Complexity and Reductions Issues in Grammatical Inference

Colin de la Higuera [1]
EURISE, Université de Saint-Etienne, 23 rue du Docteur Paul Michelon, 42023 Saint-Etienne, France
e-mail: cdlh@univ-st-etienne.fr

ABSTRACT

Grammatical inference deals with learning grammars or automata from different textual informations. A general paradigm allowing us to describe the convergence of the process is that of identification in the limit. When trying to combine this paradigm with complexity issues, problems arise. We revisit identification in the limit from a (slightly) categorial perspective, and formalise a reduction technique between problems that allows to refine previous results.

Keywords: Identification in the limit, polynomial learning, grammars, automata.

1. Introduction

Grammatical inference is concerned with learning language formalisms from some sort of information which can be text, examples and counter-examples, or anything really that should provide us insight about the language being sought. Surveys of the field are those by Sakakibara [Sak97] or de la Higuera [dlH05]. A presentation of the field for theoreticians can be found in [FdlH04]. If the problem can be stated in such an informal way and there are many papers presenting heuristics and ideas allowing to extract some grammar or automaton from all types of information, theoretical properties of convergence of the algorithms are usually desired. This requires then not just to find some grammar but **the** grammar, inducing that the problem is about searching for a hidden grammar, *i.e.* one defined *a priori*. Obviously, in practice such a hidden grammar may not exist, but the assumption is needed to study the convergence of the algorithms, and experiments (for instance [LPP98]) show that theoretically founded algorithms work better.

There have been two main lines of direction to study convergence in a formal manner:

- Gold's model of learning called *identification in the limit*, with two main variants: learning *from text* and learning *from an informant* [Gol67, Gol78]. From these pioneer works alternative models have been proposed adding complexity constraints [dlH97] or probabilistic issues [CO94b].
- The PAC model (*Probably Approximatively Correct*) introduced by Valiant [Val84] but that in the context of language learning did not allow hardly any positive results. The model has been adapted by taking into account only certain types of distributions and simple PAC learning has been considered with more success [DG97, Den01].

[1]This work was supported in part by the IST Programme of the European Community, under the Pascal Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

We will concentrate here on the former model and follow lines close to complexity theory where reductions can be found allowing to reduce one problem to another. By doing this we follow the work by Pitt and Warmuth [PW88]. The question of polynomial learning is also of interest. A first discussion has taken place in [Pit89], with discussions and further ideas in [dlH97] and [Yok03]. Original ideas concerning these questions (with a notion of stochastically polynomial) can be found in [Zeu03].

In most works concerned with identification in the limit, definitions are irksome and comparisons are hard due to a general lack of common notions and notations. We attempt in this paper to propose a general and (we hope) elegant definition and use this definition to derive reduction results with a categorical flavour.

The paper is organised as follows: in section 2 we discuss identification in the limit which we revisit in some way. We provide in section 3 a theorem which we are able to use in order to relate learning in one formalism with learning in another. This leads to our central theorem 2. Section 4 is about some results deriving from our general theorem. Section 5 is about some open problems and a conclusion.

2. Definitions

2.1. Languages and grammars

An *alphabet* Σ is a finite nonempty set of symbols called *letters*. A *string* w over Σ is a finite sequence $w = a_1a_2 \dots a_n$ of letters. Let $|w|$ denote the length of w. In the following, letters will be indicated by $a, b, c, \dots$, strings by $u, v, \dots, z$, and the empty string by λ. Let Σ^* be the set of all finite strings over alphabet Σ. We denote by Σ^∞ the set of all infinite strings defined as total functions $\mathbb{N} \to \Sigma$.

Let $\mathcal{A}$ be a class of languages.

Let $R(\mathcal{A})$ be a class of representations of languages from $\mathcal{A}$.

Call $\mathcal{L}_\mathcal{A} : \mathcal{R}(\mathcal{A}) \to \mathcal{A}$ the function that with any representation returns (in a non computable way) the corresponding language. This function is surjective: every language can be represented. But it does not have to be injective. Indeed the problem "given G_1 and G_2 $\mathcal{L}_\mathcal{A}(G_1) = \mathcal{L}_\mathcal{A}(G_2)$?" is the classical equivalence problem whose importance is known when dealing with the computational aspects of learning [dlH97].

We suppose on the other hand that the following word problem to be decidable: given $w \in \Sigma^*$ and $G \in R(\mathcal{A})$, $w \in \mathcal{L}_\mathcal{A}(G)$?.

In the above we can think of classes of languages as being the regular languages or the context-free languages, and classes of representations, in turn, as being those of context-free grammars, deterministic finite state automata or regular expressions.

2.2. Identification in the limit

Definition 1 Let $\mathcal{A}$ be a class of languages. A presentation is an element of $Pres(\mathcal{A})$ which is a set of functions $\mathbb{N} \to X$ where X is a set. In some way these presentations *denote* languages from $\mathcal{A}$, *i.e.* there exists a function $yield : Pres(\mathcal{A}) \to \mathcal{A}$. If $L = yield(f)$ then we will say that f is a presentation of L.

With this definition one should not think of presentations as text or informant, but in a broader sense as a sequence of informations of any type that hopefully inform us on the language we are to learn.

Typical presentations could be Text, Informant, Prefixes,... Here are some examples of possible presentations:

- Text=$\{f : \mathbb{N} \to \Sigma^* \cup \{\#\} : f(\mathbb{N}) = L \text{ or } \{\#\} \text{ where } L \subset \Sigma^*\}$.

- Ordered Text=$\{f : \mathbb{N} \to \Sigma^* \cup \{\#\} : f(\mathbb{N}) = L$ and $(i < j \implies f(i) < f(j))$ or $\{\#\}$ where $L \subset \Sigma^*\}$.
- Informant=$\{f : \mathbb{N} \to \Sigma^* \times \{0,1\} : f(\mathbb{N}) = L \times \{1\} \cup \overline{L} \times \{0\}\}$.
- Prefixes=$\{f : \mathbb{N} \to \Sigma^* \times \{0,1\} : f(\mathbb{N}) = Pref(L) \times \{1\} \cup Pref(\overline{L}) \times \{0\}\}$ where $L \subset \Sigma^\infty$.

Definition 2 Let $\mathcal{A}$ be a class of languages and $Pres(\mathcal{A})$ be a type of presentations for $\mathcal{A}$, with associated function *yield*. The setting is said to be *valid* when given 2 presentations f and g, if their range is equal (*i.e.* if $f(\mathbb{N}) = g(\mathbb{N})$) then $yield(f) = yield(g)$.

If a setting is not valid, $\mathcal{A}$ is not going to be learnable from $Pres(\mathcal{A})$.

Given a presentation f we denote by f_n the set $\{f(j) : j \leq n\}$. Given a presentation f we denote by $f(n) \vDash G$ (conversely $f(n) \nvDash G$) when $f(n)$ is consistent with $\mathcal{L}_{\mathcal{A}}(G)$. A *learning algorithm* $\mathbf{a}$ is a program that takes the first n elements of a presentation and returns a representation as output. $\mathbf{a} : \bigcup_{i\in\mathbb{N}}\{f_i\} \to R(\mathcal{A})$. The following definition is directly adapted from [Gol67]:

Definition 3 We say that $\mathcal{A}$ is learnable from $Pres(\mathcal{A})$ in terms of $R(\mathcal{A})$ iff there exists a learning algorithm $\mathbf{a}$ such that for all $L \in \mathcal{A}$ and for any presentation f of L (belonging to $Pres(\mathcal{A})$), there exists a rank n such that for all $m \geq n$, $\mathcal{L}_{\mathcal{A}}(\mathbf{a}(f_m)) = L$.

From this definition can be derived the well-known facts :

Theorem 1 *[Gold, 1967]*
Any class of recursive languages is learnable from an informant.
No super-finite class of languages is learnable from text. A super-finite class is a class that contains all finite languages and at least one infinte one.

2.3. Complexity aspects

These issues have been studied in various places including [Pit89, dlH97]. Nevertheless there is no general agreement between the authors in the field about which definitions should be used, neither (with the exception of [PH00]) have definitions been compared.

We first need to define the sizes of the objects we are manipulating.

- $|G|$ is the size of a grammar (number of bits);
- $|f_n|$ is the number of items in the first n+1 elements of a presentation (n+1);
- $\|f_n\|$ is the number of symbols in in the first n+1 elements of a presentation (number of bits).
- $|L| = \min\{\|G\| : \mathbb{L}(G) = L\}$ The "size" of a language is the size of the smallest grammar of the considered class that can generate it;

The following are then correct definitions of (some type of) polynomial identification in the limit:

The first definition is by far too optimistic and can only work on very limited classes of languages with extremely precise presentations. It implies that all the necessary information is given early which is a strong constraint.

Definition 4 (Overall polynomial time) If there exists a polynomial $p()$ such that $\forall G \in R(\mathcal{A}), \forall n \geq p(|G|), \forall f \in Pres(L)$, $\mathcal{L}_{\mathcal{A}}(\mathbf{a}(f_n) = L$.

The next definition just states that to produce its next hypothesis the algorithm only requires polynomial time. This definition alone is insufficient as shown in [Pit89].

Definition 5 (Polynomial update time) An algorithm **a** is said to have *polynomial update time* if there is a polynomial $p()$ such that, for every presentation f and every integer n, constructing $H_n = \mathbf{a}(f_n)$ requires $p(\|f_n\|)$ time.

Another definition that has allow

Definition 6 (Polynomial characteristic set) An algorithm **a** admits a polynomial characteristic set if there exists a polynomial $p()$ such that $\forall G \in R(\mathcal{A}) \exists W \subset X : \|W\| \leq p(|G|) \wedge W \subset f_n \implies \mathcal{L}_{\mathcal{A}}(\mathbf{a}(f_n)) = \mathcal{L}_{\mathcal{A}}(G)$.

A notion that has been used in various papers was introduced in [Pit89]:

Definition 7 (Implicit prediction errors) Given a learning algorithm **a** and a presentation f, we say that **a** *makes an implicit prediction error at time n* if $f(n) \not\models \mathbf{a}(f_{n-1})$.

Let f be a presentation for L, algorithm **a** is said to make a *polynomial number of implicit prediction errors* if there is a polynomial $p()$ such that, for each language L and each presentation f for L, $|\{k \in \mathbb{N} : f(k+1) \not\models \mathbf{a}(f_k)\}| \leq p(|L|)$.

A nice alternative to counting the number of errors is that of counting the number of changes of hypothesis one makes. On its own, this is meaningless (why change?), but if combined with identification in the limit the definition makes sense:

Definition 8 (Polynomial mind changes) Given a learning algorithm **a** and a presentation f, we say that **a** *changes its mind at time n* if $\mathbf{a}(f_n) \neq \mathbf{a}(f_{n-1})$.

Let f be a presentation for L, algorithm **a** is said to make a *polynomial number of mind changes* if there is a polynomial $p()$ such that, for each language L and each presentation f for L, $|\{k \in \mathbb{N} : \mathbf{a}(f_k) \neq \mathbf{a}(f_{k+1})\}| \leq p(|L|)$.

Combining ideas, one gets:

Definition 9 (Yokomori [Yok03]) An algorithm **a** identifies a class $\mathcal{A}$ in the limit in Yokomori polynomial time if:

- **a** identifies $\mathcal{A}$ in the limit;
- **a** has polynomial update time;
- **a** makes a polynomial number of implicit prediction errors.

Note that the first condition is not implied by the two other; in a similar way Pitt introduced a definition that could be formalized as follows:

Definition 10 (Pitt [Pit89]) An algorithm **a** identifies a class $\mathcal{A}$ in the limit in Pitt polynomial time if:

- **a** identifies $\mathcal{A}$ in the limit;
- **a** has polynomial update time;
- **a** makes a polynomial number of mind changes.

Here also, please note that the first condition is not implied by the two other.

3. Commuting diagrams

A learning/identifying situation can be understood by stating the class of languages under study, the representations one is interested in and the sort of presentations one admits.

In this section we aim at studying how situations relate to each other.

3.1. Definitions

3.1.1. Languages and grammars

Let $\mathcal{A}$ and $\mathcal{B}$ be 2 classes of languages represented by grammars from (respectively) $R(\mathcal{A})$ and $R(\mathcal{B})$.

We denote by $\mathcal{L}_{\mathcal{A}}$ (respectively $\mathcal{L}_{\mathcal{B}}$) the surjective mapping $R(\mathcal{A}) \to \mathcal{A}$ (respectively $\mathcal{L}_{\mathcal{B}} : R(\mathcal{B}) \to \mathcal{B}$).

Given a surjective mapping $\phi : \mathcal{A} \to \mathcal{B}$, we denote by ψ a (surjective) mapping $R(\mathcal{A}) \to R(\mathcal{B})$ for which the diagram commutes.

$$\begin{array}{ccc} R(\mathcal{A}) & \xrightarrow{\psi} & R(\mathcal{B}) \\ {\scriptstyle \mathcal{L}_{\mathcal{A}}}\downarrow & & \downarrow{\scriptstyle \mathcal{L}_{\mathcal{B}}} \\ \mathcal{A} & \xrightarrow{\phi} & \mathcal{B} \end{array} \tag{3.1}$$

Hence:

$$\phi \circ \mathcal{L}_{\mathcal{A}} = \mathcal{L}_{\mathcal{B}} \circ \psi \tag{3.2}$$

3.1.2. Languages and presentations

We now concentrate on presentations.

$$\begin{array}{ccc} \mathcal{A} & \xrightarrow{\phi} & \mathcal{B} \\ {\scriptstyle yield}\uparrow & & \uparrow{\scriptstyle yield} \\ Pres(\mathcal{A}) & \xrightarrow{\xi} & Pres(\mathcal{B}) \end{array} \tag{3.3}$$

Again we need the second diagram to commute. Given a surjective mapping $\phi : \mathcal{A} \to \mathcal{B}$, we denote by ξ a (surjective) mapping $Pres(\mathcal{A}) \to Pres(\mathcal{B})$ for which the diagram commutes. So we have:

$$\phi \circ yield = yield \circ \xi \tag{3.4}$$

As a presentation may not be a computable function, describing the computation aspects of function ξ is as follows:

Definition 11 A reduction between presentations $\xi : Pres(\mathcal{A}) \to Pres(\mathcal{B})$ such that $\xi(f) = g$ is computable if there exists a computable function $\overline{\xi} : X \to 2^Y$ such that $\bigcup_{i \in \mathbb{N}} \overline{\xi}(f(i)) = g(\mathbb{N})$.

Note that $\forall i \in \mathbb{N}$, $\overline{\xi}(f(i)$ is a finite set. $\overline{\xi}$ is the description at each point of function ξ.

Definition 12 A reduction between presentations $\xi : Pres(\mathcal{A}) \to Pres(\mathcal{B})$ such that $\xi(f) = g$ is polynomial if there exists a polynomial function $\overline{\xi} : X \to 2^Y$ such that $\bigcup_{i \in \mathbb{N}} \overline{\xi}(f(i)) = g(\mathbb{N})$.

So combining both diagrams we have:

$$\begin{array}{ccc} R(\mathcal{A}) & \xrightarrow{\psi} & R(\mathcal{B}) \\ {\scriptstyle \mathcal{L}_{\mathcal{A}}}\downarrow & & \downarrow{\scriptstyle \mathcal{L}_{\mathcal{B}}} \\ \mathcal{A} & \xrightarrow{\phi} & \mathcal{B} \\ {\scriptstyle yield}\uparrow & & \uparrow{\scriptstyle yield} \\ Pres(\mathcal{A}) & \xrightarrow{\xi,\overline{\xi}} & Pres(\mathcal{B}) \end{array} \tag{3.5}$$

3.2. The theorem

Theorem 2 *If $\mathcal{B}$ is learnable in terms of $R(\mathcal{B})$ from $Pres(\mathcal{B})$, and there exists a computable function $\chi : R(\mathcal{B}) \to R(\mathcal{A})$ such that $\psi \circ \chi = Id$, and ξ is a computable reduction, then $\mathcal{A}$ is learnable in terms of $R(\mathcal{A})$ from $Pres(\mathcal{A})$.*

It will therefore be more useful to represent the situation as follows:

$$\begin{array}{ccc} R(\mathcal{A}) & \xleftarrow{\chi} & R(\mathcal{B}) \\ {\scriptstyle \mathcal{L}_{\mathcal{A}}}\downarrow & & \downarrow{\scriptstyle \mathcal{L}_{\mathcal{B}}} \\ \mathcal{A} & \xrightarrow{\phi} & \mathcal{B} \\ {\scriptstyle yield}\uparrow & & \uparrow{\scriptstyle yield} \\ Pres(\mathcal{A}) & \xrightarrow{\xi,\overline{\xi}} & Pres(\mathcal{B}) \end{array} \tag{3.6}$$

Proof. Let **b** be a learning algorithm that identifies $\mathcal{B}$. Consider algorithm **a** below, that takes a presentation f by its n first items (f_n) and then executes:
$g_m \longleftarrow \overline{\xi}(f_n); G_{\mathcal{B}} \longleftarrow \mathbf{b}(g_m)$; $G_{\mathcal{A}} \longleftarrow \chi(G_{\mathcal{B}})$; return $G_{\mathcal{A}}$;
As ξ is computable set, T_n can be constructed.
Also if ξ and χ are polynomial, then **b** is polynomial if **a** is. This remark needs to be analysed in a stricter way and will depend on the definition of polynomiality one is using. □

4. Using the theorem to derive results

4.1. From text to informant

Our first example is very simple.

$$\begin{array}{ccc} R(\mathcal{A}) & \xleftarrow{\text{Id}} & R(\mathcal{A}) \\ {\scriptstyle \mathcal{L}_{\mathcal{A}}}\downarrow & & \downarrow{\scriptstyle \mathcal{L}_{\mathcal{A}}} \\ \mathcal{A} & \xrightarrow{\text{Id}} & \mathcal{A} \\ {\scriptstyle yield}\uparrow & & \uparrow{\scriptstyle yield} \\ \text{Informant} & \xrightarrow{\xi,\overline{\xi}} & \text{Text} \end{array} \tag{4.7}$$

$\overline{\xi} : \Sigma^* \to 2^{\Sigma^*}, \xi((w,0)) = \emptyset$, and $\xi((w,1)) = \{w\}$. Hence any class learnable from text is also learnable from an informant.

4.1.1. About sure languages

In [dlHJ04] de la Higuera and Janodet proved that sure languages were learnable from prefixes. These languages concern infinite strings and the result can be reached in a much simpler way through our theorem:

$$\begin{array}{ccc} \text{Büchi automata} & \xleftarrow{\chi} & \text{DFA} \\ {\scriptstyle \mathcal{L}_{\mathcal{A}}}\downarrow & & \downarrow{\scriptstyle \mathcal{L}_{\mathcal{B}}} \\ \text{Sure languages} & \xrightarrow{\phi} & \text{REC} \\ {\scriptstyle yield}\uparrow & & \uparrow{\scriptstyle yield} \\ \text{Prefix Informant} & \xrightarrow{\xi,\overline{\xi}} & \text{Informant} \end{array} \tag{4.8}$$

ξ is identity, and ϕ and χ are the natural transformations. Within the scope of this paper details of the transformations are not given, but these can be easily reconstructed from the article [dlHJ04].

4.2. Even linear languages

Even linear languages and grammars have been analysed in the context of grammatical inference in many papers, for instance [Tak88, SG94, Mäk96, KMT97].

Definition 13 (Linear context-free grammars) A context free grammar $G = (\Sigma, V, P, S)$ is a *linear* grammar if $P \subset V \times (\Sigma^* V \Sigma^* \cup \Sigma^*)$.

Definition 14 (Even linear context-free grammars) A context free grammar $G = (\Sigma, V, P, S)$ is an *even linear* grammar if $P \subset V \times (\Sigma V \Sigma \cup \Sigma \cup \lambda)$.

Thus languages like the set of all palindromes, or language $\{a^n b^n : n \in \mathbb{N}\}$ are even linear without being regular.

$$\begin{array}{ccc} \text{Even linear grammars} & \xleftarrow{\chi} & \text{DFA} \\ \Big\downarrow \mathcal{L}_{\mathcal{A}} & & \Big\downarrow \mathcal{L}_{\mathcal{B}} \\ \text{Even linear languages} & \xrightarrow{Id} & \text{REC} \\ \Big\uparrow yield & & \Big\uparrow yield \\ \text{Informant} & \xrightarrow{\xi,\overline{\xi}} & \text{Informant} \end{array} \tag{4.9}$$

$\overline{\xi}$ takes a string $a_1 a_2 ... a_{2n}$ and returns $(a_1, a_{2n})(a_2, a_{2n-1})....(a_{n-1} a_n)$ (or takes $a_1 a_2 ... a_{2n+1}$ and returns $(a_1, a_{2n+1})(a_2, a_{2n})....(a_{n-1} a_{n+1})(a_n, \#)$).

χ transforms a DFA over alphabet $(\Sigma \cup \{\#\})$ into a deterministic linear grammar over Σ.

4.3. Tree languages

There have been many results over learning from strings that have been adapted to the case where the data is trees [Sak87, Sak90, KS94, Fer02]. It would seem reasonable to have a general method to, when given a string algorithm, derive a tree algorithm. We cannot give the method here, but can propose a setting in which this method should be investigated:

$$\begin{array}{ccc} \text{Tree automata} & \xleftarrow{\chi} & \text{DFA} \\ \Big\downarrow \mathcal{L}_{\mathcal{A}} & & \Big\downarrow \mathcal{L}_{\mathcal{B}} \\ \text{Tree languages} & \xrightarrow{Id} & \text{string languages} \\ \Big\uparrow yield & & \Big\uparrow yield \\ \text{informant} & \xrightarrow{\xi,\overline{\xi}} & \text{informant} \end{array} \tag{4.10}$$

5. Conclusions

In this paper we have not presented new grammatical inference results, but there are times where results are not necessarily what a field needs. Grammatical inference requires a better understanding of what the different definitions that have appeared over the years are about. We hope that what is presented in this paper may contribute to go in that direction.

The next steps should consist in revisiting other results in this setting and perhaps answer some of the important questions for which we do not have an answer yet :

- Trees: in what case can results on string automata and grammars be derived to the case of tree grammars and automata?
- polynomial learning: can we compare the definitions and show some implications?
- Noise: learning from noisy data can be presented in this setting: just introduce the notion of noisy presentations. Does this allow us to derive new results?

Acknowledgements

Thanks to Rémi Eyraud, Jose Oncina and Jean-Christophe Janodet for different discussions that led to the diagram representations. Work with Henning Fernau on polynomial learning is where the ideas in that section come from.

References

[CO94a] R. C. Carrasco and J. Oncina, editors. *Grammatical Inference and Applications, Proceedings of ICGI '94*, number 862 in LNAI, Berlin, Heidelberg, 1994. Springer-Verlag.

[CO94b] R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In ICGI'94 [CO94a], pages 139–150.

[Den01] F. Denis. Learning regular languages from simple positive examples. *Machine Learning Journal*, 44(1):37–66, 2001.

[DG97] F. Denis and R. Gilleron. PAC learning under helpful distributions. In M. Li and A. Maruoka, editors, *Proceedings of ALT '97*, volume 1316 of *LNCS*, pages 132–145, Berlin, Heidelberg, 1997. Springer-Verlag.

[dlH97] C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning Journal*, 27:125–138, 1997.

[dlH05] C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38:1332–1348, 2005.

[dlHJ04] C. de la Higuera and J-C. Janodet. Inference of ω-languages from prefixes. *Theoretical Computer Science*, 313(2):295–312, 2004.

[FdlH04] H. Fernau and C. de la Higuera. Grammar induction: an invitation to formal language theorists. *Grammars*, 7:45–55, 2004.

[Fer02] H. Fernau. Learning tree languages from text. In J. Kivinen and R. H. Sloan, editors, *Proceedings of COLT 2002*, number 2375 in LNAI, pages 153–168, Berlin, Heidelberg, 2002. Springer-Verlag.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[Gol78] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[KMT97] T. Koshiba, E. Mäkinen, and Y. Takada. Learning deterministic even linear languages from positive examples. *Theoretical Computer Science*, 185(1):63–79, 1997.

[KS94] T. Knuutila and M. Steinby. Inference of tree languages from a finite sample: an algebraic approach. *Theoretical Computer Science*, 129:337–367, 1994.

[LPP98] K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In V. Honavar and G. Slutski, editors, *Grammatical Inference, Proceedings of ICGI '98*, number 1433 in LNAI, pages 1–12, Berlin, Heidelberg, 1998. Springer-Verlag.

[Mäk96] E. Mäkinen. A note on the grammatical inference problem for even linear languages. *Fundamenta Informaticae*, 25(2):175–182, 1996.

[PH00] R. J. Parekh and V. Honavar. On the relationship between models for learning in helpful environments. In A. de Oliveira, editor, *Grammatical Inference: Algorithms and Applications, Proceedings of ICGI '00*, volume 1891 of *LNAI*, pages 207–220, Berlin, Heidelberg, 2000. Springer-Verlag.

[Pit89] L. Pitt. Inductive inference, DFA's, and computational complexity. In *Analogical and Inductive Inference*, number 397 in LNAI, pages 18–44. Springer-Verlag, Berlin, Heidelberg, 1989.

[PW88] L. Pitt and M. Warmuth. Reductions among prediction problems: on the difficulty of predicting automata. In *3rd Conference on Structure in Complexity Theory*, pages 60–69, 1988.

[Sak87] Y. Sakakibara. Inferring parsers of context-free languages from structural examples. Technical Report 81, Fujitsu Limited, International Institute for Advanced Study of Social Information Science, Numazu, Japan, 1987.

[Sak90] Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, 76:223–242, 1990.

[Sak97] Y. Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185:15–45, 1997.

[SG94] J. M. Sempere and P. García. A characterisation of even linear languages and its application to the learning problem. In Carrasco and Oncina [CO94a], pages 38–44.

[Tak88] Y. Takada. Grammatical inference for even linear languages based on control sets. *Information Processing Letters*, 28(4):193–199, 1988.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142, 1984.

[Yok03] T. Yokomori. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 1(298):179–206, 2003.

[Zeu03] T. Zeugmann. Can learning in the limit be done efficiently? In R. Gavaldà, K. Jantke, and E. Takimoto, editors, *ALT*, number 2842 in LNCS, pages 17–38, Berlin, Heidelberg, 2003. Springer-Verlag.