notion of characteristic sample in the set-driven approach.

Definition 2.7 (Characteristic sample)**.** Given an $(\mathbb{L}, \mathbb{R})$-learning algorithm $\mathfrak{A}$, we say that a sample CS is a *characteristic sample* of a language $L \in \mathbb{L}$ if for all samples S such that $CS \subseteq S$, $\mathfrak{A}$ returns a representation R such that $\mathcal{L}(R) = L$.

Hopefully it is evident that the class of representations is especially relevant in this paradigm.

The learning paradigm can now be defined as follows.

Definition 2.8 (Set-driven identification in polynomial time and data [de la Higuera, 1997])**.** A class $\mathbb{L}$ of languages is *identifiable in polynomial time and data (IPTD)* from a class $\mathbb{R}$ of representations if and only if there exist an $(\mathbb{L}, \mathbb{R})$-learning algorithm $\mathfrak{A}$ and two polynomials $p()$ and $q()$ such that:

1. Given a sample S of size m for $L \in \mathbb{L}$, $\mathfrak{A}$ returns a consistent hypothesis $H \in \mathbb{R}$ in $\mathcal{O}(p(m))$ time ;

2. For each representation R of size k of a language $L \in \mathbb{L}$, there exists a characteristic sample of L of size at most $\mathcal{O}(q(k))$.

Notice that the first item is a reformulation of the polynomial update time requirement, which is now in terms of the size of the sample. The second item corresponds to the additional requirement that the amount of data needed to converge is computationally reasonable. By forcing the algorithm to converge to a correct hypothesis whenever a characteristic sample of reasonable size has been seen, this paradigm tackles the risk of collusion by forbidding Pitt's delaying tricks.

The main reason this unusual way to formalize identification is chosen is because formalizing learning when a set of data is available corresponds to the most common framework when real-world data is considered.

Furthermore, the set-driven approach encompasses the incremental approach since any algorithm studied in latter can easily be cast into a set-driven one. In other words, any algorithm that learns a class of languages in the sense of Definition 2.5 also learns the class in the sense of Definition 2.8.

However, it is not easy to cast set-driven learners into incremental ones. Naively one may believe that for any algorithm $\mathfrak{A}$ satisfying Definition 2.8, there exists an incremental algorithm which satisfies Definition 2.5. The idea would be, for each new

data, to run $\mathfrak{A}$ on the set of data seen so far. However, as shown in the Appendix, this simple approach will not always work. There is an algorithm for learning the substitutable context-free languages which satisfies definition 2.8 for which this incremental construction fails. In the appendix of the chapter of Eyraud et al. [2016], it is shown that unless this incremental algorithm $\mathfrak{A}$ is conservative[7], $\mathfrak{A}$ will not converge to a single grammar. However, if $\mathfrak{A}$ is conservative then there is a presentation at a point of which the characteristic set is seen but $\mathfrak{A}$ has not yet converged to the correct grammar. It remains to be seen whether for every set-driven learner satisfying Definition 2.8, there is an incremental learner satisfying Definition 2.5.

Main results

Many learning algorithms have been studied in the context of IPTD. The main positive results concern approaches that use positive and negative examples as input. In this context, regular languages are learnable [de la Higuera, 1997] using deterministic finite state automata, and so are deterministic even linear languages as the question of inferring this class of context-free grammars can be reduced to the one of inferring deterministic finite state automata [Sempere and García, 1994]. Another related class of languages that has been positively investigated in this context is the one of deterministic linear languages [de la Higuera and Oncina, 2002]. The algorithms is fed with positive and negative examples and outputs a deterministic linear grammar.

Context-free languages that are representable by delimited, almost non-overlapping string rewriting systems are also IPTD-learnable [Eyraud et al., 2007] from positive and negative examples. Comparisons of this class with the previous ones are difficult since they are not defined using the same kind of representation.

The whole class of context-free languages is learnable in the IPTD sense [Sakakibara, 1992, de la Higuera, 2010] from structural positive examples, that is to say derivation trees with no information on the internal nodes. Given a positive integer k, the target class of representation is the one of k-reversible context-free grammars [Oates et al., 2002] and the elements of the sample have to correspond to derivation trees of these grammars.

Limitations

We have already discussed one drawback to measuring the complexity of the learning problem in terms of the size of the representation: it can be unclear what counts as a 'reasonable' representation. Consequently, it may be possible to artificially inflate representations to allow learning. This is another kind of trick since the algorithm would be efficient according to the letter of the definition but not its spirit.

[7] An incremental learner is *conservative* if it changes its current hypothesis H if and only if the next datum is inconsistent with H.

Identification in polynomial time and data also suffers from the opposite kind of drawback. As we will see, for non-regular languages, there can be exponentially *compact* representations of languages. For such cases, IPTD-learning appears to give the wrong results: classes which intuitively ought to count as tractably learnable (because they return a very compact representation of the target language) can in fact be shown to not be IPTD-learnable. As IPTD was developed and studied in the context of learning regular languages, neither of these problems arose since minimal deterministic finite-state automata are considered to be reasonable representations of regular languages.

Example 3 illustrates the problem for the IPTD-learning of non-regular languages. It shows that context-free languages cannot be learned under this criterion using context-free grammars. Indeed, the characteristic sample of any grammar of the series has to contain the only string in the language, but the length of this string is exponentially greater than the size of the grammar.

Example 3. [de la Higuera, 1997] let $\mathbb{G}_1 = \cup_{n>0}\{G_n\}$ be the class of context-free grammars such that for any n, the unique axiom of G_n is N_0 and its production rules are $N_i \to N_{i+1}N_{i+1}$, for $0 \leq i < n$, and $N_n \to a$. The language of G_n is the singleton $L(G_n) = \{a^{2^n}\}$.

The reason why this example is not learnable does not come from the hardness of the languages: they are all made of only one string. But the use of any class of representations that contains $\mathbb{G}_1$ is not identifiable in the limit.

It seems that in this case the problem comes from the definition of what learning means, that is to say from the learning criterion, rather than the properties of the language. From an information theory point of view, it is obviously interesting to have an algorithm that is able to find a model explaining the data it is fed with that is exponentially smaller than the data. This is actually a desired property in many fields of machine learning (see Girosi [1998] for instance). Hence, the trouble here comes from the learning paradigm.

2.5 Recent refinements

In this section, we review two contemporary approaches that develop a definition of efficient learning which can be applied to non-regular classes of languages. They are both refinements of the identification with polynomial time and data.

2.5.1 Structurally complete set

We first introduce the following definition:

Definition 2.9 (Structurally Complete Set)**.** Given a generative grammar G, a structurally complete set (SCS) for G is a set of data SC such that for each production $\alpha \to \beta$,

there exists an element $x \in SC$, an element $\gamma \in I$ and two elements $\eta, \tau \in (\Sigma \cup N)^*$ such that $\gamma \Rightarrow^* \eta\alpha\tau \Rightarrow \eta\beta\tau \Rightarrow^* x$. The smallest structurally complete set (SSCS) S for a grammar G is the sample such that for all SCS S' for G, $S \trianglelefteq S'$.

A notion of structurally complete sample has already been defined in the context of regular language learning [Dupont et al., 1994]. However, this former definition relied on a particular representation, namely the finite state automaton, and it considered only the case of positive and negative examples. Definition 2.9 is more general as it does not depend on a particular representation and does not consider a particular type of data. Definition 2.9 is a generalization of the notion of *representative sample* [Tajima et al., 2004] that have been introduced in the context of learning, from membership queries and a sample of positive examples, a subclass of context-free languages named simple deterministic languages.

Definition 2.10 (Polynomial Structurally Complete Identification)**.** A class $\mathbb{L}$ of languages is *identifiable in polynomial time and structurally complete data (IPTscD)* for a class $\mathbb{R}$ of representations if and only if there exist an algorithm $\mathfrak{A}$ and two polynomials $p()$ and $q()$ such that:

1. Given a sample S for $L \in \mathbb{L}$ of size m, $\mathfrak{A}$ returns a consistent hypothesis $H \in \mathbb{R}$ in $\mathcal{O}(p(m))$ time ;
2. For each representation R of a language $L \in \mathbb{L}$, there exists a *characteristic sample* CS whose size is in $\mathcal{O}(q(k))$, where k is the size of the smallest structurally complete set for R.

Notice that in the case where negative data is also available, the size of the characteristic sample has to be polynomial in the size of a SCS which contains only positive examples. This implies that the amount of negative evidence has to be polynomially related to that of the positive evidence.

This paradigm shifts the perspective considerably: the efficiency does not rely anymore directly on the size of the representation but instead on the kind of elements it can generate. This move is anticipated, and pursued in part, in the approach by Yoshinaka [2008], discussed in section 2.5.2 below.

Comparison with IPTD

Consider the class of languages $\mathbb{L}_2 = \bigcup_{n\in\mathbb{N}}\{\{a^i : 0 \leq i \leq 2^n\}\}$. This class is identifiable in polynomial time and data from positive data only using the class of representations $\mathbb{G}_2 = \bigcup_{n\in\mathbb{N}}\{\langle\{a\}, \{S, A\}, \{S \to A^{2^n}, A \to a|\lambda\}, \{S\}\rangle\}$. Indeed, given a target language, the simple algorithm that returns the only grammar consistent with a sample admit the characteristic sample $\{a^{2^n}\}$ which is linear in the size of the target. However, the smallest structurally complete set of any target grammar is $\{\lambda, a\}$ which is of size 2.

As the size of the smallest SCS is constant and the class of languages infinite, $\mathbb{L}_2$ is not identifiable in polynomial time and structurally complete data.

On the other hand, let consider the class of languages of Example 3: $\mathbb{L}_1 = \bigcup_{n\in\mathbb{N}}\{\{a^{2^n}\}\}$ and its class of representations $\mathbb{G}_1 = \bigcup_{n\in\mathbb{N}}\{\langle\{a\}, N_n, P_n, \{N_0\}\rangle\}$, with $P_n = \{N_n \to a\} \cup_{0\leq i<n} \{N_i \to N_{i+1}N_{i+1}\}$. Given n, the characteristic sample is $\{a^{2^n}\}$ which is also the smallest structurally complete set for the target grammar. However, this sample is not polynomial in the size of the target grammar. Therefore $\mathbb{L}_1$ is identifiable in the limit in polynomial time and structurally complete data using $\mathbb{G}_1$ but not in polynomial time and data.

This shows that these two paradigms are thus non-comparable. However, most non-trivial language classes studied under the former paradigm are identifiable in polynomial time and structurally complete data. This is the case for instance of the regular languages from positive and negative examples and of all sub-regular classes studied in the context of grammatical inference: there is a linear link between the size of a regular grammar and what can be derived from any of its non-terminals.

2.5.2 Thickness

In a recent paper Yoshinaka [2008] introduced the identification from a characteristic sample whose size is a polynomial in the size of the target grammar and of a measure called the thickness of the grammar.

Definition 2.11 (Thickness)**.** Let $G = \langle\Sigma, N, P, I\rangle$ be a generative grammar. The *thickness* of G is $\tau_G = \max\{|\omega(\alpha)| : \exists\beta, \alpha \to \beta \in P\}$ where $\omega(\alpha) = \min_{\lhd}\{w \in \Sigma^* : \alpha \Rightarrow_G^* w\}$.

Informally, the thickness is the length of the longest element in the set of the smallest elements that can be generated from a left hand-side of a grammar rule.

This definition is an extended version of the one that was first introduced for context-free grammars in the context of model complexity [Wakatsuki and Tomita, 1993]. Notice that it has nothing to do with the usual notion of thickness in learning theory Angluin [1980a].

Definition 2.12 (Polynomial Thick Identification [Yoshinaka, 2008])**.** A class $\mathbb{L}$ of languages is *identifiable in polynomial time and thick data (IPTtD)* for a class $\mathbb{R}$ of representations if and only if there exist an algorithm $\mathfrak{A}$ and two polynomials $p()$ and $q()$ such that:

1. Given a sample S for $L \in \mathbb{L}$ of size m, $\mathfrak{A}$ returns a consistent hypothesis $H \in \mathbb{R}$ in $\mathcal{O}(p(m))$ time ;
2. For each representation R of a language $L \in \mathbb{L}$ of size k, there exists a *characteristic sample* CS whose size is in $\mathcal{O}(q(k, \tau_R))$.

IPTtD is clearly a refinement of IPTD since it simply adds the thickness as a parameter of the paradigm. It is however a fundamental move since it links the efficiency of the learning not only on the target representation but also to the kind of strings the grammar produces. This shift in perspective is a way to indirectly take into account the length of the strings in the language in the learning criterium. On the other hand, since it does not go so far as to remove the requirement that the characteristic sample be polynomial in the size of the grammar, it is still susceptible to inflation tricks.

Learnable classes

Since the size of the representation is used in Definition 2.12, it is clear that every class of languages that is IPTD is also IPTtD.

However, the converse is not true. Consider the grammars of Example 3: The thickness of any $G_n \in \mathbb{G}_1$ is 2^n.

More interesting examples are the classes of languages that have been investigated in the context of what is called distributional learning (see for instance the chapter *Distributional Learning* from Clark and Yoshinaka [2016] in the book *Topics in Grammatical Inference*, or Part II of this HDR). For instance, a context-free language is *substitutable* if whenever two substrings appear once in the same context, then they always appear in the same context in the language [Clark and Eyraud, 2007].

There exists a polynomial time algorithm that identifies the class of context-free substitutable languages from positive examples only, in the sense of Definition 2.8, but the exhibited characteristic sample might be of size exponential in the size of the target representation (this is the case for the languages of Example 3, which are substitutable). Thus, this algorithm is not IPTD. On the other hand, it is easy to see that this characteristic sample is polynomial in the size *and the thickness* of the target grammar, so the algorithm is IPTtD. This result can be extended to the more complex classes that have been studied in the context of distributional learning from positive examples only (see for instance [Yoshinaka, 2008, 2011]).

2.5.3 Comparison of the two refinements

Since the IPTD and IPTscD classes are incomparable and every IPTD class is IPTtD, clearly there is an IPTtD class which is not IPTscD (this is the case for instance of the class $\mathbb{L}_2$ introduced at the end of Section 2.5.1). However, one can show that every IPTtD class of unambiguous CFGs is IPTscD. Also, it is easy to see that every IPTscD class of context-free languages is IPTtD using the same class of representations.

The two refinements of polynomial identification share a basic idea – to measure the complexity by the size of the simplest strings that a grammar generates, rather than the description size of it. Indeed one can show that the size of the smallest SCS of G is

polynomially bounded by $\tau_G||G||$. That is, if a language class is IPTscD for a class of context-free grammars, then it is also IPTtD.

However, the converse is not necessarily true. The following discussion illustrates a particularly difficult problem for IPTscD learning: ambiguity. Let G_n consists of the following rules:

$$P_n = \{\, A \to a,\, A \to b,\, B \to b \,\} \cup \{\, S \to X_1 \dots X_n \mid X_i \in \{A, B\} \,\},$$

with S being its only axiom. It generates the language $L(G_n) = \{a, b\}^n$. Then the set $\{a^n, b^n\}$, whose size is $2n + 2$, is the smallest SCS for G_n. On the other hand, $||G_n|| \in \mathcal{O}(n2^n)$ and $\tau_{G_n} = n$. When learning the class $\mathbb{L} = \bigcup_{n\in\mathbb{N}} \mathbb{L}_n$ where $\mathbb{L}_n = \{\, L \subseteq \{a, b\}^n \,\} \setminus \{\emptyset\}$ with a positive sample, the only possible characteristic sample of $L(G_n)$ is $L(G_n)$ itself for any learning algorithm. Therefore, $\mathbb{L}$ is not IPTscD for any representation class. One can easily see that $\mathbb{L}$ is IPTtD for a reasonable class of grammars where G_n is the unique grammar for $\{a, b\}^n$.

The grammar G_n is very redundant and highly ambiguous – there are 2^n ways to derive b^n. If the redundancy is removed from G_n by deleting the nonterminal B and the rules involving B, the size of the grammar is now $O(n)$ and it is not IPTtD any more. In fact, one can show that $\tau_G||G||$ is polynomially bounded by the size of the smallest SCS when only unambiguous context-free grammars are considered.

2.6 Conclusion

The purpose of this chapter was to provide a panorama of learning paradigms when grammatical inference is of concern and to address the problem of efficiency in these framework. We claimed that the PAC framework is not the best suited one even though its efficiency requirements are well-designed. On the other hand, we argued in favor of identification in the limit paradigms provided they are adequately modified to include efficiency requirements. This survey showed doing so is more challenging than anyone may have anticipated. We discussed the challenges that have been encountered by different attempts. For regular languages, de la Higuera [1997] solution is satisfactory due to the canonical representation given by the smallest deterministic acceptors. For non-regular languages, challenges remain. We discussed two promising paths forward to address efficient learning in identification in the limit paradigm in the realm of non-regular languages. One was based on the notion of a structurally complete sample[8], and the other was based on the 'thickness' of strings generated by production rules. Both are measuring efficiency at least partly in terms of the size of particular strings generated by grammars. We believe further developments along these lines will help shape future directions in grammatical inference.

[8]Note: This paradigm is a new contribution that I have formalized for this chapter.

Bibliography

A. Ambainis, S. Jain, and A. Sharma. Ordinal mind change complexity of language identification. *Theoretical Computer Science*, pages 323–343, 1999. 22

D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980a. 19, 24, 29

D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980b. 14

D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987. 12, 16

D. Angluin, J. Aspnes, and A. Kontorovich. On the learnability of shuffle ideals. In *Proc.of the Algorithmic Learning Theory Conference*, pages 111–123, 2012. 15

L. E. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28(2):125–155, 1975. 22

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989. ISSN 0004-5411. 15

R. Book and F. Otto. *String-Rewriting Systems*. Springer Verlag, 1993. 14

J. Case. *Gold-Style Learning Theory*, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. 20

J. Case and T. Kötzing. Difficulties in forcing fairness of polynomial time inductive inference. In *Proc. of the Algorithmic Learning Theory Conference*, pages 263–277, 2009. 21

N. Chater, A. Clark, J.A. Goldsmith, and A. Perfors. *Empiricism and Language Learnability*. Oxford University Press, 2015. 17

N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124, 1956. 14

A. Clark. Distributional learning of some context-free languages with a minimally adequate teacher. In J. and P. Garcia, editors, *Proc. of the International Colloquium on Grammatical Inference*, pages 24–37. Springer-Verlag, 2010a. 16

A. Clark. Learning context free grammars with the syntactic concept lattice. In J. Sempere and P. Garcia, editors, *Proc. of the International Colloquium on Grammatical Inference*, pages 38–51. Springer-Verlag, 2010b. 17

A. Clark. Learning trees from strings: A strong learning algorithm for some context-free grammars. *Journal of Machine Learning Research*, 14:3537–3559, 2014. 22

A. Clark and R. Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007. 13, 30

A. Clark and S. Lappin. *Linguistic Nativism and the Poverty of the Stimulus*. Wiley-Blackwell, 2011. 18

A. Clark and F. Thollard. Pac-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5:473–497, 2004. 16

A. Clark and R. Yoshinaka. Distributional learning of parallel multiple context-free grammars. *Machine Learning*, 96:5–31, 2014a. 17

A. Clark and R. Yoshinaka. Distributional learning of parallel multiple context-free grammars. *Machine Learning*, 96(1):5–31, 2014b. 17

A. Clark and R. Yoshinaka. *Distributional Learning of Context-Free and Multiple Context-Free Grammars*, pages 143–172. Springer Berlin Heidelberg, 2016. 30

H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications, 2007. 14

C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138, 1997. 24, 25, 26, 27, 31

C. de la Higuera. *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010. 26

C. de la Higuera. *Learning Grammars and Automata with Queries*, pages 47–71. Springer Berlin Heidelberg, 2016. 16

C. de la Higuera and J. Oncina. Learning deterministic linear languages. In *Proc. of the Conference on Learning Theory*, pages 185–200, 2002. 26

F. Denis and R. Gilleron. Pac learning under helpful distributions. *RAIRO - Theoretical Informatics and Applications*, 35(2):129–148, 2001. 16

P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference? In *Proc. of the International Colloquium on Grammatical Inference*, pages 25–37, 1994. 28

R. Eyraud, C. de la Higuera, and J.-C. Janodet. Lars: A learning algorithm for rewriting systems. *Machine Learning*, 66(1):7–31, 2007. 26

R. Eyraud, J. Heinz, and R. Yoshinaka. *Efficiency in the Identification in the Limit Learning Paradigm*, pages 25–46. Springer Berlin Heidelberg, 2016. 12, 26

F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Comput.*, 10(6):1455–1480, 1998. ISSN 0899-7667. 27

E. M. Gold. Language identification in the limit. *Information and Control*, 10(5): 447–474, 1967. 12, 18, 19

J. Heinz. Computational theories of learning and developmental psycholinguistics. In J. Lidz, W. Synder, and J. Pater, editors, *The Cambridge Handbook of Developmental Linguistics*. Cambridge University Press, 2014. To appear. 18

J. Heinz and J. M. Sempere. *Topics in Grammatical Inference*. Springer Publishing Company, Incorporated, 2016. 12, 20

D. Hsu, S. M. Kakade, and P. Liang. Identifiability and unmixing of latent parse trees. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1520–1528, 2013. 16

M. Isberner, F. Howar, and B. Steffen. Learning register automata: from languages to program structures. *Machine Learning*, 96:65–98, 2014. 16

Y. Ishigami and S. Tani. VC-dimensions of finite automata and commutative finite automata with k letters and n states. *Discrete Applied Mathematics*, 74:123–134, 1997. 15

M. Kanazawa and R. Yoshinaka. The strong, weak, and very weak finite context and kernel properties. In *Proc. of Language and Automata Theory and Applications*, pages 77–88, 2017. 17

J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, December 2005. ISSN 1532-4435. 14

M. Li and P. Vitanyi. Learning simple concepts under simple distributions. *Siam Journal of Computing*, 20:911–935, 1991. 14

D. López and P. García. *On the Inference of Finite State Automata from Positive and Negative Data*, pages 73–112. Springer Berlin Heidelberg, 2016. 21

E. Moore. Gedanken-experiments on sequential machines. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 129–153. Princeton University Press, 1956. 16

T. Oates, D. Desai, and V. Bhat. Learning k-reversible context-free grammars from positive structural examples. In *Proc. of the International Conference in Machine Learning*, pages 459–465, 2002. 26

J. Oncina and P. García. Identifying regular languages in polynomial time. In *Advances in Structural and Syntactic Pattern Recognition*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, pages 99–108. 1992. 13, 22

T.-W. Pao and J. Carr III. A solution of the syntactical induction-inference problem for regular languages. *Computer Languages*, 3(1):53 – 64, 1978. ISSN 0096-0551. 17

L. Pitt. Inductive inference, DFA's, and computational complexity. In *Analogical and Inductive Inference*, number 397 in LNAI, pages 18–44. Springer-Verlag, 1989. 20, 22, 23, 24

D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. In *Proc. of the Conference on Learning Theory*, pages 31–40, 1995. 16

G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations*. World Scientific Publishing Co., Inc., 1997. 14

Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97:23–60, 1992. 26

H. Seki, T. Matsumura, M. Fujii, and T. Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229, 1991. 14

J. M. Sempere and P. García. A characterization of even linear languages and its application to the learning problem. In *Proc. of the International Colloquium in Grammatical Inference*, pages 38–44, 1994. 26

C. Shibata and R. Yoshinaka. Probabilistic learnability of context-free grammars with basic distributional properties from positive examples. *Theoretical Computer Science*, 620:46–72, 2016. 16

Y. Tajima, E. Tomita, M. Wakatsuki, and M. Terada. Polynomial time learning of simple deterministic languages via queries and a representative sample. *Theoretical Computer Science*, 329(1-3):203 – 221, 2004. ISSN 0304-3975. 28

L. G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142, 1984. 12, 14, 17

V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. 15

M. Wakatsuki and E. Tomita. A fast algorithm for checking the inclusion for very simple deterministic pushdown automata. *IEICE TRANSACTIONS on Information and Systems*, VE76-D(10):1224–1233, 1993. 29

T. Yokomori. On polynomial-time learnability in the limit of strictly deterministic automata. *Machine Learning*, 19:153–179, 1995. 23

T. Yokomori. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 1(298):179–206, 2003. 23

R. Yoshinaka. Identification in the limit of k, l-substitutable context-free languages. In *Proc. of the International Colloquium in Grammatical Inference*, pages 266–279, 2008. 28, 29, 30

R. Yoshinaka. Learning efficiency of very simple grammars from positive data. *Theoretical Computer Science*, 410(19):1807–1825, 2009. ISSN 0304-3975. 23

R. Yoshinaka. Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. *Theoretical Computer Science*, 412:1821–1831, 2011. 30

R. Yoshinaka and A. Clark. Polynomial time learning of some multiple context-free languages with a minimally adequate teacher. In *Proc. of Formal Grammar*, pages 192–207, 2010. 16

R. Yoshinaka and M. Kanazawa. *Distributional Learning of Abstract Categorial Grammars*, pages 251–266. Springer Berlin Heidelberg, 2011. 17

T. Zeugmann. Can learning in the limit be done efficiently? In *Proc. of the Algorithmic Learning Theory conference*, pages 17–38, 2003. 17, 20

T. Zeugmann. From learning in the limit to stochastic finite learning. *Theoretical Computer Science*, 364(1):77–97, 2006. 17

Part II

Distributional Learning

Chapter 3

On Learning from Strings

Contents

Context of this work

This work is a quite direct continuation of the one realized during my PhD [Eyraud, 2006]. It started when Alexander Clark came in Marseilles as an invited professor twelve years ago.

Alexander's research interest focusing mainly on how children acquire their native language, this chapter clearly enjoys a computational linguistic flavor. Written in collaboration with Amaury Habrard, it is based on two publications in the peer-reviewed proceedings of international conferences [Clark et al., 2008, 2009]. The content of this chapter is close to the article published in the Journal of Machine Learning Research [Clark et al., 2010]: the main differences can be found in the two first sections and the conclusion.

From a scientific point of view, it is important to notice that this work was transitional in several aspects:

- it moves from a purely set-driven approach to a more complex framework where the learner has access to both a set of examples and a membership oracle;
- it introduces two notions, namely the finite context and the finite kernel properties, that have been proven central in distributional learning since;
- it was the first time we were designing a new computational model specially shaped for learning.

This explains the success of that work: it is the starting point of several research paths that recently obtained important successes (I detail some of them in the conclusion of this chapter). However, this also has a less positive side: not all ideas were mature at the time we wrote the cited articles. For instance, the presentation of some notions is not as clear as it can be today. Nevertheless, I chose to keep our initial formulation in this HDR since it is my main contribution to the distributional learning from strings, since it can be viewed as a witness of the evolution of this field, and since there is no undeniable formulation yet.

3.1 Introduction

In natural language processing, many applications require the learning of powerful grammatical models. One of the central concerns of generative linguistics is the definition of an adequate formalism that needs to satisfy two different objectives. On the one hand, such a formalism must be expressive enough to describe natural languages. On the other hand, it has to be sufficiently constrained to be learnable from the sort of linguistic data available to the child learner [Chomsky, 1986]. In this context, there are two possible research strategies. One is to take a descriptively adequate formalism

such as *Tree Adjoining Grammars* [Joshi and Schabes, 1997] or some other mildly context sensitive computational models and try to construct learning algorithms for that class. However, such a strategy is unlikely to be successful because classes that are so powerful are difficult to handle from a machine learning point of view. The other approach, which we adopt in this chapter, consists in switching to a formalism that is in some sense intrinsically learnable, and seeing whether we can represent linguistically interesting formal languages in that representation.

Grammatical inference has obtained many learnability results for regular languages (see for instance the work of Angluin [1987] and Carrasco and Oncina [1994]). However, this class is not sufficient to correctly represent natural languages. The next class of languages to consider is the class of context-free languages (CFL). Unfortunately, there exists no learnability results for the whole class. This may be explained by the fact that this class relies on syntactic properties instead of intrinsic properties of the language like the notion of residuals for regular languages [Denis et al., 2004]. Thus, most of the approaches proposed in the literature are either based on heuristics [Langley and Stromsten, 2000, Nakamura and Matsumoto, 2005] or are theoretically well founded but concern very restricted subclasses of context-free languages [Higuera and Oncina, 2002, Yokomori, 2003, Eyraud et al., 2007]. Some of these approaches are built from the idea of *distributional learning*[1], normally attributed to Harris [1954]. The basic principle – as we reinterpret it in our work – is to look at the set of contexts that a substring can occur in. The distribution of a substring is the linguistic way of referring to this set of contexts. This idea has formed the basis of many heuristic algorithms for learning context-free grammars (see the work of Adriaans [2002] or of van Zaanen and van Noord [2014] for instance). However, in Clark and Eyraud [2007] we presented an accurate formalization of distributional learning. From this formulation, a provably correct algorithm for context-free grammatical inference was given in the IPTtD paradigm (Definition 2.12 of Chapter 2), albeit for a very limited subclass of languages, the substitutable languages (see the formerly cited paper and the work of Yoshinaka [2008] for details). From a more general point of view, the central insight is that it is not necessary to find the non-terminals of the context-free grammar (CFG): it is enough to be able to represent the congruence classes of a sufficiently large set of substrings of the language and to be able to compute how they combine. This result was quickly extended to a PAC-learning result under a number of different assumptions [Clark, 2006] for a larger class of languages, and also to a family of classes of learnable languages [Yoshinaka, 2008].

Despite their theoretical bases, these results are still too limited to form the basis for models for natural language. There are two significant limitations to this work: first it uses a very crude measure for determining the syntactic congruence, and secondly the

[1]Note here that the word distributional does not refer to stochastic distributions, but to the occurrence of strings into contexts. The distribution of a string corresponds to all the possible contexts in which the string can appear.

number of congruence classes required will in real cases be prohibitively large. If each non-terminal corresponds to a single congruence class (the NTS languages [Boasson and Senizergues, 1985]), then the problem may be tractable. However in general the contexts of different non-terminals overlap enormously: for instance the contexts of adjective phrases and noun phrases in English both contain contexts of the form (*"it is"*, *"."*). Problems of lexical ambiguity also cause trouble. Thus for a CFG it may be the case that the number of congruence classes corresponding to each non-terminal may be exponentially large (in the size of the grammar). But the situation in natural language is even worse: the CFG itself may have an exponentially large number of non-terminals to start off with! Conventional CFGs are simply not sufficiently expressive to be cognitively plausible representations of natural language: to write a CFG requires a multiplication of the numbers of non-terminals to handle phenomena like *subject verb agreement*, *gender features*, *displaced constituents*, etc. This requires the use of a formalism like GPSG (Generalized Phrase Structure Grammar) [Gazdar et al., 1985] to write a meta-grammar — a compact way of specifying a very large CFG with richly structured non-terminals. Thus we cannot hope to learn natural languages by learning one congruence class at a time: it is vital to use a more structured representation.

This is the objective of the approach introduced in this chapter: for the first time, we can bridge the gap between theoretically well founded grammatical inference methods and the sorts of structured representations required for modeling natural languages.

In this chapter, we present a family of representations for highly structured context-free languages and show how they can be learned. This is a chapter about learning, but superficially it may appear to be a chapter about computational models: much of the work is done by switching to a more tractable formalism, a move which is familiar to many in machine learning. From a machine learning point of view, it is a commonplace that switching to a better representation – for example, through a non-linear map into some feature space – may make a hard problem very easy.

The contributions presented in this chapter are as follows: we present in Section 3.3 a rich grammatical formalism, which we call *Contextual Binary Feature Grammars* (CBFG). This grammar formalism is defined using a set of contexts which play the role of features with a strict semantics attached to these features. Though not completely original, since it is closely related to a number of other formalisms such as Range Concatenation Grammars [Boullier, 2000], it is of independent interest.

We consider then the case when the contextual features assigned to a string correspond to the contexts that the string can occur in, in the language defined by the grammar. When this property holds, we call it an *exact CBFG*. The crucial point here is that for languages that can be defined by an exact CBFG, the underlying structure of the representation relies on intrinsic properties of the language easily observable on samples by looking at context sets.

The learning algorithm is defined in Section 3.4. We provide some conditions, both

on the context sets and the learning set, to ensure the learnability of languages that can be represented by CBFG. We prove that this algorithm can identify in the limit this restricted class of CBFGs from positive data and a membership oracle.

Some experiments are provided in Section 3.5: these experiments are intended to demonstrate that even quite naive algorithms based on this are efficient and effective at learning context-free languages.

Section 3.6 contains a theoretical study on the expressiveness of CBFG representations. We investigate the links with the classical Chomsky hierarchy, some well known grammatical representations used in natural language processing. An important result about the expressive power of the class of CBFG is obtained: it contains all the context-free languages and some non context-free languages. This makes this representation a good candidate for representing natural languages. However exact CBFG do not include all context-free languages but do include some non context-free ones, thus they are orthogonal with the classic Chomsky hierarchy and can represent a large class of languages. This expressiveness is strengthened by the fact that exact CBFG contains most of the existing learnable classes of languages.

3.2 Basic Definitions and Notations

We will use the notation introduced in Section 2.1.2 of Chapter 2.

Nevertheless, we need additional notations and definitions: we will write the concatenation of two strings u and v as uv, and similarly for sets of strings. $u \in \Sigma^*$ is a substring of $v \in \Sigma^*$ if there are strings $l, r \in \Sigma^*$ such that $v = lur$. Define $\texttt{Sub}(u)$ to be the set of non-empty substrings of u. For a set of strings S define $\texttt{Sub}(S) = \bigcup_{u \in S} \texttt{Sub}(u)$.

A context is an element of $\Sigma^* \times \Sigma^*$. For a string u and a context $f = (l, r)$ we write $f \odot u = lur$; the insertion or wrapping operation. We extend this to sets of strings and contexts in the natural way. We define by $\texttt{Con}(w) = \{(l, r) | \exists u \in \Sigma^+ : lur = w\}$ i.e. the set of all contexts of a word w. Similarly, for a set of strings, we define: $\texttt{Con}(S) = \bigcup_{w \in S} \texttt{Con}(w)$.

We give now a formal definition of the set of contexts since it represents a notion often used in the chapter.

Definition 3.1. The set of contexts, or context distribution, of a string u in a language L is, $C_L(u) = \{(l, r) \in \Sigma^* \times \Sigma^* | lur \in L\}$. We will often drop the subscript where there is no ambiguity.

Definition 3.2. Two strings u and v are syntactically congruent with respect to a language L, denoted $u \equiv_L v$, if and only if $C_L(u) = C_L(v)$.

The equivalence classes under this relation are the *congruence* classes of the language.

After these basic definitions and notations, we recall here the definition of a context-free grammar which is a class which is close to the language class studied in this chapter.

Definition 3.3. A context-free grammar (CFG) is a quadruple $G = (\Sigma, V, P, S)$. Σ is a finite alphabet of terminal symbols, V is a set of non terminals s.t. $\Sigma \cap V = \emptyset$, $P \subseteq V \times (V \cup \Sigma)^+$ is a finite set of productions, $S \in V$ is the start symbol.

We denote a production of P: $N \to \alpha$ with $N \in V$ and $\alpha \in (V \cup \Sigma)^+$. Following the general definitions of Section 2.1.2, we will write $uNv \Rightarrow_G u\alpha v$ if there is a production $N \to \alpha$ in G, and denote $\stackrel{*}{\Rightarrow}_G$ the reflexive transitive closure of $\Rightarrow_G$.

The language defined by a CFG G is $L(G) = \{w \in \Sigma^* | S \stackrel{*}{\Rightarrow}_G w\}$. In the following, we will consider the CFG are represented in the *Chomsky normal form* (CNF), *i.e.* with right hand side of production rules composed of exactly two non terminals or with exactly one terminal symbol.

In general we will assume that λ is not a member of any language.

3.3 Contextual Binary Feature Grammars (CBFG)

Distributional learning, in our view, involves explicitly modeling the distribution of the substrings of the language – we would like to model $C_L(w)$. Clearly a crucial element of this distribution is the empty context (λ, λ): $(\lambda, \lambda) \in C_L(w)$ if and only if $w \in L$. Our goal is to construct a representation that allows us to recursively compute a representation of the distribution of a string w, $C_L(w)$, from the (representations of) the distributions of its substrings.

The representation by *contextual binary feature grammars* relies on the inclusion relation between sets of contexts of language L. In order to introduce this formalism, we propose, for a start, to present some preliminary results on context inclusion. These results will lead us to define a relevant representation for modeling these inclusion dependencies by the notion of *contextual binary feature grammars*.

3.3.1 Preliminary Results about Context Inclusion

The objective of this section is to give some information about contexts that will help to give an intuition about the representation. The basic insight behind CBFGs is that there is a relation between the contexts of a string w and the contexts of its substrings. This is given by the following trivial lemma:

Lemma 3.1. For any language L and for any strings u, u', v, v' if $C(u) = C(u')$ and $C(v) = C(v')$, then $C(uv) = C(u'v')$.

Proof. We write out the proof completely as the ideas will be used later on. Suppose we have u, v, u', v' that satisfy the conditions. If $(l, r) \in C(uv)$, then $(l, vr) \in C(u)$ and

thus $(l, vr) \in C(u')$. As a consequence, $(lu', r) \in C(v)$ and then $(lu', r) \in C(v')$ which implies that $(l, r) \in C(u'v')$. Symmetrically, by using the same arguments, we can show that $(l, r) \in C(u'v')$ implies (l, r) in $C(uv)$. Thus $C(uv) = C(u'v')$. □

This establishes that the syntactic monoid $\Sigma^* / \equiv_L$ is well-defined; from a learnability point of view this means that if we want to compute the contexts of a string w we can look for a split into two strings uv where u is congruent to u' and v is congruent to v'; if we can do this and we know how u' and v' combine, then we know that the contexts of uv will be exactly the contexts of $u'v'$. There is also a slightly stronger result:

Lemma 3.2. For any language L and for any strings u, u', v, v' if $C(u) \subseteq C(u')$ and $C(v) \subseteq C(v')$, then $C(uv) \subseteq C(u'v')$.

Proof. See proof of Lemma 3.1. □

$C(u) \subseteq C(u')$ means that we can replace any occurrence in a sentence of u with a u', without affecting the grammaticality, but not necessarily vice versa. Note that none of these strings need to correspond to non-terminals: this is valid for any fragment of a sentence.

We will give a simplified example from English syntax: the pronoun "*it*" can occur everywhere that the pronoun "*him*" can, but not vice versa[2]. Thus given a sentence "*I gave him away*", we can substitute "*him*" for "*it*", to get the grammatical sentence "*I gave it away*", but we cannot reverse the process. For example, given the sentence "*it is raining*", we cannot substitute "*him*" for "*it*", as we will get the ungrammatical sentence "*him is raining*". Thus we observe $C(him) \subsetneq C(it)$.

Looking at Lemma 3.2 we can also say that, if we have some finite set of strings K, where we know the contexts, then:

Corollary 3.3. For any language L and for any set of strings K, we have:

$$C(w) \supseteq \bigcup_{\substack{u',v': \\ u'v'=w}} \bigcup_{\substack{u \in K: \\ C(u) \subseteq C(u')}} \bigcup_{\substack{v \in K: \\ C(v) \subseteq C(v')}} C(uv).$$

This is the basis of our representation: a word w is characterized by its set of contexts. We can compute the representation of w, from the representation of its parts u', v', by looking at all of the other matching strings u and v where we understand how they combine (with subset inclusion). Rather than representing just the congruence classes, we will represent the lattice structure of the set of contexts using subset inclusion; sometimes called Dobrušin-domination [Marcus, 1967].

The key relationships are given by context set inclusion. *Contextual binary feature grammars* allow a proper definition of the combination of context inclusion.

[2]This example does not account for a number of syntactic and semantic phenomena, particularly the distribution of reflexive anaphora.

3.3.2 Contextual Binary Feature Grammars

The formalism of *contextual binary feature grammars* has some resemblance with *Generalized Phrase Structure Grammar (GPSG)* [Gazdar et al., 1985], and most importantly the class of *Range Concatenation Grammars (RCG)* [Boullier, 2000]; these relationships will be detailed in Section 3.6. As we will see later, note that our formalism defines a class orthogonal to the class of context-free grammars, indeed the use of subsets inclusion allows to model non context-free languages while not all the context-free languages can be represented. It is worth noticing also that this formalism is not directly compatible with our notion of generative grammars though only small modifications of Definition 2.1 allow the compatibility.

Definition 3.4. A Contextual Binary Feature Grammar (CBFG) G is a tuple $\langle F, P, P_L, \Sigma \rangle$:

- F is a finite set of contexts, (i.e. $F \subset \Sigma^* \times \Sigma^*$) called features, where we write $E = 2^F$ for the power set of F defining the categories of the grammar, and where $(\lambda, \lambda) \in F$.
- $P \subseteq E \times E \times E$ is a finite set of productions that we write $x \to yz$ where $x, y, z \in E$,
- $P_L \subseteq E \times \Sigma$ is a set of lexical rules, written $x \to a$,
- Σ denotes the alphabet.

Given a CBFG G we can recursively define a function f_G from $\Sigma^* \to E$ as follows:

$$f_G(\lambda) = \emptyset \tag{3.1}$$

$$f_G(w) = \bigcup_{(x \to w) \in P_L} x \qquad \text{iff } |w| = 1 \tag{3.2}$$

$$f_G(w) = \bigcup_{u,v:uv=w} \bigcup_{\substack{x \to yz \in P: \\ y \subseteq f_G(u) \wedge \\ z \subseteq f_G(v)}} x \qquad \text{iff } |w| > 1. \tag{3.3}$$

Given a CBFG G and a string w it is possible to compute $f_G(w)$ in time $\mathcal{O}(|F||P||w|^3)$ using standard dynamic programming techniques. A straightforward modification of the Cocke-Kasami-Younger algorithm for parsing Context-Free Grammars will suffice.

Thus a CBFG, G, defines for every string u a set of contexts $f_G(u)$: this will be a representation of the context distribution. $f_G(u)$ will be a subset of F: we will want $f_G(u)$ to approximate $C_L(u) \cap F$. The natural way to define the membership of a string w in $L(G)$ is to have the context $(\lambda, \lambda) \in f_G(w)$.

Definition 3.5. The language defined by a CBFG G is the set of all strings that are assigned the empty context: $L(G) = \{u | (\lambda, \lambda) \in f_G(u)\}$.

We give here more explanation about the function f_G. A rule $x \to yz$ is applied to analyze a string w if there is a split or *cut* of the string w into two strings u and v such that $uv = w$ s.t. $y \subseteq f_G(u)$ and $z \subseteq f_G(v)$ — recall that y and z are sets of features.

One way of viewing a production $x \to yz$ is as an implication: if two strings u and v are such that they have the features y and z, then their concatenation will have the features x. As features correspond to contexts, intuitively, the relation given by the production rule is linked with Lemma 3.2: x is included in the set of features of $w = uv$. From this relationship, for any $(l, r) \in x$ we have $lwr \in L(G)$.

The complete computation of f_G is then justified by Corollary 3.3: $f_G(w)$ defines all the possible contextual features associated by G to w with all the possible cuts $uv = w$ (*i.e.* all the possible derivations).

Note the relation between the third clause above and Corollary 3.3. In general we will apply more than one production at each step of the analysis.

We will discuss the relation between this class and the class of CFGs in some detail in Section 3.6. For the moment, we will just make the following points. First, the representation is quite close to that of a CFG where the non-terminals correspond to sets of contexts (subsets of F). There are, however, crucial differences: the very fact that they are represented by sets means that the non-terminals are no longer atomic symbols but rather structures; the formalism can combine different rules together at each step. Secondly, the function f_G can combine different parsing paths. It is not the case that every feature assigned to w must come from the same split of w into u and v. Rather some features could come from one split, and some from another: these two sets of features can be combined in a single derivation even though they come from different parsings (which corresponds to different derivation trees in CFG). It is this property that takes the class of languages out of the class of context-free languages. In the special case where all of the productions involve only singleton sets then this will reduce to a CFG — the non-terminals will correspond to the individual features, and $f_G(w)$ will correspond to the set of non-terminals that can derive the string w.

Clearly by the definition of $L(G)$ we are forcing a correspondence between the occurrence of the context (λ, λ) in $C_L(w)$ and the occurrence of the feature (λ, λ) in $f_G(w)$. But ideally we can also require that f_G defines exactly the possible features that can be associated to a given string according to the underlying language. Indeed, we are interested in cases where there is a correspondence between the language theoretic interpretation of a context, and the occurrence of that context as a feature in the grammar: in this case the features will be observable which will lead to learnability.

This is formalized via the following definitions.

Definition 3.6. Given a finite set of contexts $F = \{(l_1, r_1), \ldots, (l_n, r_n)\}$ and a language

L we can define the context feature function $F_L : \Sigma^* \to 2^F$ which is just the function $u \mapsto \{(l, r) \in F | lur \in L\} = C_L(u) \cap F$.

Using this definition, we now need a correspondence between the language theoretic context feature function F_L and the representation in our CBFG, f_G.

Definition 3.7. A CBFG G is *exact* if for all $u \in \Sigma^*$, $f_G(u) = F_{L(G)}(u)$.

Example. Let $L = \{a^n b^n | n > 0\}$. Let $\langle F, P, P_L, \Sigma \rangle$ a CBFG s.t.
$F = \{(\lambda, \lambda), (a, \lambda), (aab, \lambda), (\lambda, b), (\lambda, abb)\}$. The lexical productions in P_L are:
$\{(\lambda, b), (\lambda, abb)\} \to a$ and $\{(a, \lambda), (aab, \lambda)\} \to b$. Note that these lexical productions are of the form $x \to a$, where x is a subset of F, that is to say, a set of features. The rule $\{(\lambda, b), (\lambda, abb)\} \to a$ therefore says that the letter a will be assigned both of the features/contexts (λ, b) and (λ, abb). Since this is the only lexical rule for a, we will have that $f_G(a) = \{(\lambda, b), (\lambda, abb)\}$. The productions in P, denoted by $x \to yz$, where x, y, z are again sets of contexts, are defined as: $\{(\lambda, \lambda)\} \to \{(\lambda, b)\}\{(aab, \lambda)\}$, $\{(\lambda, \lambda)\} \to \{(\lambda, abb)\}\{(a, \lambda)\}$, $\{(\lambda, b)\} \to \{(\lambda, abb)\}\{(\lambda, \lambda)\}$, $\{(a, \lambda)\} \to \{(\lambda, \lambda)\}\{(aab, \lambda)\}$.

In each of these rules, in this trivial case, the sets of contexts are singleton sets. In general, these productions may involve sets that have more than one element. This defines an exact CBFG for L. Indeed, the grammar assigns only (λ, λ) to the elements of the language; for all elements w of $\{a^n b^{n+1} : n > 1\}$ we have $f_G(w) = \{(a, \lambda)\} = F_L(w)$ and for all all elements w of $\{a^{n+1} b^n : n > 1\}$, $f_G(w) = \{(\lambda, b)\} = F_L(w)$; The lexical rules assign correct contexts to each letter.

Exact CBFGs are a more limited formalism than CBFGs themselves; without any limits on the interpretation of the features, we can define a class of formalisms that is equal to the class of *Conjunctive Grammars* (see Section 3.6.2.3). However, exactness is an important property because it allows to associate the intrinsic structure of a language to the structure of the representation. Contexts are easily observable from a sample and moreover it is only when the features correspond to the contexts that distributional learning algorithms can infer the structure of the language.

3.3.3 A Parsing Example

To clarify the relationship with CFG parsing, we will give a simple worked example. Consider the CBFG $G = \langle \{(\lambda, \lambda), (aab, \lambda), (\lambda, b), (\lambda, abb), (a, \lambda)\}, P, P_L, \{a, b\} \rangle$ with

$$P_L = \{ \quad \{(\lambda, b), (\lambda, abb)\} \to a, \quad \{(a, \lambda), (aab, \lambda)\} \to b \quad \}.$$

and

$$P = \{ \quad \{(\lambda, \lambda)\} \to \{(\lambda, b)\}\{(aab, \lambda)\}, \quad \{(\lambda, \lambda)\} \to \{(\lambda, abb)\}\{(a, \lambda)\}, \quad \{(\lambda, b)\} \to \{(\lambda, abb)\}\{(\lambda, \lambda)\}, \quad \{(a, \lambda)\} \to \{(\lambda, \lambda)\}\{(aab, \lambda)\} \quad \}.$$

If we want to parse a string w the usual way is to have a bottom-up approach. This means that we recursively compute the f_G function on the substrings of w in order to check whether (λ, λ) belongs to $f_G(w)$.

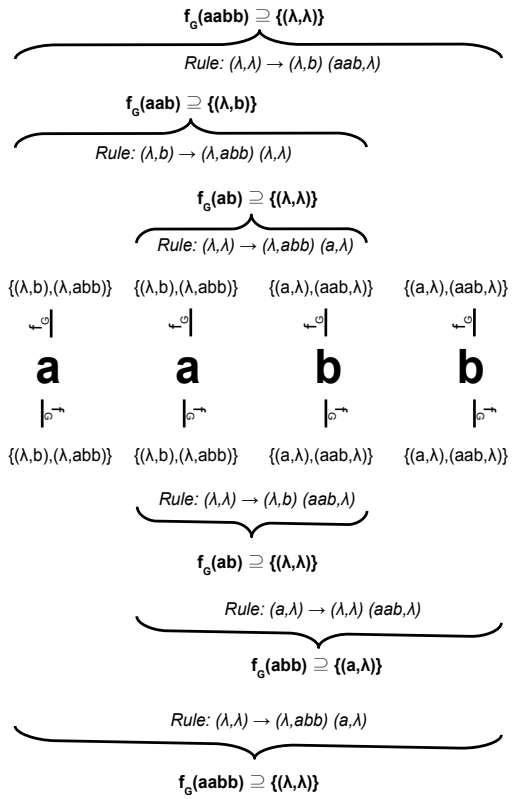

Figure 3.1: The two derivations to obtain (λ, λ) in $f_G(aabb)$ in the grammar G.

For example, suppose $w = aabb$. Figure 3.1 graphically gives the main steps of the computation of $f_G(aabb)$. Basically there are two ways to split $aabb$ that allow the derivation of the empty context: $aab|b$ and $a|abb$. The first one corresponds to the top part of the figure while the second one is drawn at the bottom. We can see for instance that the empty context belongs to $f_G(ab)$ thanks to the rule $\{(\lambda, \lambda)\} \rightarrow \{(\lambda, abb)\}\{(a, \lambda)\}$: $\{(\lambda, abb)\} \subseteq f_G(a)$ and $\{(a, \lambda)\} \subseteq f_G(b)$. But for symmetrical reasons the result can also be obtained using the rule $\{(\lambda, \lambda)\} \rightarrow \{(\lambda, b)\}\{(aab, \lambda)\}$.

As we trivially have $f_G(aa) = f_G(bb) = \emptyset$, since no right-hand side contains the concatenation of the same two features, an induction proof can be written to show that $(\lambda, \lambda) \in f_G(w) \Leftrightarrow w \in \{a^n b^n : n > 0\}$.

This is a simple example that illustrates the parsing of a string given a CBFG. This example does not fully express the power of CBFG since no element of the right hand side of a rule is composed of more than one context. A more complex example, corresponding to a context-sensitive language, will be presented in Section 3.6.1.3.

We stop here the presentation of the CBFG formalism and we present our learning algorithm in the next section. However, if the reader wishes to become more familiar with CBFGs a study on their expressiveness is provided in Section 3.6.

3.4 Learning Algorithm

We have carefully defined the representation so that the inference algorithm will be almost trivial. Given a set of strings, and a set of contexts, we can simply write down a CBFG that will approximate a particular language.

3.4.1 Building CBFGs from Sets of Strings and Contexts

Definition 3.8. Let L be a language, F be a finite set of contexts such that $(\lambda, \lambda) \in F$, K a finite set of strings, $P_L = \{F_L(u) \to u | u \in K \wedge |u| = 1\}$ and $P = \{F_L(uv) \to F_L(u)F_L(v) | u, v, uv \in K\}$. We define $G_0(K, L, F)$ as the CBFG $\langle F, P, P_L, \Sigma\rangle$.

Often K will be closed under substrings: i.e. $\texttt{Sub}(K) = K$. This grammar is a CBFG, since K and F are finite, and so P and P_L are too by construction. In general it will not be exact.

We will call K here the *basis* for the language. The set of productions is defined merely by observation: we take the set of all productions that we observe as the concatenation of elements of the small set K.

Let us explain the construction in more detail. P_L is the set of lexical productions – analogous to rules of the form $N \to a$ in a CFG in Chomsky normal form. These rules just assign to the terminal symbols their observed distribution – this will obviously be correct in that $f_G(a) = F_L(a)$. P is the interesting set of productions: these allow us to predict the features of a string uv from the features of its part u and v. To construct P we take all triples of strings u, v, uv that are in our finite set K. We observe that u has the contexts $F_L(u)$ and v has the set of contexts $F_L(v)$: our rule then states that we can combine any string that has all of the contexts in $F_L(u)$ together with any string that has the contexts in $F_L(v)$ and the result will have all of the contexts in $F_L(uv)$.

We will now look at a simple example. Let $L = \{a^n b^n \mid n > 0\}$, F, the set of features is $\{(\lambda, \lambda), (a, \lambda), (\lambda, b)\}$ and K, the basis, is $\{a, b, ab, aa, aab\}$. For each of the elements of K we can compute the set of features that it has:

- $F_L(a)$ is just $\{(\lambda, b)\}$ – this is the only one of the three contexts in F such that $f \odot a \in L$.
- $F_L(b) = \{(a, \lambda)\}$
- $F_L(aa) = \emptyset$
- $F_L(ab) = \{(\lambda, \lambda)\}$

- $F_L(aab) = \{(\lambda, b)\}$.

G_0 will therefore have the following lexical productions $P_L = \{(\lambda, b)\} \to a$, $\{(a, \lambda)\} \to b$. We can now consider the productions in P. Looking at K we will see that there are only four possible triples of strings of the form uv, u, v: these are (aa, a, a), (ab, a, b), (aab, aa, b) and (aab, a, ab). Each of these will give rise to an element of P:

- The rule given by $ab = a \circ b$: $\{(\lambda, \lambda)\} \to \{(\lambda, b)\}\{(a, \lambda)\}$,
- $aa = a \circ a$ gives $\emptyset \to \{(\lambda, b)\}\{(\lambda, b)\}$,
- $aab = aa \circ b$ gives $\{(\lambda, b)\} \to \emptyset\,\{(a, \lambda)\}$,
- $aab = a \circ ab$ gives $\{(\lambda, b)\} \to \{(\lambda, b)\}\{(\lambda, \lambda)\}$.

Given K, F and a membership oracle for L we can thus simply write down a CBFG. However, in this case, the language $L(G_0)$ is not the same as L; moreover, the resulting grammar is not exact. Applying the rules for the recursive computation of f_G, we can see that $f_{G_0}(aab) = \{(\lambda, b)\}$ and $f_{G_0}(abb) = f_{G_0}(aabb) = \{(\lambda, b), (\lambda, \lambda)\}$ but $F_{L(G_0)}(abb) = \{(a, \lambda), (\lambda, b), (\lambda, \lambda)\}$ and thus G_0 is not exact. The problem here is caused by the fact that the production $\{(\lambda, b)\} \to \emptyset\,\{(a, \lambda)\}$ allows any string to occur in the place specified by the $\emptyset$: indeed since $\emptyset \subseteq f_{G_0}(aab)$ and $\{(a, \lambda)\} \subseteq f_{G_0}(b)$ the rule holds for $aabb$ and thus $\{(\lambda, b)\} \subseteq f_{G_0}(aabb)$. This is actually caused by the fact that there are no contexts in F that correspond to the string aa in K.

Fixing L for the moment, clearly the language defined depends on two factors: the set of strings K and the set of features F. Given K and F, and access to a membership oracle, we can write down a CBFG with almost no computation, but we still have the problem of finding suitable K and F – it might be that searching for exactly the right combination of K and F is intractably hard. It turns out that it is also very easy to find suitable sets.

In the next section we will establish two important lemmas that show that the search for K and F is fundamentally tractable: first, that as K increases the language defined by $G_0(K, L, F)$ will increase, and secondly that as F increases the language will decrease.

Let us consider one example that illustrates these properties. Consider the language $L = \{a^n b \mid n \geq 0\} \cup \{ba^m \mid m \geq 0\} \cup \{a\}$.

First, let $K = \{a, b, ab\}$ and $F = \{(\lambda, \lambda)\}$; then, by the definition of G_0, we have the following productions:

- $\{(\lambda, \lambda)\} \to a$

- $\{(\lambda, \lambda)\} \to b$
- $\{(\lambda, \lambda)\} \to \{(\lambda, \lambda)\}\{(\lambda, \lambda)\}$.

It is easy to see that $L(G_0) = \Sigma^+$.

Now, suppose that $F = \{(\lambda, \lambda), (\lambda, b)\}$ with K unchanged; then, by construction G_0 will have the following productions:

- $\{(\lambda, \lambda), (\lambda, b)\} \to a$,
- $\{(\lambda, \lambda)\} \to b$,
- $\{(\lambda, \lambda)\} \to \{(\lambda, \lambda), (\lambda, b)\}\{(\lambda, \lambda)\}$.

The language defined by G_0 contains $a^n b$ and also a^n since $\{(\lambda, \lambda)\} \subset \{(\lambda, \lambda), (\lambda, b)\}$ allowing the third production to accept strings ending with an a. Thus, the language has been reduced such that $L(G_0) = \{a^n b \mid n \geq 0\} \cup \{a^m \mid m \geq 0\}$.

We continue by leaving $F = \{(\lambda, \lambda), (\lambda, b)\}$ and we enlarge K such that $K = \{a, b, ab, ba\}$. The productions in G_0 are:

- $\{(\lambda, \lambda), (\lambda, b)\} \to a$,
- $\{(\lambda, \lambda)\} \to b$,
- $\{(\lambda, \lambda)\} \to \{(\lambda, \lambda), (\lambda, b)\}\{(\lambda, \lambda)\}$; the rule given by $ab = a \circ b$,
- $\{(\lambda, \lambda)\} \to \{(\lambda, \lambda)\}\{(\lambda, \lambda), (\lambda, b)\}$; the rule given by $ba = b \circ a$.

The addition of the last rule allows the grammar to recognize ba^n and it can be easily shown that by a combination of the last two productions $a^n b a^m$ belongs to the language defined by the grammar. Then, $L(G_0)$ has been increased such that $L(G_0) = \{a^n b a^k \mid n, k \geq 0\} \cup \{a^m \mid m \geq 0\}$.

In this example, the addition of $(\lambda, b), (a, \lambda)$ and (λ, a) to F and the addition of aab and baa to K will then define the correct language. In fact this illustrates one principle of our approach: in the infinite data limit, the construction G_0 will define the correct language. In the following lemma we abuse notation and use G_0 for when we have infinite K, and F: in this lemma we let K be the set of all non-empty strings and we let F be the set of all possible contexts $(\Sigma^* \times \Sigma^*)$. Recall that in this case for every string w $C_L(w) = F_L(w)$.

Lemma 3.4. For any language L, let $G = G_0(\Sigma^+, L, \Sigma^* \times \Sigma^*)$. For all $w \in \Sigma^+$ we have $f_G(w) = C_L(w)$ and therefore $L(G) = L$.

Proof. By induction on the length of w. If $|w| = 1$, and $w = a$ then there is a lexical production $C_L(a) \to a$ and by the definition of $f_G(a) = C_L(a)$. Suppose this is true for all w with $|w| \leq k$. Let w be some string of length $k+1$. Consider any split of w into u, v such that $w = uv$. $f_G(w)$ is the union over all these splits of a function. We will show that every such split will give the same result of $C_L(w)$. By inductive hypothesis $f_G(u) = C_L(u), f_G(v) = C_L(v)$. Since u, v, w are in $K = \Sigma^+$ we will also have an element of P of the form $C_L(w) \to C_L(u)C_L(v)$, so we know that $f_G(w) \supseteq F_L(w)$. We now show that f_G will not predict any extra contexts. Consider every production in P, $F_L(u'v') \to F_L(u')F_L(v')$, that applies to u, v, *i.e.* with $F_L(u') \subseteq f_G(u) = C_L(u)$ and $F_L(v') \subseteq f_G(v) = C_L(v)$. Lemma 3.2 shows that in this case $F_L(u'v') \subseteq F_L(w)$ and thus we deduce that $f_G(w) \subseteq F_L(w)$, which establishes the lemma. □

Informally if we take K to be every string and F to be every context, then we can accurately define any language. Of course, we are just interested in those cases where this can be defined finitely and we have a CBFG, in which case L will be decidable, but this infinite limit is a good check that the construction is sound.

3.4.2 Monotonicity Lemmas

We now prove two lemmas that show that the size of the language, and more particularly the features predicted will increase or decrease monotonically as a function of the basis K, and the feature set F, respectively. In fact, they give also a framework for approaching a target language from K and F.

Lemma 3.5. Suppose we have two CBFGs defined by $G = G_0(K, L, F)$ and $G' = G_0(K, L, F')$ where $F \subseteq F'$. Then for all u, $f_G(u) \supseteq f_{G'}(u) \cap F$.

Proof. Let G' have a set of productions P', P'_L, and G have a set of productions P, P_L. Clearly if $x \to yz \in P'$ then $x \cap F \to (y \cap F)(z \cap F)$ is in P by the definition of G_0, and likewise for P_L, P'_L. By induction on $|u|$ we can show that any feature in $f_{G'}(u) \cap F$ will be in $f_G(u)$. The base case is trivial since $F'_L(a) \cap F = F_L(a)$; if it is true for all strings up to length k, then if $f \in f_{G'}(u) \cap F$; there must be a production in F' with f on the head. By the inductive hypothesis, the right hand sides of the corresponding production in P will be triggered, and so f must be in $f_G(u)$. □

Corollary 3.6. Suppose we have two CBFGs defined by $G = G_0(K, L, F)$ and $G' = G_0(K, L, F')$ where $F \subseteq F'$; then $L(G) \supseteq L(G')$.

Proof. It is sufficient to remark that if $u \in L(G')$ then $(\lambda, \lambda) \in f_{G'}(u) \subseteq f_G(u)$ and thus $u \in L(G)$. □

Conversely, we can show that as we increase K, the language and the map f_G will increase. This is addressed by the next lemma.

Lemma 3.7. Suppose we have two CBFGs defined by $G = G_0(K, L, F)$ and $G' = G_0(K', L, F)$ where $K \subseteq K'$. Then for all u, $f_{G_0(K,L,F)}(u) \subseteq f_{G_0(K',L,F)}(u)$.

Proof. Clearly the sets of productions of $G_0(K, L, F)$ will be a subset of the set of productions of $G_0(K', L, F)$, and so anything that can be derived by the first can be derived by the second, again by induction on the length of the string. □

A simple result is that when K contains all of the substrings of a word, then $G_0(K, L, F)$ will generate all of the correct features for this word.

Lemma 3.8. For any string w, if $\texttt{Sub}(w) \subset K$, and let $G = G_0(K, L, F)$, then $F_L(w) \subseteq f_G(w)$.

Proof. By recursion on the size of w. Let $G = G_0(K, L, F) = \langle F, P, P_L, \Sigma \rangle$. First, notice that if w is of length 1 then we have $F_L(w) \to w$ in P_L and thus the lemma holds. Then suppose that $|w| = k \geq 2$. Let u and v in Σ^+ be such that $w = uv$. As $\texttt{Sub}(w) \subset K$ we have u, v in K. Therefore the rule $F_L(w) \to F_L(u)F_L(v)$ belongs to P. As $|u| < |w|$ and $|v| < |w|$, by recursion we get $F_L(u) \subseteq f_G(u)$ and $F_L(v) \subseteq f_G(v)$. Thus the rule can be applied and then $F_L(w) \subseteq f_G(w)$. □

In particular if $w \in L$, and $\texttt{Sub}(w) \subseteq K$, then $w \in L(G)$. This means that we can easily increase the language defined by G just by adding $\texttt{Sub}(w)$ to K. In general we do not need to add every element of $\texttt{Sub}(w)$ – it is enough to have one binary bracketing.

To establish learnability, we need to prove that for a target language L, if we have a sufficiently large F then $L(G_0(K, L, F))$ will be contained within L and that if we have a sufficiently large K, then $L(G_0(K, L, F))$ will contain L.

3.4.3 Fiducial Feature Sets and Finite Context Property

We need to be able to prove that for any K if we have enough features then the language defined will be included within the target language L. We formalize the idea of having enough features in the following way:

Definition 3.9. For a language L and a string u, a set of features F is fiducial on u if for all $v \in \Sigma^*$, $F_L(u) \subseteq F_L(v)$ implies $C_L(u) \subseteq C_L(v)$.

Note that if F is fiducial on u and $F \subset F'$ then F' is fiducial on u. Therefore we can naturally extend this to sets of strings.

Definition 3.10. For a language L and a set of strings K, a set of features F is fiducial on K if for all $u \in K$, F is fiducial on u.

Clearly, for any string w, $C_L(w)$ will be fiducial on w; but this is vacuous – we are interested in cases where there is a finite set of contexts which is fiducial for w,

but where $C_L(w)$ is infinite. If u and v are both in K then having the same features means they are syntactically congruent. However if two strings, neither of which are in K, have the same features this does not mean they are necessarily congruent (for instance if $F_L(v) = F_L(v') = \emptyset$). For non finite state languages, the set of congruence classes will be infinite, and thus we cannot have a finite fiducial set for the set of all strings in $\texttt{Sub}(L)$, but we can have a feature set that is correct for a finite subset of strings, or more generally for an infinite set of strings, if they fall into a finite number of congruence classes.

Let us consider our running example $L = \{a^n b^n | n > 0\}$. Take the string ab. $C_L(ab)$ is infinite and contains contexts of the form $(\lambda, \lambda), (a, b), (aa, bb)$ and so on. Consider a set with just one of these contexts, say $F = \{(a, b)\}$. This set is clearly fiducial for ab, since the only strings that will have this context are those that are congruent to ab. Consider now the string aab; clearly $\{(\lambda, b)\}$ is fiducial for aab, even though the string a, which is not congruent to aab, also occurs in this context. Indeed, this does not violate fiduciality since $C_L(a) \supset C_L(aab)$. However, looking at string a, $\{(\lambda, b)\}$ is not fiducial, since aab has this context but does not include all the contexts of a such as, for example, (λ, abb).

In these trivial examples, a context set of cardinality one is sufficient to be fiducial, but this is not the case in general. Consider the finite language $L = \{ab, ac, db, ec, dx, ey\}$, and the string a. It has two contexts (λ, b) and (λ, c) neither of which is fiducial for a on its own. However, the set of both contexts is: $\{(\lambda, b), (\lambda, c)\}$ is fiducial for a.

We now define the finite context property, which is one of the two conditions that languages must satisfy to be learnable in this model; this condition is a purely language theoretic property.

Definition 3.11. A language L has the Finite Context Property (FCP) if every string has a finite fiducial feature set.

Clearly if L has the FCP, then any finite set of substrings, K, has a finite fiducial feature set which will be the union of the finite fiducial feature sets for each element of K. If $u \notin \texttt{Sub}(L)$ then any set of features is fiducial since $C_L(u) = \emptyset$.

We note here that all regular languages have the FCP. We refer the reader to the Section 3.6.1.1 about CBFG and regular languages where the Lemma 3.20 and the associated construction proves this claim.

We can now state the most important lemma: this lemma links up the definition of the feature map in a CBFG, with the fiducial set of features to show that only correct features will be assigned to substrings by the grammar. It states that the features assigned by the grammar will correspond to the language theoretic interpretation of them as contexts.

Lemma 3.9. For any language L, given a set of strings K and a set of features F, let $G = G_0(K, L, F)$. If F is fiducial on K, then for all $w \in \Sigma^*$ $f_G(w) \subseteq F_L(w)$.

Proof. We proceed by induction on length of the string. *Base case:* strings of length 1. $f_G(w)$ will be the set of observed contexts of w, and since we have observed these contexts, they must be in the language. *Inductive step:* let w a string of length k.

Take a feature f on $f_G(w)$; by definition this must come from some production $x \to yz$ and a split u, v of w. The production must be from some elements of K, u', v' and $u'v'$ such that $y = F_L(u'), z = F_L(v')$ and $x = F_L(u'v')$. If the production applies this means that $F_L(u') = y \subseteq f_G(u) \subseteq F_L(u)$ (by inductive hypothesis), and similarly $F_L(v') \subseteq F_L(v)$. By fiduciality of F this means that $C(u') \subseteq C(u)$ and $C(v') \subseteq C(v)$. So by Lemma 3.2 $C(u'v') \subseteq C(uv)$. Since $f \in C(u'v')$ then $f \in C(uv) = C(w)$. Therefore, since $f \in F$ and $C(w) \cap F = F_L(w)$, $f \in F_L(w)$, and therefore $f_G(w) \subseteq F_L(w)$. □

Corollary 3.10. If F is fiducial on K then $L(G_0(K, F, L)) \subseteq L$.

Therefore for any finite set K from an FCP language, we can find a set of features so that the language defined by those features on K is not too big.

3.4.4 Kernel and Finite Kernel Property

We will now show a complementary result, namely that for a sufficiently large K the language defined by G_0 will include the target language. We will start by formalizing the idea that a set K is large enough, by defining the idea of a *kernel*.

Definition 3.12. A finite set $K \subseteq \Sigma^*$ is a kernel for a language L, if for any set of features F, $L(G_0(K, F, L)) \supseteq L$.

Consider again the language $L = \{a^n b^n | n \geq 0\}$. The set $K = \{a, b, ab\}$ is not a kernel, since if we have a large enough set of features, then the language defined will only be $\{ab\}$ which is a proper subset of L. However $K = \{a, b, ab, aab, abb, aabb\}$ is a kernel: no matter how large a set of features we have the language defined will always include L. Consider a language $L' = L \cup \{b^{16}\}$. In this case, a kernel for L' must include, as well as a kernel for L, some set of substrings of b^{16}: it is enough to have b^{16}, b^8, b^4, bb, b.

To prove that a set is a kernel, it suffices to show that if we consider all the possible features for building the grammar, we will contain the target language; any smaller set of features defines then a larger language. In our case, we can take the infinite set of all contexts and define productions based on the congruence classes. If F is the set of all contexts then we have $F_L(u) = C_L(u)$, thus the productions will be exactly of the form $C(uv) \to C(u)C(v)$. This is a slight abuse of notation since feature sets are normally finite.

Lemma 3.11. Let $F = \Sigma^* \times \Sigma^*$; if $L(G_0(K, L, F)) \supseteq L$ then K is a kernel.

Proof. By monotonicity of F: any finite feature set will be a subset of F. □

Not all context-free languages will have a finite kernel. For example $L = \{a^+\} \cup \{a^n b^m | n < m\}$ is clearly context-free, but does not have a finite kernel. Assume that the set K contains all strings of length less than or equal to k. Assume w.l.o.g. that the fiducial set of features for K includes all features (λ, b^i), where $i \leq k+1$. Consider the rules of the form $F_L(a^k) \to F_L(a^j)F_L(a^{k-j})$; we can see that no matter how large k is, the derived CBFG will under-generalize as a^k is not congruent to a^{k-1}.

Definition 3.13. A context-free grammar $G_T = \langle V, S, P, \Sigma \rangle$ has the Finite Kernel Property (FKP) iff for every non-terminal $N \in V$ there is a finite set of strings $K(N)$ such that $a \in K(N)$ if $a \in \Sigma$ and $N \to a \in P$ and such that for all $k \in K(N), N \stackrel{*}{\Rightarrow} k$ and where for every string $w \in \Sigma^*$ such that $N \stackrel{*}{\Rightarrow} w$ there is a string $k \in K(N)$ such that $C(k) \subseteq C(w)$. A CFL L has the FKP, if there is a grammar in CNF for it with the FKP.

Notice that all regular languages have the FKP since they have a finite number of congruence classes.

Lemma 3.12. Any context-free language with the FKP has a finite kernel.

Proof. Let $G_T = \langle V, S, P, \Sigma \rangle$ be such a CNF CFG with the FKP. Define

$$K(G_T) = \bigcup_{N \in V} \left(K(N) \cup \bigcup_{X \to MN \in P} K(M)K(N) \right). \tag{3.4}$$

We claim that $K(G_T)$ is a kernel. Assume that $F = \Sigma^* \times \Sigma^*$ and let G be such that $G = G_0(K(G_T), L(G_T), F) = \langle F, (\lambda, \lambda), P, P_L, \Sigma \rangle$.

We will show, by induction on the length of derivation of w in G_T, that for all N, w if $N \stackrel{*}{\Rightarrow} w$ then there is a k in $K(N)$ such that $f_G(w) \supseteq C(k)$. If length of derivation is 1, then this is true since $|w| = 1$ and thus $w \in K(N)$: therefore $C(w) \to w \in P_L$. Suppose it is true for all derivations of length less than j. Take a derivation of length j; say $N \stackrel{*}{\Rightarrow} w$. There must be a production in G_T of the form $N \to PQ$, where $P \Rightarrow^* u$ and $Q \Rightarrow^* v$, and $w = uv$. By inductive hypothesis; we have $f_G(u) \supseteq C(k_u)$ and $f_G(v) \supseteq C(k_v)$. By construction $k_u k_v \in K(G_T)$ and then there will be a rule $C(k_u k_v) \to C(k_u)C(k_v)$ in P. Therefore $f_G(uv) \supseteq C(k_u k_v)$. Since $N \stackrel{*}{\Rightarrow} k_u k_v$ there must be some $k_{uv} \in K(N)$ such that $C(k_{uv}) \subseteq C(k_u k_v)$. Therefore $f_G(w) \supseteq C(k_u k_v) \supseteq C(k_{uv})$.

Now we can see that if $w \in L$, then $S \stackrel{*}{\Rightarrow} w$, then there is a $k \in K(S)$ such that $f_G(w) \supseteq C(k)$ and $S \stackrel{*}{\Rightarrow} k$, therefore $(\lambda, \lambda) \in f_G(w)$ since $(\lambda, \lambda) \in C(k)$, thus $w \in L(G)$ and therefore K is a kernel. □

3.4.5 Learning Algorithm

Before we present the algorithm, we will discuss the learning model that we use. The class of languages that we will learn is superfinite and thus we cannot get a positive data only identification in the limit result [Gold, 1967]. The ultimate goal is to model the learning of natural languages, where negative data - that is, strings that are not in the target language, or equivalence queries are generally not available or are computationally impossible. Accordingly, we have decided to use the framework of positive data together with membership queries. The presented algorithm runs in time polynomial in the size of the sample S. We already discussed in Chapter 2 how this is not a strong enough result and the difficulties to have efficiency constraint to such a learning paradigm, in particular for representations, such as the ones in this chapter, that are powerful enough to define languages whose shortest strings are exponentially long. Accordingly we do not require in this model a polynomial dependence on the size of the representation. We note that the situation is unsatisfactory[3], but we do not intend to propose a solution in this chapter. We merely point out that the algorithm is genuinely polynomial and processes all of the data in the sample without "delaying tricks" of the type discussed by Pitt [1989].

Definition 3.14. A class of languages $\mathbb{L}$ is identifiable in the limit from positive data and a membership oracle with polynomial time and queries if and only if there exist two polynomials $p(), q()$ and an algorithm $\mathfrak{A}$ such that for all language $L \in \mathbb{L}$, for all texts S of L:

- $\mathfrak{A}$ returns a representation $G = \mathfrak{A}(S_n)$ in time $p(||S_n||)$.
- $\mathfrak{A}$ asks at most $q(||S_n||)$ queries to build $\mathfrak{A}(S_n)$.
- there exists an index N such that for all $n \geq N$: $\mathfrak{A}(S_n) = \mathfrak{A}(S_N)$ and $L(\mathfrak{A}(S_N)) = L$.

Before we present the algorithm we hope that it is intuitively obvious how the approach will work. Figure 3.2 diagrammatically shows the relationship between K and F. When we have a large enough K, we will be to the right of the vertical line; when we have enough features for that K we will be above the diagonal line. Thus the basis of the algorithm is to move to the right, until we have enough data, and then to move up vertically, increasing the feature set until we have a fiducial set.

We can now define our learning algorithm in Algorithm 3.1. Informally, D is the list of all strings that have been seen so far and G_n is the current grammar obtained with the first n strings of D. The algorithm uses two tests: one test is just to determine if the current hypothesis under-generalizes. This is trivial, since we have a positive

[3] A recent work from Yoshinaka [2015] proposed a set-driven polynomial approach that extends the one presented here. We will discuss this work in the last section of this HDR.

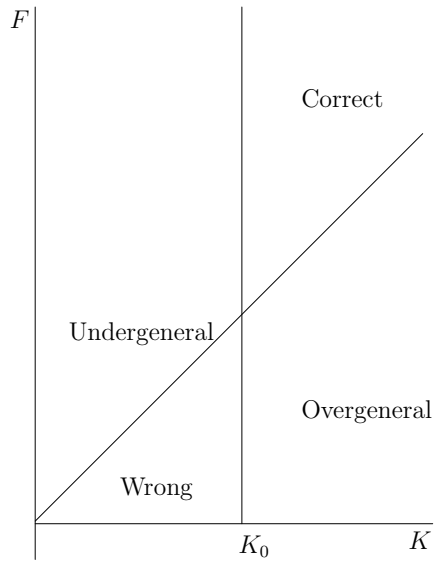

Figure 3.2: The relationship between K and F: The diagonal line is the line of fiduciality: above this line means that F is fiducial on K. K_0 is a kernel for the language.

presentation of the data, and so eventually we will be presented with a string in $L \setminus L(G_n)$. In this case we need to increase K; we accordingly increase K to the set of all substrings that we have observed so far. The second test is a bit more delicate. We want to detect if our algorithm overgeneralizes. This requires us to search through a polynomially bounded set of strings looking for a string that is in $L(G_n) \setminus L$. An obvious candidate set is $\texttt{Con}(D) \odot \texttt{Sub}(D)$; but though we conjecture that this is adequate, we have not yet been able to prove that is correct, as it might be that the over-generated string does not lie in $\texttt{Con}(L) \odot \texttt{Sub}(L)$.

Here we use a slightly stricter criterion: we try to detect whether F is fiducial for K: we search through a polynomially bounded set of strings, $\texttt{Sub}(D)$, to find a violation of the fiduciality condition. If we find such a violation, then we know that F is not fiducial for K, and so we increase F to the set of all contexts that we have seen so far, $\texttt{Con}(D)$.

In Algorithm 3.1, $G_0(K, \mathcal{O}, F)$ denotes the same construction as $G_0(K, L, F)$, except that we use membership queries with the oracle $\mathcal{O}$ to compute F_L for each element in K. We give the identification in the limit version of the algorithm, *i.e.* that admits an infinite positive presentation of strings in input.

Theorem 3.13. Algorithm 3.1 runs in polynomial time in the size of the sample, and makes a polynomial number of calls to the membership oracle.

Proof. The value of D will just be the set of observed strings; $\texttt{Sub}(D)$ and $\texttt{Con}(D)$ are

Algorithm 3.1: CBFG learning algorithm IIL

Data: A sequence of strings $S = \{w_1, w_2 \ldots, \}$, membership oracle $\mathcal{O}$
Result: A sequence of CBFGs $G_1, G_2, \ldots$
$K \leftarrow \emptyset$; $D \leftarrow \emptyset$; $F \leftarrow \{(\lambda, \lambda)\}$; $G = G_0(K, \mathcal{O}, F)$;
for w_i **do**
  $D \leftarrow D \cup \{w_i\}$; $C \leftarrow \texttt{Con}(D)$; $S \leftarrow \texttt{Sub}(D)$;
  if $\exists w \in D \setminus L(G)$ **then**
    $K \leftarrow S$; $F \leftarrow C$;
  else if $\exists v \in S, u \in K, f \in C$ *such that* $F_L(u) \subseteq F_L(v)$ *and* $f \odot u \in L$ *but* $f \odot v \notin L$ **then**
    $F \leftarrow C$;
  $G = G_0(K, \mathcal{O}, F)$;
  Output $G_i = G$;

both polynomially bounded by the size of the sample, and therefore so are $|K|$ and $|F|$. Therefore the number of calls to the oracle is clearly polynomial, as it is bounded by $|K||F|$. Computing G_0 is also polynomial, since $|P| \leq |K|^2$, and all strings involved are in $\texttt{Sub}(D)$. □

3.4.6 Identification in the limit result

In the following, we consider the class of context-free languages having the FCP and the FKP, represented by CBFG. K_n denotes the value of K at the n^{th} loop, and similarly for F, D.

Definition 3.15. $\mathcal{L}_{CFG}$ is the class of all context-free languages that satisfy the FCP and the FKP.

In what follows we assume that L is an element of this class, and that $w_1, \ldots, w_n, \ldots$ is a infinite presentation of the language. The proof is straightforward and merely requires an analysis of a few cases. We will proceed as follows: there are 4 states that the model can be in, that correspond to the four regions of the diagram in Figure 3.2.

1. K is a kernel and F is fiducial for K; in this case the model has converged to the correct answer. This is the region labeled *correct* in Figure 3.2.

2. K is a kernel and F is not fiducial for K: then $L \subseteq L(G)$, and at some later point, we will increase F to a fiducial set, and we will be in state 1: this is the region labeled *over-general*.

3. K is not a kernel and F is fiducial. Either $L(G) = L$, in which case we have converged to a correct answer or, if not, we will define a proper subset of the target language. In the later case we will change hypothesis at some later point, increase K to a kernel, and move to state 2 or state 1. This is the area labeled *under-general*.

4. K is not a kernel and F is not fiducial: in this case at some point we will move to states 1 or 2. This is the area labeled *wrong*.

We will start by making some basic statements about properties of the algorithm:

Lemma 3.14. If there is some n, such that F_n is fiducial for K_n and $L(G_n) = L$, then the algorithm will not change its hypothesis: i.e. for all $n > N$, $K_n = K_N, F_n = F_N$ and therefore $G_n = G_N$.

Proof. If $L(G_n)$ is correct, then the first condition of the loop will never be met; if F_n is fiducial for K_n, then the second condition will never be satisfied. □

Lemma 3.15. If there is some N such that K_N is a kernel, then for all $n > N$, $K_n = K_N$.

Proof. Immediate by definition of a kernel, and of the algorithm. □

We now prove that if F is not fiducial then the algorithm will be able to detect this.

Lemma 3.16. If there is some n such that F_n is not fiducial for K_n, then there is some index $n' \geq n$ at which F_n will be increased.

Proof. If F_n is not fiducial, then by definition there is some $u \in K$, $v \in \Sigma^+$ such that $F_L(u) \subseteq F_L(v)$, but there is an $f \in C_L(u)$ that is not in $C_L(v)$. By construction $F_L(u)$ is always non-empty, and so is $F_L(v)$. Thus $v \in \texttt{Sub}(L)$. Note $f \odot u \in L$, so $f \in \texttt{Con}(L)$. Let n' be the smallest index such that $v \in \texttt{Sub}(D_n)$ and $f \in \texttt{Con}(D_n)$: at this point, either F_n will have changed, or not, in which case it will be increased at this point. □

We now prove that we will always get a fiducial feature set.

Lemma 3.17. For any n, there is some n' such that $F_{n'}$ is fiducial for K_n.

Proof. If F_n is fiducial then $n' = n$ satisfies the condition. Assume otherwise. Let F be a finite set of contexts that is fiducial for K_n. We can assume that $F \subseteq \texttt{Con}(L)$. Let n_1 be the first index such that $\texttt{Con}(D_{n_1})$ contains F. At this point we are not sure that $F_{n_1} = \texttt{Con}(D_{n_1})$ since the conditions for changing the set of contexts may not be reached. Anyhow, if it is the case then F_{n_1} is fiducial, then $n_1 = n'$ satisfies the condition. If not, then by the preceding lemma, there must be some point n_2 at which we will increase the set of contexts of the current grammar; $F_{n_2} = \texttt{Con}(n_2)$ must contain F since $\texttt{Con}(D_{n_1}) \subset \texttt{Con}(D_{n_2})$, and is therefore fiducial, and so $n_2 = n'$ satisfies the condition. □

Lemma 3.18. For every positive presentation of an $L \in \mathcal{L}_{CFG}$, there is some n such that either the algorithm has converged to a correct grammar or K_n is a kernel.

Proof. Let m be the smallest number such that $\texttt{Sub}(D_m)$ is a kernel. Recall that any superset of a kernel is a kernel, and that all CFL with the FKP have a finite kernel (Lemma 3.12), and that such a kernel is a subset of $\texttt{Sub}(L)$, so such an m must exist.

Consider the grammar G_m; there are three possibilities:
1. $L(G_m) = L$, and F_m is fiducial, in which case the grammar has converged.
2. $L(G_m)$ is a proper subset of L and F_m is fiducial. Let m' be the first point at which $w_{m'}$ is in $L \setminus L(G_m)$; at this point $K_{m'}$ will be increased to include $\texttt{Sub}(D_m)$ and it will therefore be a kernel.
3. F_m is not fiducial: in this case by Lemma 3.17; there is some n at which F_n is fiducial for K_m. Either $K_n = K_m$ in which case this reduces to Case 2; or K_n is larger than K_m in which case it must be a kernel, since it will include $\texttt{Sub}(D_m)$ which is a kernel. □

We now can prove the main result of the chapter:

Theorem 3.19. Algorithm 3.1 identifies in the limit the class of context-free languages with the finite context property and the finite kernel property.

Proof. By Lemma 3.18 there is some point at which it converges or has a kernel. If K_n is a kernel then by Lemma 3.17, there is some point n' at which we have a fiducial feature set. Therefore $L(G_{n'}) = L$, and the algorithm has converged. □

3.4.7 Examples

We will now give a worked example of the algorithm.

Suppose $L = \{a^n b^n | n > 0\}$.

G_0 will be the empty grammar, with $K = \emptyset, F = \{(\lambda, \lambda)\}$ and an empty set of productions. $L(G_0) = \emptyset$.

1. Suppose $w_1 = ab$. $D = \{ab\}$. This is not in $L(G_0)$ so we set

   - $K = \texttt{Sub}(D) = \{a, b, ab\}$.
   - $F = \texttt{Con}(D) = \{(\lambda, \lambda), (a, \lambda), (\lambda, b)\}$.

   This gives us one production: $F_L(ab) \to F_L(a)F_L(b)$ which corresponds to $\{(\lambda, \lambda)\} \to \{(\lambda, b)\}\{(a, \lambda)\}$, and the lexical productions $\{(\lambda, b)\} \to a, \{(a, \lambda)\} \to b$. The language defined is thus $L(G_1) = \{ab\}$.

2. Suppose $w_2 = aabb$. $D = \{ab, aabb\}$. This is not in $L(G_1)$ so we set

   - $K = \texttt{Sub}(D) = \{a, b, ab, aa, bb, aab, abb, aabb\}$.

- $F = \texttt{Con}(D) = \{(\lambda, \lambda), (a, \lambda), (\lambda, b), (aa, \lambda), (a, b), (\lambda, bb), (aab, \lambda), (aa, b), (a, bb), (\lambda, abb)\}$.

We then have the following productions:

- $F_L(ab) \to F_L(a), F_L(b)$ which is $\{(\lambda, \lambda), (a, b)\} \to \{(a, bb), (\lambda, abb), (\lambda, b)\}, \{(aa, b), (aab, \lambda), (a, \lambda)\}$
- $F_L(aab) \to F_L(a), F_L(ab)$ which is $\{(a, bb), (\lambda, b)\} \to \{(a, bb), (\lambda, abb), (\lambda, b)\}, \{(\lambda, \lambda), (a, b)\}$
- $F_L(aab) \to F_L(aa), F_L(b)$ which is $\{(a, bb), (\lambda, b)\} \to \{(\lambda, bb)\}, \{(aa, b), (aab, \lambda), (a, \lambda)\}$
- $F_L(bb) \to F_L(b), F_L(b)$ which is $\{(aa, \lambda)\} \to \{(aa, b), (aab, \lambda), (a, \lambda)\}, \{(aa, b), (aab, \lambda), (a, \lambda)\}$
- $F_L(aa) \to F_L(a), F_L(a)$ which is $\{(\lambda, bb)\} \to \{(a, bb), (\lambda, abb), (\lambda, b)\}, \{(a, bb), (\lambda, abb), (\lambda, b)\}$
- $F_L(aabb) \to F_L(a), F_L(abb)$ which is $\{(\lambda, \lambda), (a, b)\} \to \{(a, bb), (\lambda, abb), (\lambda, b)\}, \{(aa, b), (a, \lambda)\}$
- $F_L(aabb) \to F_L(aa), F_L(bb)$ which is $\{(\lambda, \lambda), (a, b)\} \to \{(\lambda, bb)\}, \{(aa, \lambda)\}$
- $F_L(aabb) \to F_L(aab), F_L(b)$ which is $\{(\lambda, \lambda), (a, b)\} \to \{(a, bb), (\lambda, b)\}, \{(aa, b), (aab, \lambda), (a, \lambda)\}$
- $F_L(abb) \to F_L(a), F_L(bb)$ which is $\{(aa, b), (a, \lambda)\} \to \{(a, bb), (\lambda, abb), (\lambda, b)\}, \{(aa, \lambda)\}$
- $F_L(abb) \to F_L(ab), F_L(b)$ which is $\{(aa, b), (a, \lambda)\} \to \{(\lambda, \lambda), (a, b)\}, \{(aa, b), (aab, \lambda), (a, \lambda)\}$

and the two lexical productions:

- $F_L(a) \to a$ which is $\{(a, bb), (\lambda, abb), (\lambda, b)\} \to a$
- $F_L(b) \to b$ which is $\{(aa, b), (aab, \lambda), (a, \lambda)\} \to b$.

K is now a kernel and $L(G) = L$, but F is not fiducial for K, since (λ, bb) is not fiducial for aa (consider $aaab$).

3. Suppose $w_3 = aaabbb$. Now $|\texttt{Con}(D_3)| = 21$; there are now several elements of $\texttt{Con}(D_3)$ that are similar. For example $(\lambda, \lambda), (a, b)$ and (aa, bb) are identical but as it is harmless for the resulting grammar, it does not mind. Now we will detect that F is not fiducial: we will find $v = aaab$, $u = aa$ and $f = (\lambda, abbb)$; $F_L(aa) = \{(\lambda, bb)\} = F_L(aaab)$, but $f \odot aaab = aaababbb$ which is not in L. We will therefore increase F to be $\texttt{Con}(D_3)$, and then the algorithm will have converged. The final grammar will have 10 productions and 2 lexical productions; $|K| = 8$ and $|F| = 21$.

3.5 Practical Behavior of the Algorithm

In this section, we propose to study the behavior of our algorithm from a practical point of view. We focus more specifically on two important issues. The first one deals with the learning ability of the algorithm when the conditions for the theoretical learning result are not reached. Indeed, although the identification in the limit paradigm proves that with sufficient data it is possible to obtain exact convergence, it says nothing about the convergence when fewer learning examples are available: does the output get closer and closer to the target until it reaches it or does it stay far from the expected solution until receives enough data? The second question concerns the learning behavior of the algorithm: does it tend to over-generalize or to under-generalize?

For our experimental setup, we need to select appropriate datasets. In grammatical inference little has been done concerning benchmarking. The main available corpora are those of the on line competitions organized by the International Conference on Grammatical Inference. Three different competitions have recently taken place[4]: the *Abbadingo One* [Lang et al., 1998] which was about regular languages, the *Omphalos* competition on context-free languages [Starkie et al., 2004] and the *Tenjino* competition [Starkie et al., 2006] dealing with transducers learning. Note that some of these datasets correspond to extremely hard learning problems since their main objective was to push the state of the art (some problems of the *Abbadingo One* competition are still unsolved thirty years after its official end!)

However, these datasets can not be directly used for evaluating our algorithm because the solutions or the target models are not available. Our algorithm needs an oracle and thus we need a way to give answers to membership queries. In order to overcome this drawback, we chose to build synthetically some data sets following the experimental setup proposed by these competitions. More precisely, we decided to randomly generate target context-free grammars following what has been done for the *Omphalos* competition. Each grammar is then used either to generate training and test sets or as an oracle for answering membership queries.

In the following paragraphs we describe first the generation of the target context-free grammars, then the experimental setup with learning and test datasets used and finally the results and conclusions that can be drawn.

3.5.1 Generation of Target Context-free Grammars

To generate the target grammars we follow the process used for the *Omphalos* competition [Starkie et al., 2004]. We built 30 different grammars randomly according the following principles. For each grammar, we first fix the number of non-terminals and

[4]Three other competitions have been organized since this work: *Zulu* [Combe et al., 2010] on query learning, *PAutomaC* [Verwer et al., 2014] on learning probabilistic finite state models, and *SPiCe* [Balle et al., 2017] on guessing the next letter of a prefix.

terminals which are randomly chosen between 4 and 7 for the non-terminals (including the start symbol) and between 2 and 4 for terminal symbols. Then we randomly generate 20 context-free rules in Chomsky normal form such that every non-terminal appears at least once in the left hand side of a grammar rule. In order to avoid the presence of useless rules, we apply two simple procedures: if a non-terminal can not generate any terminal string, a new terminal rule generating one terminal symbol is created for this non-terminal; if a non-terminal can not be reached from the start symbol, we erase it from the grammar (*i.e.* we remove all the rules containing this non-terminal). From these grammars without useless rules, we force them to generate non finite languages by checking that the start symbol is used at least once in a right hand side of a grammar rule (in average this symbol appears in a right hand side of a rule more than 3 times per grammar).

The main difference with the *Omphalos* generation process is that we do not especially need non-regular languages. Indeed, one of the aim of these experiments is to give an idea on the behavior of the algorithm when its theoretical assumptions are not likely to be valid. From this standpoint, all randomly generated non-finite languages are good candidates as learning targets. However, with a similar principle used for the *Omphalos* competition, we checked that some of the generated grammars can not be easily solved by methods for regular languages. Although we can not decide if these grammars define non regular context-free languages, it ensures us that the target models are at least not too simple.

3.5.2 Experimental Setup

For each target grammar we generate a learning and a test sample following the *Omphalos* competition requirements. We build the learning sample by first creating a *structurally complete set* (see Definition 2.9) of strings for each grammar. Recall that this set is built such that for each rule of the target grammar, at least one string of the set can be derived using this rule [Parekh and Honavar, 1996]. This would guarantee that the complete learning set would have the minimal amount of information for finding the structure of the grammar. We then complete this learning set by randomly generating new strings from the grammar in order to have a total of 50 examples. We chose arbitrarily this value for two reasons: first it is sufficient to ensure that each sample strictly contains a structurally complete set for each target grammar and secondly we are likely to be far from the guarantees of the identification in the limit framework.

The construction of the test set needs particular attention. Since the learning phase uses a membership oracle, when the hypothesis is being constructed, some new strings may be built and queried for the oracle by picking a substring and a context from the learning sample. Thus, even if the test set does not contain any string of the learning sample, the construction G_0 may consider some strings present in the test set. In order

to avoid this drawback, *i.e.* to guarantee that no string of the test could be seen during the construction of the CBFG, each test string has a length of at least 3 times the maximal length of the strings in the learning set, which is by construction the maximal size of the strings queried. According to this procedure, we randomly generate a test set of 1000 strings over the alphabet of terminal symbols used to define the target grammar (1000 examples is twice the size of the small test sets of the *Omphalos* competition). The test sequences are then labeled positive or negative depending on their membership to the language defined by the grammar. We repeat this process until we have the desired number of strings. The ratio between strings in the language and strings outside the language is fixed to be between 40% and 60%.

In order to study the behavior of our algorithm, we define the following setup. For each target context-free grammar, we construct a CBFG by applying the construction $G_0(K, \mathcal{O}, F)$ with $K = \texttt{Sub}(S)$ and $F = \texttt{Con}(S)$ where S is a set of strings drawn from the learning set and using the target grammar as the oracle $\mathcal{O}$ for the membership queries. We generate different sets S by drawing an increasing number of learning examples (from 2 to 50) from the learning sample of the considered grammar. Then, we evaluate the learned CBFG on the test sample by measuring the accuracy of correct classification. We present the results averaged on the 30 test sets of the different target context-free grammars.

3.5.3 Results and Discussion

Figure 3.3 shows the averaged accuracy over the different target grammars according to the number of strings in the learning sample. We can note that a high correct classification rate (nearly 90%) is reached with 20 examples and with only 5 examples an accuracy of 75% is obtained. These results indicate that a relevant hypothesis can be found even with few examples. The standard deviations represented by vertical bars show a good stability of the results from learning sets of 20 strings. This confirms that our algorithm is able to learn partly correct representations even when learning sets may not have a kernel or a fiducial learning set and thus are far from the identification in the limit assumptions.

The analysis of the behavior of the algorithm in terms of false positive and false negative rates is shown in Table 3.1. The proportion of false negatives (*i.e.* positive strings classified as negative) is significantly[5] higher than the proportion of false positives (*i.e.* negative strings classified as positive), whatever the size of the learning sample is. Thus the output of the algorithm tends more to under-generalize than the converse. As it is generally admitted that over-generalization is the main trouble when learning from positive examples, this tendency confirms that the algorithm behaves well. However, it is difficult to draw firm conclusions without a natural distribution over negative examples.

[5]The significance was assessed by a Student paired t-test with a risk of 1%.

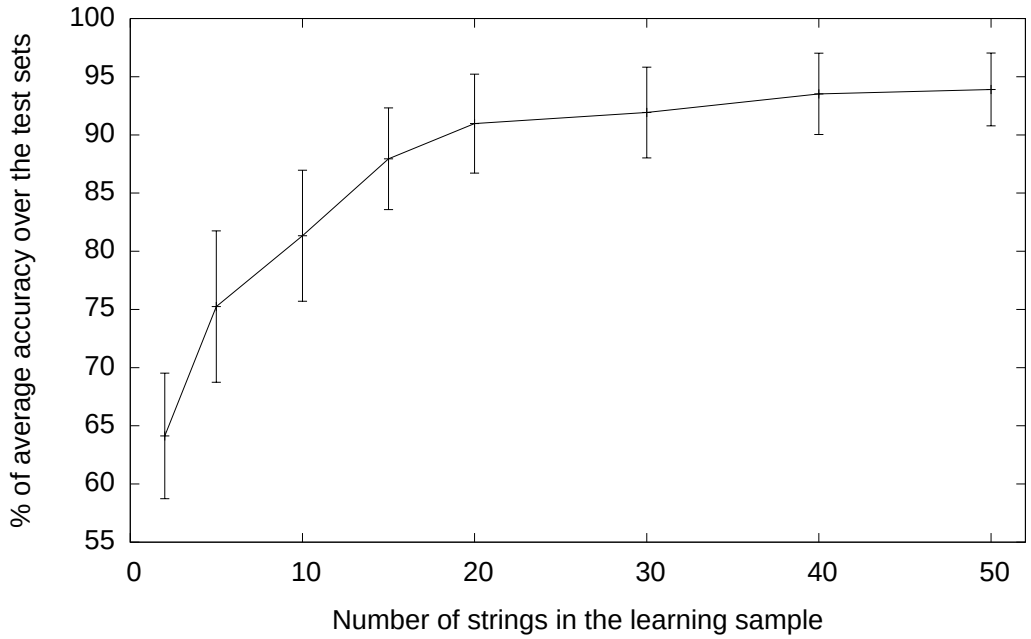

Figure 3.3: Evolution of the average percentage of correct classification according to the number of learning examples.

The preceding results show that despite its simplicity the algorithm behaved nicely during these experiments, in particular concerning over-generalization. We focus now on the amount of queries needed by the algorithm for building the CBFG. The growth of the number of requested queries according to the average size of the learning sample is shown in Figure 3.4 (recall that here the size of the sample means the sum of the string lengths of the sample). While a very worst case analysis of the grammar construction used by G_0 would lead to a complexity in $O(|S|^5)$, we can observe that the number of queries seems to be quadratic, at least in the case of the grammars we consider here. However, the volume of queries used is large, which can be explained by the simplicity of the algorithm. From a practical standpoint, it is clear that much work has to be done in order to try to minimize the number of queries needed by selecting the most informative examples, but this point is out of the scope of the chapter.

Finally, we can note that these experiments suffer of the lack of comparison with other approaches. This is due to the fact that, at the time we work on that subject, no other tractable algorithm used a positive learning sample and a membership oracle only. Indeed, since the work of Angluin [1988] about the Minimum Adequate Teacher all algorithms using membership queries were designed with the additional help of equivalence queries. The point of view adopted in this chapter is rather theoretical since our aim was to show the relevance of CBFG representations for language learning. However,

Number of strings in S	false positive	false negative
02	07.2 % ± 12.3	36.4 % ± 7.0
05	04.5 % ± 4.7	28.4 % ± 9.1
10	03.8 % ± 2.9	22.4 % ± 8.4
15	03.9 % ± 2.6	14.1 % ± 6.6
20	04.1 % ± 3.0	09.4 % ± 6.1
30	04.2 % ± 2.8	07.9 % ± 5.5
40	03.9 % ± 2.6	05.6 % ± 4.7
50	04.4 % ± 1.6	04.8 % ± 3.9

Table 3.1: Average percentage of false positive and false negative rates obtained over the test samples.

a perspective of this work is to try to avoid the use of the oracle (by using statistical or simulation methods) which will allow us to compare more easily our approach with other methods.

3.6 Expressiveness of CBFG

In this section, we compare the expressiveness of CBFG with other well known representations. As noted earlier, we are primarily interested in the class of exact CBFGs – these are CBFGs where the presence of a contextual feature in the representation corresponds exactly to the language theoretic interpretation of the context. The class of unrestricted CBFG is significantly larger, but less relevant.

The algorithm presented in this chapter cannot learn the entire class of exact CBFGs, but we conjecture that there are more powerful algorithms that can[6].

3.6.1 Exact CBFGs and the Chomsky Hierarchy

We start by examining the class of languages defined by exact CBFGs. We will show that this class

- includes all regular languages
- does not include all context free languages
- includes some non-context-free languages.

This class is thus orthogonal to the Chomsky hierarchy.

[6]This conjecture was proven true afterwards by Yoshinaka [2015] that we will discuss in the last section of this chapter.

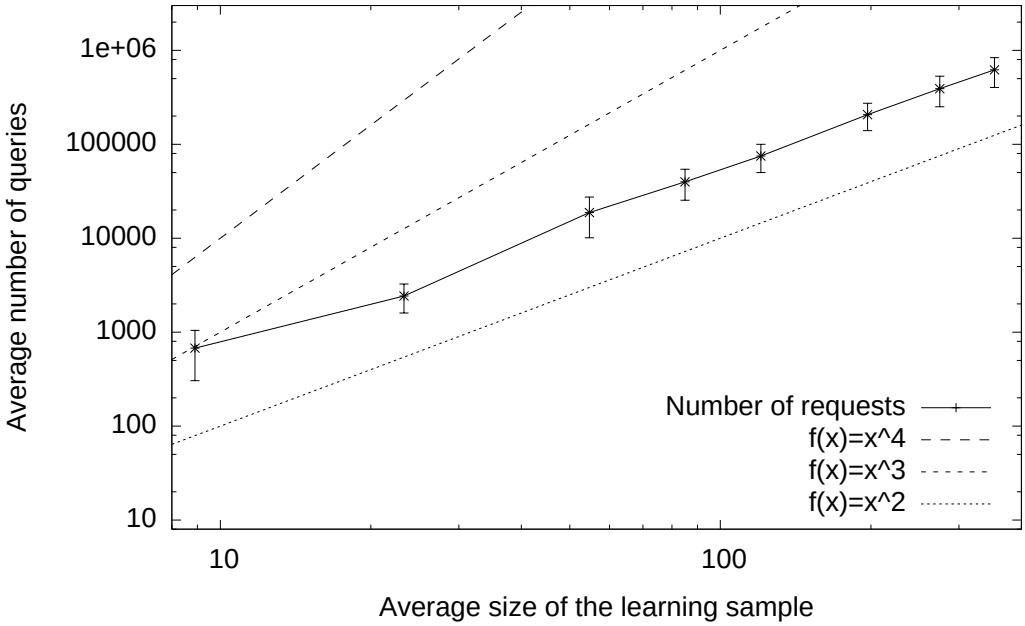

Figure 3.4: Growth of the number of membership queries versus the average of the total size of the learning sample (using log log scale).

3.6.1.1 Regular Languages

Any regular language can be defined by an exact CBFG. We will show a way of constructing an exact CBFG for any regular language. Suppose we have a regular language L: we consider the left and right residual languages:

$$u^{-1}L = \{w | uw \in L\}, \tag{3.5}$$

$$Lu^{-1} = \{w | wu \in L\}. \tag{3.6}$$

For any $u \in \Sigma^*$, let $l_{min}(u)$ be the lexicographically shortest element such that $l_{min}^{-1}L = u^{-1}L$. The number of such l_{min} is finite by the Myhill-Nerode theorem, we denote by L_{min} this set, i.e. $\{l_{min}(u) | u \in \Sigma^*\}$. We define symmetrically R_{min} for the right residuals ($Lr_{min}^{-1} = Lu^{-1}$).

We define the set of contexts as:

$$F(L) = L_{min} \times R_{min}. \tag{3.7}$$

$F(L)$ is clearly finite by construction.

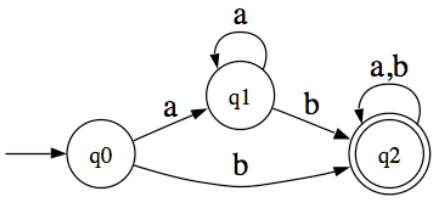

Figure 3.5: Example of a DFA. The left residuals are defined by $\lambda^{-1}L$, $a^{-1}L$, $b^{-1}L$ are the right ones by $L\lambda^{-1}$, Lb^{-1}, Lab^{-1} (note here that $La^{-1} = L\lambda^{-1}$).

If we consider the regular language defined by the deterministic finite automata of Figure 3.5, we obtain $L_{min} = \{\lambda, a, b\}$ and $R_{min} = \{\lambda, b, ab\}$ and thus
$F(L) = \{(\lambda, \lambda), (a, \lambda), (b, \lambda), (\lambda, b), (a, b), (b, b), (\lambda, ab), (a, ab), (b, ab)\}$.

By considering this set of features, we can prove the following lemma:

Lemma 3.20. For any strings u, v such that $F_L(u) \supset F_L(v)$ then $C_L(u) \supset C_L(v)$.

Proof. Suppose $F_L(u) \supset F_L(v)$ and let (l, r) be a context in $C_L(v)$. Let l' be the lexicographically shortest element of $\{u : u^{-1}L = l^{-1}L\}$ and r' the lexicographically shortest element of $\{u : Lu^{-1} = Lr^{-1}\}$. By construction we have $(l', r') \in F(L)$ and $l'vr' \in L$, as $vr' \in l'^{-1}L = l^{-1}L$. $F_L(v)$ is contained in $F_L(u)$ therefore we have $l'ur' \in L$. $l'^{-1}L = l^{-1}L$ implies $lur' \in L$. As r' is congruent to r, $lur \in L$. □

This lemma means that the set of features F is sufficient to represent context inclusion.

Note that the number of congruence classes of a regular language is finite. Each congruence class is represented by a set of contexts $F_L(u)$. Let K_L be finite set of strings formed by taking the lexicographically shortest string from each congruence class. The final grammar can be obtained by combining elements of K_L. For every pair of strings $u, v \in K_L$, we define a rule

$$F_L(uv) \to F_L(u)F_L(v) \tag{3.8}$$

and we add lexical productions of the form $F_L(a) \to a$, $a \in \Sigma$.

The following lemma shows the correctness and the exactness of the grammar.

Lemma 3.21. For all $w \in \Sigma^*$, $f_G(w) = F_L(w)$.

Proof. (Sketch) The proof is in two steps: $f_G(w) \subseteq F_L(w)$ and $F_L(w) \subseteq f_G(w)$. Each step is made by induction on the length of w and uses the rules created to build the grammar, the derivation process of a CBFG and the fiduciality for the second step.

First, we show $\forall w \in \Sigma^*, f_g(w) \subseteq F_L(w)$ by induction on the length of w. For $|w| = 1$, the inclusion is trivial since all the lexical rules $F_L(a) \to a$ are included in the grammar. Suppose that a string w, $|w| = n > 1$, is parsed by the CBFG G, then there exists a cut of w in $uv = w$ and a rule $z \to xy$ in G such that $x \subseteq f_G(u)$ and $y \subseteq f_G(v)$. By induction hypothesis, $x \subseteq F_L(u)$ and $y \subseteq F_L(v)$. By construction of the grammar, there exists two strings $u', v' \in K_L$ such that u, resp. v, belongs to same congruence class than u', resp. v' and the rule $F_L(u'v') \to F_L(u')F_L(v')$ belongs to the productions of the grammar. By induction hypothesis, $x \subseteq F_L(u) = F_L(u')$ and $y \subseteq F_L(v) = F_L(v')$ and thus $f_G(w) \subseteq F_L(w)$.
Second, we prove that $\forall w \in \Sigma^*, F_L(w) \subseteq f_G(w)$ by induction on the length of w. The key point relies on the fact that when a string w is parsed by a CBFG G, there exists a cut of w into $uv = w$ (u, $v \in \Sigma^*$) and a rule $z \to xy$ in G such that $x \subseteq f_G(u)$ and $y \subseteq f_G(v)$. The rule $z \to xy$ is also obtained from a substring from the set used to build the grammar using the F_L function. By the inductive hypothesis we obtain inclusion between f_G and F_L on u and v. □

For the language of Figure 3.5, the following set is sufficient to build an exact CBFG: $\{a, b, aa, ab, ba, aab, bb, bba\}$ (this corresponds to all the substrings of aab and bba). We have:
$F_L(a) = F(L)\backslash\{(\lambda, \lambda), (a, \lambda)\} \to a$,
$F_L(b) = F(L) \to b$,
$F_L(aa) = F_L(a) \to F_L(a)F_L(a)$,
$F_L(ab) = F(L) \to F_L(a)F_L(b) = F_L(a)F(L)$,
$F_L(ba) = F(L) \to F_L(b)F_L(a) = F(L)F_L(a)$,
$F_L(bb) = F(L) \to F_L(b)F_L(b) = F(L)F(L)$,
$F_L(aab) = F_L(bba) = F_L(ab) = F_L(ba)$.

The approach presented here gives a canonical form for representing a regular language by an exact CBFG. Moreover, this is is *complete* in the sense that every context of every substring will be represented by some element of F: this CBFG will completely model the relation between contexts and substrings.

3.6.1.2 Exact CBFGs do not include all CFLs

First, it is clear that the class of exact CBFGs includes some non-regular context-free languages: the grammar defined in Section 3.3.3 is an exact CBFG for the context-free and non regular language $\{a^n b^n | n > 0\}$, showing the class of exact CBFG has some elements properly in the class of CFGs.

We give now a context-free language L that can not be defined by an exact CBFG:

$$L = \{a^n b | n > 0\} \cup \{a^n c^m | m > n > 0\}.$$

Suppose that there exists an exact CBFG that recognizes it and let N be the length of the biggest feature (*i.e.* the longest left part of the feature). For any sufficiently large $k > N$, the sequences c^k and c^{k+1} share the same features: $F_L(c^k) = F_L(c^{k+1})$. Since the CBFG is exact we have $F_L(b) \subseteq F_L(c_k)$. Thus any derivation of $a^{k+1}b$ could be a derivation of $a^{k+1}c^k$ which does not belong to the language.

However, this restriction does not mean that the class of exact CBFG is too restrictive for modeling natural languages. Indeed, the example we have given is highly unnatural and such phenomena appear not to occur in attested natural languages.

3.6.1.3 CBFG and Non Context-Free Languages

CBFGs are more powerful than CFGs in two respects. First, CBFGs can compactly represent languages like the finite language of all $n!$ permutations of an n-letter alphabet, that have no concise representation as a CFG [Asveld, 2006]. Secondly, as we now show, there are some exact CBFGs that are not context-free. In particular, we define a language closely related to the *MIX language* (consisting of strings with an equal number of a's, b's and c's in any order) which is known to be non context-free, and indeed is conjectured to be outside the class of indexed grammars [Boullier, 2003].

Let $M = \{\{a, b, c\}^+\}$, the set of all strings of length at least one that can be built on the alphabet $\{a, b, c\}$. We consider now the language
$L = L_{abc} \cup L_{ab} \cup L_{ac} \cup \{a'a, b'b, c'c, dd', ee', ff'\}$:
$L_{ab} = \{wd | w \in M, |w|_a = |w|_b\}$,
$L_{ac} = \{we | w \in M, |w|_a = |w|_c\}$,
$L_{abc} = \{wf | w \in M, |w|_a = |w|_b = |w|_c\}$.
In order to define a CBFG recognizing L, we have to select features (contexts) that can represent exactly the intrinsic components of the languages composing L. We propose to use the following set of features for each sub-language:

- For L_{ab}: (λ, d) and $(\lambda, ad), (\lambda, bd)$.
- For L_{ac}: (λ, e) and $(\lambda, ae), (\lambda, ce)$.
- For L_{abc}: (λ, f').
- For the letters a', b', c', a, b, c we add: $(\lambda, a), (\lambda, b), (\lambda, c), (a', \lambda), (b', \lambda), (c', \lambda)$.
- For the letters d, e, f, d', e', f' we add; $(\lambda, d'), (\lambda, e'), (\lambda, f'), (d, \lambda), (e, \lambda), (f, \lambda)$.

Here, L_{ab} will be represented by (λ, d), but we will use $(\lambda, ad), (\lambda, bd)$ to define the internal derivations of elements of L_{ab}. The same idea holds for L_{ac} with (λ, e) and $(\lambda, ae), (\lambda, ce)$.

For the lexical rules and in order to have an exact CBFG, note the special case for a, b, c:
$\{(\lambda, bd), (\lambda, ce), (a', \lambda)\} \to a$
$\{(\lambda, ad), (b', \lambda)\} \to b$
$\{(\lambda, ae), (c', \lambda)\} \to c$
For the nine other letters, each one is defined with only one context, for example using the rule $\{(\lambda, d')\} \to d$.

For the production rules, the most important one is: $(\lambda, \lambda) \to \{(\lambda, d), (\lambda, e)\}, \{(\lambda, f')\}$.

Indeed, this rule, with the presence of two contexts in one of categories, means that an element of the language has to be derived so that it has a prefix u such that $f_G(u) \supseteq \{(\lambda, d), (\lambda, e)\}$. This means u is both an element of L_{ab} and L_{ac}. This rule represents the language L_{abc} since $\{(\lambda, f')\}$ can only represent the letter f.

The other parts of the language will be defined by the following rules:
$(\lambda, \lambda) \to \{(\lambda, d)\}, \{(\lambda, d')\}$,
$(\lambda, \lambda) \to \{(\lambda, e)\}, \{(\lambda, e')\}$,
$(\lambda, \lambda) \to \{(\lambda, a)\}, \{(\lambda, bd), (\lambda, ce), (a', \lambda)\}$,
$(\lambda, \lambda) \to \{(\lambda, b)\}, \{(\lambda, ad), (b', \lambda)\}$,
$(\lambda, \lambda) \to \{(\lambda, c)\}, \{(\lambda, ae), (c', \lambda)\}$,
$(\lambda, \lambda) \to \{(\lambda, d')\}, \{(d, \lambda)\}$,
$(\lambda, \lambda) \to \{(\lambda, e')\}, \{(e, \lambda)\}$,
$(\lambda, \lambda) \to \{(\lambda, f')\}, \{(f, \lambda)\}$.

This set of rules is incomplete, since for representing L_{ab}, the grammar must contain the rules ensuring to have the same number of a's and b's, and similarly for L_{ac}. To lighten the presentation here, the complete grammar is presented in Appendix.

We claim this is an exact CBFG for a context-sensitive language. L is not context-free since if we intersect L with the regular language $\Sigma^* d$, we get an instance of the non context-free MIX language (with d appended). The exactness comes from the fact that we chose the contexts in order to ensure that strings belonging to a sub-language can not belong to another one and that the derivation of a substring will provide all the possible correct features with the help of the union of all the possible derivations.

Note that the MIX language on its own is not definable by an exact CBFG: it is only when other parts of the language can distributionally define the appropriate partial structures that we can get context sensitive languages. Far from being a limitation of this formalism (a bug), we argue this is a feature: it is only in rather exceptional circumstances that we will get properly context sensitive languages. This formalism thus potentially accounts not just for the existence of non context-free natural languages but also for their rarity.

3.6.2 Inexact CBFGs

We are less interested in the class of all CBFGs: these are CBFGs where the contexts are just arbitrary features and there is no relation between $f_G(u)$ and $C_L(u)$ except for the presence of (λ, λ). However, it is important to understand the language theoretic power of this class as this upper bounds the hypothesis class of the algorithm, and is easier to analyze.

3.6.2.1 Context-free grammars

First, we note that this class contains all context-free languages. Given a context-free language, that does not include the empty string, we can take a CFG in Chomsky normal form and convert it directly into a CBFG. Let V be the set of non-terminals of such a CFG. We pick an arbitrary set of distinct contexts to represent the elements of V, subject only to the constraint that S corresponds to (λ, λ). Let $C(N)$ be the context corresponding to the non-terminal N. For every production rule in the CFG of the form $N \to PQ$, we add a CBFG production $\{C(N)\} \to \{C(P)\}, \{C(Q)\}$. For every production in the CFG of the form $N \to a$, we add a CBFG production to P_L of the form $\{C(N)\} \to a$. It is easy to see that this will define the same language.

3.6.2.2 Range Concatenation Grammars

While CBFG formalism has some relationship to a context-free grammar, and some to a semi-Thue system (also known as a string rewriting system), it is not formally identical to either of these. One exact equivalence is to a restricted subset of Range Concatenation Grammars; a very powerful formalism [Boullier, 2000]. We include the following relationship, but suggest that the reader unfamiliar with RCGs proceeds to the discussion of the relationship with the more familiar class of context-free grammars.

Lemma 3.22. For every CBFG G, there is a non-erasing positive range concatenation grammar of arity one, in 2-var form that defines the same language.

Proof. Suppose $G = \langle F, P, P_L, \Sigma \rangle$. Define a RCG with a set of predicates equal to F and the following clauses, and the two variables U, V. For each production $x \to yz$ in P, for each $f \in x$, where $y = \{g_1, \dots g_i\}$, $z = \{h_1, \dots h_j\}$ add clauses

$$f(UV) \to g_1(U), \dots g_i(U), h_1(V), \dots h_j(V).$$

For each lexical production $\{f_1 \dots f_k\} \to a$ add clauses

$$f_i(a) \to \epsilon.$$

It is straightforward to verify that $f(w) \vdash \epsilon$ iff $f \in f_G(w)$. □

3.6.2.3 Conjunctive Grammar

A tighter correspondence is to the class of Conjunctive Grammars [Okhotin, 2001], invented independently of RCGs.

Definition 3.16. A conjunctive grammar is defined as a quadruple $\langle \Sigma, N, P, S \rangle$, in which: Σ is the alphabet; N is the set of non terminal symbols; P is the set of rules, each of the form $A \to \alpha_1 \& ... \& \alpha_m$, where $A \in V$ and $\forall i < m$, $\alpha_i \in (V \cup \Sigma)^*$; $S \in N$ is the start symbol.

In this formalism, a string w is derived from $A \in V$ iff there exists a rule $A \to \alpha_1 \& ... \& \alpha_m$ in P and for all $i \leq m$, α_i derives w.

We claim that for every every language L generated by a conjunctive grammar there is a CBFG representing $L\#$ (where the special character $\#$ is not included in the original alphabet).

Suppose we have a conjunctive grammar $G = \langle \Sigma, N, P, S \rangle$ in binary normal form (as defined in [Okhotin, 2003]). We construct the equivalent CBFG $G' = \langle F, P', P_L, \Sigma \rangle$ as followed:

- For every letter a we add a context (l_a, r_a) to F such that $l_a a r_a \in L$;
- For every rules $X \to a$ in P, we create a rule $\{(l_a, r_a)\} \to a$ in P_L.
- For every non terminal $X \in N$, for every rule $X \to P_1Q_1 \& \ldots \& P_nQ_n$ we add distinct contexts $\{(l_{P_iQ_i}, r_{P_iQ_i})\}$ to F, such that for all i it exists u_i, $l_{P_iQ_i} u_i r_{P_iQ_i} \in L$ and $P_iQ_i \overset{*}{\Rightarrow}_G u_i$;
- Let $F_{X,j} = \{(l_{P_iQ_i}, r_{P_iQ_i}) : \forall i\}$ the set of contexts corresponding to the j^{th} rule applicable to X. For all $(l_{P_iQ_i}, r_{P_iQ_i}) \in F_{X,j}$, we add to P' the rules $(l_{P_iQ_i}, r_{P_iQ_i}) \to F_{P_i,k} F_{Q_i,l}$ $(\forall k, l)$.
- We add a new context (w, λ) to F such that $S \overset{*}{\Rightarrow}_G w$ and $(w, \lambda) \to \#$ to P_L;
- For all j, we add to P' the rule $(\lambda, \lambda) \to F_{S,j}\{(w, \lambda)\}$.

It can be shown that this construction gives an equivalent CBFG.

3.7 Discussion and Conclusion

One of the main objective of our approach is to provide a framework that helps to bridge the gap between theoretical methods of grammatical inference and the structured representations required in linguistics. We provide a conclusion and a discussion of our work according to these two standpoints, and we finish this chapter by giving a brief survey of the works that continued the ideas presented here.

3.7.1 Grammatical Inference

In this chapter, we have presented a new formalism, the *Contextual Binary Feature Grammars*, and shown its relevance for representing a large class of languages. We have proposed a learning algorithm using only membership queries and shown that this algorithm can identify in the limit the class of context-free languages satisfying the FCP and FKP assumptions. First of all, we should establish how large the class of languages with the FCP and the FKP is: it includes all finite languages and all regular languages, since the set of congruence classes is finite for finite state languages. It similarly includes the context-free substitutable languages, [Clark and Eyraud, 2007], since every string in a substitutable language belongs to only one syntactic congruence class. As already stated it does not include all CFLs since not all CFLs have the FCP and/or the FKP. However it does include languages like the Dyck languages of arbitrary order, Lukacevic language and most other classic simple examples. As a special case consider the equivalence relation between contexts $f \cong_L f'$ iff $\forall u$ we have that $f \odot u \in L$ iff $f' \odot u \in L$. The class of CFLs where the context distribution of every string is a finite union of equivalence classes of contexts clearly has both the FKP and the FCP.

If we now focus on the algorithm proposed: it is relatively simple but has two main drawbacks. First, the algorithm is not conservative since once we have found the correct language, the representation may change – if the feature set found is not fiducial – until the fiduciality is reached. Second, the CBFG output by the algorithm may not be consistent with some answers provided by the oracle. Indeed, when the algorithm checks the fiduciality of the feature set F, the membership of new strings is tested. These strings do not appear in the list of learning examples given to the oracle but are built from all the possible contexts and substrings that can be extracted from this list. Then, it is possible that, among these new strings, some of them belong to the target language but are not recognized by the current grammar. In this case, the output grammar is nevertheless not modified. We can imagine a procedure that changes the grammar by adding these new positive strings for building the CBFG, however this could lead to having to deal with an exponential number of strings. Thus, a more reasonable procedure is to wait for these strings in the positive data presentation. One proposal for future work, from these two remarks, is a new learning algorithm that overcomes these drawbacks.

Our approach to context-free grammatical inference is based on a generalization of distributional learning, following the work of Clark and Eyraud [2007]. The state of the art in context-free inductive inference from flat unstructured examples only was rather limited at the time of this work. When learning from stochastic data or using a membership oracle, it is possible to have powerful results, if we allow exponential computation (see for example [Horning, 1969]). The main contribution of this work is thus to show that efficient learning is possible, with an appropriate representation.

The presented work relies on using a membership oracle, but under suitable assump-

tions about distributions, it could be possible to get a PAC-learning result for this class along the lines of Clark [2006] or Shibata and Yoshinaka [2016], placing some bounds on the number of features required.

3.7.2 Linguistics

The field of grammatical inference has close relations to the study of language acquisition. Attempts to model natural languages with context-free grammars require additional machinery: natural language categories such as noun phrases contain many overlapping subclasses with features such as case, number, gender and similarly for verbal categories. Modeling this requires either an exponential explosion of the number of non-terminals employed or a switch to a richer set of features. Our formalism can be seen as a first step to integrate such features. While we have implemented the algorithm described here, and verified that it works in accordance with theory on small artificial examples, there are a number of modifications that would need to be made before it can be applied to real grammar induction on natural language. First, the algorithm is very naive; in practice a more refined algorithm could select both the kernel and the feature set in a more sophisticated way. Secondly, considering features that correspond to individual contexts may be too narrow a definition for natural language given the well known problems of data sparseness and it will be necessary to switch to features corresponding to sets of contexts, which may overlap. Thus for example one might have features that correspond to sets of contexts of the form $F(u, v) = \{(lu, vr)|l, r \in \Sigma^*\}$. This would take this approach closer to methods that have been shown to be effective in unsupervised learning in NLP [Klein and Manning, 2004] where typically $|u| = |v| = 1$. In any event, we think such modifications will be necessary for the acquisition of non context-free languages. Finally, at the moment the algorithm has polynomial update time, but in the worst case, there are deterministic finite state automata such that the size of the smallest kernel will be exponential in the number of states. There are, however, natural algorithms for generalizing the productions by removing features from the right hand sides of the rules; this would have the effect of accelerating the convergence of the algorithm, and removing the requirement for the Finite Kernel Property.

3.7.3 Following works

As already mentioned, the work presented in this chapter is the starting point of several other positive results that have been obtained since its first publication.

First, the idea of using the properties of context inclusion as the semantic of non-terminals for grammars has been proven crucial in distributional learning. Actually, this led to two parallel paths of research: approaches that define this semantics by strings are called primal and those by contexts are dual [Clark and Yoshinaka, 2016]. By relying at the same time on strings (for the decomposition of non-terminals in right handsides

of rules) and on contexts (for the semantic of non-terminals, which are set of contexts), CBFG are the only example of a mixed approach. Recent works have shown that not confusing the two gives better control during the inference and a better characterization of the learnable classes. The study of the *syntactic concept lattice* [Clark, 2015] complete the understanding of the inherent properties of substring/context distribution and its interest for learning.

The notions of Finite Kernel Property (FKP) and Finite Context Property (FCP) have also been shown to be central in distributional learning: the extension called the k-FKP is a requirement of most of the works in the primal approach, while k-FCP is needed for dual ones. It is worth noticing that the definition of the FCP given in this chapter is not the one currently used: it is now referred to as the finite fiducial set property. For a complete panorama of the existing properties and their nuances, we refer the Reader to the recent paper of Kanazawa and Yoshinaka [2017].

Then, the framework of having a set of examples of the target language and a membership oracle, though not completely new, has been proven extremely useful since its (re)introduction in the work presented in this chapter. Indeed, classes of increasing complexity have been proven learnable in this framework:

- Families of context-free classes enjoyed efficient learning results, either in the primal or the dual approaches [Clark, 2010b, Yoshinaka, 2011, 2012, Leiß, 2014]
- Yoshinaka [2010] showed that the class of multiple context-free languages with the FKP is learnable under this paradigm. This result in the primal concerns a class that contains context-free languages and context-sensitive ones.
- Clark and Yoshinaka [2013] proved the learnability in this framework of a subclass of parallel multiple context-free grammars which generate non-semilinear languages covering all known structures of natural languages, including the non context-free ones. This result relies on an extension of the notion of k-FCP, and thus takes the dual approach.

While the usual grammatical approach consists in restraining existing classes, the attitude of designing and studying representations shaped for learning that is at the core of the work presented here have also been fecund. We can cite for instance the work on lattice representations [Clark, 2009] or the one on plane graph grammars presented in the next chapter (Chapter 4).

Finally, the most direct continuation of this work is certainly the paper of Yoshinaka [2015]. Indeed, he showed in this article that exact CBFG and conjunctive grammars with the k-FCP are equivalent models and he provided a learning algorithm for this class of conjunctive grammars, proving a stronger learning result than the one presented here (providing that the learner knows the value of k). His result comes with nice efficiency guarantees:

- the time required to infer a model is polynomial in the size of the learning sample,
- the number of queries is also polynomial in this size,
- conditions for a set of examples to be characteristic are given, together with a bound on the number of elements in these samples polynomial in the size of the target representation.

In addition, although it is presented in the usual identification in the limit paradigm, the algorithm is set-driven in the sense that each hypothesis depends only on the available example and the answer of the oracle. Therefore, and despite the fact that the proof still need to be carefully written, everything needed is available to provide an *identification from polynomial time, thick data, and queries* result.

This brief panorama about the work on distributional learning that followed the one presented in this chapter does not claim to be exhaustive. For instance we did not detail the works in other learning paradigms (see for instance the papers from Clark [2010a] or Shibata and Yoshinaka [2016]), the identification of more abstract grammar formalisms [Yoshinaka and Kanazawa, 2011], or using more structured data like trees (see Kasprzik and Yoshinaka [2011] or Clark et al. [2016] for instance) or graphs (see Chapter 4 of this HDR).

Bibliography

P. Adriaans. Learning shallow context-free languages under simple distributions. *Algebras, Diagrams and Decisions in Language, Logic and Computation*, 127, 2002. 41

D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987. 41

D. Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1988. 67

P.R.J. Asveld. Generating all permutations by context-free grammars in Chomsky normal form. *Theoretical Computer Science*, 354(1):118–130, 2006. 72

B. Balle, R. Eyraud, F. M. Luque, A. Quattoni, and S. Verwer. Results of the sequence prediction challenge (SPiCe): a competition on learning the next symbol in a sequence. In S. Verwer, M. van Zaanen, and R. Smetsers, editors, *Proc. of The International Conference on Grammatical Inference*, volume 57 of *Proceedings of Machine Learning Research*, pages 132–136. PMLR, 2017. 64

L. Boasson and G. Senizergues. NTS languages are deterministic and congruential. *J. Comput. Syst. Sci.*, 31(3):332–342, 1985. 42

P. Boullier. A Cubic Time Extension of Context-Free Grammars. *Grammars*, 3:111–131, 2000. 42, 46, 74

P. Boullier. Counting with range concatenation grammars. *Theoretical Computer Science*, 293(2):391–416, 2003. 72

R.C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In R.C. Carrasco and J. Oncina, editors, *Proc. of the International Conference on Grammatical Inference*, volume 862 of *LNAI*, pages 139–150. Springer, 1994. 41

N. Chomsky. *Knowledge of Language : Its Nature, Origin, and Use*. Praeger, 1986. 40

A. Clark. PAC-learning unambiguous NTS languages. In *Proc. of the International Conference on Grammatical Inference*, pages 59–71, 2006. 41, 77

A. Clark. A learnable representation for syntax using residuated lattices. In *Proc. of Formal Grammar*, Bordeaux, France, 2009. 78

A. Clark. Distributional learning of some context-free languages with a minimally adequate teacher. In J M. Sempere and P. García, editors, *Proc. of the International Conference on Grammatical Inference*, pages 24–37. Springer, 2010a. 79

A. Clark. *Learning Context Free Grammars with the Syntactic Concept Lattice*, pages 38–51. Springer Berlin Heidelberg, 2010b. 78

A. Clark. The syntactic concept lattice: Another algebraic theory of the context-free languages? *Journal of Logic and Computation*, 25(5):1203–1229, 2015. 78

A. Clark and R. Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007. 41, 76

A. Clark and R. Yoshinaka. Distributional learning of parallel multiple context-free grammars. *Machine Learning*, 2013. 78

A. Clark and R. Yoshinaka. *Distributional Learning of Context-Free and Multiple Context-Free Grammars*, pages 143–172. Springer Berlin Heidelberg, 2016. 77

A. Clark, R. Eyraud, and A. Habrard. A polynomial algorithm for the inference of context free languages. In *Proc. of International Conference on Grammatical Inference*, pages 29–42. Springer, September 2008. 40

A. Clark, R. Eyraud, and A. Habrard. A note on contextual binary feature grammars. In *Proc. of the EACL Workshop on Computational Linguistic Aspects of Grammatical Inference*, CLAGI '09, pages 33–40. Association for Computational Linguistics, 2009. 40

A. Clark, R. Eyraud, and A. Habrard. Using contextual representations to efficiently learn context-free languages. *Journal of Machine Learning Research*, 11:2707–2744, 2010. 40

A. Clark, M. Kanazawa, G. M. Kobele, and R. Yoshinaka. Distributional learning of some nonlinear tree grammars. *Fundamenta Informaticae*, 146(4):339–377, 2016. 79

D. Combe, C. de la Higuera, and J.-C. Janodet. *Zulu: An Interactive Learning Competition*, pages 139–146. Springer Berlin Heidelberg, 2010. 64

F. Denis, A. Lemay, and A. Terlutte. Learning regular languages using rfsas. *Theor. Comput. Sci.*, 313(2):267–294, 2004. 41

R. Eyraud. *Inférence Grammaticale de Langages Hors-Contextes*. PhD thesis, Université Jean Monnet, 2006. 40

R. Eyraud, C. de la Higuera, and J.C. Janodet. Lars: A learning algorithm for rewriting systems. *Machine Learning*, 66(1):7–31, 2007. 41

G. Gazdar, E. Klein, G. Pullum, and I. Sag. *Generalised Phrase Structure Grammar*. Basil Blackwell, 1985. 42, 46

E.M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967. 58

Z. Harris. Distributional structure. *Word*, 10(2-3):146–62, 1954. 41

C. De La Higuera and J. Oncina. Inferring deterministic linear languages. In *Proc. of the Conference on Computational Learning Theory*, pages 185–200. Springer-Verlag, 2002. 41

J.J. Horning. *A Study of Grammatical Inference*. PhD thesis, Stanford University, Computer Science Department, California, 1969. 76

A.K. Joshi and Y. Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, Berlin, New York, 1997. 41

M. Kanazawa and R. Yoshinaka. The strong, weak, and very weak finite context and kernel properties. In *Proc. of Language and Automata Theory and Applications*, pages 77–88, 2017. 78

A. Kasprzik and R. Yoshinaka. Distributional learning of simple context-free tree grammars. In *Proc. of Algorithmic Learning Theory*, pages 398–412, 2011. 79

D. Klein and C.D. Manning. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proc. of the Annual Meeting on Association for Computational Linguistics*, pages 478–485, 2004. 77

K.J. Lang, B.A. Pearlmutter, and R. Price. Results of the abbadingo one dfa learning competition and a new evidence driven state merging algorithm. In *Proc. of the International Conference on Grammatical Inference*, pages 1–12. Springer-Verlag, 1998. 64

P. Langley and S. Stromsten. Learning context-free grammars with a simplicity bias. In *Proc. of the European Conference on Machine Learning*, pages 220–228. Springer-Verlag, 2000. 41

H. Leiß. Learning context free grammars with the finite context property: A correction of a. clark's algorithm. In G. Morrill, R. Muskens, R. Osswald, and F. Richter, editors, *Proc. of Formal Grammar*, pages 121–137. Springer Berlin Heidelberg, 2014. 78

S. Marcus. *Algebraic Linguistics; Analytical Models.* Academic Press, N. Y., 1967. 45

K. Nakamura and M. Matsumoto. Incremental learning of context-free grammars based on bottom-up parsing and search. *Pattern Recognition*, 38(9):1384–1392, 2005. 41

A. Okhotin. Conjunctive grammars. *J. Autom. Lang. Comb.*, 6(4):519–535, 2001. 75

A. Okhotin. An overview of conjunctive grammars. *Formal Language Theory Column, bulletin of the EATCS*, 79:145–163, 2003. 75

R. Parekh and V. Honavar. An incremental interactive algorithm for regular grammar inference. In *Proc. of the International Conference on Grammatical Inference*, pages 238–250. Springer-Verlag, 1996. 65

L. Pitt. Inductive inference, dfa's, and computational complexity. In *Lecture Notes in Artificial Intelligence*, pages 8–14. Springer-Verlag, 1989. 58

C. Shibata and R. Yoshinaka. Probabilistic learnability of context-free grammars with basic distributional properties from positive examples. *Theoretical Computer Science*, 620:46–72, 2016. 77, 79

B. Starkie, F. Coste, and M. Van Zaanen. The omphalos context-free grammar learning competition. In *Proc. of the Internation Conference on Grammatical Inference*, pages 16–27. Springer, 2004. 64

B. Starkie, M. van Zaanen, and D. Estival. The tenjinno machine translation competition. In *Proc. of the International Conference on Grammatical Inference*, volume 4201 of *Lecture Notes in Computer Science*, pages 214–226. Springer-Verlag, 2006. 64

M. van Zaanen and N. van Noord. Evaluation of selection in context-free grammar learning systems. In *Proc of the International Conference on Grammatical Inference*, pages 193–206, 2014. 41

S. Verwer, R. Eyraud, and C. dela Higuera. Pautomac: a probabilistic automata and hidden markov models learning competition. *Machine Learning*, 96(1):129–154, 2014. 64

T. Yokomori. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 298(1):179–206, 2003. 41

R. Yoshinaka. Identification in the limit of k,l-substitutable context-free languages. In *Proc. of the International Conference on Grammatical Inference*, pages 266–279. Springer-Verlag, 2008. 41

R. Yoshinaka. Polynomial-time identification of multiple context-free languages from positive data and membership queries. In *Proc. of the International Conference on Grammatical Inference*, pages 230–244, 2010. 78

R. Yoshinaka. Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In G. Mauri and A. Leporati, editors, *Proc. of Developments in Language Theory*, pages 429–440, 2011. 78

R. Yoshinaka. Integration of the dual approaches in the distributional learning of context-free grammars. In A.-H. Dediu and Carlos M., editors, *Proc. of Language and Automata Theory and Applications*, pages 538–550. Springer Berlin Heidelberg, 2012. 78

R. Yoshinaka. Learning conjunctive grammars and contextual binary feature grammars. In A.-H. Dediu, E. Formenti, C. Martín-Vide, and B. Truthe, editors, *Proc. of Language and Automata Theory and Applications*, pages 623–635. Springer International Publishing, 2015. 58, 68, 78

R. Yoshinaka and M. Kanazawa. Distributional learning of abstract categorial grammars. In *Proc. of Logical Aspects of Computational Linguistics*, pages 251–266, 2011. 79

Chapter 4

On Learning from Graphs

Contents

Context of this work

The origins of this chapter date back to a couple of discussions with Tim Oates during my visiting year at the University of Maryland. Once we made inroads into the thematic of learning from graphs, it appeared important to join our efforts with the researchers that had just proposed the new graph formalism that looked promising for our learning goal. That is how Jean-Christophe Janodet, and his PhD student Frederic Papadopoulos, joined us on that project. Two papers describing this research were accepted for

publication (and few were rejected): one in a peer-reviewed conference [Eyraud et al., 2012] and a longer article, called *Designing and Learning Substitutable Plane Graph Grammars*, in a journal [Eyraud et al., 2016].

At the core of this work, there is the will to extend distributional learning to graph data. Indeed, at the time I started this research, Kasprzik and Yoshinaka [2011] had just extended the first results on strings to trees: graphs were thus the next obvious step. However, it quickly appeared that despite numerous formalizations, no graph grammar formalism is truly shaped for learning. This is why I decided to introduce and investigate a new formalism, that we called *plane graph grammars*. This explains why an important part of this chapter deals with the definition and the study of this formalism. Despite of that, it is important to remember that the final goal is and has always been the learning from graph data.

Finally, notice that the learning framework here is simpler than the one used in Chapter 3. It corresponds to the one used for the first works on distributional learning [Clark and Eyraud, 2005, 2007, Yoshinaka, 2008]: the learner only have access to examples of the (graph) language to identify. I formalize for the first time in this chapter an identification in polynomial time and thick data result in this context.

4.1 Introduction

Graph Grammars have been defined and studied for more than four decades from a language-theoretical standpoint (see the article from Rozenberg and Ehrig [1997] 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* [Cook and Holder, 2000] and *FFSM* [Huan et al., 2003], and their extensions [Matsuda et al., 2002, Jonyer et al., 2003, Kukluk et al., 2008].

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, Jeltsch and Kreowski [1991] give an algorithm that generates the set of grammars consistent with a given set of graphs. Brijder and Blockeel [2011] 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 [Costa Florêncio, 2009]. Besides, note that new approximate learnability results are anticipated in the framework of recognizable series of (hyper-)graphs [Bailly et al., 2015].

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 [Harchaoui and Bach, 2007]. However, they generally hide the complexity of the graph structures into

abstract numerical structures such as graph kernels [Kadri et al., 2013]. These kernels mainly consist in counting the number of particular subgraphs in a new data in order to represent the graph as a vector: but doing this, machine learners forget that detecting a subgraph in a graph is generally not efficiently tractable...

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/recognize a graph with a grammar, are generally $\mathcal{NP}$-complete [Garey and Johnson, 1979]. 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 [Vishwanathan et al., 2010].

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 sub-structures, *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) [Drewes et al., 1997], 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 the ordering of given by 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 chapter, a promising candidate is the class of *plane graphs*, that is, planar graphs embedded in the plane (see Figure 4.1 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 [Fusy, 2007]. 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 [1948] 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 chapter 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 Gold [1967] 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 (see Chapter 3) and tree grammars [Kasprzik and Yoshinaka, 2011] motivate us to define an analogue of substitutable context-free languages [Clark and Eyraud, 2007] for plane graph languages.

Notice that a preliminary version of this chapter appeared in the Proceedings of the International Conference in Grammatical Inference in 2012, but the present chapter is substantially different: in [Eyraud et al., 2012] 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. Following [Eyraud et al., 2016], we consider in this chapter

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

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 4.2. The definition of Plane-Graph Grammars as well as the rewriting mechanism is detailed in Section 4.3, where we also compare them with node-replacement grammars and hyperedge-replacement grammars. We prove formal properties of these grammars in Section 4.4, in the scope of the parsing problem. Next Section 4.5 is devoted to the learning of PGG, and is thus the core of the chapter: the substitutability property is first adapted, then the learning procedure is described, and an identification result with efficiency bounds is finally proved for substitutable plane graph languages. We conclude the chapter with a discussion in Section 4.6.

4.2 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* [de la Higuera et al., 2013] below, which allow us to describe any *connected* plane graph through its faces.

Let $X \subset \mathbb{N}$ be the alphabet of vertices. Following the definitions and notations of previous chapters, X^* is the set of all *strings* over X. Given a string $x = a_1 \dots a_n$, we denote 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 $\mathring{X}$. 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 4.1. The outer face is f_1 and the 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.

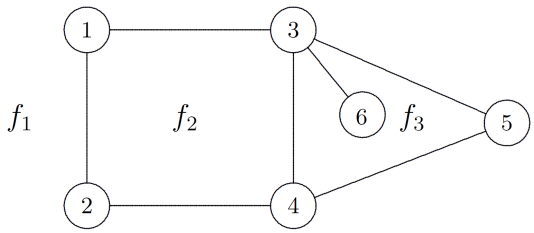

Figure 4.1: A plane graph with 3 faces.

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

Definition 4.1 (Plane Graph System [de la Higuera et al., 2013])**.** 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 [Gibbons, 1985],
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. $\mathcal{D} : F \to \mathring{X}$ 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 4.2 (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 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 chapter, the proof is given in Appendix 4.6. Notice that all the PGS we shall consider in this chapter will be valid, so this property will be kept implicit, most of the time:

Theorem 4.1. 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 4.1. 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 4.3 (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 $f = f_0, f_1, \dots f_n = f'$ of faces in K exists such that $\forall i \in \{0, 1, \dots, n-1\}$, 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 4.1, $\texttt{outer}(\{f_2, f_3\}) = f_1$ and $\texttt{outer}(\{f_3\}) = [354]$. Notice that $\texttt{outer}(K)$ can be computed in polynomial time using the *normalization* procedure introduced by de la Higuera et al. [2013].

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

Definition 4.4 (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 4.1 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/or edges, independently of faces), but they often induce intractable problems. In particular, de la Higuera et al. [2013] showed 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*.

4.2.1 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 4.5. 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 4.6 (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 $o = [\phi(a_k) y \phi(a_1) z]$.

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

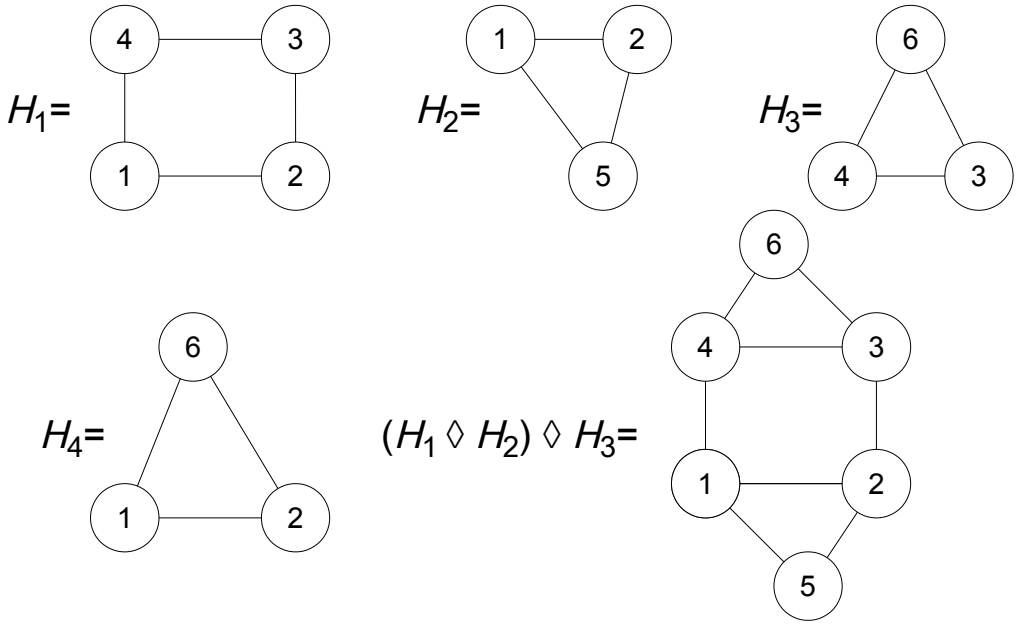

Figure 4.2: Example of concatenation. H_1 and H_2 are concatenable following the identity function and so is H_2 and H_4. The same occurs for $H_1 \diamond H_2$ and H_3. H_2 and H_3 are not concatenable following the identity function, which is also the case of H_1 and H_4.

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 4.2. 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 4.5 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 $(H_1 \diamond H_2) \diamond H_3$.

4.2.2 Plane isomorphism

We finally need a way to compare two PGS:

Definition 4.7 (Plane isomorphism [de la Higuera et al., 2013])**.** 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 [de la Higuera et al., 2013]. 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 [de la Higuera et al., 2013].

We can now define the notion of plane graph language:

Definition 4.8. 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 concatenable then graphs that are plane-isomorphic to them are also concatenable (using a different function). Moreover, the concatenated graphs are plane-isomorphic.

Lemma 4.2. 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 $a_1, \dots, a_k$ and (possibly empty) sequences of nodes y and z such that $o_{G_1} = [a_1 \dots a_k y]$ and $o_{G_2} = [\phi(a_k) \dots \phi(a_1) z]$. Moreover we have $o_{G_1'} = [\chi_1(a_1) \dots \chi_1(a_k)\hat{\chi_1}(y)]$ and $o_{G_2'} = [\chi_2 \circ \phi(a_k) \dots \chi_2 \circ \phi(a_1)\hat{\chi_2}(z)]$.

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 $o_{G_1} = [a_1' \dots a_k' \hat{\chi_1}(y)]$ and $o_{G_2} = [\phi'(a_k') \dots \phi'(a_1')\hat{\chi_2}(z)]$. 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. □

4.3 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 a 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 represent 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 [Engelfriet, 1975], we suppose that we have a ranked alphabet for soundness and type-checking reasons.

Definition 4.9 (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 4.10 (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 $N_{x_1}^1$ and $N_{x_2}^2$ are *concatenable* if PGS $[\![N_{x_1}^1]\!]$ and $[\![N_{x_2}^2]\!]$ are concatenable (following the identity). In this case, we denote by $[\![N_{x_1}^1 N_{x_2}^2]\!]$ the PGS $[\![N_{x_1}^1]\!] \diamond [\![N_{x_2}^2]\!]$.

An example of non-terminal gadget is given in Figure 4.3: 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 non-terminal gadgets. Many other examples of non-terminal gadgets appear in both the left-hand side and the right-hand side of Figure 4.4. 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 4.11 (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 4.3. 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 4.12 (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 4.4 graphically shows the plane graph production $N_{2341} \to A_{2541}\ N_{2645}\ B_{2346}$.

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

Definition 4.13 (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.

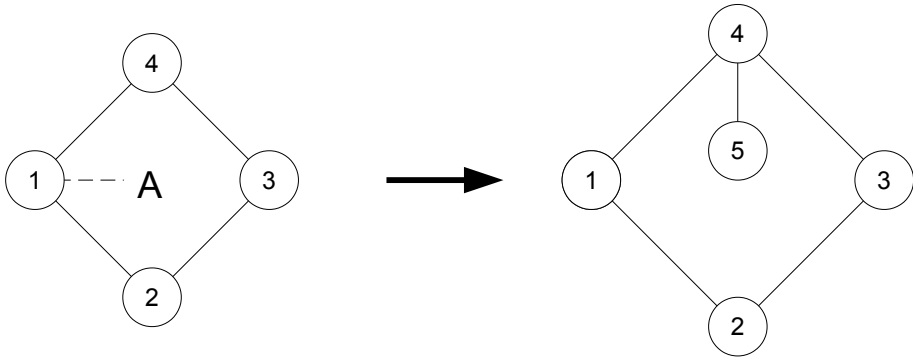

Figure 4.3: A graphical representation of the plane graph lexical rule $A_{1234} \to \langle\{1,2,3,4,5\},\{(1,2),(2,3),(3,4),(4,5)\},\{[123454],[1432]\},[1432]\rangle$. Notice that it is possible to simplify the textual notation of a lexical rule: the PGS on the right handside is fully determined by 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.

Example 4. 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 4.4 while the first of the three lexical rules is depicted in Figure 4.3. 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 4.14 (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

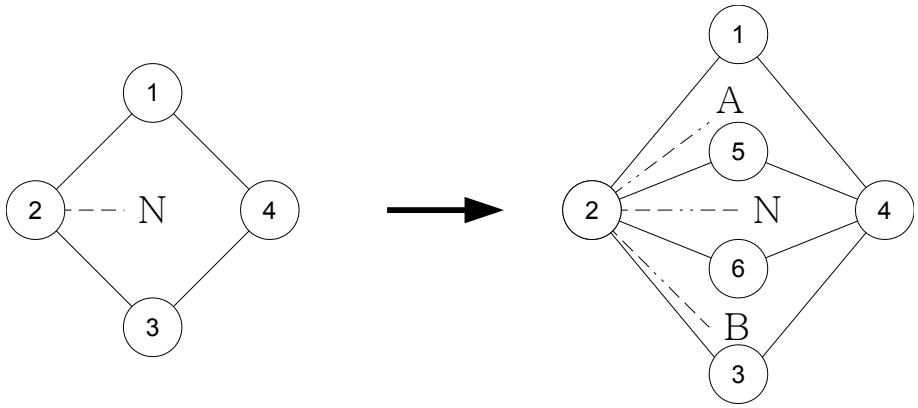

Figure 4.4: A graphical representation of the (recursive) plane graph production $N_{2341} \to A_{2541}\ N_{2645}\ B_{2346}$. Recall that the dashed lines attach each non-terminal symbol to the vertex appearing at the head of the string corresponding to that face in the non-terminal gadget.

$|\texttt{nodes}(f)| = \texttt{rank}(N)$ 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 4.5.

4.3.1 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 5. The lexical rule $A_{1234} \to [123454]$ in $\mathcal{G}_1$ of Example 4 is applicable to the sentential S_3 of Figure 4.5. Once applied, the result is the sentential form S_4 of the same figure.

4.3.2 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

- $F' = (F \setminus \{f\}) \bigcup_{1 \leq i \leq k} \{\hat{\phi}([a^i_1 \dots a^i_{n_i}])\}$
- $o' = o$;
- $X' = X \bigcup_{1 \leq i \leq k} \mathtt{nodes}(\hat{\phi}([a^i_1 \dots a^i_{n_i}]))$;
- $E' = E \bigcup_{1 \leq i \leq k} \mathrm{edges}(\hat{\phi}([a^i_1 \dots a^i_{n_i}]))$;
- New labeling function $\mathcal{L}'$ is as follows:

– $\mathcal{L}'(\hat{\phi}([a_1^i \dots a_{n_i}^i])) = (N^i, \phi(a_1^i))$ for all $1 \leq i \leq k$, and
– $\forall f' \in F, f' \neq f$, if $\mathcal{L}(f')$ is defined, then $\mathcal{L}'(f') = \mathcal{L}(f')$.

Example 6. The production $N_{2341} \to A_{2541}\ N_{2645}\ B_{2346}$ in $\mathcal{G}_1$ of Example 4 is applicable to the sentential S_1 of Figure 4.5 (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 $G_\diamond = [\![N^1_{a_1^1 \dots a_{n_1}^1} \dots N^k_{a_1^k \dots a_{n_k}^k}]\!]$ 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 4.2.1), we have $|X_\diamond| - |E_\diamond| + |F_\diamond| = 2$. Moreover, the outer face of $G_\diamond$ matches the string $a_1^0 \dots a_m^0$, 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.

4.3.3 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 $S_1, \dots, S_n$ 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.

The *length of the derivation* is $n-1$. An example of a derivation of length 4 is given in Figure 4.5. 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

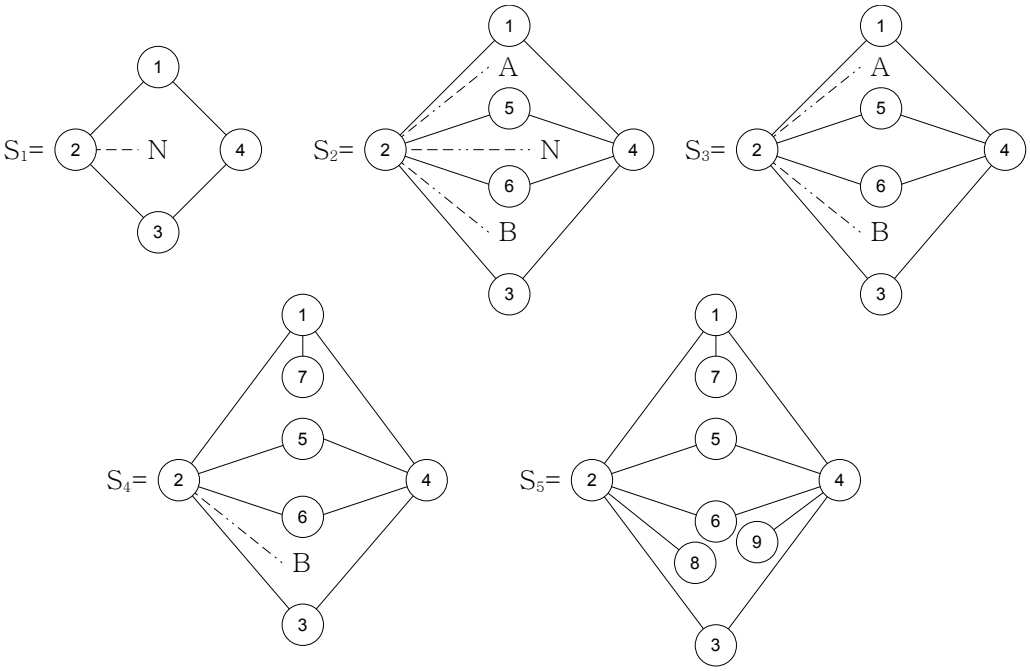

Figure 4.5: A graphical representation of an example of a derivation in the grammar $\mathcal{G}_1$ in Example 4. This derivation corresponds to the sequence of sentential forms S_1, S_2, S_3, S_4, S_5. The functions $\mathcal{L}_i$, $1 \leq i \leq 5$, are represented via dash lines.

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

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.

4.3.4 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 [Nagl, 1976] 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 [Whitney, 1931], 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 [Drewes et al., 1997] are another type of graph grammar formalism's that seems closer to the one introduced in this chapter. 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 4.3) 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 4.5. If one want the nodes to carry semantic information, the formalism is easily modifiable with the adding of a semantic 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 3G map L-systems for fruit modeling [Bohl et al., 2015]). 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

4.4 Properties of Plane Graph Grammars

4.4.1 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 [Courcelle, 1987]. 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 labels.

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 4.3 (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. $\exists N_x \to N_{x_1}^1 \dots N_{x_m}^m$ 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 $\mathcal{L}_1(o_{G_1}^R) = (N, a)$, for some $a \in \text{nodes}(o_{G_1}^R)$, 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 $N_x \to N_{x_1}^1 \dots N_{x_m}^m$ and $G_2 = [\![N_{x_1}^1 \dots N_{x_m}^m]\!] = \langle X_{G_2}, E_{G_2}, \cup_{1 \leq i \leq m} \{[x_i]\} \cup [x^R]\}, [x^R] \rangle$, $\mathcal{L}_2([x_i]) = (N^i, \texttt{first}(x_i))$, $\forall i$. 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 $H = H_1 \diamond \ldots \diamond H_m$ (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 $\exists N_x \rightarrow N^1_{x_1} \ldots N^m_{x_m}$ in P and there exist m PGS $H_1, \ldots, H_m$ such that $N^i \Rightarrow^* H_i$, for all $1 \leq i \leq m$, and $H \cong_p H_1 \diamond \ldots \diamond H_m$. 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 $(G_2, \mathcal{L}_2)$, $\ldots$, $(G_k, \mathcal{L}_k)$ 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 $N_x \rightarrow N^1_{x_1} \ldots N^m_{x_m}$ with $m > 2$, one can replace it by two rules $N_x \rightarrow N'_{x'} N^m_{x_m}$ and $N'_{x'} \rightarrow N^1_{x_1} \ldots N^{m-1}_{x_{m-1}}$, 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 chapter that all the PGG are in such a normal form.

4.4.2 A Parsing Algorithm

Theorem 4.3 provides a straightforward parsing algorithm, given as pseudo-code in Algorithm 4.1.

Proposition 4.4. For all PGS G, Algorithm 4.1 terminates and yields TRUE iff $G \in L(\mathcal{G})$.

Algorithm 4.1: 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 $N \in \mathcal{A}$ **do**
  if $DERIVE(\mathcal{G}, N, G)$ **then**
    return TRUE
return FALSE

Algorithm 4.2: 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 $N \to H \in P_L$ *and* $H \cong_p G$ **then**
  return TRUE
foreach $N_x \to N_y^1 N_z^2 \in P$ **do**
  if $\exists H_1, H_2$ *such that DERIVE*(N^1, H_1) *and DERIVE*(N^2, H_2) *and* $G \cong_p H_1 \diamond H_2$ **then**
    return TRUE
return FALSE

Proof. Algorithm 4.1 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 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 [Lautemann, 1989].

Definition 4.15 (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 4.2 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 4.5. Let $\mathcal{G}$ be a PGG. If $\text{rank}_{L(\mathcal{G})}(n)$ is polynomial in n then Algorithm 4.1 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 $G \in L(\mathcal{G})$. □

4.5 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 [Clark and Eyraud, 2007].

4.5.1 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 4.16 (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 4.17 (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 4.6 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.

Definition 4.18 (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'',

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

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.

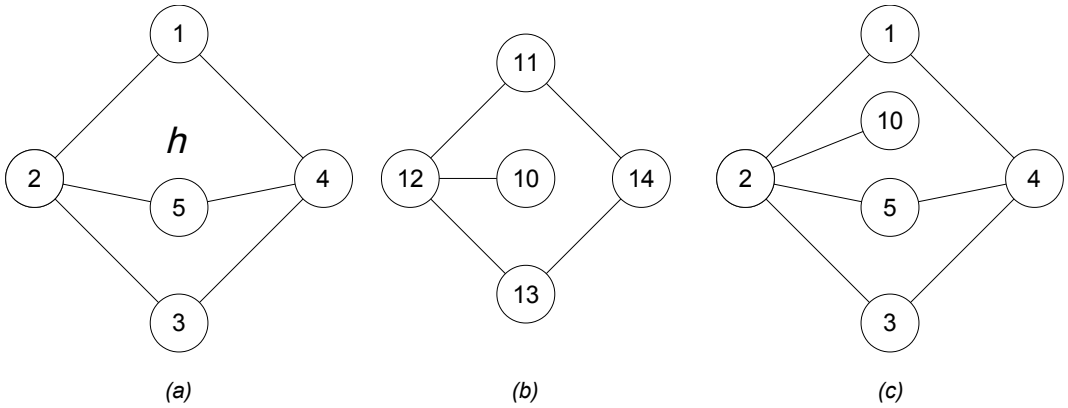

Figure 4.6: *(a)* A context whose hole is the face h. *(b)* A plane graph. *(c)* The gluing of the graph into the context following the gluing function $\phi(11) = 1, \phi(12) = 2, \phi(13) = 5, \phi(14) = 4$.

Lemma 4.6. 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 4.19 (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 4.7. 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 $C_{C\diamond\hat{\phi}(G_1)} = \langle X_C \cup \hat{\phi}(X_{G_1}) \cup \mathtt{nodes}(\hat{\phi}(o_{G_2})), E_C \cup \hat{\phi}(E_{G_1}) \cup \mathrm{edges}(\hat{\phi}(o_{G_2})), (F_C \setminus h) \cup \hat{\phi}(F_{G_1} \setminus o_{G_1}) \cup \hat{\phi}(o^R_{G_2}), \hat{\phi}(o^R_{G_2}), o_C\rangle$. 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 4.6 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 4.20 (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.

4.5.2 The Learner

Our learning algorithm is described in Algorithm 4.3.

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 [Clark and Eyraud, 2007] 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

Algorithm 4.3: 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
       $[b_1 \dots b_n] = \texttt{outer}(F_1)$, $[c_1 \dots c_p] = \texttt{outer}(F_2)$, $b_1 = b$, $c_1 = c$ and
       $\forall i : G_i = \langle X_i, E_i, F_i \cup \{\texttt{outer}(F_i)\}, \texttt{outer}(F_i) \rangle$;
return $\langle N, P_L, P, \mathcal{A} \rangle$

function $number\_nodes(C)$ returns the number of nodes of the outer face of the graphs in the observable congruence class C.

4.5.3 Learning result

Our learning paradigm is the one of identification in polynomial time and thick data introduced in Definition 2.12 of Chapter 2.

Plane Graph Grammars being not totally compatible with the notion of generative grammars (Definition 2.1), some adjustments are needed. In particular, we have to define the notion of thickness for PGG:

Definition 4.21 (Thickness)**.** Let $\mathcal{G} = \langle \mathcal{N}, P_L, P, \mathcal{A} \rangle$ be a plane graph grammar. The *thickness* of $\mathcal{G}$ is $\tau_{\mathcal{G}} = \max\{|\omega(N)| : \exists \beta, x \text{ s.t. } N_x \rightarrow \beta \in P \cup P_L\}$ where $\omega(N) = \min_{\lhd}\{G \in \mathbb{G} : N \Rightarrow_{\mathcal{G}}^{*} G\}$.

This is a straightforward extension to PGG of Definition 2.11.

The requirements of the learning paradigm imply a time polynomial in the size of

the learning sample[3], and a characteristic sample polynomial in the size of the target representation and its thickness.

4.5.3.1 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 4.4.2) 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 [de la Higuera et al., 2013]. 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 4.3 are polynomial in the size of the observable congruence classes.

4.5.3.2 Proof the hypothesis is not too large

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

Lemma 4.8. 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

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

$G'' = \langle X'', E'', \{\hat{\phi}(\texttt{outer}(F_1)), \hat{\phi}(\texttt{outer}(F_2)), o\}, o\rangle$, $\mathcal{L}''(\texttt{outer}(F_1)) = (\mathcal{N}^t(G_1), b_1)$ 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 4.2, $G \cong_p G'_1 \diamond G'_2$. □

Lemma 4.9. 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 4.6 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 4.7 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 4.10. For all samples of a language L, the output $\mathcal{G}$ of Algorithm 4.3 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 4.9 states that G'' and G' are substitutable and thus $G \equiv_S^L G'$. As G' is an element of L, $G \in L$. □

4.5.3.3 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 $L(\mathcal{G}) \supseteq L_*$.

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 4.3 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 $\langle G_1, \mathcal{L}_1 \rangle, \ldots, \langle G_k, \mathcal{L}_k \rangle$ 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 $C = C(N^i)$.

Lemma 4.11. Given any PGG $\mathcal{G}_*$, the size of $CS(\mathcal{G}_*)$ is polynomial in the size and thickness of $\mathcal{G}_*$

Proof. The cardinality of $CS(\mathcal{G}_*)$ is at most $|P_*| + |P_{L*}|$ by construction, which is clearly polynomially bounded. Moreover, the size of each element in the set is at most the thickness of the grammar times the number of rules needed to generate this element (by definition). This number is smaller than $(|P_*| + |P_{L*}|)|P_*||P_{L*}|$ by definition of $C(\cdot)$ and $G(\cdot)$ (no recursivity is possible to generate these elements). Therefore, the size of this characteristic sample is bounded by $(|P_*| + |P_{L*}|)^2 |P_*||P_{L*}|\tau_{\mathcal{G}_*}$ and then the size of the characteristic sample is polynomial in the size of the target PGG and its thickness. □

Notice also that if there exists a polynomial-sized structurally complete sample then the size of our characteristic set is polynomial in the size of the target. Indeed, one can show that the definition of $CS(\mathcal{G}_*)$ corresponds to a smallest structurally complete set of $\mathcal{G}_*$. This implies that if we show that this set validates the conditions to be characteristic (Definition 2.7), the algorithm will fulfills the requirements of the IPTscD paradigm of Definition 2.10.

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 4.12. 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 4.2).

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 4.13. Let $\mathcal{G}_*$ be a target plane graph grammar corresponding to a substitutable plane graph language. From any sample containing $CS(\mathcal{G}_*)$ Algorithm 4.3 returns a grammar $\mathcal{G}$ such that $L(\mathcal{G}) \supseteq L(\mathcal{G}_*)$.

Proof. If $G \in L(\mathcal{G}_*)$ then there exists $N \in \mathcal{A}_*$ such that $N \Rightarrow^*_{\mathcal{G}_*} G$. By Lemma 4.12, 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 $G \in L(\mathcal{G})$. □

This last theorem allowed the following learning result:

Theorem 4.14. Algorithm 4.3 identifies in polynomial time and thick data the class of substitutable plane graph languages with polynomial rank.

Proof. Let $\mathcal{G}_*$ be the target plane graph grammar corresponding to a substitutable plane graph language and $\mathcal{G}$ the returned grammar from a sample containing $CS(\mathcal{G}_*)$. By Theorem 4.13, we have $L(\mathcal{G}_*) \subseteq L(\mathcal{G})$. Theorem 4.10 states that $L(\mathcal{G}) \subseteq L(\mathcal{G}_*)$. Therefore, $L(\mathcal{G}) = L(\mathcal{G}_*)$. The discussion on time complexity of Section 4.5.3.1 and Lemma 4.11 allow to conclude. □

4.6 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 chapter 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 [Clark and Eyraud, 2007] 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 [Yoshinaka, 2011] 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 [Shibata and Yoshinaka, 2013].

Then, the works on strings grammars that required both a set of positive examples and a membership oracle (see for instance the brief overview at the end of Chapter 3) can possibly be extended to graph grammars. Indeed, the recent extension to tree grammars [Kanazawa and Yoshinaka, 2016, Clark et al., 2016] provides a good witness for the feasibility of such a goal.

Finally, due to the interest of planar graphs in image processing [Samuel et al., 2010], it is likely that the learning of plane graph grammars, and more generally grammatical inference techniques, could be used to tackle image classification challenges, such as concept detection or content-based image retrieval. The idea would be to use the transform the images into plane graphs, using Samuel et al.'s algorithm, to learn a PDD from this set of graphs, and to use the learned grammar to mine new images. Of course, demonstrating it can outperformed current deep learning heuristics on these tasks will necessitate important investments and efforts, comparable to the one artificial neural networks benefited during the past ten years.

Bibliography

R. Bailly, F. Denis, and G. Rabusseau. Recognizable series on hypergraphs. In *Proc. of the International Conference on Language and Automata Theory and Applications*, pages 639–651. LNCS 8977, 2015. 86

E. Bohl, O. Terraz, and D. Ghazanfarpour. Modeling fruits and their internal structure using parametric 3gmap l-systems. *The Visual Computer*, 31(6-8):819–829, 2015. 102

R. Brijder and H. Blockeel. On the inference of non-confluent NLC graph grammars. *Journal of Logic and Computation*, 23(4):799–814, 2011. 86

A. Clark and R. Eyraud. Identification in the limit of substitutable context-free languages. In *Proc. 16th Int. Conference on Algorithmic Learning Theory (ALT'05)*, pages 283–296. LNCS 3734, 2005. 86

A. Clark and R. Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007. 86, 88, 106, 109, 115

A. Clark, M. Kanazawa, G. M. Kobele, and R. Yoshinaka. Distributional learning of some nonlinear tree grammars. *Fundamenta Informaticae*, 146(4):339–377, 2016. 115

D. J. Cook and L. B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15 (2):32–41, 2000. 86

C. Costa Florêncio. Identification of hyperedge-replacement graph grammars. In *Proc. of the International Workshop on Mining and Learning with Graphs*, pages 19–21. Leuven, Belgium, 2009. 86

B. Courcelle. An axiomatic definition of context-free rewriting and its application to nlc graph grammars. *Theoretical Computer Science*, 55:141–181, 1987. 103

G Damiand and P. Lienhardt. *Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing*. A K Peters/CRC Press, 2014. 119

C. de la Higuera, J.-C. Janodet, E. Samuel, G. Damiand, and C. Solnon. Polynomial algorithms for open plane graph and subgraph isomorphisms. *Theoretical Computer Science*, 498:76–99, 2013. 89, 90, 91, 92, 94, 111

F. Drewes, H.-J. Kreowski, and A. Habel. Hyperedge replacement graph grammars. In *Handbook of Graph Grammars*, pages 95–162. Rozenberg and Ehrig [1997], 1997. 87, 102

J. Engelfriet. Tree automata and tree grammars, 1975. DAIMI FN-10 (Lecture Notes), Aarhus University. 95

R. Eyraud, J.-C. Janodet, and T. Oates. Learning substitutable binary plane graph grammars. In *Proc. the International Conference on Grammatical Inference*, pages 114–128. JMLR Worshops and Conference Proceedings 21, 2012. 86, 88

R. Eyraud, J.-C. Janodet, T. Oates, and F. Papadopoulos. Designing and learning substitutable plane graph grammars. *Fundamenta Informaticae*, 146(4):403–430, 2016. 86, 88

I. Fáry. On straight line representation of planar graphs. *Acta Univ Szeged. Sect. Sci. Math*, 11:229–233, 1948. 88

E. Fusy. *Combinatoire des graphes planaires et applications algorithmiques (in English)*. PhD thesis, Ecole Polytechnique - ParisTech, 2007. 88

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. 87

A. Gibbons. *Algorithmic graph theory*. Cambridge University Press, 1985. 90

E. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967. 88

Zaïd Harchaoui and Francis R. Bach. Image classification with segmentation graph kernels. In *Proc. of the Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2007. 86

J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proc. of the International Conference on Data Mining*, pages 549–552. IEEE Computer Society, 2003. 86

E. Jeltsch and H.-K. Kreowski. Grammatical inference based on hyperedge replacement. In *Proc. of the International Workshop on Graph Grammars and their Application to Computer Science*, pages 461–474. LNCS 532, 1991. 86

I. Jonyer, L. B. Holder, and D. J. Cook. MDL-based context-free graph grammar induction. *International Journal of Artificial Intelligence Tools*, 13:65–79, 2003. 86

H. Kadri, M. Ghavamzadeh, and P. Preux. A generalized kernel approach to structured output learning. In *Proc. of the International Conference in Machine Learning*, pages 471–479. JMLR Workshops and Conference Proceedings 28, 2013. 87

M. Kanazawa and R. Yoshinaka. Distributional learning and context/substructure enumerability in nonlinear tree grammars. In *Proc. of Formal Grammar*, pages 94–111, 2016. 115

A. Kasprzik and R. Yoshinaka. Distributional learning of simple context-free tree grammars. In *Proc. of the International Conference on Algorithmic Learning Theory*, pages 398–412. LNAI 6925, 2011. 86, 88

J. Kukluk, L. Holder, and D. Cook. Inference of edge replacement graph grammars. *International Journal on Artificial Intelligence Tools*, 17(3):539–554, 2008. 86

C. Lautemann. The complexity of graph languages generated by hyperedge replacement. *Acta Informatica*, 27(5):399–421, 1989. 105

Pascal Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3): 275–324, 1994. 119

T. Matsuda, H. Motoda, and T. Washio. Graph-based induction and its applications. *Advanced Engineering Informatics*, pages 135–143, 2002. 86

M. Nagl. Formal languages of labelled graphs. *Computing*, 16:113–137, 1976. 101

G. Rozenberg and H. Ehrig. *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1–3. World Scientific, 1997. 86, 116

E. Samuel, C. de la Higuera, and J.-C. Janodet. Extracting plane graphs from images. In *Proc. of the International Workshop on Structural and Syntactic Pattern Recognition*, pages 233–243. LNCS 6218, 2010. 115

C. Shibata and R. Yoshinaka. Pac learning of some subclasses of context-free grammars with basic distributional properties. In *Proc. of the International Conference in Algorithmic Learning Theory*, pages 143–157. LNCS 8139, 2013. 115

S. Vishwanathan, N. Schraudolph, R. Imre Kondor, and K. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010. 87

H. Whitney. Non-separable and planar graphs. *Proc. of the National Academy of Science, U.S.A.*, 17(2):125–127, 1931. 102

R. Yoshinaka. Identification in the limit of (k,l)-substitutable context-free languages. In *Proc. of the International Conference in Grammatical Inference*, pages 266–279. LNAI 5278, 2008. 86

R. Yoshinaka. Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. *Theoretical Computer Science*, 412(19): 1821–1831, 2011. 115

Proof of Theorem 4.1

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. 4.2 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 4.1, 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 [Lienhardt, 1994].

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.4.2, 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.4.2, 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. 4.2, 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. 4.2, the genus [Damiand and Lienhardt, 2014] 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.

Part III

Functional Learning

Chapter 5

On Learning with a Known Domain

Contents

Context of this work

This chapter concerns researches that have been done in collaboration with members of the Department of Linguistics and Cognitive Science of University of Delaware, USA. During a two and half years stay in US East Coast, and another 1 year stay in the same region, together with Jeffrey Heinz (and James Rogers of the Earlham College during the first months) we weekly animated a research group on computational linguistics and grammatical inference. This group was mainly composed of graduate students. Two of them were particularly invested in the group which allowed to deepen the collaboration: Jane Chandlee who was starting as a PhD candidate (now enjoying a tenure-track at the

linguistic department of Haverford College), and Adam Jardine who was just starting his graduate school (now an assistant professor at the linguistic department of Rutgers University). Their enthusiasm for the field led to several publications, that forms the scientific content of this part of the HDR, and to an important involvement all along their PhD and after (membership in dissertation proposal and PhD defense committees, regular discussions, on-going works, ...).

The content of this chapter is based on the first work we published together [Jardine et al., 2014]. The paper was tittled *Very efficient learning of structured classes of subsequential functions from positive data* and was published in the proceedings of the 12th International Conference on Grammatical Inference.

Here, and in the following chapter, the learning goal is different than the usual one in grammatical inference. Indeed, while we usually aim at learning the largest class possible, these works address the efficient learnability of the smallest class that contains all phonological/morphological processes.

5.1 Introduction

String transductions are functions from strings to strings. They have been extensively studied in theoretical computer science (see the survey from Filiot and Reynier [2016] for a recent overview), and can commonly be found in many applicative fields, from speech processing (see for instance the work of Mohri et al. [2002]) to software engineering (a recent example is the article from Veanes et al. [2012]).

One of the most important grammatical inference results for learning string transductions is OSTIA [Oncina et al., 1993], which identifies in the limit total subsequential functions. Its behavior is not unlike that of RPNI, upon which it is based. de la Higuera [2010] provides an up-to-date treatment of OSTIA. More recent work on learning string transductions has focused on learning the entire class of regular relations or even larger classes [Clark, 2011].

This chapter focuses on *proper subclasses* of subsequential functions and presents an algorithm that identifies each class from positive data. This result is interesting for 3 main reasons. First, OSTIA is unable to learn partial functions exactly, while the classes identified exactly in the present work include partial functions. Second, the algorithm we provide is able to learn in *linear* time and with a characteristic sample that is *quadratic* with respect to the size of the target transducer. This is an improvement over OSTIA, which, while polynomial, requires cubic computation time and its convergence is ensured only if the data available is cubic in the size of the target transducer. Third, this algorithm is applicable to infinitely many classes that contain infinitely many transductions.

The reason why this algorithm is possible is because each proper subclass comes with a significant amount of prior knowledge. In particular, we define a subclass of

subsequential functions in terms of a finite-state structure. Essentially, the states and the transitions of the transducer are provided and only the output labels on the transitions are missing. In this way, these results are analogous to those of Heinz and Rogers [2013], where the structure of the automata is fixed in advance to define an efficiently learnable class of formal languages.

Such prior knowledge may seem undesirable, especially for cases where it is not available. However, for domains where such prior knowledge *is* available, it would be senseless to not use it. We provide such domains from linguistics, in particular phonology and morphology. Specifically, we show that such a structure characterizes the Input Strictly k-Local functions introduced by Chandlee [2014] and detailed in Chapter 6. This class, which Chandlee shows includes significantly many phonological and morphological processes, represents one way to generalize the Strictly k-Local formal languages [McNaughton and Papert, 1971] to transductions. The few phonological processes that are not Strictly k-Local involve long-distance dependencies, and we show that these too are learnable, again provided *a priori* knowledge of the relevant finite-state structure.

In addition, when such prior knowledge is not available, one can expect to learn the domain from the data: it corresponds to regular languages for which grammatical inference provides several learning algorithms that used only positive examples for subclasses. This is the case for instance of the Strictly k-Local Languages [Garcia et al., 1990] and of k-Reversible Languages Angluin [1982] which are identifiable from positive data.

This chapter is organized as follows. Section 5.2 presents the mathematical notations that will be used throughout. Section 5.3 introduces subsequential finite state transducers, including the traditional definition, an alternative definition that we will be assuming throughout, and the particular type known as onward transducers, which will be crucial for the learning algorithm. Section 5.4 shows how to convert any subsequential FST into an onward one. Section 5.5 presents the adaptation of the learning paradigm of identification in polynomial time and data (Definition 2.8) to transduction and provides a stronger reformulation. Section 5.6 introduces empty transducers, which will provide the learner's *a priori* knowledge of the structure of the target transducer. Section 5.7 presents our Structured Onward Subsequential Function Inference Algorithm (SOSFIA). Section 5.8 proves that this set-driven algorithm identifies the target subclasses in the sense of our strong formulation of the paradigm of de la Higuera [1997]. Section 5.9 presents several demonstrations of the SOSFIA using linguistically-motivated examples. Section 5.10 concludes.

5.2 Preliminaries

We use throughout this chapter the same definitions and notations than in the previous chapters, to which we add the following ones.

Given three strings w, u and v such that $w = uv$, we write $u^{-1} \cdot w = v$ iff $w = uv$ and $w \cdot v^{-1} = u$. It trivially occurs that $\lambda^{-1} \cdot w = w = w \cdot \lambda^{-1}$ and $uu^{-1} \cdot w = w = w \cdot v^{-1}v$.

The *prefixes* of w are $\texttt{pref}(w) = \{u \in \Sigma^* \mid (\exists v)[uv = w]\}$. We extend the notion of prefixes to sets with $\texttt{pref}(S) = \cup_{w \in S}\texttt{pref}(w)$. The *shared prefixes* of S are the prefixes shared by all strings in S: $\texttt{sh\_pref}(S) = \cap_{w \in S}\texttt{pref}(w)$. The *longest common prefix* (`lcp`) of S is then $\texttt{lcp}(S) = w \in \texttt{sh\_pref}(S)$ such that $\forall v \in \texttt{sh\_pref}(S) : |v| \leq |w|$. We set the longest common prefix of an empty set to be λ.

Given an input alphabet Σ and an output alphabet Δ, a *relation* from Σ to Δ is a subset of $\Sigma^* \times \Delta^*$. Given a relation R, we use $\texttt{dom}(R)$ to refer to its left projection and $\texttt{image}(R)$ its right projection. So $\texttt{dom}(R) = \{w \mid (\exists v)[(w, v) \in R\}$ and $image(R) = \{v \mid (\exists w)[(w, v) \in R\}$. If, for all $w \in \Sigma^*, v, v' \in \Delta^*$, $(w, v), (w, v') \in R \Rightarrow v = v'$ then R is a *function*, and we write $R(w) = v$ instead of $(w, v) \in R$. We also write $R : \Sigma^* \to \Delta^*$ instead of $R \subseteq \Sigma^* \times \Delta^*$. A function f is *total* iff for all $w \in \Sigma^*$ there exists $v \in \Delta^*$ such that $f(w) = v$. A function that is not total is *partial*.

For any function $t : \Sigma^* \to \Delta^*$ and $w \in \Sigma^*$, we define the *tails* of w with respect to t as

$$\texttt{tails}_t(w) = \left\{(x, v) \mid wx \in \texttt{dom}(t) \text{ and } t(wx) = uv \text{ and } u = \texttt{lcp}(t(w\Sigma^*))\right\} .$$

Two strings $w, w' \in \Sigma^*$ are *tail-equivalent* with respect to t iff $\texttt{tails}_t(w) = \texttt{tails}_t(w')$, in which case we write $w \sim_t w'$. Clearly, $\sim_t$ is an equivalence relation which partitions Σ^*.

Definition 5.1. A function f is *subsequential* iff $\sim_f$ partitions Σ^* into *finitely* many blocks.

5.3 Representations of Subsequential Functions

5.3.1 Traditional Subsequential Transducers

We give here the usual definition of subsequential transducer:

Definition 5.2. A *subsequential finite-state transducer (SFST)* is a 6-tuple $\langle Q, q_0, \Sigma, \Delta, \delta, \rho\rangle$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, Σ and Δ are finite alphabets of symbols,[1] $\delta \subseteq Q \times \Sigma \times \Delta^* \times Q$ is the transition function, $\rho : Q \times \Delta^*$ is the output function, and the following holds: if $(q, \sigma, w, r), (q, \sigma, v, s) \in \delta$ then $(r = s) \wedge (w = v)$, i.e. the transition function is deterministic.

The transition and output functions of a SFST may be partial. We extend recursively the transition function to δ^* in the usual way: $(q, \lambda, \lambda, q) \in \delta^*$; if $(q, u, v, q') \in \delta^*$ and $(q', \sigma, w, q'') \in \delta$ then $(q, u\sigma, vw, q'') \in \delta^*$.

[1] Notice that Σ can be equal to Δ: in this case we only specify one alphabet in the definition of a SFST.

The relation described by a SFST $\mathcal{T}$ is $R(\mathcal{T}) = \{(w, vv') \mid (\exists q)[(q_0, w, v, q) \in \delta^* \wedge \rho(q) = v']\}$. However, since they are deterministic, SFSTs only describe functions: for every SFST, each element of Σ^* corresponds to at most one element of Δ^*. Therefore we write $\mathcal{T}(w) = v$ instead of $(w, v) \in R(\mathcal{T})$. We note that SFSTs compute *total* functions iff $\forall q \in Q, \forall \sigma \in \Sigma$, there exists $q' \in Q, v \in \Delta^*$ such that $(q, \sigma, v, q') \in \delta$ and $\rho(q')$ is defined. The size of a SFST $\mathcal{T} = \langle Q, q_0, \Sigma, \Delta, \delta, \rho \rangle$ is $|\mathcal{T}| = |Q| + |\delta| + \sum_{(q,\sigma,u,q') \in \delta} |u|$.

The following is a well-known result about subsequential functions:

Theorem 5.1 (Oncina and Garcia, 1991)**.** A function f is subsequential iff there exists a SFST $\mathcal{T}$ such that $R(\mathcal{T}) = f$.

This result can be seen as the functional analogue to the Myhill-Nerode theorem for regular languages. Recall that for any language L, the tails of a word w w.r.t. L is defined as $\texttt{tails}_L(w) = \{u \mid wu \in L\}$. These tails (often called residuals in this context and sometime written $w^{-1}L$) can be used to partition Σ^* into a finite set of equivalence classes iff L is regular. These equivalence classes are the basis for constructing the (unique up to isomorphism) smallest deterministic acceptor for a regular language. Likewise, Oncina and Garcia's proof of Theorem 5.1 shows how to construct the (unique up to isomorphism) smallest SFST for any subsequential function.

5.3.2 Delimited Subsequential Transducers

Here we present an alternative representation of subsequential functions, which we call *delimited subsequential transducers* and which are a slight variant of traditional SFSTs.

Definition 5.3. A *delimited subsequential finite-state transducer (DSFST)* is a 6-tuple $\langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $q_f \in Q$ is the final state, Σ and Δ are finite alphabets of symbols, $\delta \subseteq Q \times \Sigma \cup \{\rtimes, \ltimes\} \times \Delta^* \times Q$ is the transition function (where $\rtimes \notin \Sigma$ is a special symbol indicating the 'start of the input' and $\ltimes \notin \Sigma$ is a special symbol indicating the 'end of the input'), and the following hold:

1. if $(q, \sigma, u, q') \in \delta$ with $\sigma \in \Sigma$ then $q \neq q_f$ and $q' \neq q_0$,
2. if $(q, \ltimes, u, q') \in \delta$ then $q' = q_f$ and $q \neq q_0$,
3. if $(q_0, \sigma, u, q') \in \delta$ then $\sigma = \rtimes$ and if $(q, \rtimes, u, q') \in \delta$ then $q = q_0$,
4. if $(q, \sigma, w, r), (q, \sigma, v, s) \in \delta$ then $(r = s) \wedge (w = v)$.

In words, in DSFSTs, the unique initial state q_0 has no incoming transitions (1) and exactly one outgoing transition for input $\rtimes$ (3) which leads to a non-final state (2), and the unique final state q_f has no outgoing transitions (1) and every incoming transition comes from a non-initial state and has input $\ltimes$ (2). DSFSTs are also deterministic on the input (4).

In addition, the transition function may be partial. We extend the transition function to δ^* recursively again in the usual way: δ^* is the smallest set containing δ and which is closed under the following condition: if $(q, w, u, q') \in \delta^*$ and $(q', \sigma, v, q'') \in \delta$ then $(q, w\sigma, uv, q'') \in \delta^*$. Note no elements of the form $(q, \lambda, \lambda, q')$ are elements of δ^*.

The size of a DSFST $\mathcal{T} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ is $|\mathcal{T}| = |Q| + |\delta| + \sum_{(q,\sigma,u,q') \in \delta} |u|$.

The difference between DSFSTs and SFSTs is that the delimiters $\rtimes, \ltimes$ are employed to mark the beginnings and ends of input strings when computing the transducer function. Oncina et al. [1993] observe that an 'end of the input' delimiter can be used in place of the output function ρ without changing the class of functions being described. Therefore, the only real difference introduced here is the use of the 'start of the input' delimiter. It is trivially true that the class of functions describable with DSFSTs is exactly the class representable by SFSTs (i.e., the subsequential functions) If a DSFST is such that $(q, \ltimes, v, q_f) \in \delta \Rightarrow v = \lambda$ then the function it represents is called *sequential*.[2] From now on, we use the term 'transducer' to refer only to DSFSTs.

Furthermore, the proof of Theorem 5.1 shows that the tail-equivalence classes are the basis for constructing the (unique up to isomorphism) smallest SFST for a regular language. This result can be adapted to DSFSTs in a straightforward way to construct the smallest DSFST recognizing a subsequential function f. We refer to this DSFST as the *canonical* DSFST for f and denote it $\mathcal{T}_C(f)$. (If f is understood from context, we may write $\mathcal{T}_C$.) States of $\mathcal{T}_C(f)$ which are neither initial nor final are in one-to-one correspondence with $\texttt{tails}_f(x)$ for all $x \in \Sigma^*$ [Oncina and García, 1991]. To construct $\mathcal{T}_C(f)$ we first let, for all $x \in \Sigma^*$ and $\sigma \in \Sigma$, the *contribution* of σ w.r.t. x be $\texttt{cont}_f(\sigma, x) = \texttt{lcp}(f(x\Sigma^*)^{-1} \cdot \texttt{lcp}(f(x\sigma\Sigma^*))$. Then,

- $Q = \{\texttt{tails}_f(x) \mid x \in \Sigma^*\} \cup \{q_0, q_f\}$,
- $\big(q_0, \rtimes, \texttt{lcp}(f(\Sigma^*)), \texttt{tails}_f(\lambda)\big) \in \delta$
- $\forall x \in \Sigma^*$, $\big(\texttt{tails}_f(x), \ltimes, \texttt{lcp}(f(x\Sigma^*))^{-1} \cdot f(x), q_f\big) \in \delta$ iff $x \in \texttt{dom}(f)$
- $\forall x \in \Sigma^*, \sigma \in \Sigma$, if $\exists y \in \Sigma^*$ with $x\sigma y \in \texttt{dom}(f)$ then $\big(\texttt{tails}_f(x), \sigma, \texttt{cont}_f(\sigma, x), \texttt{tails}_f(x\sigma)\big) \in \delta$.
- Nothing else is in δ.

Observe that unlike the traditional construction, the initial state q_0 is not $\texttt{tails}_f(\lambda)$. The single outgoing transition from q_0, however, goes to this state with the input $\rtimes$. Canonical DSFSTs have an important property called *onwardness* (detailed in Section 5.3.3).

We end this section with the following definitions:

Definition 5.4 (Trimmed Transducers)**.** A state q in a DSFST τ is *reachable* iff there exists $w \in \Sigma^*, v \in \Delta^*$ such that $(q_0, \rtimes w, v, q) \in \delta^*$. A state q in a DSFST τ is *useful*

[2] Sakarovitch [2009] inverts these terms and uses 'sequential' to mean our 'subsequential' and 'subsequential' to mean our 'sequential.'

iff there exists $w \in \Sigma^*, v \in \Delta^*$ such that $(q, w\ltimes, v, q_f) \in \delta^*$ and q is reachable. A transducer is *trimmed* iff every states other than q_0 and q_f are *useful*.

Henceforth, the only DSFSTs we consider are trimmed.

5.3.3 Onward Transducers

An important concept for both SFSTs and DSFSTs that is also important to learning subsequential functions is the property of *onwardness*, which we define here for DSFSTs.

Definition 5.5. A DSFST $\tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ is *onward* iff for all $(q, \sigma, u, r) \in \delta$, $q \neq q_0$, either $\exists (q, \sigma', v, s)$ in δ such that $\sigma \neq \sigma'$ and $\texttt{lcp}(u, v) = \lambda$, or $u = \lambda$.

An equivalent definition states that a DSFST τ is onward iff for every $w \in \Sigma^*$, $u \in \Delta^*$, $(q_0, \rtimes w, u, q) \in \delta^* \iff u = \texttt{lcp}(\{f(w\Sigma^*)\})$.

Intuitively, a transducer is onward iff there is no delay in writing the output strings. As the input symbols are consumed from left to right, the output is written the moment it is determined. Onward SFSTs are discussed in detail by Oncina et al. [1993], as onwardness plays a large role in the OSTIA learning algorithm. However, Definition 5.5 generalizes that notion of onward transducers to partial functions. For total functions, the definitions are equivalent. In the next section we show that any DSFST can be made onward without changing its structure or the function it represents.

Figures 5.1 and 5.2 (from Oncina and García [1991]) illustrate the differences between SFSTs and Onward SFSTs: these SFSTs represent the same function but Figure 5.1 is onward while Figure 5.2 is not (and cannot be made) onward. Figure 5.3 is a DSFST and Figure 5.4 is an onward DSFST. They both represent the same function shown in Figures 5.1 and 5.2.

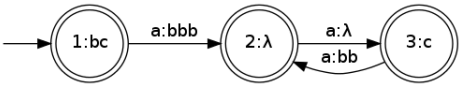

Figure 5.1: Onward SFST

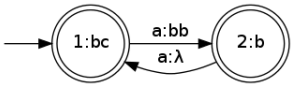

Figure 5.2: SFST

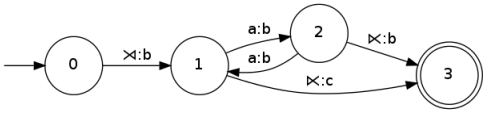

Figure 5.3: A DSFST

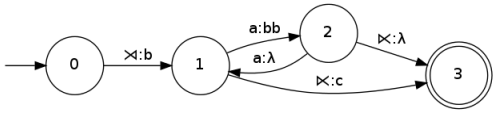

Figure 5.4: An onward DSFST

Readers are referred to Oncina and García [1991], Oncina et al. [1993], and Mohri [1997a] for more on subsequential transducers, and Eisner [2003] for generalizations regarding onwardness.

5.4 Deriving an Onward DSFST

We prove the following theorem which says for any DSFST τ, there is an onward DSFT with the exact same structure which recognizes the same function described by τ.

Theorem 5.2. For any trimmed DSFST $\tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$, there is an onward trimmed DSFST $\tau' = \langle Q, q_0, q_f, \Sigma, \Delta, \delta' \rangle$ such that $R(\tau) = R(\tau')$ and $(q, a, u, q') \in \delta$ iff there exists $u' \in \Delta^*$ and $(q, a, u', q') \in \delta'$.

The proof of the above theorem makes use of a function `push_back` which takes as arguments a transducer τ and a state q of τ and returns a transducer τ' in which the longest common prefix of the outputs of the transitions leaving q is pushed as a suffix onto the outputs of the transitions entering q (if they exist). More formally, for a trimmed DSFST, $\delta' = \delta$ if $q = q_0$, else $\delta' = \{(q, \sigma, x^{-1} \cdot ux, q) \mid (q, \sigma, u, q) \in \delta\} \cup \{(q, \sigma, x^{-1} \cdot u, q') \mid (q, \sigma, u, q') \in \delta\} \cup \{(q', \sigma, ux, q) \mid (q', \sigma, u, q) \in \delta\} \cup \delta \setminus \{(q_1, \sigma, u, q_2) \mid q_1 = q \text{ or } q_2 = q\}$, where $x = \texttt{lcp}(\{u \mid (q, \sigma, u, q) \in \delta\})$.

First we observe that pushing does not change the function described by a DSFST.

Lemma 5.3. Let $\tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ be a trimmed DSFST. For all $q \in Q$, we have $R(\tau) = R(\texttt{push\_back}(\tau, q))$.

Proof. Let $q \in Q$ and $(v, w) \in R(\tau)$. Then there exist $\sigma_1, \ldots, \sigma_n$ in Σ such that $v = \sigma_1 \ldots \sigma_n$, $\forall i, 1 \leq i \leq n$ $(q_i, \sigma_i, u_i, q_{i+1}) \in \delta$, $q_1 = q_0$, $q_{n+1} = q_f$, and $w = u_1 \ldots u_2$. Consider $\tau' = \texttt{push\_back}(\tau, q)$.

As the result trivially holds for $q = q_0$, suppose $q \neq q_0$ and let $w' = \tau'(v)$. Since $q \neq q_0$ and τ is trim, there is always at least one incoming transition into q and hence a place for the `lcp` to be pushed back to. Since `push_back` only modifies the output string of some transitions, it is the case that $w' = u'_1 \ldots u'_n$, with $\forall i, 1 \leq i \leq n$ $(q_i, \sigma_i, u'_i, q_{i+1}) \in \delta'$. Let $x = \texttt{lcp}(\{u : (q, \sigma, u, q') \in \delta\})$ and $i_1, \ldots, i_k$ $(k < n)$ the indexes such that $q_{i_j} = q$, for $1 \leq j \leq k$. We have $u'_{i_j - 1} = u_{i_j - 1} x$ and $u'_{i_j} = x^{-1} \cdot u_{i_j}$. Thus $w' = u_1 \ldots (u_{i_1 - 1} x)(x^{-1} \cdot u_{i_1}) \ldots (u_{i_k - 1} x)(x^{-1} \cdot u_{i_k}) \ldots u_n = w$ □

Notice that this proof is not possible without the initial transition with the start delimiter. In Figure 5.2, applying `push_back` to the state 1 modifies the represented function: for instance, (λ, bc) is replaced in the relation by (λ, c).

Proof. (Theorem 5.2). Since τ is a DSFST, q_0 has only one transition to one other state. So we can always push back to this transition without creating non-onwardness. We can cycle through the other states pushing back the `lcp`. The number of times we have to push back is finite. This is because the number of times we might need to push back from a state q is limited by the length of the output function of state q. Therefore, pushing will lead to an onward DSFST without, by Lemma 5.3, changing the language. □

The DSFST of Figure 5.3 corresponds to the SFST of Figure 5.2 after applying `push_back` once to state 1, and the DSFST of Figure 5.4 shows the onward result of recursively applying `push_back` according to the proof of Theorem 5.2.

In the rest of the chapter, we are only considering onward, trimmed, delimited transducers.

5.5 Learning Paradigm

The capability of the algorithm detailed in Section 5.8 is demonstrated in paradigm of identification in polynomial time and data [de la Higuera, 1997], studied in Chapter 2. We discussed the limitations of this IPTD paradigm there. However, since the structures considered here are of regular type, non of the drawbacks we described are of consideration in the present context.

We have to extend Definition 2.8 for set-driven algorithms that aim at learning functions. We first need to define the following notion:

Definition 5.6. Let $\mathbb{T}$ be a class of functions represented by some class $\mathbb{R}$ of representations.

1. A *sample* S for a function $t \in \mathbb{T}$ is a finite set of data consistent with t, that is to say $(w, v) \in S$ iff $t(w) = v$. The size of a sample S is the sum of the length of the strings it is composed of: $|S| = \sum_{(w,v)\in S} |w| + |v|$.

2. A $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathfrak{A}$ is a program that takes as input a sample for a function $t \in \mathbb{T}$ (possibly together with additional prior knowledge) and outputs a representation from $\mathbb{R}$.

The paradigm relies on the notion of characteristic sample, straightforwardly adapted here for functions:

Definition 5.7 (Characteristic sample)**.** Given a $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathfrak{A}$, we say that a sample CS is a *characteristic sample* of a transduction $t \in \mathbb{T}$ if for all samples S such that $CS \subseteq S$, $\mathfrak{A}$ returns a representation r such that $r(w) = t(w)$, for all $w \in \mathtt{dom}(t)$.

This definition is the one used in the proof of the OSTIA algorithm. We are actually using a strongest version:

Definition 5.8 (Strong characteristic sample)**.** Given a $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathfrak{A}$, we say that a sample CS is a *strong characteristic sample* of a transducer $\tau \in \mathbb{R}$ if for all samples S such that $CS \subseteq S$, $\mathfrak{A}$ returns a representation r such that $r = t$.

The learning paradigm can now be defined as follows.

Definition 5.9 (Strong identification in polynomial time and data)**.** A class $\mathbb{T}$ of functions is *strongly identifiable in polynomial time and data (SIPTD)* if and only if there exist a $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathfrak{A}$ and two polynomials $p()$ and $q()$ such that:

1. Given a sample S of size m for $t \in \mathbb{R}$, $\mathfrak{A}$ returns a hypothesis $r \in \mathbb{R}$ in $\mathcal{O}(p(m))$ time.
2. For each representation $t \in \mathbb{R}$ of size k, there exists a strong characteristic sample of L of size at most $\mathcal{O}(q(k))$.

For a discussion about (weak and) strong identification, the reader is referred to the work of Clark [2014]. Section 5.8 shows how out SOSFIA meets these criteria.

5.6 Target Classes

Empty transducers provide the *a priori* structural information that will allow learning of the function. An *empty DSFST* (henceforth 'empty transducer') is a DSFST whose transition outputs are all blanks ($\square$). An empty transducer τ_E defines a *class* of functions $\mathcal{T}$ which is exactly the set of functions which can be created by taking the states and transitions of τ_E and replacing the blanks with output strings, maintaining onwardness.

Definition 5.10. An *empty transducer* τ_E is a DSFST $\langle Q, q_0, q_f, \Sigma, \{\square\}, \delta\rangle$ such that for all $(q, a, u, q') \in \delta$, $u = \square$.

Definition 5.11. The class of functions $\mathcal{T}_{\tau_E}$ described by an empty DSFST $\tau_E = \langle Q, q_0, q_f, \Sigma, \{\square\}, \delta\rangle$ is:

$$\mathcal{T}_{\tau_E} = \Big\{ t \mid \exists \tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta_\tau\rangle \quad \text{s.t. } \forall (q,\sigma,v,r) \in \delta_\tau, (q,\sigma,\square,r) \in \delta \text{ and } \forall (q,\sigma,\square,r) \in \delta, \exists v \in \Delta^* : (q,\sigma,v,r) \in \delta_\tau; \; t(\tau) \text{ and } \mathit{onward}(\tau) \Big\}$$

Lemma 5.4. For all τ_E, all $t \in \mathcal{T}_{\tau_E}$ are subsequential functions.

Proof. From Definitions 5.2, 5.10, and 5.11. □

We observe there are infinitely many DSFSTs and therefore infinitely many such classes. Also, for each τ_E, each class $\mathcal{T}_{\tau_E}$ contains infinitely many functions since there is no upper bound on the length of output strings.

5.7 Learning Algorithm

Given a class of functions $\mathcal{T}_{\tau_E}$, a target function $t \in \mathcal{T}_{\tau_E}$, and a sample S of t, the Structured Onward Subsequential Function Inference Algorithm (SOSFIA) iterates through

the states of τ_E. At each state, it sets the output of each outgoing transition to be the *minimal change in the output* generated by this transition, according to S. To calculate this, we first define the *common output* of an input prefix w.

Definition 5.12. The *common output* of an input prefix w in a sample $S \subset \Sigma^* \times \Delta^*$ is the `lcp` of all $t(wv)$ that are in S: $\texttt{CommonOut}_S(w) = \texttt{lcp}(\{u \in \Sigma^* \mid \exists v \text{ s.t. } (wv, u) \in S\})$

The minimal change in the output is then simply the difference between the common outputs of w and $w\sigma$.

Definition 5.13. The *minimal change in the output* in $S \subset \Sigma^* \times \Delta^*$ from w to $w\sigma$ is:

$$\texttt{Min\_Change}_S(\sigma, w) = \begin{cases} \texttt{CommonOut}_S(\sigma) & \text{if } w = \lambda \\ \texttt{CommonOut}_S(w)^{-1}\texttt{CommonOut}_S(w\sigma) & \text{otherwise} \end{cases}$$

This gives us exactly the output needed to maintain onwardness, which will in turn guarantee that the SOSFIA converges to the correct function, provided that the sample contains enough information. Note that the minimal change is calculable for S because it is finite. The algorithm is presented below.

Algorithm 1 Structured Onward Subsequential Function Inference Algorithm (SOSFIA)

Data: A sample $S \subset \rtimes\Sigma^* \ltimes \times\Delta^*$, an empty DSFST $\tau_E = \langle Q, q_0, q_f, \Sigma, \{\square\}, \delta\rangle$
Result: τ_E as a DSFST with filled transitions

```
F ← empty_Queue
Push(F, (q_0, λ))
mark(q_0)
while F is not empty do
    (q, w) ← Shift_First(F)
    for σ ∈ Σ ∪ {⋊, ⋉} in lexicographic order do
        for δ_i = (q, σ, □, q') ∈ δ do
            if there exists σ' ≠ σ such that (q, σ', u, q'') ∈ δ then
                Change δ_i to (q, σ, v, r), where v = Min_Change_S(w, σ)
            else
                Change δ_i to (q, σ, λ, r)
            if q' is not marked then
                Push(F, (q', wσ))
                mark(q')
return τ_E;
```

The SOSFIA does a breadth-first parsing of the empty DSFST, storing in the queue F (a First-In/First-Out data type) seen but untreated states together with the smallest

prefix that leads to each of these states: this is ensured by the lexicographic order in which the letters are considered and by the fact that only unmarked states (i.e., states that have not been discovered so far) are pushed into F. When a state is treated, all its outgoing transitions are considered. The output of this transition is set to be the result of the `Min_Change` function on the shortest prefix arriving at this state and the input letter of the transition, unless it is the only transition leaving this state in which case the output is set to be λ[3].

5.8 Learning Result

Let $\tau = \langle Q, q_0, q_f, \Sigma, \delta \rangle$ be a DSFST. We define $MinPref(q) = min\{w \in \rtimes\Sigma^*\ltimes : (q_0, w, u, q) \in \delta^*\}$.

Lemma 5.5 (Characteristic Sample)**.** Let $\tau_* = \langle Q, q_0, q_f, \Sigma, \delta_* \rangle$ be the target DSFST. We define requirements on a sample CS as followed:

- For all $(q, \sigma, u, q') \in \delta_*$, $q \neq q_0$, there exists $(v\sigma v', w)$ in CS, with $v = MinPref(q)$,
- For all $q \in Q$ for which at least two outgoing transitions exist, for all $(q, \sigma, s, q') \in \delta_*$, if there exists $q_1, \ldots, q_n \in Q$ such that
  1. $q' = q_1$, and
  2. $\forall i < n$, q_i has only one outgoing transition and $(q_i, \sigma_i, \lambda, q_{i+1}) \in \delta_*$, and
  3. $(q_{n-1}, \sigma_n, s, p) \in \delta$, and
  4. there exists $\sigma' \neq \sigma_n$ such that $(q_{n-1}, \sigma', s', p') \in \delta_*$ and $\mathtt{lcp}(s, s') = \lambda$

  then there exists $(v\sigma\sigma_1 \ldots \sigma_n w, z)$ and $(v\sigma\sigma_1 \ldots \sigma_{n-1}\sigma' w', z')$ in CS, with both being an element of the target function and $v = MinPref(q)$.

Any sample fulfilling these requirements is a characteristic sample for the SOSFIA algorithm.

Notice that the third and fourth items are possible only for onward transducers.

Proof. (Sketch) Suppose that a sample CS fulfilling the requirements of Theorem 5.5 is contained in S. Let $\tau_A = \langle Q, q_0, q_f, \Sigma, \delta_A \rangle$ be the transducer returned by the algorithm on S. Let $(q, \sigma, s, q') \in \delta_*$ and $v = MinPref(q)$.

If there is no other outgoing transition from q, then $s = \lambda$ as τ_A is onward. By the definition of the algorithm, $(q, \sigma, \lambda, q') \in \delta_A$.

[3]Readers may wonder whether the entire if/then/else statement can be replaced with only the part between *then* and *else* (so changing δ_i to (q, σ, v, r)). We believe this is the case and while it makes for a more aesthetically pleasing algorithm, it significantly complicates the identification proof; hence the current presentation of the algorithm.

Suppose now that there exist several transitions from the state q. By construction of the sample (first item), there exists $(v\sigma v', u)$ in S. Therefore, by the definition of the learning algorithm: $(q, \sigma, \texttt{Min\_Change}_S(v, \sigma), q') \in \delta_A$.

Suppose $q \neq q_0$. As q admits several outgoing transitions, there exists (q, σ_1, s_1, q_1) and (q, σ_2, s_2, q_2) in δ_* such that $\texttt{lcp}(s_1, s_2) = \lambda$. By construction of the sample, $\exists v_1, v_2, u_1, u_2$ such that $(v\sigma_1 v_1, u_1)$ and $(v\sigma_2 v_2, u_2)$ are in the sample. We then have $\texttt{lcp}(u_1, u_2) = x$ and therefore $\texttt{CommonOut}_S(v) = x$.

By the definition of a function, $xs \in \texttt{pref}(\texttt{CommonOut}_S(v\sigma))$. If $q' = q_f$ then $(v\sigma, xs)$ is in S as there is no outgoing transition from the final state. Therefore $xs = \texttt{CommonOut}_S(v\sigma))$ and then $\texttt{Min\_Change}(v, \sigma) = s$.

Suppose now that $q \neq q_f$. By construction, there exists $(v\sigma\sigma_1 \dots \sigma_n w, u)$ and $(v\sigma_1 \dots \sigma_{n-1}\sigma' w', u')$ in the sample. We have $(q', \sigma_1 \dots \sigma_{n-1}, \lambda, q'') \in \delta_*^*$, therefore $u = xss_1v_1$ and $u' = xss_2v_2$, where $(q'', \sigma_n, s_1, p) \in \delta_*$ and $(q'', \sigma', s_2, p') \in \delta_*$ and $\texttt{lcp}(s_1, s_2) = \lambda$. Therefore $xs = \texttt{CommonOut}_S(v\sigma)$ and then $\texttt{Min\_Change}(v, \sigma) = s$.

The proof for the case $q = q_0$ is of similar nature. □

Time Complexity. The *while* loop is executed exactly $|Q|$ times since every state is considered once and only once. Then the two *for* loops are executed overall $|\Sigma| \cdot card(\delta)$ times since each transition is modified only once ($card(\delta)$ is the cardinality of δ). In the worst case, the algorithm launches `Min_Change` for each transition, which corresponds to the computation of two `lcp`. Each of these calculations is doable in $O(n \cdot m)$ with an adapted structure for storing strings (e.g., prefix tree acceptor), where m is the length of the longest string in the right projection of the sample, and n is the number of strings in the left projection of the sample. The overall time complexity of the algorithm is thus in $O(n \cdot m)$ since $|Q|$ and $card(\delta)$ are fixed.

Data Complexity. The size of the characteristic sample defined in Lemma 5.5 for a target DSFST $\tau = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ is in $O(|\tau|^2)$.

Indeed, one needs at most $|Q|$ elements to fulfill the first requirement. Each of these string pairs can have a left projection of length at most $|Q|$ and a right projection of length at most $\sum_{(q,\sigma,u,q') \in \delta} |u|$. This is due to the fact that in the worst case these elements correspond to a parsing through all states and all transitions of τ (in case τ admits cycles, longest elements can be considered, but we focus only on the smallest ones in this proof). Therefore the size of this part of the characteristic sample is in $O(|Q| \cdot (|Q| + |\tau|)) = O(|Q| \cdot |\tau|) = O(|\tau|^2)$.

The number of elements needed for a sample to fulfill the second item of the definition is at most $2 \cdot card(\delta)$. For the same reason as previously, these elements can have a left projection of length at most $|Q|$ and a right projection of length at most $\sum_{(q,\sigma,u,q') \in \delta} |u|$. The size of this second part of the characteristic sample is thus in $O(card(\delta) \cdot |\tau|) = O(|\tau|^2)$.

Theorem 5.6. SOSFIA strongly identifies in polynomial time and data any class of functions definable by a given empty transducer.

Proof. Immediate from the previous complexity results. □

5.9 Demonstrations

The applicability of subsequential transducers to natural language phonology and morphology is discussed in detail in [Mohri, 1997b, Beesley and Karttunen, 2003]. Briefly, morphological operations can be thought of as a transformation from a bare form to an affixed form, and a foundational principle of modern generative phonology is that there is a phonological mapping from abstract, lexical 'underlying' representations of words and morphemes to their concrete surface pronunciations [Hayes, 2011]. In this section we present several demonstrations of the SOSFIA using linguistically-motivated examples.

5.9.1 Input Strictly Local Functions

As mentioned above, Chandlee [2014] shows that the input-output mapping of many phonological processes can be modeled with Input Strictly Local (ISL) functions. The automata-theoretic characterization of these functions provides sufficient structure for the SOSFIA to identify them in the limit. We tested the learner on three example ISL functions, all of which can be thought of as phonological 'repairs' that prevent the underlying sequence D⋉ from surfacing. The examples are summarized in (1).

(1)

Process	Rule
Final devoicing	D → T / __⋉
Deletion	D → ∅ / __ ⋉
Epenthesis	∅ → V / D__ ⋉

These notations are very common in computational linguistic but they might require some explanations here: the first line means that a D is replaced by a T if it is at the end of a sequence; the second states that a D is erased if it happens at the end of a sequence; the last implies that a V is created after a final D.

Using the alphabet $\Sigma = \{\text{D, T, N, V}\}$, where D is a voiced obstruent, T is a voiceless obstruent, N is a sonorant consonant, and V is a vowel, we constructed a data set of 1365 string pairs. The left projection of the data set is $\Sigma^{\leq 5}$; each string in the left projection was paired with an output string according to the target function. The same empty FST was used in all three test cases, as the FST for each rule only differs in terms of the output strings. The output of the SOSFIA is correct in all three cases; as an example, the resulting machine for the final devoicing test case is

given in Figure 5.5. The machines for the deletion and epenthesis cases were identical, except for that in the deletion case the transition from state 2 to state 6 has an output of λ, and in the epenthesis case the transitions out of state 2 are as follows: $(2, D, D, 2), (2, T, T, 3), (2, V, V, 5), (2, N, N, 4), (2, \ltimes, V, 6)$.

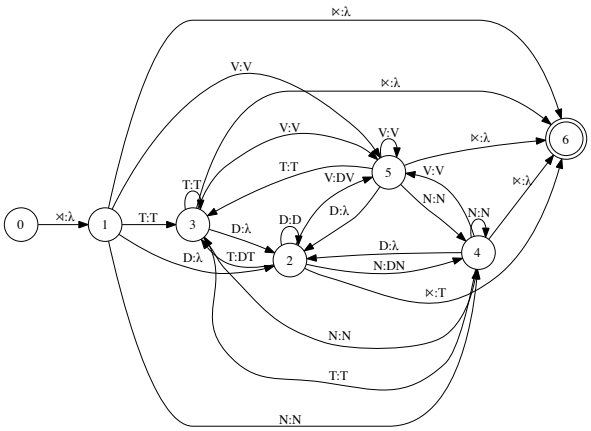

Figure 5.5: Final devoicing test case

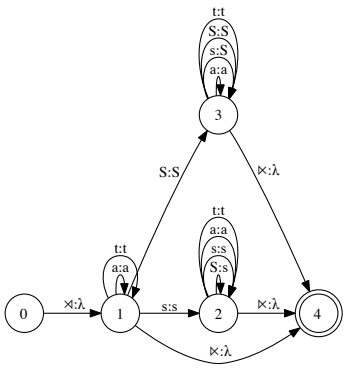

Figure 5.6: Sibilant harmony test case

5.9.2 Non-ISL Phonological Processes

There also exist phonological processes that cannot be modeled with ISL functions. These are 'long-distance' processes such as the sibilant harmony process attested in Samala. In this language, all sibiliant sounds (i.e., {s, ʃ}) in a word must be the same as the rightmost one [Applegate, 1972, Hansson, 2010]. An example is shown in (2).

(2) /hasxintilawaʃ/ ↦ [haʃxintilawaʃ] 'his former gentile name'

Using the (simplified) alphabet $\Sigma = \{s, ʃ, t, a\}$, we constructed a dataset of string pairs in which the left projection is $\Sigma^{\leq 4}$. Each of these strings was paired with one in which all sibiliants (if any) assimilate to the rightmost one. As shown in Figure 5.6, the output of the SOSFIA correctly models this process.

Two things should be noted about this test case. One is that this process is right subsequential, which means the input and output strings must be reversed to get the correct mapping. Two, the empty FST given to the learner includes separate states for words in which 's' is the rightmost sibiliant and words in which 'ʃ' is the rightmost sibilant (i.e., the *a priori* knowledge given to the learner identifies the set of segments involved in the process). This is akin to phonological theories in which certain classes of segments (here the sibilants) are represented on distinct tiers (e.g., vowel harmony;

[Clements, 1976]).

5.9.3 Morphological Parsing

The learner was also tested on a morphology-to-phonology function, as discussed in [Beesley and Karttunen, 2003], in which meanings of morphemes are mapped to their pronunciation. The data were from Swahili [Hayes, 2011] verbs, which show a series of prefixes indicating person, number, and tense. The following example shows the morphological breakdown of *nimenipenda* 'I have liked myself':[4]

(3)

ni	+	me	+	ni	+	penda	'I have liked myself'
1st-acc		perf		1st-nom		like	

The learner was given an empty transducer representing the possible morpheme orders and 90 pairs of the shape <*morpheme string, phoneme string*>, as exemplified in (4).

(4)

1st-nom+perf+1st-acc+like	,	nimenipenda
3rd-nom+pres+1st-acc+like	,	ananipenda
2nd-nom+perf+1st-pl-acc+beat	,	umetupiga

The learner correctly learned the morphological function; i.e., for any transition on '1st-nom' it learned that the output should be 'ni'.

5.10 Conclusion

This chapter contains two main contributions. First, we introduced DSFSTs and showed that, unlike SFSTs, they can be made onward without modifying their structure. Second, we introduced a learning algorithm for proper subclasses of subsequential functions in which all member functions share a SFST structure. There are infinitely many such classes, each with infinitely many member functions. While the learner is given a significant amount of prior knowledge in the form of the empty transducer representing the class of functions, this allows for extremely efficient learning measured in both time and data. We have also shown specific applications for the learner in the linguistic domains of phonology and morphology.

The algorithm presented in this chapter returns a function even given incomplete data. With a sacrifice in added time complexity linear to the sum of the length of the left projection of the sample, the algorithm can be modified so that it checks if any element of the sample uses a transition it is about to modify. If it is not the case it can stop since it is sure that the data is not sufficient for learning. We have opted not

[4] Abbreviations: 1st-acc = first person accusative ('me'); 1st-nom = first person nominative ('I'); 1st-pl-acc = first person plural accusative ('us'); 2nd-nom = second person nominative ('we'); 3rd-nom = third person nominative ('he'); pres = present tense; perf = perfect tense.

to include this functionality, as this is not enough to guarantee that there is sufficient data to calculate the correct output for the transition (as it is still unknown whether *all* the input/output pairs necessary to calculate the correct output for that transition are in the data).

Bibliography

D. Angluin. Inference of reversible languages. *J. ACM*, 29(3):741–765, July 1982. 125

R. Applegate. *Inseño Chumash grammar*. PhD thesis, University of California, Berkeley, 1972. 137

K. R. Beesley and L. Karttunen. *Finite State Morphology*. CSLI Publications, 2003. 136, 138

J. Chandlee. *Strictly Local Phonological Processes*. PhD thesis, University of Delaware, 2014. 125, 136

A. Clark. Inference of inversion transduction grammars. In *Proc. of the International Conference on Machine Learning*, pages 201–208, 2011. 124

A. Clark. Learning trees from strings: A strong learning algorithm for some context-free grammars. *Journal of Machine Learning Research*, 14:3537–3559, 2014. 132

G. N. Clements. *Vowel harmony in nonlinear generative phonology: an autosegmental model*. Indiana University Linguistics Club, 1976. 138

C de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning Journal*, 27:125–138, 1997. 125, 131

C. de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010. 124

J. Eisner. Simpler and more general minimization for weighted finite-state automata. In *Proc. of the Joint Meeting of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics*, pages 64–71, 2003. 129

E. Filiot and P.-A. Reynier. Transducers, logic and algebra for functions of finite words. *ACM SIGLOG News*, 3(3):4–19, 2016. 124

P. Garcia, E. Vidal, and J. Oncina. Learning locally testable languages in the strict sense. In *Proc. of the Workshop on Algorithmic Learning Theory*, pages 325–338, 1990. 125

G. Ó. Hansson. *Consonant harmony: long-distance interaction in phonology*. Berkeley, CA: University of California Press, 2010. 137

B. Hayes. *Introductory Phonology*. John Wiley & Sons, 2011. 136, 138

J. Heinz and J. Rogers. Learning subregular classes of languages with factored deterministic automata. In A. Kornai and M. Kuhlmann, editors, *Proc. of the Meeting on the Mathematics of Language (MoL 13)*, pages 64–71. Association for Computational Linguistics, 2013. 125

A. Jardine, J. Chandlee, R. Eyraud, and J. Heinz. Very efficient learning of structured classes of subsequential functions from positive data. In A. Clark, M. Kanazawa, and R. Yoshinaka, editors, *Proc. of the International Conference on Grammatical Inference*, volume 34 of *PMLR*, pages 94–108, 2014. 124

R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, 1971. 125

M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997a. 129

M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997b. 136

M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69 – 88, 2002. 124

J. Oncina and P. García. Inductive learning of subsequential functions. Technical Report DSIC II-34, Univ. Politécnia de Valencia, 1991. 128, 129

J. Oncina, P. García, and Enrique Vidal. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458, May 1993. 124, 128, 129

Jaques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009. Translated by Reuben Thomas from the 2003 edition published by Vuibert, Paris. 128

M. Veanes, P. Hooimeijer, B. Livshits, D. Molnar, and N. Bjorner. Symbolic finite state transducers: Algorithms and applications. *SIGPLAN Not.*, 47(1):137–150, 2012. 124

Chapter 6

On Learning with Locality Constraints

Contents

Context of this work

This chapter presents the main results obtained during the on-going collaboration with Jeffrey Heinz and Jane Chandlee. In her PhD dissertation of 2014, Jane introduced two classes of transductions with phonological motivations. We then focused on learning these classes, which implied the redefinition of one of them and required a characterization in terms of finite state models for both of them. Learning algorithms have been designed and successfully studied, and an encompassing class of transductions has been defined and studied.

The work on the two first classes has been published in the *Transaction of the Association for Computational Linguistics* [Chandlee et al., 2014] and in the proceedings

of the conference on *Mathematics of Language* [Chandlee et al., 2015]. The work on the more general class has not been published yet.

6.1 Introduction

A foundational principle of modern generative phonology is that systematic variation in morpheme pronunciation is best explained with a single *underlying* representation of the morpheme that is transformed into various *surface* representations based on context [Kenstowicz and Kisseberth, 1979, Odden, 2014]. Thus, much of generative phonology is concerned with the nature of these transformations, more than just the surface language: this is our main motivation to introduce and study new classes of transductions.

Indeed, one way to better understand the nature of linguistic phenomena is to develop strong computational characterizations of them. Discussing SPE-style phonological rewrite rules [Chomsky and Halle, 1968], Johnson [1972, p. 43] expresses the reasoning behind this approach:

> It is a well-established principle that any mapping whatever that can be computed by a finitely statable, well-defined procedure can be effected by a rewriting system (in particular, by a Turing machine, which is a special kind of rewriting system). Hence any theory which allows phonological rules to simulate arbitrary rewriting systems is seriously defective, for it asserts next to nothing about the sorts of mappings these rules can perform.

This leads to the important question of what kinds of transformations ought a theory of phonology allow?

Earlier work suggests that phonological theories ought to exclude non-regular relations [Johnson, 1972, Kaplan and Kay, 1994, Frank and Satta, 1998, Graf, 2010]. More recently, it has been hypothesized that phonological theory ought to only allow certain subclasses of the regular relations [Gainor et al., 2012, Chandlee et al., 2012, Chandlee and Heinz, 2012, Payne, 2017, Luo, 2014, Heinz and Lai, 2013]. This research places particular emphasis on *subsequential* functions, which can informally be characterized as functions definable with a weighted, deterministic finite-state acceptor where the weights are strings and multiplication is concatenation. The aforementioned work suggests that this hypothesis enjoys strong support in segmental phonology, with interesting and important exceptions in the domain of tone [Jardine, 2016].

Recent research has also showed an increased awareness and understanding of subregular classes of formal languages and their importance for theories of *phonotactics*, that is, the study of the surface form [Heinz, 2007, 2009, 2010, Rogers et al., 2010, Rogers and Pullum, 2011, Rogers et al., 2013]. While many of these classes and their properties were studied much earlier [McNaughton and Papert, 1971, Thomas, 1997],

little to no attention has been paid to similar classes properly contained within the subsequential functions. Thus, at least within the domain of segmental phonology, there is an important question of whether stronger computational characterizations of phonological *transformations* are possible, as seems to be the case for phonotactics.

Additionally, an important thread in much of the aforementioned research was that the stronger computational characterizations of phonological phenomena can help address the learning problem. Theoretical learning results have existed for many years for Strictly Local (SL) languages [Garcia et al., 1990], total subsequential functions [Oncina et al., 1993], and many other classes (see for instance the work of García and Ruiz [2004]). Recent work generalizes some of these results [Heinz et al., 2012] and develops variants that apply generally to a wide range of subclasses at the cost of some expressibility [Heinz and Rogers, 2013, Jardine et al., 2014].

In this chapter, we focus on defining, studying, and learning subclasses of regular transduction based on the properties of well-studied Strictly Local formal languages [McNaughton and Papert, 1971, Rogers and Pullum, 2011, Rogers et al., 2013]. These are languages that can be defined with grammars of substrings of length k (called k-factors), such that a string is in the language only if its own k-factors are a subset of the grammar.

It has been shown that these languages can model natural language phonotactic constraints which pick out contiguous substrings bounded by some length k [Heinz, 2007, 2010].

To extend this work to phonological transformations, we detail here three classes of transductions, namely the Input Strictly Local functions (ISL), the Output Strictly Local functions (OSL), and their generalization: the Input Output Strictly Local functions (IOSL). They correspond to three different ways to generalize to functions the SL languages.

Section 6.2 provides the needed definitions and preliminary remarks. ISL and OSL functions are introduced in Section 6.3 together with the main results obtained for each of them. Section 6.4 deals with the more general IOSL functions: formal definitions, finite state characterization, and relations between classes are detailed. Section 6.5 contains the learning algorithm for IOSL functions and the proof that it identifies the class in polynomial time and data while Section 6.6 concludes.

6.2 Preliminaries

We use throughout this chapter the same definitions and notations than in the previous chapters, to which we add the following ones.

The set of all possible strings of length n or less over an alphabet Σ is denoted $\Sigma^{\leq n}$. Similarly to the set of prefixes defined in Section 5.2, the set of suffixes of w is $\texttt{suff}(w) = \{s \in \Sigma^* \mid (\exists p \in \Sigma^*)[w = ps]\}$. For all $w \in \Sigma^*$ and $n \in \mathbb{N}$, $\texttt{suff}^n(w)$ is

the single suffix of w of length n if $|w| \geq n$; otherwise $\texttt{suff}^n(w) = w$. The reduction expressed in the following remark will prove useful later:

Remark 6.1. For all $w, v \in \Sigma^*, n \in \mathbb{N}$, $\texttt{suff}^n\big(\texttt{suff}^n(w)v\big) = \texttt{suff}^n(wv)$.

The transducers of consideration in this chapter are the Delimited Subsequential Finite State Transducers introduced in Definition 5.3. As it is already the case in Chapter 5, onwardness (see Definition 5.5) is an important and desired property for transducers. The following technical lemma will be of crucial interest, for which we need to define for each state q of a DSFST $\langle Q, q_0, q_f, \Sigma, \Delta, \delta\rangle$, the *outputs* of the edges out of q: $\texttt{outputs}(q) = \big\{u \mid (\exists \sigma \in \Sigma \cup \{\rtimes, \ltimes\})(\exists q' \in Q)[(q, \sigma, u, q') \in \delta]\big\}$.

Lemma 6.2. If a DSFST $\mathcal{T}$ recognizes f and is onward then $\forall q \neq q_0\ \texttt{lcp}(\texttt{outputs}(q)) = \lambda$ and $\texttt{lcp}(\texttt{outputs}(q_0)) = \texttt{lcp}(f(\Sigma^*))$.

Proof. By construction of a DSFST, only one transition leaves q_0: $(q_0, \rtimes, u, q)$. This implies $(q_0, \rtimes\lambda, u, q) \in \delta^*$ and as the transducer is onward we have $\texttt{lcp}(\texttt{outputs}(q_0)) = \texttt{lcp}(u) = u = \texttt{lcp}(f(\lambda\Sigma^*)) = \texttt{lcp}(f(\Sigma^*))$. Now take $q \neq q_0$ and $w \in \Sigma^*$ such that $(q_0, \rtimes w, u, q) \in \delta^*$. Suppose $\texttt{lcp}(\texttt{outputs}(q)) = v \neq \lambda$. Then v is a prefix of $\texttt{lcp}(\{f(w\sigma x) \mid \sigma \in \Sigma \cup \{\ltimes\}, x \in \Sigma^*\})$ which implies uv is a prefix of $\texttt{lcp}(f(w\Sigma^*))$. But $v \neq \lambda$, contradicting the fact that $\mathcal{T}$ is onward. □

6.3 Input and Output Strictly Local functions

Before defining the more general Input-Output Strictly Local functions (in Section 6.4), we first defined Input Strictly Local (ISL) functions and Output Strictly Local (OSL) functions. ISL were originally introduced in Chandlee [2014] PhD dissertation and in a paper by Chandlee et al. [2014], while OSL were proposed by Chandlee et al. [2015].

These classes (and the one of Section 6.4) generalize the Strictly Local languages to functions based on a defining property of SL languages, the Suffix Substitution Closure [Rogers and Pullum, 2011].

Theorem 6.3 (Suffix Substitution Closure)**.** L is Strictly Local iff for all strings u_1, v_1, u_2, v_2, there exists $k \in \mathbb{N}$ such that for any string x of length $k-1$, if u_1xv_1, $u_2xv_2 \in L$, then $u_1xv_2 \in L$.

An important corollary of this theorem follows.

Corollary 6.4 (Suffix-defined Residuals)**.** L is Strictly Local iff for all strings u_1, u_2, there exists $k \in \mathbb{N}$ such that for any string x of length $k-1$ it is the case that the *residuals* (the language equivalent of tails) of u_1, u_2 with respect to L are the same; formally, that $\{v \mid u_1xv \in L\} = \{v \mid u_2xv \in L\}$.

Input and Output Strictly Local functions are defined in the manner suggested by the corollary.

6.3.1 Input Strictly Local functions

The definition of ISL functions is straightforward:

Definition 6.1 (Input Strictly Local Functions)**.** A function $f : \Sigma^* \to \Delta^*$ is Input Strictly Local (ISL) if there is a k such that for all $u_1, u_2 \in \Sigma^*$, if $\texttt{suff}^{k-1}(u_1) = \texttt{suff}^{k-1}(u_2)$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$.

Informally, a function is k-ISL if the output of every input string $a_0 a_1 \cdots a_n$ is $u_0 u_1 \cdots u_n$ where u_i is a string which only depends on a_i and the $k-1$ input symbols before a_i (so $a_{i-k+1} a_{i-k+2} \cdots a_{i-1}$).

The theorem below establishes an automata-theoretic characterization of ISL functions.

Theorem 6.5. A function f is ISL iff there is some k such that f can be described with a SFST for which

1. $Q \subseteq \Sigma^{\leq k-1} \cup \{q_0, q_f\}$
2. $(q_0, \rtimes, \texttt{lcp}(f(\Sigma^*)), \lambda) \in \delta$
3. $(\forall q \in Q \setminus \{q_0\}, \forall a \in \Sigma, \forall u \in \Gamma^*)$
   $\big[(q, a, u, q') \in \delta \Rightarrow q' = \texttt{suff}^{k-1}(qa)\big]$.

Proof. The complete proof of this theorem is given by Chandlee et al. [2014]. We do not reproduce it here since the proof of Theorem 6.13 is a generalization of it. □

This theorem helps make clear how ISL functions are Markovian: the output for input symbol a depends on the last $(k-1)$ input symbols. Also, since the transducer defined in Theorem 6.5 is deterministic, it is unique and we refer to it as $\mathcal{T}_f^{\texttt{ISL}}$. $\mathcal{T}_f^{\texttt{ISL}}$ may not be isomorphic to $\mathcal{T}_{Cf}$. Figure 6.1 shows $\mathcal{T}_f^{\texttt{ISL}}$ (with $k = 2$) and $\mathcal{T}_{Cf}$ for the identity function.[1]

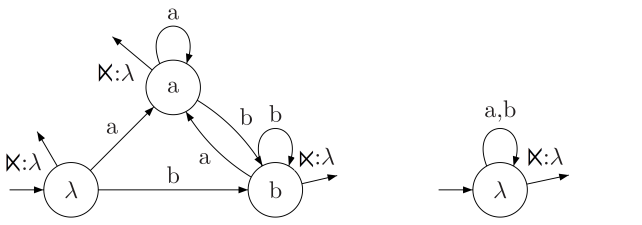

Figure 6.1: Non-isomorphic $\mathcal{T}_f^{\texttt{ISL}}$ (left) and $\mathcal{T}_{Cf}$ (right) (traditional SFSTs)

[1] In all figures, single-symbol transition labels indicate that the input and output are identical, and the final output function is represented as a transition on the end-of-word symbol $\rtimes$.

In an already cited paper [Chandlee et al., 2014] we provided a learning algorithm called ISLFLA for ISL functions using a state merging approach, and proved the following theorem:

Theorem 6.6. ISLFLA identified in polynomial time and data the class of k-ISL functions in the sense of Definition 5.9.
Its time complexity is in $\mathcal{O}(n \cdot m \cdot k \cdot |\Sigma|)$ where n is sum of the length of the input strings of the learning sample and m is the length of the longest output string in this sample.
The size of its exhibited characteristic sample is in $\mathcal{O}(|Q_\diamond| \cdot |\Sigma| \cdot k \cdot m + |Q_\diamond|^2 \cdot m + |Q_\diamond| \cdot |\Sigma| \cdot p)$ where $Q_\diamond$ is the set of states of the target transducer, and $p = max\{|v| \mid (q, \ltimes, v, q_f) \in \delta_\diamond\}$, with $\delta_\diamond$ the transition function of the target DSFST.

6.3.2 Output Strictly Local functions

To design OSL we first need to define the prefix function associated to a subsequential function.

Definition 6.2 (Prefix function)**.** Let $f : \Sigma^* \to \Delta^*$ be a subsequential function. We define the prefix function f^p associated to f as a function from Σ^* to Δ^* such that $f^p(w) = \texttt{lcp}\{f(w\Sigma^*)\}$.

This definition allows for separation of what happens during the computation from what happens at its end.

Two simple but useful remarks can be done about prefix functions:

Remark 6.7. if $\mathcal{T}$ is the canonical transducer of f then $f^p(w) = u \iff \exists q, (q_0, w, u, q) \in \delta^*_{\mathcal{T}}$.

Remark 6.8. If f is sequential then $f = f^p$.

We can now give the definition of OSL functions.

Definition 6.3 (Output Strictly Local Function)**.** We say that a subsequential function f is k-OSL if for all w_1, w_2 in Σ^*, $\texttt{Suff}^{k-1}(f^p(w_1)) = \texttt{Suff}^{k-1}(f^p(w_2)) \Rightarrow \texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$.

The Definition of k-OSL transducers is slightly more technical than the one of k-ISL transducers:

Definition 6.4 (k-OSL transducer)**.** An onward DSFST $\mathcal{T} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ is k-OSL if

1. $Q = S \cup \{q_0, q_f\}$ with $S \subseteq \Delta^{\leq k-1}$
2. $(\forall u \in \Delta^*)\big[(q_0, \rtimes, u, q') \in \delta \Longrightarrow q' = \texttt{Suff}^{k-1}(u)\big]$

3. $(\forall q \in Q \backslash \{q_0\}, \forall a \in \Sigma, \forall u \in \Delta^*)$
$\big[(q, a, u, q') \in \delta \Longrightarrow q' = \texttt{suff}^{k-1}(qu)\big]$.

We proved in a recent paper [Chandlee et al., 2015] that:

- Any k-OSL transducers computes a k-OSL function;
- The following theorem holds, which implies that k-OSL functions and functions represented by k-OSL DSFSTs exactly correspond.

Theorem 6.9. Let f be a k-OSL function. The DSFST $\mathcal{T}$ defined as followed computes f and is a k-OSL transducer:

- $Q = S \cup \{q_0, q_f\}$ with $S \subseteq \Delta^{\leq k-1}$
- $(q_0, \rtimes, u, \texttt{Suff}^{k-1}(u)) \in \delta \Longleftrightarrow u = f^p(\lambda)$
- $a \in \Sigma$, $(q, a, u, \texttt{Suff}^{k-1}(qu)) \in \delta$, $\Longleftrightarrow (\exists w)\big[\texttt{Suff}^{k-1}(f^p(w)) = q \wedge f^p(wa) = vqu$ with $v = f^p(w) \cdot q^{-1}\big]$,
- $(q, \ltimes, u, q_f) \in \delta \Longleftrightarrow u = f^p(w_q)^{-1} \cdot f(w_q)$, where $w_q = \min_{\lhd}\{w \mid \exists u, (q_0, \rtimes w, u, q) \in \delta^*\}$.

The diagram below helps express pictorially how the transitions are organized per the second and third bullets above. The input is written above the arrows, and the output written below.

$$q_0 \xrightarrow[f^p(w)=vq]{\rtimes w} q \xrightarrow[u]{a} q'$$

The detailed proof can be found in the already cited paper and we also refer the Reader to the proof of Theorem 6.13 that generalizes the one of Theorem 6.9.

A learning algorithm called OSLFIA was designed and the following theorem proven [Chandlee et al., 2015]:

Theorem 6.10. OSLFIA identifies the k-OSL functions in polynomial time and data. Its time complexity is in $\mathcal{O}(n + m(n + |S|))$ where $n = \sum_{(w,u)\in S} |w|$ and $m = max\{|u| : (w, u) \in S\}$. It is thus sub-quadratic.
The size of its exhibited characteristic sample is in $\mathcal{O}((m_\diamond + |Q_\diamond|)(|Q_\diamond| + |\Sigma||\delta_\diamond|)$ where $\mathcal{T}^\diamond = \langle Q_\diamond, q_{0_\diamond}, q_{f_\diamond}\Sigma, \Delta, \delta_\diamond, \rangle$ is the target transducer, and $m_\diamond = \sum_{(q,\sigma,u,q')\in\delta_\diamond} |u|$. It is thus quadratic in the size of $\mathcal{T}^\diamond$.

Once again we omit the proofs here since they can be found in the cited paper and since the ones for IOSL functions in Section 6.4 are generalized versions.

6.4 Input-Output Strictly Local functions

6.4.1 Definition

By mixing the ISL and OSL approaches, one can define an encompassing class of functions:

Definition 6.5 (Input-Output Strictly Local Function)**.** We say that a subsequential function f is (k,ℓ)-IOSL if for all w_1, w_2 in Σ^*, $\texttt{Suff}^{k-1}(w_1) = \texttt{Suff}^{k-1}(w_2)$ and $\texttt{Suff}^{\ell-1}(f^p(w_1)) = \texttt{Suff}^{\ell-1}(f^p(w_2)) \Rightarrow \texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$.

6.4.2 Automata Characterization

We define (k,ℓ)-IOSL transducers.

Definition 6.6 ((k,ℓ)-IOSL transducer)**.** We say that an onward DSFST $\mathcal{T} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta, \rangle$ is (k,ℓ)-IOSL if

1. $Q = S \cup \{q_0, q_f\}$ with $S \subseteq \Sigma^{\leq k-1} \times \Delta^{\leq \ell-1}$
2. $(\forall u \in \Delta^*)\big[(q_0, \rtimes, u, (q, q')) \in \delta \Longrightarrow q = \lambda \text{ and } q' = \texttt{Suff}^{\ell-1}(u)\big]$
3. $(\forall (q,q') \in Q \setminus \{q_0\}, \forall a \in \Sigma, \forall u \in \Delta^*)$
   $\big[((q,q'), a, u, (r,r')) \in \delta \Longrightarrow r = \texttt{Suff}^{k-1}(qa) \text{ and } r' = \texttt{suff}^{\ell-1}(q'u)\big]$.

We show next that (k,ℓ)-IOSL functions and (k,ℓ)-IOSL DSFST exactly correspond.

Lemma 6.11 (Extended transition function)**.** Let $\mathcal{T} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta\rangle$ be a (k,ℓ)-IOSL DSFST. We have

$$(q_0, \rtimes w, u, (q, q')) \in \delta^* \Longrightarrow q = \texttt{Suff}^{k-1}(w) \text{ and } q' = \texttt{Suff}^{\ell-1}(u)$$

Proof. The proof is done by recursion on the size of w The initial case is valid for $|w| = 0$ since if $(q_0, \rtimes, u, (q, q')) \in \delta^*$ then $(q_0, \rtimes, u, (q, q')) \in \delta$ and by Definition 6.6, $q = \lambda = \texttt{Suff}^{k-1}(w)$ and $q' = \texttt{Suff}^{\ell-1}(u)$. Suppose now that the lemma holds for inputs of size n. Let w be of size n such that $(q_0, w, u, (r, r')) \in \delta^*$ and suppose $((r,r'), a, v, (q,q')) \in \delta$ (i.e. $(q_0, wa, uv, (q,q')) \in \delta^*$). By recursion, we know that $r = \texttt{Suff}^{k-1}(w)$ and $r' = \texttt{Suff}^{\ell-1}(u)$. By definition of (k,ℓ)-IOSL SFST, $q = \texttt{Suff}^{k-1}(ra) = \texttt{Suff}^{k-1}(\texttt{Suff}^{k-1}(wa)) = \texttt{Suff}^{k-1}(wa)$ and $q' = \texttt{Suff}^{\ell-1}(rv) = \texttt{Suff}^{\ell-1}(\texttt{Suff}^{\ell-1}(u)v) = \texttt{Suff}^{\ell-1}(uv)$ (by Remark 6.1). □

Lemma 6.12. Any (k,ℓ)-IOSL DSFST corresponds to a (k,ℓ)-IOSL function.

Proof. Let $\mathcal{T} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta\rangle$ be a (k,ℓ)-IOSL DSFST for a function f and let w_1, w_2 in Σ^* such that $\texttt{Suff}^{k-1}(w_1) = \texttt{Suff}^{k-1}(w_2)$ and $\texttt{Suff}^{\ell-1}(f^p(w_1)) = \texttt{Suff}^{\ell-1}(f^p(w_2))$. By remark 6.7 there exists $(q,q'), (r,r') \in Q$ such that

$(q_0, \rtimes w_1, f^p(w_1), (q, q')) \in \delta^*$ and $(q_0, \rtimes w_2, f^p(w_2), (r, r')) \in \delta^*$. By lemma 6.11 $q = \texttt{Suff}^{k-1}(w_1) = \texttt{Suff}^{k-1}(w_2) = r$ and $q' = \texttt{Suff}^{\ell-1}(f^p(w_1)) = \texttt{Suff}^{\ell-1}(f^p(w_2)) = r'$. Thus $(q, q') = (r, r')$ which implies $\texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$. Therefore f is a (k, ℓ)-IOSL function. □

We now need to show that every (k, ℓ)-IOSL function can be represented by a (k, ℓ)-IOSL DSFST. An issue here is that one cannot work from the canonical transducer since its states are defined in terms of its tails, which themselves are defined only in terms of *input* strings, not output strings. Hence, the proof below is constructive.

Theorem 6.13. Let f be a (k, ℓ)-IOSL function. The DSFST $\mathcal{T}$ defined as followed computes f:

- $Q = S \cup \{q_0, q_f\}$ with $S \subseteq \Sigma^{\leq k-1} \times \Delta^{\leq \ell-1}$,
- $(q_0, \rtimes, u, (q, \texttt{Suff}^{\ell-1}(u))) \in \delta \iff q = \lambda$ and $u = f^p(\lambda)$,
- $((q, q'), a, u, (\texttt{Suff}^{k-1}(qa), \texttt{Suff}^{\ell-1}(q'u))) \in \delta \iff \exists w : \texttt{Suff}^{k-1}(w) = q \wedge \texttt{Suff}^{\ell-1}(f^p(w)) = q' \wedge f^p(wa) = vq'u$ with $v = f^p(w) \cdot {q'}^{-1}$,
- $((q, q'), \ltimes, u, q_f) \in \delta \iff u = f^p(w_{(q,q')})^{-1} \cdot f(w_{(q,q')})$ where $w_{(q,q')} = \min_{\lhd}\{w \mid \exists u, (q_0, \rtimes w, u, (q, q')) \in \delta^*\}$.

The diagram below helps express pictorially how the transitions are organized per the second bullet above. The input is written above the arrows, and the output written below.

$$q_0 \xrightarrow[f^p(w)=vq']{\rtimes w} (q, q') \xrightarrow[u]{a} (r, r') = (\texttt{Suff}^{k-1}(qa), \texttt{Suff}^{\ell-1}(q'u))$$

Note that $\mathcal{T}$ is a (k, ℓ)-IOSL transducer by construction.

To prove this result, we are going to show first the following lemma:

Lemma 6.14. Let $\mathcal{T}$ be the transducer defined in Theorem 6.13. We have:

$$\exists (r, r') \in Q, (q_0, \rtimes w, u, (r, r')) \in \delta^* \iff f^p(w) = u$$

Proof. ($\Rightarrow$) Suppose $(q_0, \rtimes w, u, (r, r')) \in \delta^*$. The proof is by recursion on the length of w.

If $|w| = 0$, we have $(q_0, \rtimes, u, (r, r')) \in \delta$ which implies $r = \lambda$, $r' = \texttt{Suff}^{\ell-1}(u)$ and $u = f^p(\lambda)$ by the second item of the construction of the DSFST.

Suppose the result holds for w of size n and pick such a w. Suppose then that $(q_0, \rtimes wa, u, (r, r')) \in \delta^*$. By the definition of the extended transition function, there exists $u_1, u_2, (q, q')$ such that $u = u_1u_2$, $(q_0, \rtimes w, u_1, (q, q')) \in \delta^*$ and $((q, q'), a, u_2, (r, r')) \in \delta$. We have $f^p(w) = u_1$ (by recursion) and thus $q' = \texttt{Suff}^{\ell-1}(f^p(w))$ (by Lemma 6.11).

By construction of the transducer, $r' = \texttt{Suff}^{\ell-1}(q'u_2)$ and thus $f^p(wa) = vq'u_2$ with $v = f^p(w) \cdot \texttt{Suff}^{\ell-1}(f^p(w))^{-1}$. Therefore $f^p(wa) = vq'u_2 = f^p(w) \cdot$

$\texttt{Suff}^{\ell-1}(f^p(w))^{-1}q'u_2 = f^p(w) \cdot \texttt{Suff}^{\ell-1}(f^p(w))^{-1}\texttt{Suff}^{\ell-1}(f^p(w))u_2 = f^p(w)u_2 = u_1u_2 = u$.
($\Leftarrow$) The proof is again by recursion on the length of w.

Let $|w| = 0$, and let $u = f^p(\lambda)$. By construction of $\mathcal{T}$, $(q_0, \rtimes, u, (\lambda, \texttt{Suff}^{\ell-1}(u))) \in \delta$, which validates the base case.

Suppose the result holds for w of size n and pick such a w. Suppose now that $f^p(wa) = u$. As f is subsequential, there exists u_1 such that $f^p(w) = u_1$. By recursion, there exists (q, q') such that $(q_0, \rtimes w, u_1, (q, q')) \in \delta^*$. By Lemma 6.11, $q' = \texttt{Suff}^{\ell-1}(u_1) = \texttt{Suff}^{\ell-1}(f^p(w))$. By definition $f^p(wa) = u_1u_1^{-1} \cdot u$ and then $f^p(wa) = u_1 \cdot \texttt{Suff}^{\ell-1}(u_1)^{-1}\texttt{Suff}^{\ell-1}(u_1)u_1^{-1} \cdot u$. We thus have $f^p(wa) = vq'u'$, with $v = u_1 \cdot \texttt{Suff}^{\ell-1}(u_1)^{-1} = f^p(w) \cdot \texttt{Suff}^{\ell-1}(f^p(w))^{-1}$ and $u' = u_1^{-1} \cdot u$. By construction we then have $((q, q'), a, u', (\texttt{Suff}^{k-1}(qa), \texttt{Suff}^{\ell-1}(q'u'))) \in \delta$. As $u_1u' = u_1u_1^{-1} \cdot u = u$, we have $(q_0, \rtimes wa, u, (\texttt{Suff}^{k-1}(qa), \texttt{Suff}^{\ell-1}(q'u'))) \in \delta^*$. □

We now provide the proof of Theorem 6.13.

Proof. Let $\mathcal{T}$ be the transducer defined in Theorem 6.13 and let $t_\mathcal{T}$ be the function it computes. We show that $\forall w \in \Sigma^*$,

$$t_\mathcal{T}(w) = u \iff f(w) = u$$

By Lemma 6.14 we know that $(q_0, \rtimes w, u, (r, r')) \in \delta^* \iff f^p(w) = u$. By construction of the DSFST, we have $((r, r'), \ltimes, f^p(w_{(r,r')})^{-1} \cdot f(w_{(r,r')}), q_f)$ where $w_{(r,r')} = \min_\lhd\{w \mid \exists u, (q_0, \rtimes w, u, (r, r')) \in \delta^*\}$. Therefore $t_\mathcal{T}(w_{(r,r')}) = uf^p(w_{(r,r')})^{-1} \cdot f(w_{(r,r')}) = f^p(w_{(r,r')})f^p(w_{(r,r')})^{-1} \cdot f(w_{(r,r')}) = f(w_{(r,r')})$.

We have $\texttt{Suff}^{\ell-1}(u) = \texttt{Suff}^{\ell-1}(f^p(w_{(r,r')})) = \texttt{Suff}^{\ell-1}(f^p(w))$ and $\texttt{Suff}^{k-1}(w) = r = \texttt{Suff}^{k-1}(w_{(r,r')})$ (By Lemma 6.11). As f is a (k, ℓ)-IOSL function, we thus have $\texttt{tails}_f(w_{(r,r')}) = \texttt{tails}_f(w)$. This implies in particular that $(\lambda, f^p(w_{(r,r')})^{-1} \cdot f(w_{(r,r')})) \in \texttt{tails}_f(w)$. Thus $f(w) = f^p(w)f^p(w_{(r,r')})^{-1} \cdot f(w_{(r,r')}))$. By construction, we have $t_\mathcal{T}(w) = uf^p(w_{(r,r')})^{-1} \cdot f(w_{(r,r')}) = f^p(w)f^p(w_{(r,r')})^{-1} \cdot f(w_{(r,r')}) = f(w)$. □

6.4.3 Relations among classes

Trivially any $(k, 1)$-IOSL functions is a k-ISL function (and vice versa) and any $(1, \ell)$-IOSL function is a ℓ-OSL function (and reciprocally).

Lemma 6.15. The classes of ISL functions and of OSL functions are strictly included in the class of IOSL functions.

Proof. We need to exhibit a (k, ℓ)-IOSL function that is not k'-ISL [resp. ℓ'-OSL] for any k' [resp. any ℓ'].

Given k and ℓ, we pick the function that rewrites waw' to wbw' if $w' = a^{k-1}$ and $\texttt{Suff}^{\ell-1}(f^p(w)) = b^{\ell-1}$, and to waw' otherwise (the function does not change the b's).

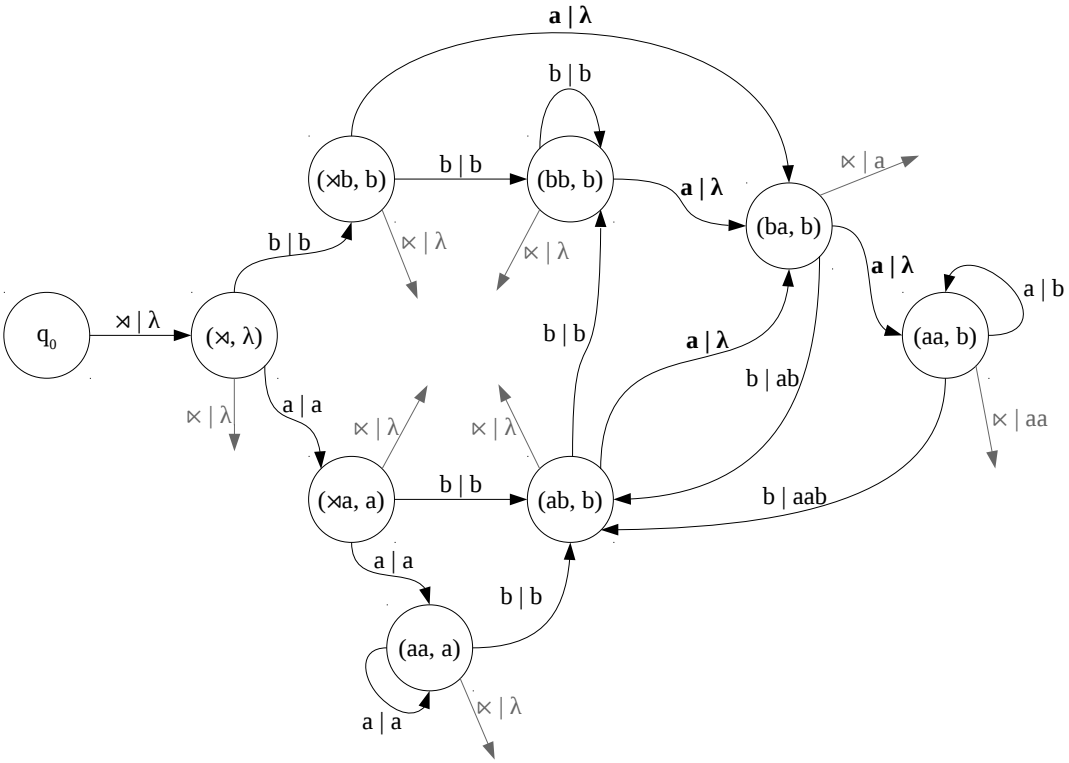

Figure 6.2: A graphical representation of an IOSL transducer (the final state is not drawn for readability reasons) It rewrites an a into b if it is followed by 2 other a's and if the suffix of the current output is made of one b: it is thus a (2,1)-IOSL transducer. Proof of Lemma 6.15 shows that the corresponding function is neither ISL nor OSL.

This function is obviously (k, ℓ)-IOSL. An example of the DSFT for the corresponding $(3, 2)$-IOSL function is given in Figure 6.2.

We now prove that given k and ℓ there does not exists any k' such that the function is k'-ISL. Indeed, for any k', we have $\texttt{Suff}^{k'-1}(b^\ell a^{k'-1}) = \texttt{Suff}^{k'-1}(a^{k'-1})$ but $(a^k, b^{k'}a^{k-1}) \in tail(b^\ell a^{k'-1})$ while $(a^k, a^k) \in tail(a^{k'-1})$, and thus $\texttt{tails}(b^\ell a^{k'-1}) \neq \texttt{tails}(a^{k'-1})$.

Finally, we prove that given k and ℓ there does not exists any ℓ' such that the function is ℓ'-OSL. For all $\ell' > 0$, we have $\texttt{Suff}^{\ell'-1}(f^p(b^{\ell-1})) = \texttt{Suff}^{\ell'-1}(b^{\ell-1}) = \texttt{Suff}^{\ell'-1}(f^p(b^{\ell-1}a^{k-1}))$ but $(a, a) \in \texttt{tails}(b^{\ell-1})$ while $(a, ba^{k-1}) \in \texttt{tails}(b^{\ell-1}a^{k-1})$ and therefore $\texttt{tails}(b^{\ell-1}) \neq \texttt{tails}(b^{\ell-1}a^{k-1})$ □

6.5 Learning IOSL functions

6.5.1 The Learning Algorithm

In Algorithm 2 we present an algorithm which we show learns the IOSL functions under the criterion introduced in Definition 5.9. We call this the Input-Output Strictly Local Function Inference Algorithm (IOSLFIA). We assume Σ, Δ, k, and l are fixed and not part of the input to the learning problem.

We define the observable prefix function on a finite set S as followed $f^p_S(w) = \texttt{lcp}(\{u \in \Delta^* \mid x \in \Sigma^* : (wx, u) \in S\})$.

Essentially, the algorithm computes a breadth-first search through the states that are reachable given the learning sample: the set C contains the states already checked while R is a queue made of the states that are reachable but have not been treated yet. `smallest` stores the input string that allow the discovery of the corresponding state. At each step of the main loop, IOSLFIA treats the first state (q, q') that has been put in the queue but has not been treated and computes whenever possible the transitions that leave that state. The output associated with each added transition is the longest common prefixes of the outputs associated with the smallest input prefix in the sample that allows the state to be reachable. This ensures that at every point in the construction of the transducer, it is onward. Consequently, the output of the algorithm is a (k, ℓ)-IOSL transducer.

6.5.2 Theoretical Results

Here we establish theoretical results, which culminate in the theorem that IOSLFIA identifies the (k, ℓ)-IOSL functions in polynomial time and data (Definition 5.9).

We first prove the following lemma:

Lemma 6.16. For any input sample S, IOSLFIA produces its output in time polynomial in the size of S.

Algorithm 2 IOSLFIA

Data: A sample $S \subset \Sigma^* \times \Delta^*$ and $k, \ell \in \mathbb{N}$

$D \leftarrow \texttt{pref}(\{w \mid \exists u, (w,u) \in S\})$;
Compute $f_S^p(\cdot)$;
$(q, q') \leftarrow (\lambda, \texttt{Suff}^{\ell-1}(f_S^p(\lambda)))$; $\texttt{smallest}(q, q') \leftarrow \lambda$;
$\delta \leftarrow \{(q_0, \rtimes, f_S^p(\lambda), (q, q'))\}$;
$R \leftarrow \{(q, q')\}$; $C \leftarrow \emptyset$;
while $R \neq \emptyset$ **do**
  $(q, q') \leftarrow First(R)$;
  $s \leftarrow \texttt{smallest}(q, q')$;
  if $\exists u$ *s.t.* $(s, u) \in S$ **then**
    $\delta \leftarrow \delta \cup \{((q, q'), \ltimes, f_S^p(s)^{-1} \cdot u, q_f)\}$;
  for *all* a *in* Σ **do**
    if $sa \in D$ **then**
      $(r, r') \leftarrow (\texttt{Suff}^{k-1}(sa), \texttt{Suff}^{\ell-1}(f_S^p(sa)))$;
      $u \leftarrow f_S^p(s)^{-1} \cdot f_S^p(sa)$;
      $\delta \leftarrow \delta \cup ((q, q'), a, u, (r, r')$;
      if $(r, r') \notin R \cup C$ **then**
        $R \leftarrow R \cup \{(r, r')\}$;
        $\texttt{smallest}(r, r') \leftarrow sa$;
  $R \leftarrow R \setminus \{(q, q')\}$; $C \leftarrow C \cup \{(q, q')\}$;
return $\langle C \cup \{q_0, q_f\}, q_0, q_f, \Sigma, \Delta, \delta \rangle$;

Proof. The computation of $f^p_S(\cdot)$ can be easily done using a prefix tree transducer (PTT): this is a prefix tree acceptor on the input, that also contains the output information. The cost to build such a transducer and to make in onward is in $\mathcal{O}(mn\log(n))$ [Oncina et al., 1993] where $n = \sum_{(w,u)\in S} |w|$ and $m = max\{|u| : (w,u) \in S\}$. Once the PTT is built, computing $f^p_S(w)$ for any w can be done in $\mathcal{O}(|w|)$. Using a hash-table for storing $f^p_S(\cdot)$ costs thus $\mathcal{O}(n)$ to be constructed and then $f^p_S(\cdot)$ is accessible in $\mathcal{O}(1)$.

The main loop is used at most $|\Sigma|^{k-1}|\Delta|^{\ell-1}$ which is constant since Σ, Δ, k and l are fixed for any learning sample. The first conditional can be tested in $\mathcal{O}(n)$ using the PTT and the computation of the eventual transition then takes $\mathcal{O}(m)$. The smaller loop is executed $|\Sigma|+1$ times. At each execution: the first conditional can be tested in time linear in n; the computation of (r,r') and of u can be done in $O(m)$; computing the suffixes requires at most $|sa|+m$ steps; The second conditional can be tested and executed in $O(1)$; all the other instructions can be done in constant time. The overall computation time is thus in $\mathcal{O}(mn\log(n)+n+|\Delta|^{k-1}(|\Sigma|+1)(n+m+n+m+n+m)) = \mathcal{O}(mn\log(n)))$ which is polynomial in the size of the learning sample. □

Next we show that for any (k,ℓ)-IOSL function f, there is a finite kernel of data consistent with f (a 'seed') that is a characteristic sample for IOSLFIA.

Definition 6.7 (A IOSLFIA seed)**.** Given a (k,ℓ)-IOSL transducer $\langle Q, q_0, q_f, \Sigma, \Delta, \delta\rangle$ computing a (k,ℓ)-IOSL function f, a sample S is a IOSLFIA *seed* for f if

- For all $(q,q') \in Q$ such that $\exists v \in \Delta^*$ $((q,q'), \ltimes, v, q_f) \in \delta$, $(w_{(q,q')}, f(w_{(q,q')})) \in S$, where $w_{(q,q')} = \min_{\lhd}\{w \mid \exists u, (q_0, \rtimes w, u, (q,q')) \in \delta^*\}$
- For all $((q,q'),a,u,(r,r')) \in \delta$ and $a \in \Sigma$, for all $b \in \Sigma$ such that there exists $((r,r'),b,u',(s,s')) \in \delta$, there exists $(w, f(w)) \in S$ and $x \in \Sigma^*$ such that $w = w_{(q,q')}abx$ and $f(w)$ is defined. Also, if there exists v such that $((r,r'), \ltimes, v, q_f) \in \delta$ then $(w_{(q,q')}a, f(w_{(q,q')}a)) \in S$.
- For $(q_0, \rtimes, u, (r,r')) \in \delta$, For all $b \in \Sigma$ such that there exists $((r,r'),b,u',(s,s')) \in \delta$, there exists $(w, f(w)) \in S$ and $x \in \Sigma^*$ such that $w = abx$ and $f(w)$ is defined. Also, if there exists v such that $((r,r'), \ltimes, v, q_f) \in \delta$ then $(a, f(a)) \in S$.

The first item ensures that the prefix that allows the discovery of a state is the smallest in the target reaching the corresponding state. The second item is a bit more technical but it aims at ensuring that while considering a transition $((q,q'),a,u,(r,r'))$, the algorithm will have enough information to correctly compute u. This requires a witness for all transitions leaving (r,r'), which allows the use of the onwardness of the target to ensure the correctness of $f^p_S(\cdot)$ for the smallest prefix reaching a state.

In what follows, we set $\mathcal{T}^\diamond = \langle Q_\diamond, q_{0_\diamond}, q_{f_\diamond}, \Sigma, \Delta, \delta_\diamond\rangle$ be the target (k,ℓ)-IOSL transducer, f the function it computes, and $\mathcal{T} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta\rangle$ be the outputted transducer on a sample.

Lemma 6.17. If a seed is contained in a sample S, then for any prefix w such that $\exists (q,q') \in Q_\diamond$, $w = \min_\lhd\{w' \mid \exists u, (q_0, \rtimes w', u, (q,q')) \in \delta_\diamond^*\} : f_S^p(w) = f^p(w)$.

Proof. Let $(q,q') \in Q_\diamond$ and $w = \min_\lhd\{w' \mid \exists u, (q_0, \rtimes w', u, (q,q')) \in \delta_\diamond^*\}$. We have $f_S^p(w) = \texttt{lcp}(\{u \mid \exists x, (wx,u) \in S\}) = \texttt{lcp}(\{u \mid \exists x, f(wx) = u \wedge (wx,u) \in S\})$. The second item of Definition 6.7 ensures that every transition leaving (q,q') to a non-final state is used by at least one element of S (whose input part is prefixed by w), while the first item ensures that if a transition leaves for q_f then $(w, f(w)) \in S$. As the target in onward, by Lemma 6.2, $\texttt{lcp}(\texttt{outputs}(q,q')) = \lambda$. Therefore $f_S^p(w) = \texttt{lcp}(f(w\Sigma^*)) = \texttt{lcp}(\{u \mid \exists x, f(wx) = u\}) = f^p(w)$. □

Lemma 6.18. If a seed is contained in a sample S, then for any prefix w such that $\exists (q,q') \in Q_\diamond$, $w = \min_\lhd\{w' \mid \exists u, (q_0, \rtimes w', u, (q,q')) \in \delta_\diamond^*\}$, for any $a \in \Sigma$ such that it exists $((q,q'), a, u, (r,r')) \in \delta_\diamond$: $f_S^p(wa) = f^p(wa)$.

Proof. Let $(q,q') \in Q_\diamond$, $w = \min_\lhd\{w' \mid \exists u, (q_0, \rtimes w', u, (q,q')) \in \delta_\diamond^*\}$, and $a \in \Sigma$ such that $\exists ((q,q'), a, u, (r,r')) \in \delta_\diamond$. We have $f_S^p(wa) = \texttt{lcp}(\{u \mid \exists x, (wax,u) \in S\})$. The second item of Definition 6.7 ensures that there is at least one element of the seed for each transition leaving (r,r') whose input part is prefixed by wa. Therefore $f_S^p(wa) = \texttt{lcp}(\{u \mid \exists b \in \Sigma, x : (wabx, f(wabx)\} \cup \{(wa, f(wa)) \mid wa \in \texttt{dom}(f)\})$.

We have $f^p(wa) = \texttt{lcp}(f(wa\Sigma^*)) = \texttt{lcp}(\{u \mid \exists b \in \Sigma, x \in \Sigma^*, (wabx, f(wabx))\} \cup \{(wa, f(wa)) \mid wa \in \texttt{dom}(f)\})$. As the $\texttt{lcp}$ of the outputs of (r,r') is λ (by onwardness) it implies $f^p(wa) = f_S^p(wa)$. □

Lemma 6.19. If a seed is contained in a learning sample S, $(q_0, \rtimes w, u, (r,r')) \in \delta^* \iff (q_{0_\diamond}, \rtimes w, u, (r,r')) \in \delta_\diamond^*$

Proof. ($\Rightarrow$). The proof is by induction on the length of w. If $|w| = 0$ then $(q_0, \rtimes, u, (r,r')) \in \delta$, $r = \lambda$, $r' = \texttt{Suff}^{\ell-1}(f_S^p(\lambda))$, and $u = f_S^p(\lambda)$ (initial steps of the algorithm). By lemma 6.17, we have $f_S^p(\lambda) = f^p(\lambda)$ as λ is the smallest input reaching this state. By the definition of a (k,ℓ)-IOSL transducer, we have $(q_{0_\diamond}, \rtimes, f^p(\lambda), (\lambda, \texttt{Suff}^{\ell-1}(f^p(\lambda)))) \in \delta_\diamond$.

Suppose the lemma is true for strings of length less than or equal to n. We refer to this as the first Inductive Hypothesis (IH1). Let wa be of size $n+1$ such that $(q_0, \rtimes wa, u, (r,r')) \in \delta^*$. By definition of δ^*, it exists $u_1, u_2, (q,q')$ such that $(q_0, \rtimes w, u_1, (q,q')) \in \delta^*$, $((q,q'), a, u_2, (r,r')) \in \delta$, and $u = u_1u_2$. By IH1 $(q_{0_\diamond}, \rtimes w, u_1, (q,q')) \in \delta_\diamond^*$. We want to show $(q_{0_\diamond}, wa, u, (r,r')) \in \delta_\diamond^*$ i.e. $((q,q'), a, u_2, (r,r')) \in \delta_\diamond$.

First we show that the IH1 also implies that $s = \texttt{smallest}((q,q'))$ is such that $s = w_{(q,q')}$. Since the algorithm works in a breadth-first search, $\rtimes s$ is the smallest input that reaches (q,q') in the constructed DSFST. If $w_{(q,q')} \lhd s$ then $\exists (x,x') \neq (q,q')$ such that $(q_0, \rtimes w_{(q,q')}, u', (x,x')) \in \delta^*$ because $w_{(q,q')}$ is a prefix of an input string

of the sample S (since S contains a seed). Since $w_{(q,q')} \lhd s$ and $|s| \leq n$, by IH1 then $(q_{0_\diamond}, \rtimes w_{(q,q')}, u', (x, x')) \in \delta_\diamond^*$ which implies $(q, q') = (x, x')$ which contradicts the supposition that $w_{(q,q')} \lhd s$. If $s \lhd w_{(q,q')}$, then again as $(q_0, \rtimes s, u', (q, q')) \in \delta^*$ it implies $(q_{0_\diamond}, \rtimes s, u', (q, q')) \in \delta_\diamond^*$ by IH1. This contradicts the definition of $w_{(q,q')}$. Therefore $s = w_{(q,q')}$.

We now show the result. Recall that we have $((q, q'), a, u_2, (r, r')) \in \delta$. By construction of the algorithm, we have $u_2 = f_S^p(\texttt{smallest}(q, q'))^{-1} \cdot f_S^p(\texttt{smallest}(q, q')a) = f_S^p(w_{(q,q')})^{-1} \cdot f_S^p(w_{(q,q')}a)$ which is equal to $f^p(w_{(q,q')})^{-1} \cdot f^p(w_{(q,q')}a)$ by Lemmas 6.17 and 6.18. By construction of the algorithm, we also have $r = \texttt{Suff}^{k-1}(w_{(q,q')})$ and $r' = \texttt{Suff}^{\ell-1}(f^p(w_{(q,q')})$. As $(q_{0_\diamond}, \rtimes w, u_1, (q, q')) \in \delta_\diamond^*$ (by IH1) and as $wa \in \texttt{pref}(\texttt{dom}(f))$ there exists $((q, q'), a, v, (x, x'))$ in $\delta_\diamond$. As $\mathcal{T}_\diamond$ is onward we have $v = f^p(w)^{-1} \cdot f^p(wa) = u_2$ and as it is a (k, ℓ)-IOSL transducer $x = \texttt{Suff}^{k-1}(wa) = r$ and $x' = \texttt{Suff}^{\ell-1}(f^p(wa)) = r'$. Therefore $(q_{0_\diamond}, \rtimes wa, u, (r, r')) \in \delta_\diamond^*$.

($\Leftarrow$). This is also by induction on the length of w. If $|w| = 0$ as $\mathcal{T}^\diamond$ is onward we have $\texttt{lcp}(\texttt{outputs}(q_{0_\diamond})) = \texttt{lcp}(f(\Sigma^*)) = f^p(\lambda)$ (Lemma 6.2) and thus $(q_{0_\diamond}, \rtimes, f^p(\lambda), (r, r')) \in \delta_\diamond$ with $r = \texttt{Suff}^{k-1}(\lambda) = \lambda$ and $r' = \texttt{Suff}^{\ell-1}(f^p(\lambda))$ as $\mathcal{T}^\diamond$ is (k, ℓ)-IOSL. As w is the smallest input string reaching (r, r'), and as a seed in contained in the sample, by Lemma 6.17 $f_S^p(w) = f^p(w)$. Therefore, thanks to the initials steps of the algorithm $(q_0, \rtimes, f^p(\lambda), (r, r')) \in \delta$.

Suppose the lemma is true for all strings up to length n. We refer to this as the second Inductive Hypothesis (IH2). Pick wa of length $n+1$ such that $(q_{0_\diamond}, \rtimes wa, u, (r, r')) \in \delta_\diamond^*$. By definition of the extended transition function, $(q, q'), u_1, u_2$ exist such that $(q_{0_\diamond}, \rtimes w, u_1, (q, q')) \in \delta_\diamond^*$ and $((q, q'), a, u_2, (r, r')) \in \delta_\diamond$, with $u_1 u_2 = u$. By IH2, we have $(q_0, \rtimes w, u_1, (q, q')), (q_0, \rtimes w_{(q,q')}, u_1', (q, q')) \in \delta^*$ (since $w_{(q,q')} \lhd w$).

We want to show $((q, q'), a, u_2, (r, r')) \in \delta$.

We first show that $s = \texttt{smallest}((q, q')) = w_{(q,q')}$. Suppose $s \lhd w_{(q,q')}$. By construction of the SFST s is a prefix of an element of S which means there exists (x, x') such that $(q_{0_\diamond}, \rtimes s, f^p(s), (x, x')) \in \delta_\diamond^*$. But by IH2, this implies that $(x, x') = (q, q')$ and the definition of $w_{(q,q')}$ contradicts $s \lhd w_{(q,q')}$. Suppose now that $w_{(q,q')} \lhd s$. By the construction of the seed, $w_{(q,q')}$ is a prefix of an element of the sample, which implies it is considered by the algorithm. As $(q_0, \rtimes w_{(q,q')}, u_1', (q, q')) \in \delta^*$ by IH2, $w_{(q,q')}$ is a smaller prefix than s that reaches the same state which is impossible as s is the earliest prefix that makes the state (q, q') reachable. Therefore $w_{(q,q')} = s$ and thus the transition from state (q, q') reading a is created when $s = w_{(q,q')}$.

Now let $v = f_S^p(sa) = f_S^p(w_{(q,q')}a)$. As the sample contains a seed, we have $v = f^p(sa)$ by Lemma 6.18 and thus $(q_{0_\diamond}, \rtimes w_{(q,q')}a, v, (r, r')) \in \delta_\diamond^*$ (Lemma 6.14).

As $f_S^p(s) = f^p(s)$ by Lemma 6.17, $f_S^p(sa) = f^p(sa)$ by Lemma 6.18, and as $s = w_{(q,q')}$, we have $u_2 = f^p(w_{(q,q')})^{-1} \cdot f^p(w_{(q,q')}a) = f_S^p(w_{(q,q')})^{-1} \cdot f_S^p(w_{(q,q')}a)$.

As the target is a (k, ℓ)-IOSL transducer (and thus deterministic) $\texttt{Suff}^{\ell-1}(q'u_2) = r'$. Therefore the transition $((q, q'), a, f_S^p(s)^{-1} \cdot f_S^p(sa), (r, r'))$ that is added to δ is the

same as the transition $((q, q'), a, u_2, (r, r'))$ in $\delta_\diamond$. This implies $(q_0, \rtimes wa, u, (r, r')) \in \delta^*$ and proves the lemma. □

Lemma 6.20. Any seed for IOSLFIA is a characteristic sample for this algorithm.

Proof. A corollary of Lemma 6.19 is that if a seed is contained into a learning sample we have $(q_0, \rtimes w, u, (q, q')) \in \delta^* \iff f^p(w) = u$ (Lemma 6.11) as the target transducer is (k, ℓ)-IOSL. For all states (q, q') of the target where $\exists v$, $((q, q'), \ltimes, v, q_{f_\diamond}) \in \delta_\diamond$, we have $(w_{(q,q')}, f(w_{(q,q')}))$ in the seed, which implies the algorithm will add $((q, q'), \ltimes, f^p_S(w_{(q,q')})^{-1} \cdot f(w_{(q,q')}), q_f)$ to δ. As $f^p_S(w_{(q,q')}) = f^p(w_{(q,q')})$ (Lemma 6.17) the added transition is exactly the output transition of the target. As every state is treated only once, this holds for any learning set containing a seed. Therefore, from any superset of a seed, for any w, the function computed by the outputted transducer of Algorithm 2 is equal to $f^p(w)f^p(w)^{-1} \cdot f(w) = f(w)$ □

Lemma 6.21. Given any (k, ℓ)-IOSL transducer $\mathcal{T}^\diamond$, there exists a seed for the IOSL learner that is of size polynomial in the size of $\mathcal{T}^*$.

Proof. Let $\mathcal{T}^\diamond = \langle Q_\diamond, q_{0_\diamond}, q_{f_\diamond}, \Sigma, \Delta, \delta_\diamond \rangle$ be the target transducer. There are at most $|Q_\diamond|$ pairs $(w_{(q,q')}, f(w_{(q,q')}))$ in a seed that corresponds to the first item of Definition 6.7, each of which being such that $|w_{(q,q')}| \leq |Q_\diamond|$ and $|f(w_{(q,q')})| \leq max\{|u| \mid \exists (q, q') \in Q_\diamond ((q, q'), \ltimes, u, q_f)\} + \sum_{((q,q'),a,u,(r,r')) \in \delta_\diamond} |u|$. We denote by $m_\diamond$ this last quantity and note that $m_\diamond = \mathcal{O}(|\mathcal{T}^\diamond|)$.

For the elements of the second item of Definition 6.7 we restrict ourselves without loss of generality to pairs $(w_{(q,q')}abw', f(w_{(q,q')}abw'))$ where $w' = min_\lhd\{y : f(w_{(q,q')}aby)$ *is defined*$\}$. We have $|w'| \leq |Q_\diamond|$ and $|f(w_{(q,q')}abw')|$ is in $\mathcal{O}(m_\diamond)$. There are at most $|\Sigma| + 1$ pairs $(w_{(q,q')}abw', f(w_{(q,q')}abw'))$ for a given transition $((q, q'), a, u, (r, r'))$ which implies that the overall bound on the number of such pairs is in $\mathcal{O}(|\Sigma||\delta|)$. Adding the pairs $(w_{(q,q')}a, f(w_{(q,q')}a))$, the overall length of the elements in the seed that fulfill the second item of the definition is in $\mathcal{O}((|Q_\diamond| + m_\diamond)(|\delta_\diamond|(|\Sigma| + 1) + |Q_\diamond|$.

The size of the seed studied in this proof is thus in $\mathcal{O}((m_\diamond + |Q_\diamond|)(|Q_\diamond| + |\Sigma||\delta|)$ which is polynomial in the size of the target transducer. □

Theorem 6.22. IOSLFIA identifies the (k, ℓ)-IOSL functions in polynomial time and data.

Proof. Follows immediately from the previous lemmas in this section. □

6.6 Conclusion and future works

The algorithm presented here is the simplest expression of the learning idea behind this work. If one is not interested by theoretical results and simple proofs, it can easily

be modified to obtain a more robust version. For instance, $w_{(q,q')}$ does not have to be the smallest input string to reach the state (q, q'): it can be for instance the more informed input string, that is, the one reaching this state for which the most data is available. The important point is that it is unique for each state. This variant is likely to work better in practice but the size of the characteristic sample is not ensured to be polynomial: we presented IOSLFIA in this way for sake of simplicity and for complexity reasons.

We are currently working on continuations of this work following different directions: closure properties of the classes, a decision procedure for deciding whether any regular relation is (k, ℓ)-IOSL or not, and if so returns the smallest (k, ℓ) transducer, etc. On a longer term, we would like to consider the learning of probabilistic [Akram and de la Higuera, 2013] semi-deterministic transducers [Beros and de la Higuera, 2016].

Bibliography

H. Ibne Akram and C. de la Higuera. Learning probabilistic subsequential transducers from positive data. In *Proc. of the International Conference on Agents and Artificial Intelligence*, pages 479–486, 2013. 158

A. A. Beros and C. de la Higuera. A canonical semi-deterministic transducer. *Fundamenta Informaticae*, 146(4):431–459, 2016. 158

J. Chandlee. *Strictly Local Phonological Processes*. PhD thesis, The University of Delaware, 2014. 144

J. Chandlee and J. Heinz. Bounded copying is subsequential: Implications for metathesis and reduplication. In *Proc. of the Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 42–51. Association for Computational Linguistics, 2012. 142

J. Chandlee, A. Athanasopoulou, and J. Heinz. Evidence for classifying metathesis patterns as subsequential. In *Proc. of the West Coast Conference on Formal Linguistics*, pages 303–309. Cascadilla Press, 2012. 142

J. Chandlee, R. Eyraud, and J. Heinz. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics*, 2:491–503, 2014. 141, 144, 145, 146

J. Chandlee, R. Eyraud, and J. Heinz. Output strictly local functions. In *Proc. of the Meeting on Mathematics of Language*, pages 112–125, 2015. 142, 144, 147

N. Chomsky and M. Halle. *The Sound Pattern of English*. New York: Harper & Row, 1968. 142

R. Frank and G. Satta. Optimality Theory and the generative complexity of constraint violability. *Computational Linguistics*, 24(2):307–315, 1998. 142

B. Gainor, R. Lai, and J. Heinz. Computational characterizations of vowel harmony patterns and pathologies. In *Proc. of the West Coast Conference on Formal Linguistics*, pages 63–71. Cascadilla Press, 2012. 142

P. García and J. Ruiz. Learning k-testable and k-piecewise testable languages from positive data. *Grammars*, 7:125–140, 2004. 143

P. Garcia, E. Vidal, and J. Oncina. Learning locally testable languages in the strict sense. In *Proc. of the Workshop on Algorithmic Learning Theory*, pages 325–338, 1990. 143

T. Graf. Logics of phonological reasoning. Master's thesis, University of California, Los Angeles, 2010. 142

J. Heinz. *The Inductive Learning of Phonotactic Patterns*. PhD thesis, University of California, Los Angeles, 2007. 142, 143

J. Heinz. On the role of locality in learning stress patterns. *Phonology*, 26(2):303–351, 2009. 142

J. Heinz and R. Lai. Vowel harmony and subsequentiality. In *Proc. of the Meeting on the Mathematics of Language*, pages 52–63, 2013. 142

J. Heinz and J. Rogers. Learning subregular classes of languages with factored deterministic automata. In *Proc. of the Meeting on the Mathematics of Language*, pages 64–71. Association for Computational Linguistics, 2013. 143

Jeffrey Heinz. Learning long-distance phonotactics. *Linguistic Inquiry*, 41(4):623–661, 2010. 142, 143

Jeffrey Heinz, Anna Kasprzik, and Timo Kötzing. Learning with lattice-structured hypothesis spaces. *Theoretical Computer Science*, 457:111–127, October 2012. 143

A. Jardine. Computationally, tone is different. *Phonology*, 33(2):247–283, 2016. 142

A. Jardine, J. Chandlee, R. Eyraud, and J. Heinz. Very efficient learning of structured classes of subsequential functions from positive data. In *Proc. of the International Conference on Grammatical Inference*, volume 34, pages 94–108. PMLR, 2014. 143

C. D. Johnson. *Formal Aspects of Phonological Description*. The Hague: Mouton, 1972. 142

R. Kaplan and M. Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, 1994. 142

M. Kenstowicz and C. Kisseberth. *Generative Phonology*. Academic Press, Inc., 1979. 142

Huan Luo. Long-distance consonant harmony and subsequantiality, 2014. Qualifying paper for the University of Delaware's Linguistics PhD Progam. 142

R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, 1971. 142, 143

D. Odden. *Introducing Phonology*. Cambridge University Press, 2nd edition, 2014. 142

J. Oncina, P. García, and E. Vidal. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458, 1993. 143, 154

Amanda Payne. All dissimilation is computationally subsequential. 93:e353–e371, 12 2017. 142

J. Rogers and G. Pullum. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20:329–342, 2011. 142, 143, 144

J. Rogers, J. Heinz, G. Bailey, M. Edlefsen, M. Visscher, D. Wellcome, and S. Wibel. On languages piecewise testable in the strict sense. In *Proc . of the Meeting on Mathematics of Language*, volume 6149 of *LNAI*, pages 255–265, 2010. 142

J. Rogers, J. Heinz, M. Fero, J. Hurst, D. Lambert, and S. Wibel. Cognitive and sub-regular complexity. In *Proc. of Formal Grammar*, volume 8036 of *LNCS*, pages 90–108, 2013. 142, 143

W. Thomas. Languages, automata, and logic. volume 3, chapter 7. Springer, 1997. 142

Part IV

Conclusion

Chapter 7

Conclusion

Contents

The work presented in this document compiles the main positive results I obtained during the past ten years. This research covers a wide part of the field called Grammatical Inference. I detail in this conclusion two main points, on two complementary subjects, that can be made to provide a global overview of these studies. Then, this conclusion briefly describes other works that were conducted during the past years, and finishes with some general personal remarks.

7.1 On congruences

Apart from the introducing part, that interrogates from a formal standpoint the notion of learnability for computational models (Chapter 2), the works of this manuscript share common characteristics. They all propose learning algorithms for specifically shaped computational models, study the abilities and limits of these algorithms, and provide characterizations for the learned classes.

These works actually share more than a common method and structure: a global learning task resolution can be drawn from these results. Indeed, they all rely on the observation of congruence classes in a set of data, and the outputted computational model is generated by examining how these classes are combined in the data.

In details, the distributional learning algorithms (Part II) focus on the congruence defined by contexts. For strings, this notion is well-understood which allows the studies on complex relations between classes, in the work presented here (Chapter 3) and its continuations. For graphs (Chapter 4), it was not studied previously and we had to carefully defined the needed notions and to limit our research to simple classes combination.

In functional learning (Part III), the congruence used is defined over a right continuation, called tails, rather than a two sides context. This implies a finite number of classes that allows the use of finite state machines. It is worth noticing that this is a functional extension of the learning algorithms for languages: works on learning regular languages, like the well-known RPNI algorithm [Oncina and García, 1992], also fulfill this global behavior by focusing on congruence defined by sets of suffixes, called *residuals*.

Furthermore, this global learning problem resolution can also be observed when stochastic computational models are considered, a GI framework not described in this manuscript despite some personal works on the matter (see Section 7.3 for some references). Regardless of its technicality, the work on residual automata [Esposito et al., 2002] is a straightforward continuation of the one of regular languages: the notion of stochastic residual is introduced as an analogue of the syntactic congruence and the learning algorithm aims at observing these stochastic classes in a multi-set of examples.

Even the recently introduced and successful approach named spectral learning [Balle et al., 2014] can be analyzed following this grid: the learning algorithms correspond to a principal component analysis [Bailly et al., 2009] in the space of stochastic residuals and (partially) observable in a learning sample. The aim is thus to find the non co-linear residuals that define the stochastic languages in an analogy to finding the non-redundant residuals that define the canonical automata in regular languages learning algorithms.

7.2 On representation hierarchies

Another general point can be made to conclude this manuscript. It concerns the way computer scientists conceive and structure the universe of all computational models.

The usual hierarchy of formal languages is the one of Chomsky [Chomsky, 1959]: 4 classes defined by restrictions on the syntax of grammatical rewriting rules. However, there exists good reasons to move beyond this hierarchy that is based only on the syntactic form of rules: for instance, the Chomsky hierarchy tends to group into classes representations that are fundamentally different since crucial properties like equivalence decidability are not shared by all elements of a class [Sénizergues, 2001]. The same occurs when one considers natural languages: it is commonly accepted that they are all contained in an intermediate class in the hierarchy, called mildly context-sensitive [Joshi, 1985], that may even be orthogonal to the classes (as not all context-free structures may be required for natural languages). The fact that no cognitive evidence exists to differentiate these classes [Öttl et al., 2015] is also a strong argument against the hierarchy.

Despite its drawbacks, few have been done to overcome the Chomsky hierarchy and to propose a new one. An exception is the work of Kallmeyer [2010] that argues in favor of a hierarchy defined at a (natural) language level and not at a representational

one. Though her point of view is biased toward her field of computational linguistics, the idea of changing the support of the hierarchy is appealing.

A possibility that the work presented in this manuscript allows to formulate is to propose a hierarchy based on the identifiability of the representation classes. Indeed, we have argued in the previous section that learning results of computational model classes can be seen through the prism of the type of congruence sought in the data: this provides a natural way to classify classes of formal languages. This promising sketch for a new hierarchy can be refined by specifying subclasses based on the complexity of the structural combination between congruence classes.

Importantly, the simplest class of this hierarchy would be the same than the one of Chomsky: the regular grammars are the one identifiable when the simplest congruence is of interest, that is the suffix/prefix one. The outlines of the following classes have still to be carefully drawn, but the work on distributional learning and the recent decidability result of classes defined in this context [Kanazawa and Kappé, 2018] provide strong bases to build the next level of the hierarchy.

This positions identification on an orthogonal standpoint than the one discussed in the introduction of this manuscript. Instead of seeing it as a rival of approximation, we can move away from machine learning considerations and understand identification as a structuring property for the set of computational models. In this context, identification is the process of decoding the information contained in a set of data in order to recover the corresponding model. One can even reverse the perspective and see the notion of characteristic sample as a way to code a language in an adversarial framework.

7.3 To be exhaustive

This document has been thought as a witness of the research work handled since my PhD. Obviously, it cannot be exhaustive: all researchers know that a lot of dead-ends have to be investigated before finding a conclusive path. And we all have on-going works not mature enough, or too partial, to be written down.

Moreover, even some interesting positive projects that have been investigated are not presented in this HDR. It is the case for instance of the two international challenges I co-organized: the Probabilistic Automata Learning Competition (PAutomaC) [Verwer et al., 2014] and the Sequence PredIction ChallengE (SPiCe) [Balle et al., 2017]. The success of both challenges mostly relied on a wide understanding of the state of the art on the corresponding task, and on a well-designed process for the data generation.

Another non described work concerns the spectral learning of weighted automata. This approach is, together with the distributional learning presented in Part II, the main success in grammatical inference over the past ten years. It provides a way to infer a finite state machine that represents a probability distribution from a multi-set of data. The theoretical study of this approach was initiated independently by Hsu

et al. [2009] and Bailly et al. [2009] and enjoyed tremendous developments since then (see for instance the survey from Balle et al. [2014] for more details). In this context, I participated to the development of the first toolbox that allows the easy use of the different variants of the algorithm [Arrivault et al., 2016, 2017].

Following this first excursion in statistical machine learning, I recently opened a new line of research by proposing to use the spectral learning algorithm to extract a computational model (a weighted automaton) from any black box computing a real value from sequential data. In particular, together with Stéphane Ayache, we show during Spring 2018, thanks to a 5 month last year Master student internship, that this idea is promising for explaining and interpreting the behavior of Recurrent Neural Networks [Ayache et al., 2018], the main deep learning model used for sequential data.

7.4 Some personal remarks

It is important to conclude this document by pointing out the scientific context of the work presented here. Indeed, it has mainly be conducted in a team of brilliant and humanly exceptional researchers, namely the QARMA team of the ex Laboratoire d'Informatique Fondamentale in Marseille (now merged into the Laboratoire d'informatique et Systèmes). However, the main focus of the members of QARMA is statistical machine learning, and so was the description of my tenure position. Given my background and my main research interests, this has led to scientific difficulties for integration. For instance, none of the works presented in the core of this manuscript has been done in collaboration with a current member of the team. This also explains, at least partly, why no co-mentoring of PhD students from Marseille have been done (my two experiences of PhD tutoring were all done via collaborations with foreign universities, on project handled by them). To be clear on the subject: I do not regret this inadequacy between my background and the central scientific concern of the team, *au contraire*, this is a remarkable richness that allowed me to grow and to deepen my knowledge of a very dynamic - and crucial - field: it has just slowed down the course of the research presented here.

A more important issue endangered - and still threatens - the type of work presented here: the impossibility to found it. Indeed, I spent the last 5 years asking all possible French funding agencies for money to develop promising research, without any success. For instance, I tried several times to fund a research aiming at investigating the links between the spectral and the distributional learnings, since these approaches share analogue starting points. Of course, my ability to write down a convincing grant can be questioned, but the fact that most of these grant proposals were written in collaboration with researchers who were used to succeed tends to dismiss this point. It is thus more likely to be a consequence of the combination of two factors: the recent effervescence around deep learning that is draining ML scientific funding despite its strong engineering

flavor, and the chronic under-funding of French academic research, accentuated by the ideological and out of touch policies of its Ministry of Research.

This is the reason why this general conclusion does not contain future research direction: hundreds of pages on that matter have been written in the form of grant proposals, but none of this will be achieved due to the absence of funding. I thus chose to not describe here potential continuations or promising research directions that are very unlikely to be developed.

In this context, this manuscript can be seen as a way to close the page of my research about learning computational models. One should never say never, and it is likely that on-going collaborations will continue on some of the topics presented here, but it will no longer be at the core of my research work. As I have been moving to more mainstream machine learning for a couple of years, the transition should not be too hard to achieve. Notice however that this move is not due to a lack of interest for the thematic, or to the lack of its potential impact: I am convinced this research is promising, useful, and has its role to play in the future of Machine Learning. It is just that, after swimming against the tide during ten years, I decided to seize the opportunity caused by the explosion of interest around the domain to move to more mainstream considerations.

To conclude on a positive note, I want to emphasize how glad I am to be able to participate with my modest means to the blooming of the field of Machine Learning: if computability and decidability have been at the core of Computer Science during the last 70 years, the future of this science will undeniably pass by the development of a theory of learnability. For reasons exposed throughout this manuscript and this conclusion, I hope GI will play a decisive role in this evolution.

Bibliography

D. Arrivault, D. Benielli, F. Denis, and R. Eyraud. Sp2Learn: A Toolbox for the Spectral Learning of Weighted Automata. In *Proc. of the International Conference on Grammatical Inference*, 2016. 166

D. Arrivault, D. Benielli, F. Denis, and R. Eyraud. Scikit-SpLearn: a toolbox for the spectral learning of weighted automata compatible with scikit-learn. In *Conférence francophone en Apprentissage*, 2017. 166

S. Ayache, R. Eyraud, and N. Goudian. Explaining black-boxes on sequential data using weighted automata. In *Proc. of the International Conference on Grammatical Inference*, 2018. To appear. 166

R. Bailly, F. Denis, and L. Ralaivola. Grammatical inference as a principal component analysis problem. In *International Conference on Machine Learning*, pages 33–40, 2009. 164, 166

B. Balle, X. Carreras, F. Luque, and A. Quattoni. Spectral learning of weighted automata. *Machine Learning*, 96(1-2):33–63, 2014. 164, 166

B. Balle, R. Eyraud, F. M. Luque, A. Quattoni, and S. Verwer. Results of the sequence prediction challenge (SPiCe): a competition on learning the next symbol in a sequence. In *Proc. of the International Conference on Grammatical Inference*, volume 57 of *PMLR*, pages 132–136, 2017. 165

N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2 (2):137–167, 1959. 164

Y. Esposito, A. Lemay, F. Denis, and P. Dupont. Learning probabilistic residual finite state automata. In *Grammatical Inference: Algorithms and Applications*, pages 77–91. Springer Berlin Heidelberg, 2002. 164

D. Hsu, S. Kakade, and T. Zhang. A spectral algorithm for learning hidden markov models. In *Conference on Computational Learning Theory*, 2009. 165

A.K. Joshi. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In Karttunen L. Dowty, D. and A. Zwicky, editors, *Natural Language Parsing*. Cambridge University Press, 1985. 164

L. Kallmeyer. *Parsing Beyond Context-Free Grammars*. Springer Publishing Company, Incorporated, 2010. 164

M. Kanazawa and T. Kappé. Decision problems for clark-congruential languages. In *Proc. of the International Conference on Grammatical Inference*, 2018. to appear. 165

J. Oncina and P. García. Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis*, 1992. 164

B. Öttl, G. Jäger, and B. Kaup. Does formal complexity reflect cognitive complexity? investigating aspects of the chomsky hierarchy in an artificial language learning study. *PLoS ONE*, 10, 2015. 164

G. Sénizergues. L(a) = l(b)? decidability results from complete formal systems. *Theor. Comput. Sci.*, 251(1-2):1–166, 2001. 164

S. Verwer, R. Eyraud, and C. de la Higuera. PAutomaC: a probabilistic automata and hidden markov models learning competition. *Machine Learning*, 96(1-2):129–154, 2014. 165

Part V

Annexes

Curriculum Vitae

Current Situation

Ph.D in Computer Science
Maitre de Conférences at the Aix-Marseilles University (France) - tenure
Member of the steering committee of the International Community in Grammatical Inference (chair between 2014 and 2018)

Academic Degrees

Doctoral Dissertation. "Grammatical Inference of Context-Free Languages", University Jean Monnet of Saint-Etienne, 2006.
Under the co-supervision of Pr. Colin de la Higuera (University of Nantes) and Pr. Jean-Christophe Janodet (University of Evry).

MSc. in Computer Science. Ecole Nationale des Mines de Saint-Etienne, 2003.
Concentration: multi- agent systems, machine learning, grammatical inference.
Final thesis: "Learning automata with a splitting-state process".
First year thesis: "Simulations of social networks using cellular automata".

Licence in Computer Science. University Jean-Monnet, Saint-Etienne, 2001. Major in Computer Science, minor in Mathematics.

Publications

All publications can be found at http://pageperso.lis-lab.fr/~remi.eyraud/.

Book Chapter

"Efficiency in the identification in the limit paradigm", Rémi Eyraud, Jeffrey Heinz, Ryo Yoshinaka, in Topics in Grammatical Inference, Springer, 2016

International Journals

- "Designing and Learning Substitutable Plane Graph Grammars", Rémi Eyraud, Jean-Christophe Janodet, Tim Oates, Frédéric Papadopoulos, Fundamenta Informaticae, 2016

- "Learning Strictly Local Subsequential Functions", Jane Chandlee, Rémi Eyraud, Jeffrey Heinz, Transactions of the Association for Computational Linguistics, 2014.

- "PAutomaC: a probabilistic automata and hidden Markov models learning competition", Sicco Verwer, Rémi Eyraud, Colin de la Higuera, Machine Learning Journal, 2014

- "Using Contextual Representations to Efficiently Learn Context-Free Languages", Alexander Clark, Rémi Eyraud, Amaury Habrard, Journal of Machine Learning Research, 2010

- "Polynomial Identification of Substitutable Context-Free Languages", Alexander Clark, Rémi Eyraud, Journal of Machine Learning Research, 2007

- "LARS: a Learning Algorithm for Rewriting Systems", Rémi Eyraud, Colin de la Higuera, Jean-Christophe Janodet, Machine Learning Journal, 2007

Peer-Reviewed International Conferences

- "Learning with Partially Ordered Representations", Jane Chandlee, Rémi Eyraud, Jeffrey Heinz, Adam Jardine, and Jonathan Rawski, proceedings of the sixteenth Mathematics of Languages, ACL, 2019

- "Explaining Black Boxes on Sequential Data using Weighted Automata", Stéphane Ayache, Rémi Eyraud, and Noé Goudian, proceedings of the 14th International Conference on Grammatical Inference, PMLR, vol. 88, to appear, 2018

- "Sp2Learn: A Toolbox for the spectral learning of weighted automata", Denis Arrivault, Dominique Benielli, FranÃ§ois Denis and Rémi Eyraud, proceedings of the 13th International Conference on Grammatical Inference, pages 3 to 17, JMLR W&C, vol. 57, 2016

- "Output Strictly Local Functions", Jane Chandlee, Rémi Eyraud, Jeffrey Heinz, proceedings of the 14th Meeting on the Mathematics of Language, pages 112 to 125, Association for Computational Linguistics, 2015

- "Very efficient learning of structured classes of subsequential functions from positive data", Adam Jardine, Jane Chandlee, Rémi Eyraud, Jeffrey Heinz , proceedings of the 12th International Conference on Grammatical Inference, pages 94 to 108, JMLR W&C, vol. 34, 2014

- "Learning Substitutable Binary Plane Graph Grammars", Rémi Eyraud, Jean-Christophe Janodet, Tim Oates, proceedings of the 11th International Conference on Grammatical Inference, pages 114 to 128, JMLR W&C, vol. 21, 2012
- "A Polynomial Algorithm for the Inference of Context-Free Languages", Alexander Clark, Rémi Eyraud, Amaury Habrard, proceedings of the 9th International Colloquium on Grammatical Inference, pages 29 to 42, LNAI 5278, Springer, 2008
- "Learning Auxiliary Fronting with Grammatical Inference", Alexander Clark, Rémi Eyraud, proceedings of the 10th Conference on Computational Natural Language Learning, pages 125 to 132, 2006
- "Identification in the Limit of Substitutable Context-Free Languages", Alexander Clark, Rémi Eyraud, proceedings of the 16ith International Conference on Algorithmic Learning Theory, pages 283 to 296, LNAI 3734, Springer, 2005
- "Representing Languages by Learnable Rewriting Systems", Rémi Eyraud, Colin de la Higuera, Jean-Christophe Janodet, proceedings of the 7th International Colloquium on Grammatical Inference, pages 139 to 150, LNAI 3264, Springer, 2004

Short papers, posters, workshops

- "Scikit-SpLearn: A Toolbox for the spectral learning of weighted automata compatible with scikit-learn", Denis Arrivault, Dominique Benielli, FranÃ§ois Denis and Rémi Eyraud, proceedings of the Conference francophone en Apprentissage (French Machine Learning Conference), 2017
- "Results of the Sequence PredIction ChallengE (SPiCe): a Competition on Learning the Next Symbol in a Sequence ", Borja Balle, Rémi Eyraud, Franco M. Luque, Ariadna Quattoni, Sicco Verwer, proceedings of the 11th International Conference on Grammatical Inference, pages 243 to 248, JMLR proceedings, 2016
- "New Polynomial Bounds for the Identification in the Limit Paradigm using Generative Grammars", Rémi Eyraud, Jeffrey Heinz, ICALP Satellite Workshop on Learning Theory and Complexity, 2013, Riga, Latvia.
- "Results of the PAutomaC Probabilistic Automaton Learning Competition ", Sicco Verwer, Rémi Eyraud, Colin de la Higuera, proceedings of the 11th International Conference on Grammatical Inference, pages 243 to 248, JMLR proceedings, vol. 21, 2012.
- "A note on contextual binary feature grammars", Alexander Clark, Rémi Eyraud, Amaury Habrard, EACL 2009 workshop on Computational Linguistic Aspects of Grammatical Inference, Athens, Greece, 2009.

- "Spring School in Machine Learning. Teaching experiences", Cécile Capponi, FranÃ§ois Denis, Rémi Eyraud, Amaury Habrard, Liva Ralaivola, PASCAL Workshop: Teaching Machine Learning, Saint-Etienne, France, 2008.
- "Two methods to learn context-free languages", Rémi Eyraud, workshop ML4NLP, Amsterdam, 2007.
- "Learning Auxiliary Fronting with Grammatical Inference", Alexander Clark, Rémi Eyraud, proceedings of the 28th Annual Conference of the Cognitive Science Society, pages 1127 to 1132, Vancouver, Canada, 2006.
- "Deux Techniques d'Apprentissage de Langages Hors-Contextes", Rémi Eyraud, proceedings of the 8th Conférence francophone d'Apprentissage Automatique, pages 186 and 187, Trégastel, France, 2006.

Career

September 2007 - present: Maitre de conférences (Junior Professor - tenure), Aix-Marseille University, LIS UMR CNRS 7020, France

March 2018: Invited lecturer at the Stony Brook University, New York, USA

July 2014 - December 2014: Invited researcher at the Delaware University, USA

July 2012 - December 2013: Sabbatical founded by the CNRS; Invited researcher at the University of Maryland, Baltimore County, USA

January 2007 - September 2007: non-permanent researcher at the University of Amsterdam, The Netherlands

September 2006 - January 2007: non-permanent full-time lecturer (ATER) at University of Saint-Etienne, France

September 2003 - August 2006: Ministry of Research founded Ph.D program with teaching duties (Allocataire-Moniteur)

Research Activities

Principal collaborations

- Alexander Clark, King's College, London, UK (7 co-authored papers, long visit)

- Jeffrey Heinz, University of Delaware, USA (5 co-authored papers, long visits, co-organised events)
- Colin de la Higuera, Nantes University, France (4 co-authored papers, co-organised events)
- Jean-Christophe Janodet, Evry University, France (4 co-authored papers)
- Sicco Verwer, Delft University, The Netherlands (3 co-authored papers, co-organised international challenges)
- Tim Oates, University of Maryland, USA (2 co-authored papers, co-organised event)
- Jane Chandlee, Haverford College, USA (3 co-authored papers, on-going work)
- FranÃ§ois Denis, Aix-Marseille University, France (2 co-authored papers, ongoing project)
- Amaury Habrard, Saint-Etienne University, France (2 co-authored papers)
- Borja Balle, Amazon Research, UK, (1 co-authored paper, co-organised events)
- Ryo Yoshinaka, Kyoto University, Japan (1 co-authored paper, short visits)

Invited talks

- Invited speaker at the ACL workshop on Deep Learning and Formal Languages: Building Bridges, Florence, Italy, 2019. Title: Distilling computational models from Recurrent Neural Networks.
- Invited speaker at the Journées annuelles de Vérification, GdR IM, Grenoble, 2018. Title: Spectral Learning of Weighted Automata, from theory to a toolbox.
- Invited speaker at the Linguistic departement of the Stony Brook University, USA, 2018. Title: Recent advances in Grammatical Inference.
- Invited speaker at the ELC Workshop on Learning Theory and Complexity, Kyoto, 2014. Title: Efficiency in the identification in the limit paradigm
- Invited seminar at the Computer and Information Science department of the University of Delaware, 2013. Title: Recent advances in grammatical inference of non-regular languages
- Invited seminar at the Computer Science department of the University of Maryland, Baltimore County, 2012. Title: Learning context-free grammars

Editing & Reviewing

- Co-editor of a special issue on grammar learning for Fundamenta Informaticae, 2016
- Member of the program committee of the international conferences EMNLP-CoNLL'07, ICGI'08, ICGI'12, ICGI'14, ICGI'16, ICGI'18
- Reviewer for the Machine Learning Journal (2014, 2015), Theoretical Computer Science (2013, 2014, 2015, 2016), Journal of Machine Learning Research (2010, 2012, 2015), Information Processing Letters (2007), Advances in Complex Systems (2006)
- Reviewer for the international conferences ECML'05, ECML'10, ICML'15, NIPS'16, ICLR'18
- Reviewer for the francophone conference CAp'05, CAp'06, CAp'16, CAp'18, CAp'19

Tutoring

- **PhD**
  - Member of the PhD proposal and defense committees of Adam Jardine (University of Delaware, USA), 2013-2016
  - Co-tutoring with Tim Oates of John Clemens (University of Maryland, Baltimore County, USA), 2012-2015
- **MSc.**
  - Final internship and thesis of a MSc student. Subject: Inside the black-box: extracting WA from RNN, 2018
  - Last year project of a MSc. Student. Subject: Automatic detection of typewriter model (in collaboration with French national Forensic team), 2017
  - Final internship of an applied MSc. student. Subject: Image classification via graph features, 2016
  - Last year project of 4 applied MSc. students. Subject: Image indexation by content, 2011
  - Last year project of 4 applied MSc. students. Subject: Development of a Firefox plug-in for film recommendation (allocine.fr), 2010
  - First year thesis of a MSc. student. Subject: learning formal languages: theory and practice, 2009

  - Final internship and thesis of a MSc. student. Subject: statistical grammatical inference: from theory to implementation, 2008
  - First year project of 3 students. Subject: Implementation of a grammatical inference algorithm, 2008
  - First year thesis of a student. Subject: Learning Languages defined with the use of String Rewriting Systems, 2006
  - First year thesis of a student. Subject: Implementation of a learning algorithm: LARS, 2005

- **Undergraduate.** Two to three months internships of third year students. Subjects: Automatic Cancer detection from blood samples & Development of a toolbox for spectral learning & Image classification via graph feature extraction & Machine Learning: teaching via examples, 2014, 2015, 2018, 2019

Past projects

- ANR DECODA, 2010-2013, Speech Analytics in recorded call-center conversations
- ANR LAMPADA, 2009-2014, Learning Algorithms, Models and sPArse representations for structured DAta
- ANR SEQUOIA, 2009-2012, Probabilistic syntactic analysis with large coverage of French.
- ANR MARMOTA, 2005-2008, MAchine learning for pRobabilistic MOdels Tree lAnguages
- Member of the European networks of excellence PASCAL: Pattern Analysis, Statistical modelling and ComputAtional Learning (2003-2007) and PASCAL2 (2008-2013)

Software

Implementation of 2 toolboxes for spectral learning, Sp2Learn and Scikit-SpLearn, during a 1 year LabEx Archimède project (both in production), Free BSD licensed

Teaching

This teaching was done during a 3 years formation (monitorat) between 2003 and 2006, then during a 6 month full-time lecturer contract at the university

Jean-Monnet (2006) and finally during my current position (≈ 220 hours per year).

2nd year MSc. Computer Science	XML*	(2 years)
	Classification, Learning, Decision	(3 years)
	Data Mining*	(4 years)
	Research Methodology*	
	Machine Learning*	(3 years)
1st year MSc. Computer Science	Theoretical Computer Science*	
	Introduction to Data Science*	
	Data Science	
3rd year License Computer Science	Algebraic Languages	(2 years)
	Data Bases	(10 years)
	Compilation	(6 years)
	Translation & Semantic	(3 years)
	Advanced Algorithmic	(3 years)
	Introduction to Machine Learning*	
2nd year License Computer Science	Rational Languages*	(2 years)
2nd year License Physics	Imperative Programming*	(2 years)
1st year License Computer Science	Functional Programming: Caml	(3 years)
	Introduction to Algorithms	
	Programming in C	(2 years)

Note: In charge of the courses "Advanced Algorithmic", "Algebraic Languages", "XML: principles and tools", "Programming in C", "Data mining", "Compilation", "Data Bases", "Translation & Semantic", "Classification, Learning, Decision", "Data Science", "Introduction to Machine Learning, "Introduction to data science", and "Theoretical Computer Science".
All courses correspond roughly to 60-70 hours, except those noted with a star, most of which are half unit of about 30 hours.

A 4 days professional formation titled "Machine Learning: from Bases to Deep Learning" was given at Leroy Merlin Headquater in April 2018, in the context of CNRS Formation Entreprise.

Collective Responsibilities

- Co-chair of the LearnAut 2018 workshop at the Federated Logic Conference (FloC 2018), Oxford, UK, 2018
- Co-chair of the LearnAut workshop at the Logic In Computer Science (LICS 2017) conference, Reykjavik, Iceland, 2017
- Co-organiser of the SPiCe on-line competition about learning the next symbol in

sequences, 2016

- Co-organiser of the PAutomaC on-line competition about learning probabilistic finite state machines, 2012
- In charge of the Master program in computer science "Advanced Data Bases" 2009-2011
- In charge of the Master program in computer science "Reliability and Security in Computer Science" 2010-2012
- Head of the organisation committee and member of the scientific committee of the PASCAL2 Bootcamp 2010, Marseilles, France
- Elected at the council of the UFR ($\simeq$ French faculty) Mathematics - Computer Science - Mechanics of the University of Provence (2008-2012). Member of the research and pedagogic council of this UFR.
- Elected at the executive committee of the national association for computer science SPECIF, 2009-2011
- Member of the organisation committee of the first French summer school on Machine Learning, EPIT 2008, in charge of communication
- Member of the organisation committee of the SPECIF congress 2006 on Ph. D cursus, Saint-Etienne
- Member of the organisation committee of the grammatical inference workshop 2006, Saint-Etienne
- Elected at the council of the doctoral school of Saint-Etienne, 2002-2007
- President of the Association Stéphanoise des jEunes Chercheurs, 2003-2004 Vice-president of the Association Stéphanoise des jEunes Chercheurs, 2004-2005
- Member of the executive committee of the Confédération des Jeunes Chercheurs, 2005-2007

Science popularisation

- Member of the organization committee of the Treize Minutes Marseille, a science popularization event, 2014, 2015, 2016, 2017, 2018
- Co-creator and animator of the network Communauté Française des Docteurs in Marseilles, 2015

- Animation of a debate ("Causerie") at the Fêtes de la science 2015, Villa mediterranée, Marseilles
- Talk titled "Computer science: education, jobs and dangers" at the French high school La Condamine of Quito (Ecuador), January 2007
- Animation of a debate with high school students, titled "To talk and to discuss with a computer", during the Vogue du Net organized by the town of Saint-Etienne in may 2006
- Animation of a debate, "the new dangers of internet", at the occasion of the Science Party 2006, at La Rotonde

Mentoring

Given the research path followed during these years, it was hard to find funding: no grant proposal has been accepted despite months passed writing them down (for the past 5 years: 5 ANR projects, 4 of which at the second round; 2 CNRS PEPS; a IUF project; 1 AMIdex Pépinière; 3 Archimède Development team - 1 accepted for officially 4 months; 2 GdR ISIS). Usual feedbacks are that the proposed research is of interest but is too risky and/or suffer from a potential lack of impact.

If one adds to this the statistical machine learning nature of the QARMA team, the lack of supervision of PhD in Marseilles is easily explainable. This has two immediate consequences: to slow down the pace of this research and to require more personal investment to achieve the results presented in this HDR.

Nevertheless, it is worth noticing that an important amount of tutoring has been realized. First, 10 graduate/master students have been supervised during the past ten years (details are given in the joined Curriculum Vitae). This includes Research internships of 3 last year Master students. Then, several works have been realized and published with the participation of PhD candidates from my collaborators' team. This implied deep exchanges and important advising.

Finally, two PhD mentoring were officially done. First, at the University of Maryland, Baltimore County (UMBC), I co-supervised John Clemens with Tim Oates when he was a PhD student. We worked on the idea to extract graphs from images, with the final goal of learning Plane Graph Grammars from these graphs. John being a part time student at that time, we did not achieve that goal, but instead used frequent subgraphs counting to use classical SML algorithms. This allowed John to write his first research paper and to validate this way his portfolio in order to become a PhD candidate[1]. As I had to leave UMBC and as John was getting interested with more mainstream ML, the tutoring ended here.

The second one happened at the Delaware University, within the team of Jeffrey Heinz. After an unofficial co-mentoring of Jane Chandlee during her last PhD years, I closely co-supervised Adam Jardine when he was a PhD candidate. As already described, papers were published together. More importantly, frequent exchanges happened and it is unlikely to be a lack of humility to say that I actively participated to the successes of the works he authored alone. This mentoring was made official by the participation to his PhD proposal and his PhD defense committees. Now with a tenure-track at the linguistic department of the Rutgers University, the interaction is continuing and new collaborating works are about to be published.

[1] This - unpublished - work can be found at http://pageperso.lis-lab.fr/~remi.eyraud/john-portfolio.pdf)

Main Acronyms and Notations Index

Grammatical Inference : Learning Computational Models from Various Types of Structured Data

Rémi Eyraud

Abstract:
Computational models, such as the ones of the Chomsky hierarchy, are at the core of the Computer Science theory and, at the same time, are massively used in real-life applications. The field of Grammatical Inference aims at studying the learnability of such models, mainly by providing learning algorithms, investigating their properties, and characterizing the learned classes. This usually raises questions that questioned the notion of learning itself, providing a deep insight into our understanding of machine learning.
In this manuscript, we focus on three developments of this field. In a first part, we discuss different formalizations of the notion of learning and their interest in the context of computational models. This allow us to formulate new paradigms and to refute the usability of well-known ones.
The second part regroups several published works that share a common idea, known under the name of distributional learning. The core of this approach is to observe in the data the distribution of congruence classes and to infer a grammatical model that captures the observed structures. We show the potential of this idea by providing first an algorithm that provably learns from strings a class of rewriting models that contains all regular languages and some context-free ones, and, second, a study of a specifically designed graph grammar class for which a simple learning algorithm can be designed.
The last part of the document concentrates on learning string to string functions using computational models known under the name of transducers. Two different works are detailed, both initiated by linguistic motivations. In the first one, we focus on learning efficiently functions whose domain is known a priori. In the second one, we restrict ourselves to classes of transductions that benefit from a locality property. In both cases, we provide learning algorithms together with a theoretical learning study.

Keywords: Grammatical Inference, Language Theory, Machine Learning, Learning Theory, Distributional Learning, Graph Grammars, Transducers.