Unambiguous automata inference by means of state-merging methods

François Coste, Daniel Fredouille

IRISA-INRIA,
Campus Universitaire de Beaulieu
35042 Rennes Cedex
{francois.coste|daniel.fredouille}@irisa.fr
http://www.irisa.fr/

Abstract We consider inference of automata from given data. A classical problem is to find the smallest compatible automaton, i.e. the smallest automaton accepting all examples and rejecting all counter-examples. We study unambiguous automata (UFA) inference, an intermediate framework between the hard nondeterministic automata (NFA) inference and the well known deterministic automata (DFA) inference. The search space for UFA inference is described and original theoretical results on both the DFA and the UFA inference search space are given. An algorithm for UFA inference is proposed and experimental results on a benchmark with both deterministic and nondeterministic targets are provided showing that UFA inference outperforms DFA inference.

Introduction

Motivations: We consider inference of *nondeterministic automata* (NFA) from given data. A classical problem is to find the smallest compatible automaton, i.e. the smallest automaton accepting all examples and rejecting all counter-examples. When automata are *deterministic* (DFA), the problem has been extensively studied and is NP-complete [Gol78,PW89]. However, if enough examples and counter-examples are provided, polynomial inference algorithms using *state-merging method* perform well [OG92,Lan92,LPP98].

NFA inference is known to be harder than DFA inference [Hig97]. But, in the Occam's razor paradigm, it is worth noticing that NFA may be exponentially smaller than DFA. NFA also represent some structures - like "gaps" in genomic - more explicitly than DFA, and therefore are better suited to be interpreted by an expert of the application domain. Experimental results of [CF00,DLT01] show that inferring regular languages using classes of automata containing nondeterministic representations is a promising approach.

Nevertheless, all the complexity of NFA is not necessary to take advantage of nondeterminism. We propose to study the inference of an intermediate class of automata, the *unambiguous automata* (UFA). As we will show in this article, inferring UFA enables to introduce a reasonable amount of nondeterminism while keeping some advantages of the DFA representation.

To tackle UFA inference, we consider this problem as a search of a particular UFA in a space of NFA. We propose to adapt states-merging methods - which have been proven successful for DFA inference - to realize UFA inference. We first describe the search space for NFA inference in the state-merging framework by revisiting results of [DMV94] (section 1). Then, we propose operators allowing to explore this search space by considering only unambiguous automata (section 2). Thanks to operators defined in section 2, different strategies for exploring the search space can be applied. We have implemented a greedy strategy together with a heuristic inspired from classical DFA inference algorithms. This algorithm is shown to perform better on a benchmark of the domain than the original DFA algorithm. A comparison with the DeLeTe2 algorithm which infer residual finite state automata (RFSA) [DLT01] showing that each algorithm is more adapted to different subparts of the benchmark is also given.

Definitions and Notations: We denote by $|E|$ the cardinality of a set E. A *partition* of a set E is a set of subsets of E such that the intersection of each pair of subsets is empty and the union of all subsets is E. An element of a partition is called a *block*. Let Σ be a finite alphabet, we denote by Σ^* the set of words on Σ, by ϵ the empty word and by $|u|$ the length of a word u of Σ^*.

Definition 1. *A* nondeterministic automaton*, or NFA, is a 5-tuple* $\langle \Sigma, Q, I, \delta, F \rangle$ *where* Σ *is the* input alphabet*,* Q *is a finite set of states,* $I \subseteq Q$ *is the set of* initial states*,* δ *is the* transition mapping *defined from* $Q \times \Sigma$ *to* 2^Q*,* F *is the set of* final states*. The* δ *function is classically extended to words by:* $\forall q \in Q,\ \forall a \in \Sigma,\ \forall w \in \Sigma^*,\ \delta(q, \epsilon) = \{q\},\ \delta(q, aw) = \bigcup_{q' \in \delta(q,a)} \delta(q', w)$*. A tuple* $\langle q, a, q' \rangle$ *with* $q' \in \delta(q, a)$ *is called a* transition

The regular language recognized by an automaton A is $L(A) = \{w \in \Sigma^* \mid \exists q_i \in I,\ \delta(q_i, w) \cap F \neq \emptyset\}$. We associate two languages to each state q of an automaton, its *prefix language* which is the set of words w such that $q \in \delta(I, w)$; and its *suffix language* which is the set of words w such that $\delta(q, w) \cap F \neq \emptyset$. NFA are considered *trimmed* (i.e. no state has an empty prefix or suffix language). The size of a NFA A is defined as its number of states.

A *deterministic finite automaton*, or DFA is a NFA $\langle \Sigma, Q, I, \delta, F \rangle$ such that: $|I| = 1$ and $\forall q \in Q, \forall a \in \Sigma, |\delta(q, a)| \leq 1$. Some particular DFA can be defined. The *canonical automaton* of the regular language L, denoted by $A(L)$, is the unique minimal DFA accepting L. The *universal automaton*, $UA(\Sigma)$ or more simple UA, is the canonical automaton $A(\Sigma^*)$ accepting all words on Σ (figure 2).

An *acceptance* for a word $w \in \Sigma^*$ - with $w = a_1 \dots a_{|w|}$ - in an automaton $A = \langle \Sigma, Q, I, \delta, F \rangle$ is a sequence $(q_0, \dots, q_{|w|})_w$ of $|w|+1$ states such that $q_0 \in I$, $\forall i \in [1, |w|]$, $q_i \in \delta(q_{i-1}, a_i)$, $q_{|w|} \in F$. Transitions $\langle q_{i-1}, a_i, q_i \rangle$ are said *reached* by the acceptance. The *ambiguity degree* of an automaton A is the maximum number of acceptances that exist in A for a word of Σ^*. An *unambiguous finite automaton*, or UFA, is a NFA with an ambiguity degree inferior or equal to one (figure 1). When a NFA is not a UFA, we say it is ambiguous.

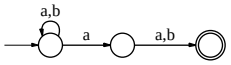

Figure1. An example of UFA, representing the language $\Sigma^* a \Sigma$

Notice that the class of DFA is included in the class of UFA. DFA and UFA are obviously included in the class of NFA. NFA, UFA and DFA can represent any regular language.

This document includes theorems for which only hints of proofs are provided; for complete proofs the reader can consult [CF03].

1 Search space for automata inference

The search space we want to explore is the restriction to UFA of the search space for NFA inference by means of state-merging methods. This first section presents and revisits the NFA search space described by [DMV94,Dup96]. Next section will study its restriction to UFA.

In the framework of inference from given data [Gol78], we try to infer languages from a *training sample*. In this paper, we define a training sample of a language L to be a couple of finite sets $\langle S_+, S_- \rangle$, where $S_+ \subseteq L$ is called the *positive training sample* and $S_- \subseteq \Sigma^* \setminus L$ is called the *negative training sample*. For the sake of clarity we consider only the positive training sample in sections 1 and 2. However, results of these sections can be easily extended to consider *unbiased inference* [AS95,Cos99], i.e. to consider symmetrically the two parts of the sample.

An underlying assumption for inference of an automaton is that the positive training sample is "representative enough" of the language to learn. This can be formalized by the notion of *structural completeness* which intuitively means that all constituents of target automaton are useful for the sample recognition. More formally:

Definition 2. *A positive training sample S_+ is said to be* structurally complete *with respect to an automaton A iff there exists an acceptance set $\mathcal{A}$ containing exactly one acceptance for each word of S_+ such that:*
- every transition of A is reached by an acceptance of $\mathcal{A}$,
- every initial state of A is the first state of an acceptance of $\mathcal{A}$,
- every final state of A is the last state of an acceptance of $\mathcal{A}$.

Structural completeness hypothesis enables one to restrict the search space to a finite ordered set[1] of automata with a top and a bottom element. The top element of this set is the universal automaton and the bottom element is the *Maximal Canonical Automaton* (figure 2).

Definition 3. *The* maximal canonical automaton *with respect to a positive sample $S_+ = \{w_1, \ldots, w_{|S_+|}\}$, denoted by $MCA(S_+)$ or more simply MCA, is the union of canonical automata $A(\{w_i\})$ for each word of the sample ($i \in [1, |S_+|]$).*

[1] Vocabulary of this paper concerning ordered sets is taken from [DP90].

The MCA realizes a learning by rote of the positive sample. Inference in the state-merging framework consists in generalizing the language recognized by the MCA by *merging* its states (or unifying them, see [DMV94] for a constructive definition). Given an automaton A, and a partition π on states of A, we can construct an automaton A/π. A/π is constructed by merging the states of A being in a same block of the partition . We say that A/π *is derived* from A with respect to partition π.

We denote the set of partitions on the states of MCA by $\boldsymbol{P}(MCA)$. An order on the partitions of $\boldsymbol{P}(MCA)$ can be defined as follows: we say that a partition π_2 *directly derives* from partition π_1, denoted by $\pi_1 \prec \pi_2$, if π_2 can be constructed from π_1 as follows: $\exists b_1, b_2, \in \pi_1, b_1 \neq b_2, \pi_2 = (\pi_1 \setminus \{b_1, b_2\}) \cup \{b_1 \cup b_2\}$. The transitive closure of $\prec$ is denoted by $\prec^*$. $\boldsymbol{P}(MCA)$ is a complete lattice of partitions under the $\prec^*$ order relation.

The relation $\prec^*$ between partitions is extended to the relation $\prec_A^*$ between automata as follows: $A_1 \prec_A A_2 \Leftrightarrow \exists \pi_1,\ \pi_2 \in \boldsymbol{P}(MCA),\ \pi_1 \prec \pi_2,\ A_1 = MCA/\pi_1,\ A_2 = MCA/\pi_2$. The transitive closure of $\prec_A$, denoted by $\prec_A^*$, defines an order relation on automata. An automaton A inferior in the sense of $\prec_A^*$ to an automaton A' is said to be *derivable* from A'.

Let $\boldsymbol{A}_{\mathrm{NFA}}(MCA)$, or more simply $\boldsymbol{A}(MCA)$, denote the set of NFA derivable from MCA. In the following sections, we extend this notation to any classes of automata and any automata, for example $\boldsymbol{A}_{\mathrm{DFA}}(A)$ will denote the set of deterministic automata derived from automaton A. The following theorem holds (illustrated by figure 2).

Theorem 11 *The search space for NFA under the hypothesis of structural completeness of a positive training sample S_+ is $\boldsymbol{A}(MCA(S_+))$.*
Hint of the proof: *The proof is an extension of the proof provided by [DMV94] taking into account our more precise definition of structural completeness and NFA having more than one initial state.*□

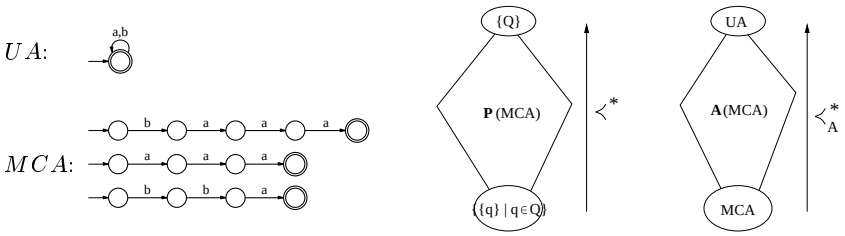

Figure2. Universal automaton (UA), Maximal canonical automaton (MCA) for $S_+ = \{aaa, bba, baaa\}$, $\boldsymbol{P}(MCA)$ and $\boldsymbol{A}(MCA)$.

Let us remark that, as illustrated by figure 3, even if $\boldsymbol{P}(MCA)$ is a lattice of partitions, $\boldsymbol{A}(MCA)$ is not a lattice of automata under $\prec_A^*$ order relation.

This shows clearly a misuse of terms used in the regular grammatical inference community.

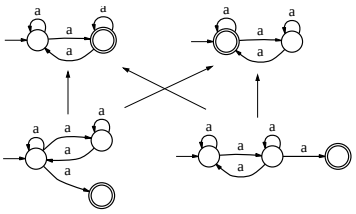

Figure3. We cannot deduce that $\boldsymbol{P}(MCA)$ being a lattice, $\boldsymbol{A}(MCA)$ is also a lattice because an automaton of $\boldsymbol{A}(MCA)$ can be derived from more than one partition of $\boldsymbol{P}(MCA)$. The figure illustrates this point by exhibiting a couple of automata - the two on top - without greatest lower bound under $\prec_A^*$ order relation (relation is represented by arrows).

2 Search space for UFA inference

2.1 From DFA to UFA

The inference search space has often been restricted to DFA, and NFA inference can be considered to be harder than DFA inference (indeed, NFA do not have a canonical form and are not polynomially learnable from given data whereas DFA are [Hig97]). We show, in this section, that properties known for the restriction of the search space to DFA are also valid for the restriction of the search space to UFA.

The bottom element of the search space of DFA is the prefix tree acceptor denoted by $PTA(S_+)$ or more simply PTA [DMV94] and is obtained by determinisation of MCA. For UFA, MCA being unambiguous, the bottom element stays MCA like for the search space of NFA.

A first link between DFA and UFA search space is given by theorem 21:

Theorem 21 *Let A be a UFA and S_+ a positive training sample structurally complete with respect to A. There exists one and only one partition π in $\boldsymbol{P}(MCA)$ such that $A = MCA/\pi$.*

Hint of the proof: *There exists a partition π such that $A = MCA/\pi$ (entailed by UFA $\subset$ NFA and theorem 11). We have to show that this partition is unique.*

For each word $w \in S_+$, A being a UFA, there exists only one *acceptance acc_1 for this word in A. The acceptance acc_2 for w in MCA defines a mapping function from states of MCA to states of A, every ith state of acc_2 being mapped to ith state of acc_1. This mapping defines for every state of MCA the unique block of partition it can be in, and therefore the unique possible partition.* □

This property was known for DFA and theorem 21 replaces it in the more general framework of UFA. From theorem 21 and as illustrated by figure 4, UFA have the advantage over NFA of being represented by only one partition. DFA have the advantage over NFA and UFA of having a canonical form.

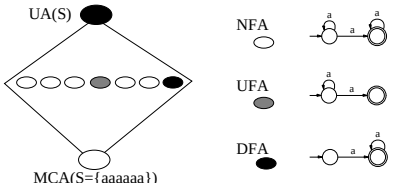

Figure4. The figure shows partitions of minimum size 2 representing the same language $L = a^+$ in $\boldsymbol{P}(MCA)$ with $S_+ = \langle\{aaaaaa\}\rangle$. When looking at derived automata from these partitions, we count a unique DFA, two UFA and 7 NFA, 5 being isomorphic.

To explore the search space of UFA, we could consider only state-merging from the MCA leading to other UFA. Indeed, we show in [CF03] - extending a theorem of [Dup96] for DFA - that all UFA of the search space can be reached from the MCA by a sequence of merge considering only UFA.

Nevertheless, we focus in this paper on another state-merging operator for UFA inference called *unambiguous merging*. This operator can be considered as the counterpart of the *deterministic merging* operator which has been extensively used in DFA inference algorithms (e.g. [OG92,LPP98]).

2.2 From deterministic merging to unambiguous merging

The deterministic merging operator is based on a procedure called *merging for determinisation*. After introducing a few definitions and a property, we present the dual *merging for disambiguisation* procedure and then the unambiguous merging operator[2].

Two states q_1 and q_2 are said to be in *common prefix relation* (resp. in *common suffix relation*) if the intersection of their prefix languages (resp. their suffix languages) is not empty. Two states q_1 and q_2 simultaneously in common prefix relation and in common suffix relation are said in *parallel acceptance relation*, denoted by $q_1 \parallel q_2$.

Property 21 *An automaton $A = \langle \Sigma, Q, I, \delta, F\rangle$ is ambiguous iff it has two different states in parallel acceptance relation.*
Hint of the proof: *For every couple of different states $\langle q_1, q_2\rangle$ in parallel acceptance relation, there exists a word u common to their prefix languages and a word v common to their suffix languages. This is equivalent to the existence of two acceptances for the word uv, the first reaching q_1 and the second q_2 by the word u.* □

The sets of common prefix and common suffix relations can be computed and incrementally maintained after each merge [CF00]. Common suffix relation is presented here for the first time but can be maintained exactly like incompatibility relation presented in [CF00]. Parallel acceptance relation is directly deduced from the previous.

[2] Formal properties of the deterministic merging operator have never been formalized, this section provides both its extension to the unambiguous case and a formalization of this operator properties.

Algorithm 1 Merging for disambiguisation of $A = \langle \Sigma, Q, I, \delta, F \rangle$

1: **while** $\exists q_1, q_2 \in Q, q_1 \parallel q_2,\ q_1 \neq q_2$ **do**
2: $\quad A \leftarrow merge(A, q_1, q_2)$

By using these relations, we can now define the *merging for disambiguisation* procedure (algorithm 1). This procedure consists in merging pair of states in parallel acceptance relation. Each merge possibly entailing new relations, the procedure stops merging when no more couple of states are in parallel acceptance relation.

Compared to merging for determinisation, which can be defined as merging of all states in common prefix relation, merging for disambiguisation merges all states both in common prefix relation and common suffix relation. Therefore merging for disambiguisation realizes only a subset of the merging needed by merging for determinisation and allows a finer exploration of the search space.

Merging for disambiguisation (resp. merging for determinisation) does all necessary and sufficient merging to reach the "closest" UFA (resp. DFA) derived from a NFA. We formalize this fact for UFA by property 22:

Property 22 *Let A be a NFA, and A' the UFA obtained by merging for disambiguisation of A. Then every UFA of $\boldsymbol{A}_{UFA}(A)$ is in $\boldsymbol{A}_{UFA}(A')$.*
Hint of the proof: *Let $A = A_1, A_2, \ldots, A_n = A'$ be the sequence of automata created by the merging for disambiguisation procedure. From property 21 we can show that there is no UFA in $\boldsymbol{A}_{UFA}(A_i) - \boldsymbol{A}_{UFA}(A_{i+1})$. The theorem can then be proven by induction on $i \in [1, n[$.* □

Let us remark that property 22 entails that whatever the order of merging realized by merging for disambiguisation (or merging for determinisation), these merging always lead to the same automaton.

We now introduce the operator of *unambiguous merging* (resp. *deterministic merging*. Unambiguous (resp. deterministic) merging consists in merging two states of an automaton and applying merging for disambiguisation (resp. determinisation) to the resulting automaton.

We will denote by $A_1 \prec_{dis} A_2$ (resp. $A_1 \prec_{det} A_2$) if automaton A_2 can be obtained by applying one unambiguous (resp. deterministic) merging on A_1. Relations $\prec^*_{dis}$ and $\prec^*_{det}$ will denote respectively the transitive closure of $\prec_{dis}$ and $\prec_{det}$.

As shown by theorem 22, every UFA derived from a given UFA A - i.e. automata of $\boldsymbol{A}_{\mathrm{UFA}}(A)$ - can be reached by a sequence of unambiguous merging from A. More formally: $\forall A_1, A_2 \in \mathrm{UFA},\ A_1 \in \boldsymbol{A}(A_2) \Rightarrow A_2 \prec^*_{dis} A_1$.

Theorem 22 *Let A be a UFA, for all UFA A' of $\boldsymbol{A}_{UFA}(A)$, there exists a sequence of automata $A_0, \ldots, A_n$ such that $A_0 \prec_{dis} A_1 \prec_{dis} \ldots \prec_{dis} A_n$ and $A = A_0$, $A' = A_n$.*
Hint of the proof: *This can be proven as a consequence of property 22.* □

The counterpart of this theorem for DFA and deterministic merging is also true, i.e.: $\forall A_1, A_2 \in \mathrm{DFA},\ A_1 \in \boldsymbol{A}(A_2) \Rightarrow A_2 \prec^*_{det} A_1$.

Section 3 presents the use of the operator of unambiguous merging to explore the space of UFA.

3 Experimental comparison

3.1 Algorithms and benchmarks

Section 2 detailed both the search space for UFA and operators available to explore it. Different strategies can be applied when using these operators. Our experimental results are based on a greedy search - presented by algorithm 2 - which is the classical approach applied for DFA inference (e.g. [OG92,LPP98]). The `choose-two-states` method of this algorithm represents the heuristic, i.e.

Algorithm 2 Principle of greedy state-merging algorithms.

Function `greedy-state-merging-algorithm`$(S = \langle S_+, S_- \rangle)$
$A \leftarrow MCA(S_+)$ (or $A \leftarrow PTA(S_+)$ for DFA inference)
while `choose-two-states`(A, q_1, q_2) **do**
  $A' \leftarrow$ `state-merging`(A, q_1, q_2)
  if A' is compatible with S_- **then** $A \leftarrow A'$
return A

the order used to try state-mergings. The `state-merging` method depends on which class of automata is inferred: we use deterministic merging and unambiguous merging for respectively DFA and UFA inference.

We compared the best heuristic known for DFA inference, called EDSM [LPP98], a hill-climbing strategy for UFA (detailed in subsection 3.2) and inference of RFSA (Residual Finite State Automata) with the DeLeTe2 algorithm [DLT01]. The experimental comparison of DFA, UFA and RFSA inference is based on benchmarks provided in [DLT00,DLT01]. These benchmarks contain training and testing sets for languages generated using different methods: construction of random DFA, random NFA and random regular expressions. We added to this benchmark languages generated by a UFA generator.

The UFA generator takes five parameters: a number of states N, a probability p_i for a state to be initial, a probability p_f to be final, an alphabet Σ and a number of transition t. After constructing the N states of the generated automaton A, each state is set initial with probability p_i; then each state is set final with probability p_f, except if this entails that A became ambiguous; and then, we try t times to insert a new transition $\langle q_1, a, q_2 \rangle$ between states of A (q_1, q_2 and a being chosen uniformly in $Q \times \Sigma \times Q$), this transition insertion is rejected if it entails A to be ambiguous. UFA of the benchmark have been generated with parameters: $N = 20$, $p_i = 0.3$, $p_f = 0.3$, $t = 60$ and $\Sigma = \{0, 1\}$.

Training and testing samples are generated with the method used in [DLT01]: for each word w of the sample, its length is chosen uniformly in $[0, 29]$ and w is

chosen uniformly between words of this length. w is labeled by '+' if it is in the generated language and by '-' otherwise. 30 languages are generated for each size of training sample (50, 100, 150 or 200). The generated language is kept only if the corresponding training sample contains at most 80%, and at least 20%, of words labeled by '+'. Testing sample of each language contains 1000 examples and counter-examples.

3.2 Heuristics for UFA and DFA inference

Heuristic: For UFA inference, we use a hill-climbing heuristic, i.e. we choose the unambiguous merging leading to the smallest automaton (which is equivalent to the unambiguous merging entailing the most state-mergings). For DFA inference we used the EDSM heuristic (for Evidence Driven State-Merging). This heuristic has been proposed by [LPP98] and won the grammatical inference competition Abbadingo [Abb98]. EDSM chooses the deterministic merging that entails the most number of merge between final states by merging for determinisation.

In practice, these two heuristics need the computation respectively of each possible deterministic mergings and unambiguous mergings of two states. A score is given to each state pair (consisting in the number of merged states for UFA, and of the number of merged final states for DFA), and the states pair with the best score is choosen.

Even if *a priori* different, these two heuristics may be seen as closely related to each other with respect to the notion of acceptance.

Indeed, the choice of counting merge between final states instead of merge between every states in EDSM can be seen as a measure of the "size" of the intersection of suffixes languages of the two scored states. The prefix languages of states of a DFA being disjoint, this measure can also be seen as a measure of the number of acceptances being unified by the state-merging.

This idea is also present in the hill-climbing heuristic for UFAs. Each state-merging computed by the merging for disambiguisation procedure is due to the existence of two acceptances for a word. These state-mergings therefore enable to unify acceptances, and the number of merged states can can be considered as a measure of the number of unified acceptances.

Use of counter-examples: Counter-examples may be used in different ways. We can consider biased inference which consists in generalizing examples and stopping the generalization with the counter-examples [DMV94]. We can also consider unbiased inference [AS95,Cos99], which consider that the couple S_+ and S_- are examples respectively of the target language L and of $L_- = \Sigma^* \setminus L$. In this context, the two languages L and L_- are inferred by generalizing simultaneously S_+ and S_- using a *classifier automaton*. Generalization is stopped with the constraint $L \cap L_- = \emptyset$ (instead of $L \cap S_- = \emptyset$).

The DeLeTe2 algorithm works in the biased inference paradigm. The EDSM heuristic has been presented in [LPP98] in the unbiased inference paradigm but can also be applied to biased inference (as presented in the previous paragraph). In this paper we compare the use of the EDSM heuristic for DFA inference both

in the unbiased and biased paradigm, hill-climbing for UFA inference both in the unbiased and biased paradigm and DeLeTe2. Corresponding algorithms will be denoted respectively D_{edsm}, Db_{edsm}, U_{hc}, Ub_{hc}, and DLT2. We will also consider the majority vote denoted by MAJ.

3.3 Inference results

The evaluation consists in scoring each algorithm for each benchmark thanks to its average recognition level on the test sets (figure 5). Like [DLT01], we also compare algorithms thanks to matches (noted algo1-algo2 in figure 6). A match consists in counting the number of time an algorithm is better than another (in term of recognition rate), and we count a tie when the difference is not significant (using the Mac Nemar test [Die98]). Those matches are noted as tuple: wonByAlgo1,nbTie,wonByAlgo2. Since experiments have been made on different machines with different implementations, comparing running time is difficult. Nevertheless, in these experimentations, our algorithm Ub_{hc} seems to be 2 orders of magnitude slower than DLT2, which is slower than Db_{edsm}. The symbol * in the cell designates when some experiments have not finished due to the time limit of 100h on 750 Mhz cpu (more precisely, two runs on the 480 runs of the algorithm U_{hc} did not finished on time).

Generator	NFA				Regular Expressions			
Sample size	50	100	150	200	50	100	150	200
MAJ	69.0	66.2	65.7	67.8	64.7	66.7	62.3	62.8
Db_{edsm}	67.1	70.0	73.1	73.3	**83.7**	85.5	91.8	92.1
D_{edsm}	67.0	67.6	70.7	71.0	79.5	81.7	89.7	93.1
Ub_{hc}	**70.4**	70.8	74.0	73.1	75.8	82.9	91.7	91.2
U_{hc}	67.0	71.2	73.7	71.3	76.0	81.5	88.8	91.0
DLT2	69.8	**74.8**	**77.1**	**79.4**	81.7	**91.7**	**92.3**	**95.9**
Generator	UFA				DFA			
Sample size	50	100	150	200	50	100	150	200
MAJ	83.8	82.1	81.4	81.9	70.7	71.0	72.5	73.8
Db_{edsm}	89.2	91.1	**94.3**	93.2	69.1	**73.3**	74.8	76.3
D_{edsm}	79.1	81.0	89.6	90.2	65.7	68.3	70.4	74.7
Ub_{hc}	**90.7**	**91.9**	**94.2**	**93.8**	70.4	**73.4**	74.5	77.5
U_{hc}	89.7	89.8	92.5*	91.6	**71.1**	72.9	**75.9**	**77.8***
DLT2	88.6	90.4	91.9	92.7	61.9	65.1	68.3	70.7

Figure5. Average recognition level on test sets.

We remark that algorithms based on UFA inference have better recognition scores on the benchmark than the original DFA algorithm (Ub_{hc} won 170 times against 119 for Db_{edsm}). This result was hoped for NFA, regular expressions and UFA benchmarks. More surprisingly, UFA inference performs better than

Generator	NFA				Regular Expressions			
Sample size	50	100	150	200	50	100	150	200
D_{edsm}-Db_{edsm}	10,8,12	5,13,12	5,10,15	6,12,12	7,12,11	5,13,12	5,16,9	3,23,4
Ub_{hc}-Db_{edsm}	16,9,5	11,12,7	12,10,8	12,10,8	5,12,13	11,6,13	10,14,6	6,10,14
U_{hc}-Ub_{hc}	4,11,15	4,20,6	5,18,7	6,11,13	9,13,8	9,5,16	5,13,12	6,13,11
DLT2-Ub_{hc}	9,6,15	12,13,5	14,10,6	16,11,3	15,8,7	17,8,5	10,10,10	18,9,3
Generator	UFA				DFA			
Sample size	50	100	150	200	50	100	150	200
D_{edsm}-Db_{edsm}	1,4,25	1,6,23	2,13,15	6,6,18	4,10,16	2,7,21	2,12,16	7,10,13
Ub_{hc}-Db_{edsm}	17,6,7	13,14,3	7,15,8	13,9,8	13,13,4	11,14,5	8,16,6	15,11,4
U_{hc}-Ub_{hc}	5,15,10	4,14,12	3,15,11*	3,12,15	8,16,6	4,21,5	10,18,2	10,16,3*
DLT2-Ub_{hc}	6,11,13	4,14,12	3,13,14	6,13,11	1,7,22	1,2,27	2,6,22	2,5,23

Figure6. Matches between algorithms.

DFA inference on the DFA benchmark. We explain this by considering that choosing the wrong unambiguous merging at a step of the algorithm causes less constraints on future mergings than choosing a wrong deterministic merging. A wrong unambiguous merging can therefore be "partly corrected" by future mergings.

We can also remark that the biased versions of the algorithms are much better than the unbiased one on benchmarks for which L and $\Sigma^* \setminus L$ are not generated symmetrically (Ub_{hc} won 135 times against 63 for U_{hc}, and Db_{edsm} won 168 times against 56 for D_{edsm} on these benchmarks).

When comparing UFA and RFSA inference, tables of figures 5 and 6 show that UFA inference and RFSA inference are each better suited to different subpart of the benchmark: Ub_{hc} performs better than DLT2 on UFA and DFA based benchmarks. However, DLT2 is the best algorithm on benchmarks based on NFA and regular expressions. Thus, we suppose that the class of UFA is "closer" to DFA than the class of RFSA is "close" to NFA and regular expressions.

Conclusion

We have revisited the search space for automata inference. We formalized properties known on the DFA search space, and extended them to the UFA search space. This work leaded to the extension of the well known deterministic merging operator to the unambiguous merging operator, which seems very promising for automata inference. Indeed, this new operator allows us to propose a heuristic closely related to EDSM [LPP98]. The use of the unambiguous merging operator together with this heuristic has been shown to perform well on benchmarks of the domain.

Deeper studies on the deterministic merging operator and on the unambiguous merging operator have shown that these operators give a lattice structure to the search space [CF03]. Therefore, practical results presented in this paper use

only part of the available theoretical properties. Integrating these properties in inference algorithms is an open perspective to our research.

References

[Abb98] Abbadingo one, 1998. http://abbadingo.cs.unm.edu/.

[AS95] R. Alquézar and A. Sanfeliu. Incremental grammatical inference from positive and negative data using unbiased finite state automata. In Shape, Structure and Pattern Recognition, *Proc. Int. Workshop on Structural and Syntactic Pattern Recognition, SSPR'94, Nahariya (Israel)*, pages 291–300, 1995.

[CF00] F. Coste and D. Fredouille. Efficient ambiguity detection in C-NFA, a step toward inference of non deterministic automata. *Grammatical Inference: Algorithms and Applications, ICGI'00*, pages 25–38, 2000.

[CF03] F. Coste and D. Fredouille. What is the search space for the inference of nondeterministic, unambiguous and deterministic automata ? Technical report, IRISA, to appear, download: http://www.irisa.fr/prive/dfredoui/down/report.ps.gz, 2003.

[Cos99] F. Coste. State merging inference of finite state classifiers. Technical Report INRIA/RR-3695, IRISA, September 1999.

[Die98] Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[DLT00] F. Denis, A. Lemay, and A. Terlutte. Learning regular languages using non deterministic finite automate. *Grammatical Inference: Algorithms and Applications, ICGI'00*, 2000.

[DLT01] F. Denis, A. Lemay, and A. Terlutte. Learning regular languages using RFSA. In *Proceedings of the 12th International Conference on Algorithmic Learning Theory, ALT'01*, pages 348–363, 2001.

[DMV94] P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference ? *Grammatical inference and Applications, ICGI'94*, pages 25–37, 1994. Springer Verlag.

[DP90] B. Davey and A. Priesley. *Introduction to lattices and order*. Cambridge mathematical textbooks, 1990.

[Dup96] P. Dupont. *Utilisation et apprentissage de modèles de langages pour la reconnaissance de la parole continue*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, 1996.

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

[Hig97] C. Higuera (de la). Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138, 1997.

[Lan92] K. J. Lang. Random dfa's can be approximately learned from sparse uniform examples. *5th ACM workshop on Computation Learning Theorie*, pages 45 – 52, 1992.

[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. *Lecture Notes in Computer Science*, 1433:1–12, 1998.

[OG92] J. Oncina and P. García. Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis*, pages 49 – 61, 1992.

[PW89] L. Pitt and M. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. In *21st ACM Symposium on Theory of Computing*, pages 421–444, 1989.