What is the search space for the inference of non deterministic, unambiguous and deterministic automata ?

F. Coste, D. Fredouille

N° 4907

Juillet 2003

THÈME 1

ISSN 0249-6399

What is the search space for the inference of non deterministic, unambiguous and deterministic automata ?

F. Coste, D. Fredouille

Thème 1 — Réseaux et systèmes
Projet Symbiose

Rapport de recherche n°4907 — Juillet 2003 — 41 pages

Abstract: A classical problem in inferring automata from data is to find the smallest compatible automaton, i.e. the smallest automaton accepting all examples and rejecting all counter-examples. We are interested in Non-deterministic Finite Automata (NFA) inference which is better suited to Occam's razor optimization principle than Deterministic Finite Automata (DFA). We revisit results of [DMV94] on the search space in the state-merging framework, focusing on nondeterminism. We introduce unambiguous automata (UFA) inference, an intermediate between the difficult NFA inference framework and the weaker DFA inference framework. The UFA search space, though containing nondeterministic automata, shares many properties with the DFA search space. Our conclusion is that the search spaces of UFA and DFA are lattices when using adequate operators.

Key-words: Grammatical inference, deterministic finite automata (DFA), non deterministic finite automata (NFA), unambiguous finite automata (UFA), lattice, state-merging inference

Quel est l'espace de recherche pour l'inférence d'automates non déterministes, non ambigus et déterministes ?

Résumé : Lors de l'inférence d'automates à partir de données fixées, un problème classique est de trouver le plus petit automate compatible, i.e. le plus petit automate acceptant tous les exemples et rejetant tous les contre-exemples. Nous sommes intéressés par l'inférence d'automates non déterministes (NFA) qui est plus adaptée au principe du razoir d'Occam que les automates déterministes (DFA). Nous revisitons les résultats de [DMV94] sur l'espace de recherche, en détaillant les propriétés liées au non déterminisme. Nous introduisons l'inférence d'automates non ambigus (UFAs), un intérmédiaire entre le cadre de travail difficile de l'inférence de NFAs et celui bien connu de l'inférence de DFAs. L'espace de recherche pour les UFAs, tout en contenant des automates non déterministes, partage de nombreuses propriétés avec l'espace de recherche pour les DFAs. Nous montrons que ces deux espaces de recherche s'organisent en treillis si l'on utilise des opérateurs de parcours adéquats.

Mots-clés : Inférence grammaticale, automate fini déterministe (DFA), automate fini non déterministe (NFA), automate fini non ambigu (UFA), treillis, inférence par fusion d'états

Contents

1 Definitions & Notations **4**

2 Search space for automata inference **6**
- 2.1 Sample 6
- 2.2 Structural completeness hypothesis 7
- 2.3 Search space under structural completeness hypothesis 8
- 2.4 Extracting automata from the search space 10
  - 2.4.1 Order relation on partitions 11
  - 2.4.2 Visiting the search space 13

3 Search space for UFA and DFA **14**
- 3.1 Automata of the search space 14
- 3.2 Structure of UFA and DFA search space under simple merge 16
- 3.3 From ordered sets to lattices 17
  - 3.3.1 Merging for determinisation and disambiguisation 17
  - 3.3.2 Order relation based on merging for determinisation/disambiguisation 21
  - 3.3.3 Lattice of automata 23

A Theorem 1 : set of structurally complete automata **28**

B State-merging and language generalization **29**

C Relation conservation with derivation **30**

D Theorem 4 : sequence of partitions representing unambiguous automata **31**
- D.1 Algorithm 3 31
- D.2 Algorithm ends 32
- D.3 Invariant : obtained automata are k-ambiguous 34
  - D.3.1 Splitting computed on line 7 34
  - D.3.2 Splitting computed on line 11 37
  - D.3.3 Splitting computed on line 15 38

Introduction

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. When automaton 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 algorithm using *state-merging method* can give good results [OG92, Lan92, LPP98].

NFA inference is a few studied area of grammatical inference, the intersest on this class of automata is recent [Yok94, CF00, DLT00, DLT01]. Indeed, the use of DFAs is often prefered: this class possess a canonical form, and operations on these automata are often polynomial (it is not the case for NFAs). But, in the Occam's razor paradigm, it is worth noticing that NFAs may be exponentially smaller than DFAs. NFAs are also better suited than DFAs to represent some structures, like "gaps" in genomics. We pursue the work on the inference of nondeterministic representations by revisiting results of [DMV94] on the search space in the state-merging framework, focusing on nondeterminism. We exhibit a subclass of NFA, the *unambiguous finite automata* (UFA), sharing with DFA many interesting search space properties while being nondeterministic (section 3). We conclude by showing that the search space for UFA and DFA are lattices when using adequate operators (section 3.3).

1 Definitions & Notations

We denote by $|E|$ the cardinal of a set E. Let Σ be a finite alphabet and Σ^* be the set of words on Σ. Let ϵ denote the empty word and $|u|$ the length of a word u of Σ^*.

Definition 1 (NFA) *A* Non-deterministic Finite state Automaton*, or NFA, is a quintuplet* $\langle \Sigma, Q, I, \delta, F\rangle$ *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*; a tuple* $\langle q_1, a, q_2\rangle$ *such that* $q_2 \in \delta(q_1, a)$ *is called a* transition *and* F *is the set of final states.*

We extend δ *to words and to set of states, i.e. from* $2^Q \times \Sigma^*$ *to* 2^Q *by:* $\forall Q' \subseteq Q$, $\forall a \in \Sigma$, $\forall w \in \Sigma^*$, $\delta(Q', \epsilon) = Q'$, $\delta(Q', a) = \bigcup_{q \in Q'} \delta(q, a)$, $\delta(Q', aw) = \delta(\delta(Q', a), w)$.

The regular language L recognized by an automaton is $L = \{w \in \Sigma^* \mid \delta(I, w) \cap F \neq \emptyset\}$.

NFA in this paper are considered *trimmed* (i.e.: $\forall q \in Q, \exists w \in \Sigma^*, \delta(q, w) \cap F \neq \emptyset$ and $\forall q \in Q, \exists q_0 \in Q_0, \exists w \in \Sigma^*, q \in \delta(q_0, w)$).

Let $|A| = |Q|$ denote the size of an automaton A.

$\mathit{Pref}_A(q)$ will denote the prefix language of a state q of an automaton $A = \langle \Sigma, Q, I, \delta, F\rangle$ defined by $\mathit{Pref}_A(q) = \{w \in \Sigma^* \mid q \in \delta(Q_0, w)\}$. $\mathit{Suff}_A(q)$ will denote the suffix language of q defined by $\mathit{Suff}_A(q) = \{w \in \Sigma^* \mid \delta(q, w) \cap F \neq \emptyset\}$.

Definition 2 (Acceptance) *An* acceptance *for a word* $w \in \Sigma^*$*, where* $w = a_1 \ldots a_{|w|}$*, in an automaton* $A = \langle \Sigma, Q, I, \delta, F\rangle$*, is a sequence* $(q_0, \ldots, q_{|w|})_w$ *of* $|w| + 1$ *states such that*

$q_0 \in Q_0$, $\forall i \in [1, |w|]$, $q_i \in \delta(q_{i-1}, a_i)$, $q_{|w|} \in F$. q_0 *is said to be the* initial state *and* $q_{|w|}$ *is said to be the* final state *of the acceptance.*

Transitions $\rangle q_{i-1}, a_i, q_i \langle$ *are said* reached *by the acceptance. The set of acceptances of a word* w *in an automaton* A *is denoted by* $Acc_A(w)$.

Some properties and subclasses of NFA can be defined as follows:

Definition 3 (DFA) *A* deterministic finite state automaton*, denoted by 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 DFAs can be defined. The *canonical automaton* of a language L, denoted by $A(L)$, is the sole minimal DFA accepting L. The *universal automaton*, $UA(L)$, is the canonical automaton $A(\Sigma^*)$ accepting all words (figure 7).

Definition 4 (Ambiguity[1], UFA) *The* ambiguity degree *of an automaton A, Degree(A), is the maximum number of acceptances that exist for a word, i.e.* $Degree(A) = max_{w \in \Sigma^*}(|Acc_A(w)|)$. *An* unambiguous finite state automaton*, denoted by UFA is a NFA A such that* $Degree(A) \leq 1$ *(figure 1). When a NFA is not a UFA, we say it is ambiguous.*

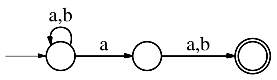


Figure 1: An example of UFA, representing the language $\Sigma^* a \Sigma$ with $\Sigma = \{a, b\}$.

Let us notice that since determinism implies unicity of acceptance for a word of the language, the class of DFA is included in the class of UFA. Obviously these classes are, by definition, both included in the class of NFA. NFA, UFA and DFA can represent any regular language.

In this paper, we consider the search space for NFA. This search space is a partially ordered set. We present here the basic definitions related to this notion (see [DP90] for a detailed presentation).

An *ordered set* is a set E together with a binary relation, named *strict order relation* and denoted by $<$. The strict order relation is non reflexive and transitive. We also consider the reflexive *order relation*, denoted by $\leq$, and defined by : $\forall x, y \in E \;\; : (x \leq y) \Leftrightarrow (x < y) \vee (x = y)$.

Definition 5 (Minimal and maximal, minimum and maximum, bottom and top) *Let* E *be an ordered set and* $E' \subseteq E$. *Then :*

1. $x \in E'$ *is a* minimal element *(resp.* maximal element*) of* E' *if* $\forall y \in E'$, $(y \leq x) \Rightarrow (y = x)$ *(resp.* $(x \leq y) \Rightarrow (y = x)$*).*

[1] Let us notice that in order to conform with the classical definition (g.e. [SH85]), "ambiguity" in this paper refers to ambiguity of parsing and not ambiguity of classification as we did in previous paper [Cos99, CF00].

2. $x \in E'$ is the minimum *(resp.* maximum*) of E' if $\forall y \in E', \ x \leq y$ (resp. $y \leq x$).*

If it exists, the minimum (resp. maximum) element of E is called the bottom *(resp.* top*) of E.*

Some ordered sets have more precise properties, for example *lattices*, for which there exist particular elements for each of their subsets : *meet* and *join*.

Definition 6 (Down-set and up-set, meet and join, lattice) *Let E be an ordered set and $E' \subseteq E$. The* down-set *and the* up-set *of E', denoted by $down(E')$ and $up(E')$ are defined by:*

- $down(E') = \{x \in E \mid \forall y \in E', \ x \leq y\}$,
- $up(E') = \{x \in E \mid \forall y \in E', \ y \leq x\}$.

The meet *of E', denoted by $\bigwedge E'$, is the maximum of $down(E')$ if it exists; the* join *of E', denoted by $\bigvee E'$, is the minimum of $up(E')$ if it exists.*

An ordered set E is a complete lattice *if each $E' \subseteq E$ has a meet and a join.*

In this document, we use the term *lattice* for complete lattice. Indeed, these notions are equivalent when the ordered set is finite.

We consider the lattice of *partitions* of a set. A partition on a set E is a set of non empty subsets of E, called *blocks*; any two of the subsets are disjoint and the union of all the subsets is E. The block B of a partition π on E containing the element $e \in E$ is denoted by $B_\pi(e)$. The set of all partitions of a set E is denoted by $\boldsymbol{P}(E)$. The order relation on $\boldsymbol{P}(E)$, conferring it a lattice structure, is described in section 2.4.

2 Search space for automata inference

The search space for automata inference has been studied in [DMV94, Dup96] and extended to classifier automata in [Cos99]. For the sake of clarity we consider only NFA, but results presented apply also to classifier automata. We revisit the results of [DMV94, Dup96, Cos99], focussing on nondeterminism and the structure of the search space.

2.1 Sample

We consider that information on inference target language, consisting of its finite subpart, is available. We formalize this by the definition of *learning sample.*

Definition 7 (Learning sample of a language) *A* learning sample *S of a language L is a finite multi-set of words from L, i.e. $\forall w \in S, w \in L$.*

We propose a definition of a learning sample as a multi-set of words. In such a sample, it is possible to find the same word several times as an exemple of a language. We denote by $Nb_S(w)$ the number of repetitions of the word w in a sample S. Repetitions in the sample are naturally available for many applications (g.e. [GBE96]). They are used in different ways in algorithms :

- In [BV96], $Nb_S(w)$ provides information on the structure of the inferred machine. Indeed, in this algorithm, a word w has exactly $Nb_S(w)$ acceptances in the target machine (the considered machines of this algorithm is a generalization of NFAs called *multiplicity automata*).
- Repetitions can also have probabilistic interpretations; in this context they are considered as frequencies used to estimate probabilities. This interpretation is used in algorithms for inference of probabilistic automaton, like Alergia [CO94].
- Finally, repetitions can be integrated in heuristic construction, which are used to guide a visit of the search space. This is the case in [Tho01] where a probabilistic heuristic is used, and where probabilities are based on repetitions.

2.2 Structural completeness hypothesis

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

This section discusses and redefines the structural completeness hypothesis so as to take into account particularities of non deterministic automata. We give below a definition of structural completeness extending [DMV94, Dup96] to consider repetitions, and the use of more than one initial states in our definition of automata.

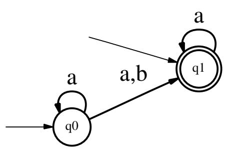
Definition 8 (Structural completeness) *A training sample S of a language L is said to be* structurally complete *with respect to an automaton A iff there exists an acceptance multi-set $\mathcal{A}cc$ containing exactly $Nb_S(w)$ acceptances for each word w of S such that:*

1. *every transition of A is reached by an acceptance of $\mathcal{A}cc$,*
2. *every initial state of A is the initial state of an acceptance of $\mathcal{A}cc$,*
3. *every final state of A is the final state of an acceptance of $\mathcal{A}cc$.*

Figure 2 illustrates this notion of structural completeness.

Let us notice that this definition entails two independent constraints between a sample and an automaton :

1. A constraint from the automaton on the sample: the sample must contain a set of words large enough to reach all automaton components.



The sample $S = \{aa, aa, b, \epsilon\}$ is structurally complete with respect to the automaton of the figure. The sample $S' = \{aaa, b\}$ is not structurally complete because there does not exist any acceptance with q_1 as initial state. The sample $S'' = \{aa, b, \epsilon\}$ is not structurally complete because no acceptance multi-set of S'' in the automaton satisfies the first condition of structural completeness definition: neither transition by letter a from q_0 to itself, nor from q_1 to itself is reached. This last example shed light on the fact that the higher the value of $Nb_s(w)$, the more the number of automata that respect structural completeness conditions.

Figure 2: Automaton and structurally complete sample.

2. A constraint from the sample on the automaton: the automaton must accept all words of the sample (there exists an acceptance for each word of S).

The first constraint can be seen as the need for having a "representative" sample of target automaton. This is this kind of constraint that motivated the introduction of structural completeness [FB75]. The second constraint is linked to the use of a non noisy sample: every word of the sample is supposed accepted by the target automaton; an acceptance then exists for each of these words. The second constraint could be suppressed by specifying that $\mathcal{A}cc$ has to contain $Nb_S(w)$ *or less* acceptances for each word of S. This definition modification is an open research direction to deal with noisy data.

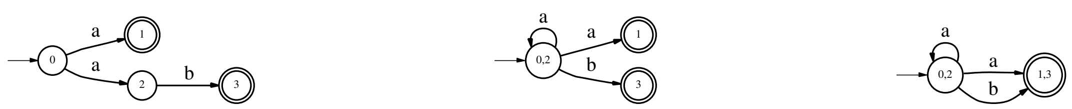
2.3 Search space under structural completeness hypothesis

If it is assumed that the sample is structurally complete with respect to inference target automaton, the corresponding search space is finite. We describe this search space and the way of visiting it by means of state-merging methods.

To define the search space, we use the operation of automata *derivation* with respect to a partition. We denote by $\boldsymbol{P}(A)$ the set of partitions defined on the states of an automaton A. From a given partition $\pi \in \boldsymbol{P}(A)$, we can construct a new automaton, denoted by A/π, thanks to the *derivation* operation :

Definition 9 (Derivation) *Let $A = \langle \Sigma, Q, I, \delta, F \rangle$ be an automaton and π a partition on A. The derived automaton $A/\pi = \langle \Sigma, Q', I', \delta', F' \rangle$ is defined by:*

1. $Q' = \pi = \{B_\pi(q) \mid q \in Q\}$
2. $I' = \{B_\pi(q_i) \mid q_i \in I\}$
3. $\delta' : Q' \times \Sigma \to 2^{Q'} : \forall B \in Q',\ \forall a \in \Sigma,\ \ \delta'(B, a) = \bigcup_{q \in B, q' \in \delta(q,a)} \{B_\pi(q')\}$
4. $F' = \{B_\pi(q_f) \mid q_f \in F\}$



An automaton A (on the left) and two automata derived from A by partitions $\{\{0,2\},\{1\},\{3\}\}$ and $\{\{0,2\},\{1,3\}\}$. Deriving an automaton with respect to a partition is done by grouping states of a same block of the partition. The automaton A derived from the partition $\{\{0\},\{1\},\{2\},\{3\}\}$ contains one state per block and is then equal to A.

Figure 3: Derivation examples.

This notion is illustrated by figure 3.

By convenience, we also say that an automaton A' is derivable from an automaton A if there exists a partition π on states of A such that $A' = A/\pi$.

We denote by $\boldsymbol{A}(A)$ the set of NFAs derived from an automaton A. More formally:

$$\boldsymbol{A}(A) = \{A' \mid \pi \in \boldsymbol{P}(A), A' = A/\pi\}.$$

Notations $\boldsymbol{P}(A)$ and $\boldsymbol{A}(A)$ are extended to take into account subclasses of NFA. We denote by $\boldsymbol{P}_{\text{DFA}}(A)$ partitions on states of A representing DFAs, and by $\boldsymbol{A}_{\text{DFA}}(A)$ the set of DFAs derived from A. More formally, for every automaton A and every subclass of automata $\mathcal{C} \subseteq$ NFA:

- $\boldsymbol{P}_{\mathcal{C}}(A) = \{\pi \mid \pi \in \boldsymbol{P}(A),\ A/\pi \in \mathcal{C}\}$
- $\boldsymbol{A}_{\mathcal{C}}(A) = \{A' \mid \pi \in \boldsymbol{P}(A),\ A' = A/\pi,\ A' \in \mathcal{C}\}$

The search space under structural completeness hypothesis is the set of derived automata from a particular one: the *maximal canonical automaton.* We extend the classical definition of maximal canonical automaton from [DMV94] to take into account automata with more than one initial states and repetitions.

Definition 10 (Maximal canonical automaton) *The* maximal canonical automaton *related to a sample S, denoted by $MCA(S)$ or more simply MCA, is the union, for each word w of sample S, of canonical automata $A(\{w\})$.*

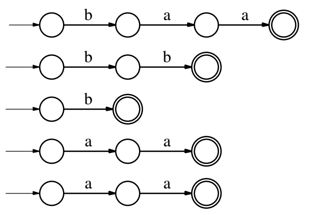
An example of *MCA* is given in figure 4.

The maximal canonical automaton consists of one "branch" per word of the sample. Its structure enables to represent the set of words of the sample but also their repetitions because, in the *MCA*, $Nb_S(w) = |Acc_{MCA}(w)|$.

Theorem 1 formalizes the relation between structural completeness, maximal canonical automaton and derived automata.

Theorem 1 *Let $\mathcal{A}$ denote the set of automata for which a sample S is structurally complete. Then $\mathcal{A}$ is equal to $\boldsymbol{A}(MCA(S))$.*

Proof: *We reformulate the proof of [Dup96] taking into account automaton with more than*



The MCA for $S = \{aa, aa, b, bb, baa\}$. The number of branches enabling to accept a word w is its number of repetitions.

Figure 4: Maximal canonical automaton.

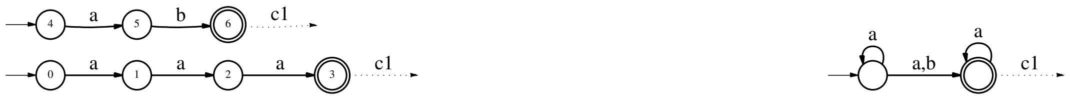
one initial states, repetitions and our new structural completeness definition. Complete proof is given in appendix A.

This theorem is the basis of inference under structural completeness hypothesis. It enables to consider inference as the visit of a finite search space, $\boldsymbol{A}(MCA)$, so as to extract one or several automata.

2.4 Extracting automata from the search space

Size of the search space is very large. Indeed, the number of partitions on MCA is $|\boldsymbol{P}(MCA)| = \mathcal{B}(|MCA|)$ where $\mathcal{B}(n)$ denotes the Bell number [Rot64]. $\mathcal{B}(n)$ being a non polynomial fonction of n, it is not possible to scrutinize each element of the set $\boldsymbol{P}(MCA)$ (except for a very small sample) for extracting the one(s) that is(are) the result of inference algorithm.

Let us notice that an automaton can be derived from more than one partitions (figure 5); the set $\boldsymbol{A}(MCA)$ is then of smaller size than $\boldsymbol{P}(MCA)$. However, for some samples



The MCA for $S = \{aaa, ab\}$ and an automaton A of $\boldsymbol{A}(MCA)$ represented by two partitions of $\boldsymbol{P}(MCA)$. A is equal both to MCA/π_1 with $\pi_1 = \{\{0,4,5\},\{1,2,3,6\}\}$ and to MCA/π_2 with $\pi_2 = \{\{0,1,4,5\},\{2,3,6\}\}$.

Figure 5: Two partitions representing the same automaton.

it is known that the set $\boldsymbol{A}_{\text{DFA}}(MCA)$ has the same size order as $|\boldsymbol{P}(MCA)|$ [Hén95]. As $\boldsymbol{A}_{\text{DFA}}(MCA)$ is included in $\boldsymbol{A}(MCA)$, an exhaustiv visit of $\boldsymbol{A}(MCA)$ is not possible either.

The search space can therefore be only partially visited. To realize this visit, we can use an order relation between its elements. The goal is to determine an order relation meaningful for languages and to visit the search space by considering inferior or superior elements to the current one. We consider an order relation reflecting a specialization/generalization of languages.

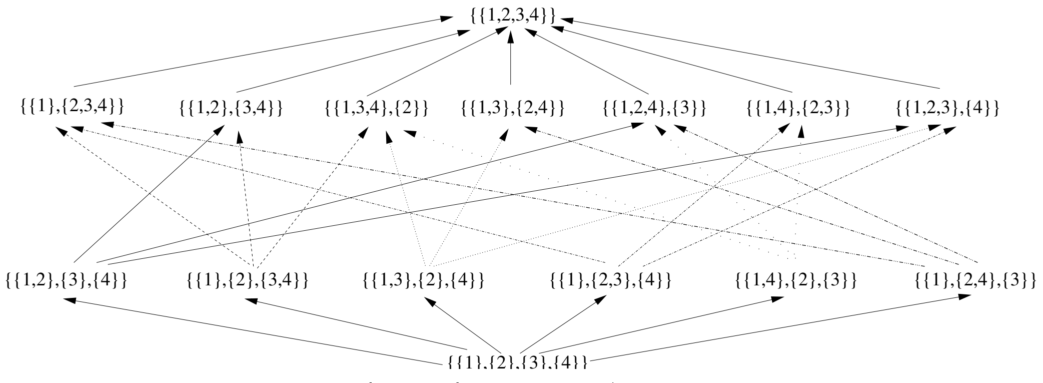
2.4.1 Order relation on partitions

A partition π_1 is said to be *directly inferior* to a partition π_2, and noted $\pi_1 \prec \pi_2$, if π_1 and π_2 satisfy the condition:

$$\exists\ b, b' \in \pi_1, b \neq b', \quad \pi_2 = (\pi_1 \setminus \{b, b'\}) \cup \{b \cup b'\}.$$

For exemple, in figure 3, $\{\{1\},\{2\},\{3\},\{4\}\} \prec \{\{1,2\},\{3\},\{4\}\} \prec \{\{1,2\},\{3,4\}\} \prec \{\{1,2,3,4\}\}$.

The transitive closure of $\prec$ is denoted $\prec^*$. The set of partitions on a set E ordered by $\prec^*$ has a minimum : the partition $\{\{q\} \mid q \in E\}$, and a maximum : the partition $\{E\}$. This set is also a lattice. It is illustrated by figure 6.



The lattice of partitions of the set $\{1,2,3,4\}$ *ordered by* $\prec^*$*, each arrow represents a direct inferiority under* $\prec^*$ *order relation.*

Figure 6: Lattice of partitions.

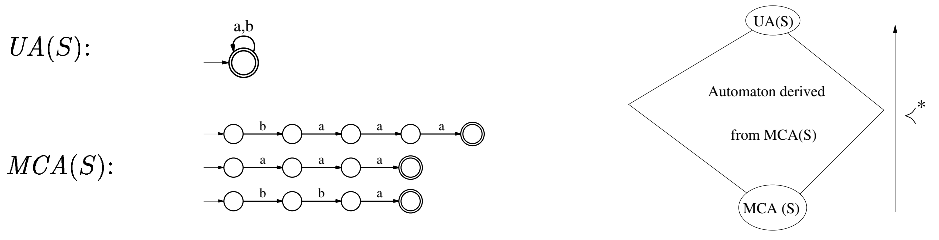
The minimum partition $\pi_s = \{\{q\} \mid q \in Q\}$ (Q being states of *MCA*) derives the *MCA* in itself. The maximum partition $\pi_g = \{Q\}$ derives the universal automaton. Figure 7 represents the search space ordered by $\prec^*$.

The relationship between derivation and languages has been established for the grammar representation [Rey68] (cited by [FB75]). We generalize his results to automata in the folowing theorem.

Theorem 2 Derivation and languages :
Let $A = \langle \Sigma, Q, I, \delta, F \rangle$ *be an automaton and* π *a partition on* A*. Then* $L(A) \subseteq L(A/\pi)$.
Proof: *In [FB75], derivation on grammar rules is used with respect to a partition on these rules. A complete proof in the case of automata is presented in appendix B.*

Theorem 2 implies the following corollary :

Corollary 1 Order relation $\prec^*$ and languages :
Let $A = \langle \Sigma, Q, I, \delta, F \rangle$ *be an automaton and* π_1, π_2 *two partitions on* A *such that* $\pi_1 \prec^* \pi_2$.



Universal automaton (UA), maximal canonical automaton (MCA) for $S = \{aaa, bba, baaa\}$ and search space $\boldsymbol{A}(MCA)$ under structural completeness hypothesis.

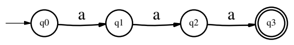
Figure 7: UA, MCA and $\boldsymbol{A}(MCA)$.

Then:

$$\forall\ A = \langle \Sigma, Q, I, \delta, F\rangle, \forall\ \pi_1, \pi_2 \in \boldsymbol{P}(A), \quad \pi_1 \prec^* \pi_2 \Rightarrow L(A/\pi_1) \subseteq L(A/\pi_2)$$

Proof: *This is a direct consequence of theorem 2; indeed, A/π_2 can be derived from A/π_1 if $\pi_1 \prec^* \pi_2$.*

The corollary 1 means that the order relation $\prec^*$ is at least as fine as the order relation defined by language inclusion. In fact, $\prec^*$ is finer than inclusion : two incomparable partitions can represent included languages (figure 8).



The two partitions $\{\{q_0, q_1\}, \{q_2\}, \{q_3\}\}$ and $\{\{q_0\}, \{q_1, q_2, q_3\}\}$ on this automata represent languages a^*aa and a^*a. These partitions are incomparable under $\prec^*$ but $a^*aa \subseteq a^*a$.

Figure 8: two incomparable partitions under $\prec^*$ representing included languages.

The order relation defined by languages inclusion is refined in two ways in the search space : first, because of the representation of languages by automata (a language can be represented by more than one automata), and second, because of the use of partitions to represent automata (an automaton can be represented by more than one partitions).

This refinment of the order relation is not desired : it affects the meaning of the ordering relation with respect to languages. Partial solutions to this problem can be obtained by restricting the search space to unambiguous automata or deterministic automata as discussed in section 3.3.

Appendix B gives a detailed proof of theorem 2 bringing into light the fact that not only languages, but also acceptances are preserved by derivation. The intrinsic interest of these properties is to enable better understanding of the ordering of search space. These properties will also be used in section 3.2 to proof properties on unambiguous automaton.

2.4.2 Visiting the search space

We now consider the operator that enables us to visit the search space with respect to $\prec^*$ order:

Definition 11 (State-merging) *Let π be a partition on $\boldsymbol{P}(MCA)$, and let b_1, b_2 be two blocks of π. We say that partition π' is obtained from π by* state-merging*, or more simply by* merging *the blocks b_1 and b_2 (denoted by $\pi \overset{\prec}{\longrightarrow} \pi'$), if and only if $\pi' = (\pi \setminus \{b_1, b_2\}) \cup \{b_1 \cup b_2\}$).*

Unlike the notation $\prec^*$, which represents an order relation, the notation $\overset{\prec}{\longrightarrow}$ is an operator which can be seen as a function, taking a partition and two of its blocks as its arguments and returning a new partition. With an appropriate representation of partition, this operation can be done efficiently [Tar75]. Trivially, a sequence of calls to $\overset{\prec}{\longrightarrow}$ operator is able to construct, depending on the pair of blocks on which it is called, every and only partition π' such that $\pi \prec^* \pi'$.

Extending the definition of state-merging definition, we will say that two states grouped in the same block of a partition are *merged*. Some mergings are illustrated in figure 3, from left to right, each partition is obtained by some state-merging on the previous partition.

Since every partition of $\boldsymbol{P}(MCA)$ is superior under $\prec^*$ to the partition π_s representing the MCA, every partition of the search space can be reached by using the state-merging operator on π_s. $\boldsymbol{P}(MCA)$ can therefore be visited entierly, and without redundancy, thanks to the state-merging operator as described in [Cos00, CF00].

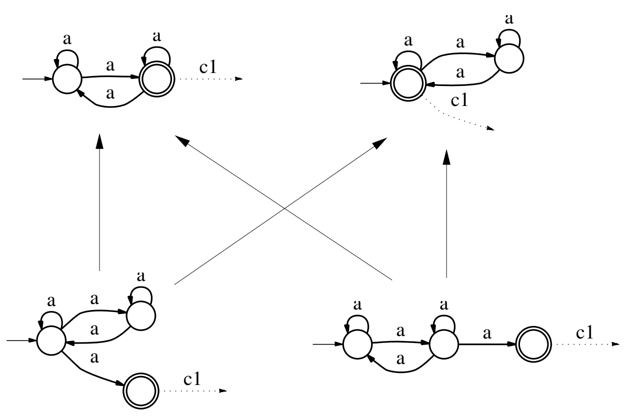
In practice, the goal is to visit the set of automata derived from partitions, and not the set of partitions. Visiting the space of partitions without redundancy does not ensure a visit of the NFA space without redundancy. Indeed, automata derivable from more than one partitions will be visited more than one time. Visiting without redundancy is important not only for a complete visit of the search space, but also for partial visits. Actually, partial visits are often used to obtain and compare a set of solutions to the problem; obtaining several identical solutions slows down this process.

This phenomenon can be formalized by considering the inferiority between automata induced by the one existing on partitions. More formally, this inferiority denoted by $\prec_A^*$ can be defined by :

$$A_1 \prec_A^* A_2 \Leftrightarrow \exists\ \pi_1, \pi_2 \in \boldsymbol{P}(MCA),\ \pi_1 \prec^* \pi_2,\ A_1 = MCA/\pi_1,\ A_2 = MCA/\pi_2$$

Under $\prec_A^*$ order relation, the space $\boldsymbol{A}(MCA)$ *is not a lattice*, as illustrated by figure 9. This shows clearly the misuse of terms used in regular grammatical inference [PC78, DMV94]. This important point implies that this space does not possess the same properties as $\boldsymbol{P}(MCA)$ and, among other things, the algorithms of [Cos00, CF00] replacing partitions by automata, blocks by states and operator $\overset{\prec}{\longrightarrow}$ by a merging directly applied on automaton states can not be used to visit this space without redundancy.

In practice, it seems less costly to explore the space of partitions than to ensure non redundancy of visited automata. However, a deeper study realized in section 3.3 shows that some subsets of $\boldsymbol{A}(MCA)$ - more precisely, $\boldsymbol{A}_{\mathrm{DFA}}(MCA)$ and $\boldsymbol{A}_{\mathrm{UFA}}(MCA)$ - are lattices



An automata pair of $\boldsymbol{A}(MCA)$ *with* $S = \{aaaaaa\}$ *(at bottom) (resp. at top) having two minimal elements in their up-set (resp. two maximal elements in their down-set) under* $\prec^*_A$ *order relation; this pair has therefore no meet (resp. join). An arrow represents a direct inferiority under* $\prec^*_A$.

Figure 9: NFAs are not organized as a lattice under $\prec^*_A$.

under an appropriate order relation. Algorithms presented in [Cos00, CF00] can therefore be used to visit these spaces without redundancy.

3 Search space for UFA and DFA

The restriction of NFA inference to DFA inference led to different algorithms [OG92, Lan92, LPP98]. We focus in this section on the fact that there exist a subclass of NFA - the UFA - that has a search space very similar to DFA while containing nondeterministic automata.

3.1 Automata of the search space

Let us first extend the proof of a well known property for DFAs to UFAs:

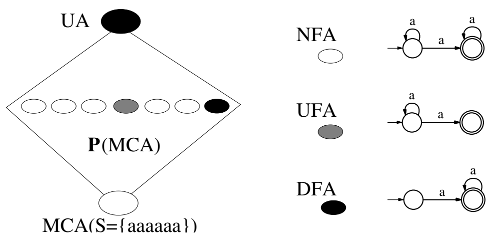
Theorem 3 *Let* A *be a UFA and* S *a training set structurally complete with respect to* A*. There exists one and only one partition* π *in* $Lat(MCA)$ *such that* $A = MCA/\pi$.
Proof: *There exists a partition* π *such that* $A = MCA/\pi$ *(entailed by* $UFA \subset NFA$ *and theorem 1). We have to show that this is the only one.*

Let $A = \langle \Sigma, Q, I, \delta, F\rangle$ *and* $MCA = \langle \Sigma, Q', I', \delta', F'\rangle$*. The acceptances are conserved after a derivation (property 8, appendix B). More formally, for all acceptances* $a = (q'_0, q'_1, \ldots, q'_{|w|})_w$ *for a word* $w \in \Sigma^*$ *in MCA, and for all partitions* π *on MCA states,* $(B_\pi(q'_0), B_\pi(q'_1), \ldots, B_\pi(q'_{|w|}))_w$ *is an acceptance in* MCA/π.

A being unambiguous, for every word it possesses at most one acceptance. Each state of an acceptance, for a word w in MCA, can then only be inside the block corresponding to this state in the acceptance for w in A. Moreover, for all states q' of Q' there exists an acceptance for a word w, that contains q'. Then, to each state of Q' corresponds only one state of Q, which defines the partition.

Thanks to theorem 3, we can forbid the distinction between partitions and automaton when considering UFA. Therefore properties demonstrated on $\boldsymbol{A}_{\mathrm{DFA}}(MCA)$ (resp. $\boldsymbol{A}_{\mathrm{UFA}}(MCA)$) are also valid for $\boldsymbol{P}_{\mathrm{DFA}}(MCA)$ (resp. $\boldsymbol{P}_{\mathrm{UFA}}(MCA)$). The converse is also true.

From theorem 3 and as illustrated by figure 10, UFA has the advantage over NFA of being represented by only one partition like DFA. DFA has the advantage, over NFA and UFA, of having a canonical form, which is represented by only one partition.



This 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 automata derived from these partitions, we count a single DFA, two UFAs and 7 NFAs, 5 being isomorphic.

Figure 10: Search space for DFA, UFA and NFA.

The *MCA* is neither deterministic nor unambiguous (except for very particular samples). It is therefore not part of the search space for deterministic or unambiguous automata. We define two automata, the *prefix tree acceptor* and the *unambiguous maximal canonical automaton*, that will "replace" the *MCA* when considering respectively deterministic and unambiguous automata inference.

Definition 12 (PTA) *The* prefix tree acceptor *on a sample S, $PTA(S)$ or more simply PTA, is the deterministic automaton obtained when merging every states of MCA with identical prefix languages. More formally, let $MCA = \langle \Sigma, Q, I, \delta, F\rangle$ and let $\pi \in \boldsymbol{P}(MCA)$ be the partition defined by:*

$$\forall q, q' \in Q \ : \ Pref_{MCA}(q) = Pref_{MCA}(q') \Leftrightarrow B_\pi(q) = B_\pi(q').$$

Then, $PTA(S) = MCA(S)/\pi$.

Definition 13 (Unambiguous MCA) *The* unambiguous maximal canonical automaton *related to a sample S, denoted by $MCA_u(S)$ or more simply MCA_u, is the union of canonical* *automata $A(\{w\})$ over all distinct words $w \in S$. More formally, let us denote $MCA = \langle \Sigma, Q, I, \delta, F\rangle$ and let $\pi \in \boldsymbol{P}(MCA)$ be the partition defined by:*

$$\forall q, q' \in Q \;:\; Pref_{MCA}(q) = Pref_{MCA}(q') \wedge Suff_{MCA}(q) = Suff_{MCA}(q') \Leftrightarrow B_\pi(q) = B_\pi(q').$$

Then, $MCA_u(S) = MCA(S)/\pi$.

These two automata are illustrated in figure 11.

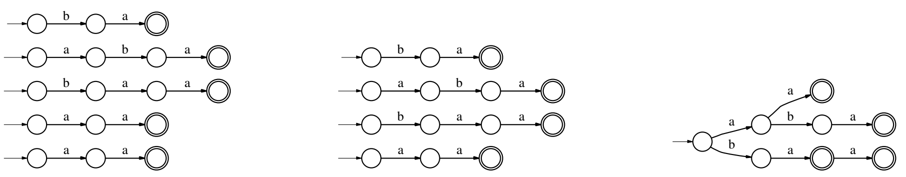


Figure 11: MCA, PTA and MCA_u with $S = \{aa, aa, ba, aba, baa\}$

The PTA (resp. MCA_u) has the noteworthly property that every DFA (resp. UFA) in $\boldsymbol{A}(MCA)$ can be derived from it. More formally, $\boldsymbol{A}_{\text{DFA}}(MCA) = \boldsymbol{A}_{\text{DFA}}(PTA)$ and $\boldsymbol{A}_{\text{UFA}}(MCA) = \boldsymbol{A}_{\text{UFA}}(MCA)$. For DFA, this was demonstrated in [Dup96]; and for UFA, this property is a direct consequence of property 5 demonstrated page 20 of this paper.

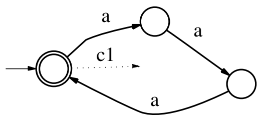
3.2 Structure of UFA and DFA search space under simple merge

We now focus on the structure of the search space with respect to merging of couples of states. Let us introduce the order relations operators $\prec_U^*$ and $\prec_D^*$ defined by:
$\pi_1 \prec_U \pi_2 \Leftrightarrow (\pi_1 \prec \pi_2) \wedge (MCA/\pi_1 \in \text{UFA},\; MCA/\pi_2 \in \text{UFA})$
$\pi_1 \prec_D \pi_2 \Leftrightarrow (\pi_1 \prec \pi_2) \wedge (MCA/\pi_1 \in \text{DFA},\; MCA/\pi_2 \in \text{DFA})$
$\prec_U^*$ and $\prec_D^*$ are defined as the transitive closure of $\prec_U$ and $\prec_D$.

The order relation $\prec_U^*$ (resp. $\prec_D^*$) has the noteworthy property that all unambiguous (resp. deterministic) automata of the search space are superior to the MCA_u (resp. PTA). This result is known for DFA [Dup96]; we extend it in theorem 4 to UFA.

Theorem 4 Sequence of partitions representing UFA in $\boldsymbol{P}(MCA)$: *Let A be a UFA in $\boldsymbol{A}_{UFA}(MCA_u)$. Then there exists, for every partition π_n satifying $MCA_u/\pi_n = A$, a sequence of partitions $\pi_0 \prec_U \pi_1 \prec_U \ldots \prec_U \pi_n$ such that $MCA_u = MCA_u/\pi_0$.*
Hint of the proof: *Proof is constructive. We propose an algorithm providing a sequence of partitions from A to MCA_u representing UFA and which are comparable to each other in the sense of $\prec_U^*$. The idea of the algorithm is to start from partition π_n and to show that blocks of this partition can be split in two parts such that the resulting partition corresponds to a UFA. Splitting blocks is done in a way depending on outgoing and incoming transitions of the block. This way of splitting ensures that no new acceptance is created. After n repetition of the operation, we obtain a partition π_0 representing MCA_u. The algorithm and the complete proof are given in appendix D.*

From theorem 4, $\boldsymbol{A}_{\mathrm{UFA}}(MCA)$ is an ordered set under order relation $\prec_U$ with a bottom (the MCA). This set may have no top, as shown by the counter-example of figure 12.



The UFA of the figure is such that every merge of two states gives rise to an ambiguous automaton. Therefore there cannot exist a sequence of directly derived UFA between this automaton and the UA.

Figure 12: A minimal UFA under $\prec_U^*$.

The operators of generalisation following $\prec_D^*$ or $\prec_U^*$ order relations are trivial in the sense that they are the direct restriction of state-merging on NFA to state-merging on UFA and DFA.

In a more general view of the search space, theorem 4 can also be generalized to k-ambiguous automata, i.e. automata such that $Degree(A) \leq k$ (UFA are the special case of k-ambiguous automata with $k = 1$). Therorem 4 is proved in this more general case.

3.3 From ordered sets to lattices

We show in this section that the search spaces for DFA and UFA (i.e. $\boldsymbol{A}_{\mathrm{DFA}}(MCA)$ and $\boldsymbol{A}_{\mathrm{UFA}}(MCA)$) are lattices under some particular order relations that correspond to particular state-merging operators.

For $\boldsymbol{A}_{\mathrm{DFA}}(MCA)$, the operator is based on the well known procedure of *merging for determinisation* [Ang82, OG92]. For $\boldsymbol{A}_{\mathrm{UFA}}(MCA)$, we introduce a new procedure: the *merging for disambiguisation.* Sub-section 3.3.1 introduces these procedures and some of their properties; sub-section 3.3.2 establishes the link between these procedures and the relation orders giving the lattice structure to $\boldsymbol{A}_{\mathrm{DFA}}(MCA)$ and $\boldsymbol{A}_{\mathrm{UFA}}(MCA)$.

3.3.1 Merging for determinisation and disambiguisation

The procedure of merging for determinisation (resp. disambiguisation) can be used to transform a NFA into a DFA (resp. UFA) by applying state-merging. These procedures compute the smallest number of state-merging to realize this transformation. They can be seen as a determinisation (resp. disambiguisation) of one automaton of the search space into another automaton of the search space. This determinisation (resp. disambiguisation) is not a "real" one since state-merging can generalize the language recognized by the automaton. In fact, the language of the obtained automaton includes or is equal to the language of the original one.

To introduce merging for determinisation and disambiguisation, we define several relations between states of an automaton.

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

The following properties link the common prefix relation to determinism and the between parallel acceptance relation to unambiguousness.

Property 1 Determinism and $\parallel_p$: *An automaton $A = \langle \Sigma, Q, I, \delta, F\rangle$ is deterministic if and only if it does not possess two different states in common prefix relation. More formally :*

$$A = \langle \Sigma, Q, I, \delta, F\rangle \textit{ deterministic} \Leftrightarrow (\forall q, q' \in Q,\ q \neq q' \Rightarrow \neg(q \parallel_p q'))$$

Proof: *Intuitively, for all state-pairs in common prefix relation, there exists a word u common to their prefix languages. If theses states are different, it means that two states can be reached by the same prefix, which is contradictory to the definition of determinism. More formally :*
$\forall q, q' \in Q,\ q \neq q' \Rightarrow \neg(q \parallel_p q')$
$\Leftrightarrow \forall q, q' \in Q,\ q \neq q' \Rightarrow \mathit{Pref}_A(q) \cap \mathit{Pref}_A(q') = \emptyset$
$\Leftrightarrow \neg(\exists q, q' \in Q,\ q \neq q',\ \exists w \in \Sigma^*,\ w \in \mathit{Pref}_A(q) \wedge w \in \mathit{Pref}_A(q'))$
$\Leftrightarrow \neg(\exists q, q' \in Q,\ q \neq q',\ \exists q_0, q_0' \in I,\ \exists w \in \Sigma^*,\ q \in \delta(q_0, w), q' \in \delta(q_0', w))$
By separating the cases where $q_0 \neq q_0'$ and where $q_0 = q_0'$
$\Leftrightarrow \neg((\exists q_0, q_0' \in I, q_0 \neq q_0',\ \exists q, q' \in Q,\ q \neq q',\ \exists w \in \Sigma^*, q \in \delta(q_0, w),\ q' \in \delta(q_0', w)) \vee$
$\quad (\exists q, q' \in Q,\ \exists q_0 \in I,\ q \neq q',\ \exists w \in \Sigma^*, q \in \delta(q_0, w),\ q' \in \delta(q_0, w)))$
$\Leftrightarrow \neg((\exists q_0, q_0' \in I, q_0 \neq q_0') \vee (\exists q_0 \in I,\ \exists w \in \Sigma^*, |\delta(q_0, w)| > 1))$
$\Leftrightarrow \neg((|I| > 1) \vee (\exists q \in Q, \exists a \in \Sigma, |\delta(q, a)| > 1))$
$\Leftrightarrow (|I| \leq 1) \wedge (\forall q \in Q,\ |\delta(q, a)| \leq 1)$
$\Leftrightarrow$ *A deterministic*

Property 2 *An automaton $A = \langle \Sigma, Q, I, \delta, F\rangle$ is ambiguous iff two of its states q and q', such that $q \neq q'$ are in parallel acceptance relation.*
Proof: *Intuitively, for all state-pairs in parallel acceptance relation, there exists a word u common to their prefix languages and a word v common to their suffix languages. If theses states are different, existence of the parallel acceptance relation is equivalent to the existence of two acceptances for the word uv. More formally :*
$\exists q, q' \in Q,\ q \neq q',\ q \parallel q'$.
By definition of $q \parallel q'$:
$\Leftrightarrow \exists q, q' \in Q,\ q \neq q',\ \exists u, v \in \Sigma^*,\ u \in \mathit{Pref}_A(q) \cap \mathit{Pref}_A(q'),\ v \in \mathit{Suff}_A(q) \cap \mathit{Suff}_A(q')$
By constructing acceptances of word $w = uv$ (for $\Rightarrow$) and by noticing that the n-th ($0 \leq n \leq |w|$) states of acceptances of a word w are always in parallel acceptance relation (for $\Leftarrow$) :
$\Leftrightarrow \exists q, q' \in Q,\ q \neq q',\ \exists u, v \in \Sigma^*,$
$\quad \exists acc_1, acc_2 \in \mathit{Acc}_A(uv),\ acc_1 = (q_0, \ldots, q_{|uv|}),\ acc_2 = (q_0', \ldots, q'_{|uv|}),\ q_{|u|} = q,\ q'_{|u|} = q'$
$\Leftrightarrow \exists w \in \Sigma^*,\ \exists acc_1, acc_2 \in \mathit{Acc}_A(w),\ acc_1 \neq acc_2$
By definition of ambiguity :
$\Leftrightarrow$ *A ambiguous.*

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. Parallel acceptance relation is directly deduced from the previous.

By using these relations, we can now define the *merging for determinisation* and *merging for disambiguisation* procedures (algorithm 1 and 2). The merging for determinisation (resp. disambiguisation) procedure consists in merging pair of states in common prefix relation (resp. parallel acceptance relation). Each merge possibly entailing new relations, the procedure stops merging when no more different states are in common prefix relation (resp. parallel acceptance relation).

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

```
1: while $\exists q_1, q_2 \in Q, q_1 \parallel_p q_2,\ q_1 \neq q_2$ do
2:     $A \leftarrow merge(A, q_1, q_2)$
3: end while
```

Algorithm 2 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:     $A \leftarrow merge(A, q_1, q_2)$
3: end while
```

Compared to merging for determinisation, which merges 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.

It can be checked formally that algorithm 1 (resp. algorithm. 2) returns an automaton which is deterministic (resp. unambiguous) such that its language includes or is equal to $L(A)$ (property 3 and 4).

Property 3 *The algorithm 1 executed on an automaton A ends, and the returned automaton A' is a DFA derived from A.*
Proof: *Property 1 demonstrates that the loop condition of line 2 being false is equivalent to determinism. If the algorithm ends, the obtained automata is therefore a DFA.*

At each iteration, the state-merging of line 3 is made. If the algorithm realizes $|A|$ state-mergings, the obtained automaton A' is UA; and this automaton has only one state. The condition on line 1 can therefore not be verified and the algorithm ends with at most $|A|$ iterations.

Trivially, A' is derived from A because A' is obtained by a sequence of mergings on A.

Property 4 *The algorithm 2 executed on an automaton A ends and the returned automaton A' is a UFA derived from A.*
Proof: *The proof is similar to that of property 3.*

A trivial algorithm always returning the universal automaton has also properties 3 and 4. The real interest of merging for determinisation (resp. disambiguisation) is that this procedure realizes all the state-mergings that are strictly necessary. In other words, if A is a NFA and A' the result of merging for determinisation (resp. disambiguisation) on A, all DFAs (resp. UFAs) derivable from A are also derivable from A'. Properties 5 and 6 formalize this point.

Property 5 Let A be a NFA, and let A' be the DFA returned by merge for determinisation of A; then $\boldsymbol{A}_{\mathbf{DFA}}(A) = \boldsymbol{A}_{\mathbf{DFA}}(A')$.

Proof:
If A is a DFA, it has no pair of different states in common prefix relation (property 1). Algorithm 1 therefore does not compute any merging, and $A = A'$; the property is trivially true.

Let us suppose now that A is not deterministic. Let $A_1, A_2, \ldots, A_n$, with $A = A_1$ and $A' = A_n$, be the sequence of automata created during the merging for determinisation procedure. We are going to show that $\boldsymbol{A}_{DFC}(A_i) \setminus \boldsymbol{A}_{DFC}(A_{i+1})$ is empty for $i \in [1, n[$; in other words, that $\boldsymbol{A}_{NFA}(A_i) \setminus \boldsymbol{A}_{NFA}(A_{i+1})$ does not contain any deterministic automata.

Let q_1 and q_2 be the states of A_i merged in A_{i+1}. Let us consider a partition π on A_i: the automaton A_i/π is derivable from A_{i+1} if and only if $B_\pi(q_1) = B_\pi(q_2)$. Therefore, automata of $\boldsymbol{A}_{NFA}(A_i) \setminus \boldsymbol{A}_{NFA}(A_{i+1})$ are represented by partitions π such that $B_\pi(q_1) \neq B_\pi(q_2)$.

The algorithm 1 merges only the states in common prefix relation; we therefore have $q_1 \parallel_p q_2$. Relations of common prefix being conserved after a merge (property 11, appendix C), we have $B_\pi(q_1) \parallel_p B_\pi(q_2)$ for all partition π on A_i.

Therefore, for all partitions π on A_i representing an automaton of $\boldsymbol{A}_{NFA}(A_i) \setminus \boldsymbol{A}_{NFA}(A_{i+1})$, we have: $B_\pi(q_1) \neq B_\pi(q_2)$ and $B_\pi(q_1) \parallel_p B_\pi(q_2)$. From property 1, the corresponding automaton is not deterministic; hence $\boldsymbol{A}_{DFC}(A_i) \setminus \boldsymbol{A}_{DFC}(A_{i+1}) = \emptyset$.

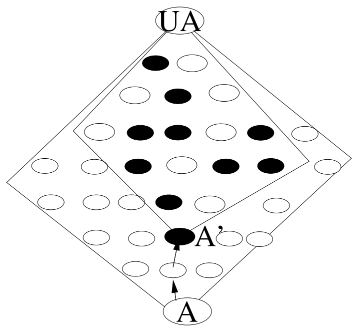
By induction on i, $\boldsymbol{A}_{DFC}(A_1) \setminus \boldsymbol{A}_{DFC}(A_n) = \emptyset$ is empty. $\boldsymbol{A}_{DFC}(A_n)$ being included in $\boldsymbol{A}_{DFC}(A_1)$, the subtraction is proper and $\boldsymbol{A}_{DFC}(A_1) = \boldsymbol{A}_{DFC}(A_n)$.

Property 6 Let A be a NFA, and let A' be the UFA returned by merge for disambiguisation of A, then $\boldsymbol{A}_{\mathbf{UFA}}(A) = \boldsymbol{A}_{\mathbf{UFA}}(A')$.

Proof: *The proof is similar to that of property 5.*

Property 5 and 6 are illustrated by figure 13.

Let us notice that property 5 (resp. 6) implies that whatever the order of merging realized by merging for determinisation or merging for disambiguisation, these mergings always lead to the same automaton. Let us also notice that applying merging for determinisation (resp. disambiguisation) on *MCA* returns the *PTA* (resp. the MCA_u). Associated with property 5 (resp. 6), it shows that all deterministic (resp. unambiguous) automata of $\boldsymbol{A}_{\mathrm{DFA}}(MCA)$ (resp. $\boldsymbol{A}_{\mathrm{UFA}}(MCA)$) can be derived from the *PTA* (resp. the MCA_u).



UA
A'
A
Merging for determinisation (resp. disambiguisation) enables to obtain a deterministic (resp. unambiguous) automaton A' from a non deterministic (resp. ambiguous) one A; and all DFA (resp. UFA) derived from A are also derived from A'. In the figure, deterministic (resp. unambiguous) automata are in black, the others are in white.

Figure 13: Merging for determinisation and disambiguisation.

3.3.2 Order relation based on merging for determinisation/disambiguisation

To visit the search space of DFA (resp. UFA), we are looking for an operator that takes a DFA (resp. a UFA) as its arguments and returns another DFA (resp. UFA), which is more general in the sense of $\prec_A^*$, and such that every DFA (resp. UFA) can be reached, if possible without redundancy. This operator, denoted by $\xrightarrow{det}$ (resp. $\xrightarrow{dis}$) can be constructed thanks to the merge for determinsation (resp. disambiguisation) procedure as follows.

Definition 14 (Deterministic state-merging and unambiguous state-merging) *Let A be a DFA (resp. UFA), and let q_1, q_2 be two states of A. Applying the* deterministic state-merging *(resp.* unambiguous state-merging*) operator on the states q_1 and q_2 of A consists in merging q_1 and q_2 and applying the merging for determinisation (resp. disambiguisation) procedure to the resulting automaton. If A is a DFA (resp. UFA), and A' is the result of a deterministic (resp. unambiguous) merging from A, we write $A \xrightarrow{det} A'$ ($A \xrightarrow{dis} A'$).*

These operators have the very noteworthy property, for deterministic (resp. unambiguous) merging, that every DFA (resp. UFA) derived from a given DFA (resp. UFA) A can be reached by a sequence of deterministic (resp. unambiguous) mergings from A. We formalize these properties by theorems 5 and 6.

Theorem 5 *Let A be a DFA. For all DFA A' of $\boldsymbol{A}_{DFA}(A)$, there exists a sequence of automata $A_0, \ldots, A_n$ such that $A_0 = A$, $A_n = A'$ and:*

$$A_0 \xrightarrow{det} A_1 \xrightarrow{det} \ldots \xrightarrow{det} A_n$$

Proof: *We prove this theorem by induction on the size of the set $\boldsymbol{A}_{DFA}(A_i)$ with $i \in [0, n]$.*

Point **0.** *: the starting point of induction is that A' is included in $\boldsymbol{A}_{DFA}(A)$ (by hypothesis).*

The induction step consists in showing that, for each automaton A_i such that $A' \in \boldsymbol{A}_{DFA}(A_i)$:

- *either $A' = A_i$ (point* **1.***),*
- *or $A' \neq A_i$ and (point* **2.***) there exists an automaton A_{i+1}, with $A_i \xrightarrow{det} A_{i+1}$ such that (point* **3.***) $A' \in \boldsymbol{A}_{DFA}(A_{i+1})$ and (point* **4.***) $|\boldsymbol{A}_{DFA}(A_{i+1})| < |\boldsymbol{A}_{DFA}(A_i)|$.*

We deduce from points **0.** *to* **4.** *that there exists a sequence $A_0 \xrightarrow{det} A_1 \xrightarrow{det} \dots \xrightarrow{det} A_n$ (point* **2.***) with $A = A_0$ (point* **0.***) and such that A' can be derived from all elements (point* **3.***). This sequence is finite because, from the constraint of point* **4.***, the size of the set $\boldsymbol{A}_{DFA}(A_i)$ decreases at each induction step (and $|\boldsymbol{A}_{DFA}(A_0)|$ is finite). The last element A_n of this sequence exists - because the sequence $A_0, \dots, A_n$ is finite - and this element is equal to A' because point* **1.** *is the only possible end of induction.*

We now prove the induction step (points **1.** *to* **4.***), and illustrate it by figure 14. At step i, either we have $A_i = A'$ (point* **1.***) and in that case induction ends with $i = n$; or we have $A_i \neq A'$. In the latter case, let us denote by E_i the set of automaton that can be obtained by realizing a state-merging on A_i.*

Every automaton derived from A_i, except A_i itself, is (by definition of derivation) derivable from at least one automaton of E_i. In other words, since $A_i \neq A'$, there exists an automaton $A'_i \in E_i$ such that $A' \in \boldsymbol{A}_{NFA}(A'_i)$; A' being a DFA by hypothesis we also have $A' \in \boldsymbol{A}_{DFA}(A'_i)$.

Let A_{i+1} be the automaton obtained by applying merging for determinisation on A'_i. By definition of deterministic merging, we have (point **2.***) $A_i \xrightarrow{det} A_{i+1}$. Point* **3.** *can be*

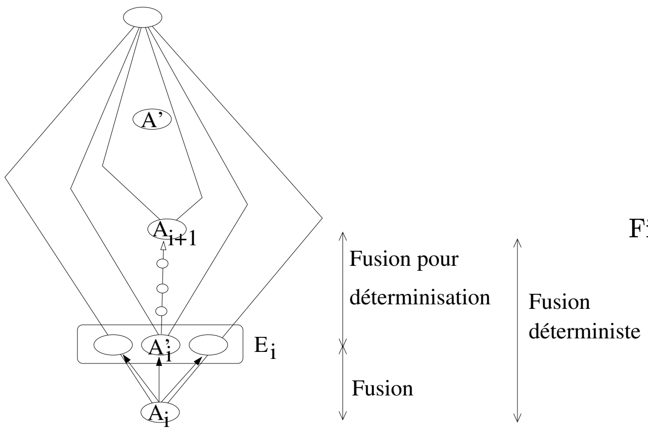


Figure 14: Deterministic merging

deduced from property 5 which implies that $\boldsymbol{A}_{DFA}(A'_i) = \boldsymbol{A}_{DFA}(A_{i+1})$, and therefore that $A' \in \boldsymbol{A}_{DFA}(A_{i+1})$. Finally, point **4.** *is deduced from the following : because $A_i \xrightarrow{det} A_{i+1}$ is a procedure using state-merging, the set of automata derived from A_{i+1} is included or equal to the set of those derived from A_i, i.e. $\boldsymbol{A}_{DFA}(A_{i+1}) \subseteq \boldsymbol{A}_{DFA}(A_i)$. Moreover, because A_i is deterministic and because the deterministic merging operator computes at least one state-merging, the set $\boldsymbol{A}_{DFA}(A_i)$ contains A_i and the set $\boldsymbol{A}_{DFA}(A_{i+1})$ does not contain A_i. Therefore, $\boldsymbol{A}_{DFA}(A_{i+1}) \subsetneq \boldsymbol{A}_{DFA}(A_i)$ and $|\boldsymbol{A}_{DFA}(A_{i+1})| < |\boldsymbol{A}_{DFA}(A_i)|$.*

Theorem 6 *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 $A_0 = A$, $A_n = A'$ and:*

$$A_0 \xrightarrow{dis} A_1 \xrightarrow{dis} \ldots \xrightarrow{dis} A_n.$$

Proof: *Proof is similar to the proof of theorem 5.*

The theorem 5 (resp. 6) brings into light, among other things, that all deterministic (resp. unambiguous) automata of the search space are reachable by successive applications of $\xrightarrow{det}$ (resp. $\xrightarrow{dis}$) from the PTA (resp. MCA_u). Compared to the theorem 4 - giving the previous result for state-merging restricted to determinism (resp. unambiguousness) - the theorem 5 (resp. 6) takes also into account automata that are different from the PTA (resp. the MCA_u).

Let us now consider the order relation related to deterministic (resp. unambiguous) merging. We show that this order relation is the restriction of $\prec^*$ to deterministic (resp. unambiguous) automata, defined by :

- $\forall \pi_1, \pi_2 \in \boldsymbol{P}(MCA) \ : \ \pi_1 \prec^*_{det} \pi_2 \Leftrightarrow \pi_1 \prec^* \pi_2, \ MCA/\pi_1 \in \text{DFA}, \ MCA/\pi_2 \in \text{DFA}$
- $\forall \pi_1, \pi_2 \in \boldsymbol{P}(MCA) \ : \ \pi_1 \prec^*_{dis} \pi_2 \Leftrightarrow \pi_1 \prec^* \pi_2, \ MCA/\pi_1 \in \text{UFA}, \ MCA/\pi_2 \in \text{UFA}$

Let denote $<$ the order relation related to deterministic (resp. unambiguous) merging; theorem 5 (resp. theorem 6) can be interpreted as :
$\forall \pi_1, \pi_2 \in \boldsymbol{P}_{\text{DFA}}(MCA) \ : \pi_1 \prec^* \pi_2 \Rightarrow \pi_1 < \pi_2$
(resp. $\forall \pi_1, \pi_2 \in \boldsymbol{P}_{\text{UFA}}(MCA) \ : \pi_1 \prec^* \pi_2 \Rightarrow \pi_1 < \pi_2$)

Moreover, the operator $\xrightarrow{det}$ (resp. $\xrightarrow{dis}$) can, by definition, be decomposed into a set of state-mergings, which, combined with the previous equation implies :
$\forall \pi_1, \pi_2 \in \boldsymbol{P}_{\text{DFA}}(MCA) \ : \pi_1 \prec^*_{det} \pi_2 \Leftrightarrow \pi_1 < \pi_2$
(resp. $\forall \pi_1, \pi_2 \in \boldsymbol{P}_{\text{UFA}}(MCA) \ : \pi_1 \prec^*_{dis} \pi_2 \Leftrightarrow \pi_1 < \pi_2$).

3.3.3 Lattice of automata

This section formalizes the fact that $\boldsymbol{A}_{\text{DFA}}(MCA)$ (resp. $\boldsymbol{A}_{\text{UFA}}(MCA)$) is strongly structured by $\prec^*_{det}$ (resp. $\prec^*_{dis}$) order relation.

Theorem 7 The set $\boldsymbol{A}_{\text{DFA}}(MCA)$ ordered by $\prec^*_{det}$ is a complete lattice.
Proof: *From the fact that $\prec^*_{det}$ is the restriction of $\prec^*$ on deterministic automata, we can deduce that UA is superior, in the sense of $\prec^*_{det}$, to all DFA of $\boldsymbol{A}_{DFA}(MCA)$. From theorem 5, we also know that PTA is inferior to all elements of $\boldsymbol{A}_{DFA}(MCA)$. $\boldsymbol{A}_{DFA}(MCA)$ possesses therefore a top (the UA) and a bottom element (the PTA).*

We also know that $\boldsymbol{A}_{DFA}(MCA)$ is finite. Based on these facts, we only have to prove that each pair $\{A_1, A_2\}$ of automata of $\boldsymbol{A}_{DFA}(MCA)$ has a join (or a meet) under $\prec^*_{det}$ order relation [DP90]. We will prove it by exhibiting the join.

As A_1 and A_2 are DFAs, each of them is represented in $\boldsymbol{P}(MCA)$ by a uniqie partition (theorem 3). Let π_1 and π_2 be these partitions. Let π be the join of π_1 and π_2 for the order relation $\prec^*$ (this element exists because $\boldsymbol{P}(MCA)$ is a lattice under $\prec^*$). Let A be the automaton obtained by applying the merging for determinisation on MCA/π. We are going to show that A is the join of $\{A_1, A_2\}$ under $\prec^*_{det}$ order relation (notations are illustrated by figure 15).

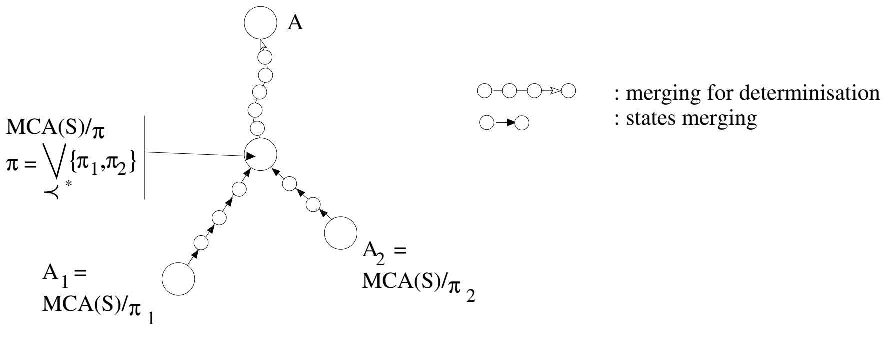


Figure 15: Join under $\prec^*_{det}$ order relation.

This proof is carried out in two steps: at step **1.** *we show that A_1 and A_2 are comparable and inferior to A under $\prec^*_{det}$ relation order, i.e. $A_1 \prec^*_{det} A$ and $A_2 \prec^*_{det} A$; and, at step* **2.**, *we prove that all DFAs A' superior to both A_1 and A_2 (i.e. $A_1 \prec^*_{det} A'$ and $A_2 \prec^*_{det} A'$) are superior to A (i.e. $A \prec^*_{det} A'$). The fulfilment of these two conditions entails, by definition, that A is the join of $\{A_1, A_2\}$ under $\prec^*_{det}$ order relation.*

- *Step* **1.** *- The join construction under $\prec^*$ relation order and merging for determinisation compute an automaton only by using state-merging. Therefore A is derived from A_1 (resp. A_2). Moreover, the use of merging for determinisation ensures that A is a DFA (property 3). Therefore we have $A \in \boldsymbol{A}_{DFA}(A_1)$ (resp. $A \in \boldsymbol{A}_{DFA}(A_2)$) and, from theorem 5 $A_1 \prec^*_{det} A$, $A_2 \prec^*_{det} A$.*
- *Step* **2.** *- Let us consider any partition π' representing a DFA and superior to both π_1 and π_2 in the sense of $\prec^*_{det}$ order relation. This partition is superior to π under $\prec^*$ order relation; otherwise it would not be comparable to both π_1 and π_2 under $\prec^*_{det}$. As π' is derivable from π (property 5), it follows that MCA/π' is also derivable from A because A is obtained by merging for determinisation from MCA/π. Therefore MCA/π' is superior to A under $\prec^*_{det}$.*

Theorem 8 The set $\boldsymbol{A}_{\mathbf{UFA}}(MCA)$ ordered by $\prec^*_{dis}$ is a complete lattice.
Proof: *The proof is similar to the one of theorem 7.*

Every subset $\mathcal{A}$ of an automata lattice possesses both a join and a meet. The proof of theorem 7 shed the light on the construction of the join of $\mathcal{A}$, which consists in : taking the partitions corresponding to automata belonging to $\mathcal{A}$, computing their join under $\prec^*$ and applying merging for determinisation.

The join of partitions under $\prec^*$ is the partition composed of the smallest blocks that are the union of original blocks of partitions. More formally :

$$\bigvee_{\prec^*}\{\pi_1,\ldots,\pi_n\} = \{B \mid q,q' \in B \Leftrightarrow (\exists i \in [1,n], B_{\pi_i}(q) = B_{\pi_i}(q'))\}$$

Figure 16 illustrates the need for the step of merging for determinisation (resp. disambiguisation) because the join under $\prec^*$ of two partitions representing deterministic (resp. unambiguous) automata does not always represent a deterministic (resp. unambiguous) automaton.

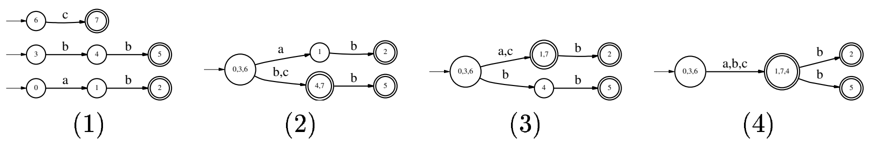


Figure 16: The MCA for $S = \{c, bb, ab\}$ (1); two deterministic automata, (2) and (3), such that their join under $\prec^*$ (4) is not deterministic. Automaton (4) being ambiguous, this example is also valuable for UFA.

The proof of theorem 7 (resp 8) could have been obtained by exhibiting the construction of the meet. This construction is described by theorem 9 (resp. 10).

Theorem 9 *Let A_1 and A_2 be two DFAs of $\boldsymbol{A}_{DFA}(MCA)$ and let π_1 and π_2 be their sole corresponding partitions (property 3) in $\boldsymbol{P}(MCA)$. The meet of $\{A_1, A_2\}$ under order relation $\prec^*_{det}$ is $A = MCA/\pi^\wedge$ where $\pi^\wedge$ is the meet of $\{\pi_1,\pi_2\}$ under $\prec^*$ order relation.*

Proof: *Let $\pi^\wedge$ be the meet of $\{\pi_1,\pi_2\}$ under $\prec^*$ order relation (figure 17), we will show*

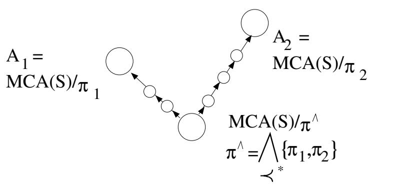


Figure 17: The meet under $\prec^*_{det}$ is the meet under $\prec^*$.

that automaton $MCA/\pi^\wedge$ is deterministic (point **1.***), that it is inferior under $\prec^*_{det}$ to A_1 and A_2 (point* **2.***), and that every automaton inferior to A_1 and A_2 under $\prec^*_{det}$ is also inferior to $MCA/\pi^\wedge$ (point* **3.***). We deduce from points* **1.***,* **2.** *and* **3.** *that $MCA/\pi^\wedge$ is the meet of $\{A_1, A_2\}$ under $\prec^*_{det}$.*

- *Point* **1.** - *$MCA/\pi^\wedge$* is deterministic : *let π be the partition obtained by applying merging for determinisation on $\pi^\wedge$; from property 3, (a) MCA/π is deterministic and* *(b) $\pi^{\wedge} \prec^{*} \pi$. From property 5, π_1 and π_2 are derivable from π, but from the definition of meet under $\prec^{*}$, $\pi^{\wedge}$ is the bigger partition from which π_1 and π_2 can be derived. Therefore, (c) $\pi \prec^{*} \pi^{\wedge}$. From (b) and (c), we deduce that $\pi = \pi^{\wedge}$; and from (a), $\pi = \pi^{\wedge}$ represent a deterministic automata.*

- *Point* **2.** - $MCA/\pi^{\wedge} \prec^{*}_{det} A_1$ and $MCA/\pi^{\wedge} \prec^{*}_{det} A_2$: *From point* **1.**, *$MCA/\pi^{\wedge}$ is deterministic, and by definition of meet under $\prec^{*}$, π_1 and π_2 are derivable from $\pi^{\wedge}$. According to theorem 5 under these conditions, $MCA/\pi^{\wedge} \prec^{*}_{det} A_1$ and $MCA/\pi^{\wedge} \prec^{*}_{det} A_2$.*
- *Point* **3.** - All automata inferior to both A_1 and A_2 under $\prec^{*}_{det}$ are inferior to $MCA/\pi^{\wedge}$: *By definition of the meet under $\prec^{*}$ of π_1 and π_2, every partition inferior to π_1 and π_2 under $\prec^{*}$ is also inferior to $\pi^{\wedge}$. Therefore, for all π inferior to π_1 and π_2 under $\prec^{*}$ and representing a deterministic automaton, $\pi^{\wedge}$ is derivable from π. Both π and $\pi^{\wedge}$ representing deterministic automata, from theorem 5 we have $MCA/\pi \prec^{*}_{det} MCA/\pi^{\wedge}$.*

Theorem 10 *Let A_1 and A_2 be two UFAs of $\boldsymbol{A}_{UFA}(MCA)$ and π_1 and π_2 be their sole corresponding partitions (property 3) in $\boldsymbol{P}(MCA)$. The meet of $\{A_1, A_2\}$ under order relation $\prec^{*}_{det}$ is $A = MCA/\pi^{\wedge}$ with $\pi^{\wedge}$ the meet of $\{\pi_1, \pi_2\}$ under $\prec^{*}$ order relation.*
Proof: *The proof is similar to proof of theorem 9.*

Theorem 9 (resp. 10) enables us to construct the meet of partitions under $\prec^{*}_{det}$ (resp. $\prec^{*}_{dis}$) since this meet is equal to the one of $\prec^{*}$. This meet is the partition composed of the set of non empty intersections of original blocks of partitions. More formally :

$$\bigwedge_{\prec^{*}}\{\pi_1, \ldots, \pi_n\} = \{B \mid B \neq \emptyset,\ \exists B_1 \in \pi_1, \ldots, B_n \in \pi_n\ ,\ B = \bigcap_{i \in [1,n]} B_i\}$$

Join and meet operators enable to visit the search space by revisiting not only one hypothesis π by constructing a partition π' such that $\pi' \prec \pi$ or $\pi \prec \pi'$, but also by combining a set of hypothese $\pi_1, \ldots, \pi_n$ to obtain a new one ($\bigvee\{\pi_1, \ldots, \pi_n\}$ or $\bigwedge\{\pi_1, \ldots, \pi_n\}$).

This kind of operators have never been used, to our knowledge, for regular language inference. However, it is used in logic inductiv programming. The notion of *smallest generalisation* [Plo70] of clauses corresponds to our notion of join.

Conclusion

We revisited the search space for NFA and DFA, clarifying some properties of the search space for NFA and bringing to light properties of the search space for DFA concerning merging for determinisation.

We also studied the intermediate UFA class that seems very promising for the introduction of some amount of nondeterminism in regular inference. Indeed, this class shares many search space properties with DFA while containing nondeterministic automata.

Defining a clear inference framework is useful only if one is able to search effectively in the corresponding hypothesis space. We have formalized and introduced various operators enabling to visit the search space for NFA, UFA and DFA. One of them - the unambiguous merging operator - shares many properties with the deterministic merging which is very frequently used for DFA inference.

Theoretical results of the last section open yet other perspectives for inference: UFA and DFA search space being lattices, new operators are available. These operators, the join and meet construction, are not unary but take a set of automata to construct a new one.

A Theorem 1 : set of structurally complete automata

Theorem 1 *Let $\mathcal{A}$ denote the set of automata for which a sample S is structurally complete. Then $\mathcal{A}$ is equal to $\boldsymbol{A}(MCA(S))$.*
Proof: *Proof is a consequence of lemma 1, saying that a sample S is structurally complete for every automata of $\boldsymbol{A}(MCA(S))$ and of lemma 2 saying that every automata for which S is structurally complete are in $\boldsymbol{A}(MCA(S))$.*

Lemma 1 *If an automaton A is in $\boldsymbol{A}(MCA)$ then S is structurally complete with respect to it.*
Proof: *This lemma is an extension of the one presented in [FB75] on grammars that we extend here to automata.*

By construction, S is structurally complete with respect to MCA. We show that derivation conserves the property of structural completeness and, hence, that for all automata of $\boldsymbol{A}(MCA)$, S is structurally complete.

Derivation operation conservs the acceptances (property 8, appendix C). To the multi-set of acceptances enabling structural completeness of MCA, corresponds a multi-set of acceptances in each derived automata. From the derivation definition, this multi-set satisfies the conditions of structural completeness (initial states, final states and transitions). S is therefore structurally complete with respect to any automaton of $\boldsymbol{A}(MCA)$.

Lemma 2 *Let S be a sample of a regular language L, and let A be an automaton accepting L. If S is structurally complete with respect to A, then A is in $\boldsymbol{A}(MCA(S))$.*
Proof: *The proof consists in reconstructing MCA from a given multi-set of acceptances $\mathcal{AC}$ of S respecting structural completeness condition. From these acceptances, we deduce the partition enabling to derive an automaton isomorphic to A from MCA.*

Let $A = \langle \Sigma, Q', I', \delta', F' \rangle$ and $MCA = \langle \Sigma, Q, I, \delta, F \rangle$. Let w_i denote the i-th word of S. We consider repetitions in this numbering, i.e. if there exist n repetitions of a word w in S, there exist n different indices for this word. Let us denote by $a_{i,j}$ the j-th letter of word w_i.

Each state of MCA can be described with respect to the sample from two indices: the state $q_{i,j}$ denotes the target state of transition representing the j-th letter of i-th word of the sample. States of indices $q_{i,0}$ are the initial states enabling the acceptance of i-th word of the sample.

Since S is structurally complete with respect to A, there exists an acceptance multi-set $\mathcal{AC}$ which defines for each word w_i ($i \in [1, |S|]$) of S an acceptance $(q'_{i,0}, \ldots, q'_{i,|w_i|})_{w_i}$ composed of states of Q'. These acceptances satisfy the following conditions by definitions of acceptance and of structural completeness, for $i \in [1, |S|], j \in [1, |w_i|]$:

- $q'_{i,0} \in I'$,
- $q'_{i,j} \in \delta'(q'_{i,j-1}, a_{i,j})$,
- $q'_{i,|w_i|} \in F$.

We define a function $\psi : Q \to Q'$ as follows :

$$\forall i \in [1, |S|], \forall j \in [0, |w_i|], \quad \psi(q_{i,j}) = q'_{i,j}.$$

This function defines a partition π on MCA states by :

$$\forall q_1, q_2 \in Q, \quad B_\pi(q_1) = B_\pi(q_2) \Leftrightarrow \psi(q_1) = \psi(q_2).$$

Let us observe that A is isomorphic to MCA/π : indeed, by associating with each state $q'_{i,j}$ of A the state $B_\pi(q_{i,j})$ of MCA/π, we obtain, by definition of derivation and from structural cxompleteness property :

1. *for each state $q'_{i,j}$ of A, the state $B_\pi(q_{i,j})$ of MCA/π is initial if and only if $q'_{i,j}$ is initial,*
2. *for each state $q'_{i,j}$ of A, the state $B_\pi(q_{i,j})$ of MCA/π is final if and only if $q'_{i,j}$ is initial,*
3. *for each state-pair $q'_{i,j}$, $q'_{l,m}$ of A and for each letter of Σ, we have $B_\pi(q_{i,j}) \in \delta(B_\pi(q_{l,m}), a)$ if and only if $q'_{i,j} \in \delta'(q'_{l,m}, a)$.*

Which entails that A is isomorphic to MCA/π, and therefore, A is in $\boldsymbol{A}(MCA)$.

B State-merging and language generalization

This section develops a set of proofs on generalization of the language recognized by an automaton after the application of the state-merging operator. These proofs are interesting in themselves; they show that prefix languages, suffix languages and acceptances are conserved or enlarged after a state-merging, which clarify the way the state-merging operator works. Results of this section are also used in proving theorem 4 and some other results.

To prove these properties, let us introduce the definition of a *way*:

Definition 15 (Way) *A* way *for a word $w \in \Sigma^*$, with $w = a_1 \dots a_{|w|}$ ($a_i \in \Sigma$), in an automaton $A = \langle \Sigma, Q, I, \delta, F \rangle$ between two states q_0 and $q_{|w|}$ is a sequence $(q_0, \dots, q_{|w|})_w$ of $|w| + 1$ states such that $\forall i \in [1, |w|]$, $q_i \in \delta(q_{i-1}, a_i)$. q_0 is said to be the* initial state *and $q_{|w|}$ the* final state *of the way.*

Acceptances are particular ways such that q_0 is initial and $q_{|w|}$ final.

Definition 16 Projection of a way : *let $c = (q_0, \dots, q_{|w|})_w$ be a way for a word w between states q_0 and $q_{|w|}$ of an automaton A. We denote by $\phi_\pi(c)$ the* projection *of the way c into the automaton A/π, defined by $\phi_\pi(c) = (B_\pi(q_0), \dots, B_\pi(q_{|w|}))_w$.*

Property 7 The projection of a way is a way : *Let A be an automaton. For all ways $c = (q_0, q_1, \dots, q_{|w|})_w$ in A, and for all partitions π on A, the projection $\phi_\pi(c) = (B_\pi(q_0), B_\pi(q_1), \dots, B_\pi(q_{|w|}))_w$ is a way in A/π.*
Proof: *To each transition between two states q and q' of A corresponds a transition between states $B_\pi(q)$ and $B_\pi(q')$ in the automaton A/π (point 3 of definition 9 of derivation). The projection of a way is therefore a way.*

Property 8 Projection of an acceptance is an acceptance : *Let A be an automaton. Then for all acceptances $\alpha = (q_0, q_1, \dots, q_{|w|})_w$ for a word $w \in \Sigma^*$ in A, and for all partitions π on A, the projection $\phi_\pi(\alpha) = (B_\pi(q_0), B_\pi(q_1), \dots, B_\pi(q_{|w|}))_w$ is an acceptance in A/π.*
Proof: *From property 7, the projection of the acceptance α is a way in A/π. We shall show that it is also an acceptance, i.e. that $B_\pi(q_0)$ is initial and $B_\pi(q_{|w|})$ is final : q_0 and $q_{|w|}$ are, by definition of acceptance, respectively initial and final, point 2 (respectively 4) of definition 9 of derivation implies that $B_\pi(q_0)$ is initial (respectively $B_\pi(q_{|w|})$ is final).*

The properties of the language of automata can be deduced from those of the ways.

Property 9 Prefix and suffix languages are conserved by derivation : *Let $A = \langle \Sigma, Q, I, \delta, F \rangle$ be an automaton and π_1, π_2 partitions on A. If $\pi_1 \prec^* \pi_2$, then $\forall q \in Q$:*

- $Pref_{A/\pi_1}(B_{\pi_1}(q)) \subseteq Pref_{A/\pi_2}(B_{\pi_2}(q))$,
- $Suff_{A/\pi_1}(B_{\pi_1}(q)) \subseteq Suff_{A/\pi_2}(B_{\pi_2}(q))$.

Proof: *From property 7 for all ways $(B_{\pi_1}(q_0), \dots, B_{\pi_1}(q_{|w|}))_w$ existing in A/π_1, there exists a way $(B_{\pi_2}(q_0), \dots, B_{\pi_2}(q_{|w|}))_w$ in A/π_2. $B_{\pi_1}(q_0)$ is initial which entails that $B_{\pi_2}(q_0)$ is also initial. Therefore, all words in the prefix language of a state q of the automaton A/π_1 are also in the prefix language of the projection of q in A/π_2. Proof for suffix languages is totally similar.*

Property 10 Languages are conserved or augmented by derivation : *Let $A = \langle \Sigma, Q, I, \delta, F \rangle$ be an automaton and π_1, π_2 be partitions on A, $\pi_1 \prec^* \pi_2$. Then, $L(A/\pi_2) \subseteq L(A/\pi_1)$.*
Proof: *This property is a direct consequence of property 8 (or of property 9).*

C Relation conservation with derivation

We show in this section that common prefix and common suffix relations are conserved after a derivation:

Property 11 Relations $\|_s$ and $\|_p$ are conserved by derivation:
Let $A = \langle \Sigma, Q, I, \delta, F \rangle$ be an automaton and $\pi_1 \prec^* \pi_2$ two partitions on A. If $q \in Q$, we have :

$$B_{\pi_1}(q) \|_p B_{\pi_1}(q') \Rightarrow B_{\pi_2}(q) \|_p B_{\pi_2}(q')$$
$$B_{\pi_1}(q) \|_s B_{\pi_1}(q') \Rightarrow B_{\pi_2}(q) \|_s B_{\pi_2}(q')$$

Proof: *Let $A_1 = A/\pi_1$ and $A_2 = A/\pi_2$ and let us suppose the existance of two states q and q' of A such that :*
$B_{\pi_1}(q) \|_p B_{\pi_1}(q')$.
By definition of $\|_p$: $Pref_{A_1}(B_{\pi_1}(q)) \cap Pref_{A_1}(B_{\pi_1}(q')) \neq \emptyset$. (1)
By property 9, we also know that:

$Pref_{A_1}(B_{\pi_1}(q)) \subseteq Pref_{A_2}(B_{\pi_2}(q))$ *and* $Pref_{A'_1}(B_{\pi'_1}(q')) \subseteq Pref_{A'_2}(B_{\pi'_2}(q'))$ *(2)*
From (1) *and* (2)*:* $Pref_{A_2}(B_{\pi_2}(q)) \cap Pref_{A'_2}(B_{\pi'_2}(q')) \neq \emptyset$
And by definition of common prefix relation : $B_{\pi_2}(q) \parallel_p B_{\pi'_2}(q')$*.*

The proof for the common suffix relation is similar.

D Theorem 4 : sequence of partitions representing unambiguous automata

The theorem 4 is proved in the more general case of k-ambiguous automata. k-ambiguous automata, or AFA$_k$, is the subset of NFA containing all automata with ambiguity degree equal to or less than k. Unambiguous automata is therefore equal to the class of AFA$_1$.

For the clarity of demonstration, we will consider in the sequel that the *MCA* is k-ambiguous. This can be done without loss of generality since, from a sample, we can always obtain a reduced sample generating a k-ambiguous *MCA* by limiting the number of repetitions in the sample to k.

We generalise the $\prec_U^*$ order relation to AFA$_\text{k}$ by : $\pi_1 \prec_{k-amb} \pi_2 \Leftrightarrow (\pi_1 \prec \pi_2) \wedge (MCA/\pi_1 \in \text{AFA}_k,\ MCA/\pi_2 \in \text{AFA}_k)$

Theorem 11 Sequence of partitions of $P_{\text{AFA}_k}(MCA)$ leading to each partition of $P_{\text{AFA}_k}(MCA)$: *Let A be an AFA$_k$ of $A_{AFA_k}(MCA)$. If MCA is k-ambiguous then, there exists, for every partition π_n such that $MCA/\pi_n = A$, a sequence of partitions $\pi_0, \pi_1, \ldots, \pi_n$ such that:*

- $MCA = MCA/\pi_0$*,*
- $\pi_0 \prec_{k-amb} \pi_1 \prec_{k-amb} \cdots \prec_{k-amb} \pi_n$*.*

Proof: *The proof is given in the following subsections.*

The proof of the theorem is constructive. We propose an algorithm providing a sequence of partitions from A to *MCA* representing AFA$_\text{k}$ which are comparable to each other in the sense of $\prec_{k-amb}^*$. We first show in section D.2 that the algorithm terminates and that the computed sets $\pi_n, \ldots, \pi_0$ are partitions. Secondly we show that these partitions represent k-ambiguous automata (section D.3).

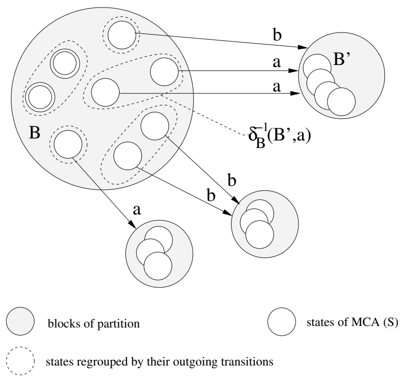
D.1 Algorithm 3

We introduce below some notations used in the algorithm. Let $MCA = \langle \Sigma, Q, I, \delta, F \rangle$ and let π be a partition on the *MCA*:

- $ntrans_\pi(B)$ (resp. $ntrans_\pi^{-1}(B)$), $B \in \pi$, is the number of outgoing (resp. incoming) transitions of B in automaton MCA/π if B is not final (resp. initial), or this number plus one if B is final (resp. initial).

- Let B and B' be two blocks of π. Then $\delta_B(B', a)$ (resp. $\delta_B^{-1}(B', a)$) is the subset of B such that $\delta_B(B', a) = \delta(B', a) \cap B$ (resp. $\delta_B^{-1}(B', a) = \delta^{-1}(B', a) \cap B$). Intuitivelly, $\delta_B(B', a)$ (resp. $\delta_B^{-1}(B', a)$) contains the states of MCA which are in B and that have an incoming (resp. outgoing) transition by a from (resp. to) block B'.

Figure 18 illustrates these notations.



We consider a particular splitting of the blocks of a partition π in disjoint subsets. Each subset groups the states of MCA with an outgoing transition by the same symbol and such that the target states of these transitions all belong to the same block of π. States of MCA without outgoing transitions (and thus final) are also grouped in the same subset. $\delta_B^{-1}(B', a)$ is one of these subsets and $nbtrans_\pi(B)$ is the number of these subsets in B. These subsets are disjoint because each state of MCA has zero or one outgoing transition. In the same way, a block can be split into subsets considering incoming transitions, which define $ntrans_\pi^{-1}(B)$ and $\delta_B(B', a)$.

Figure 18: Illustration of functions $ntrans_\pi$, δ_B, $ntrans_\pi^{-1}$ and δ_B^{-1}

D.2 Algorithm ends

We show in this section that the algorithm realizes a finite number of iterations, and that the partition obtained at the end is π_0 (representing the MCA).

Lemma 3 At each step, the set π_{i-1} is a partition on the states of MCA such that $\pi_{i-1} \prec \pi_i$.

Proof: *The condition on line 3 ensures that the algorithm goes either into the* **if** *on line 4, the* **else if** *on line 8 or into the* **else if** *on line 12. Therefore, at each step, either lines 5 through 7, or lines 9 through 11, or lines 13 through 15 are executed.*

If lines 14 through 16 are executed, then by construction π_{i-1} is a partition and this partition satisfies $\pi_{i-1} \prec \pi_i$ (the block B always containing at least two states - condition of the loop on line 4).

If lines 5 through 7 or 9 through 11 are executed, we will consider two cases. First, let us consider the case of lines 5 through 7; the choice of an ingoing transition in B (line 6) is always possible. Indeed, from the condition $ntrans_\pi^{-1}(B) \geq 2$ (line 4), the state B

Algorithm 3 Algorithm constructing partitions $\pi_0, \ldots, \pi_n$ of k-ambiguous automata

```
1: Let $\pi_n$ be one of the partitions such that $A = MCA/\pi_n$
2: $i \leftarrow n$ /* curent partition index */
3: while $\exists B \in \pi_i,\ |B| \geq 2$ do
4:    if $\exists B \in \pi_i,\ ntrans^{-1}_{\pi_i}(B) \geq 2$ then
5:       /* splitting based on ingoing transitions */
6:       Choose an ingoing transition of $B$; let $a$ be its symbol, $S$ be its source
7:       $\pi_{i-1} \leftarrow (\pi_i \setminus B) \cup \{\delta_B(S,a),\ B - \delta_B(S,a)\}$
8:    else if $\exists B \in \pi_i,\ ntrans_{\pi_i}(B) \geq 2$ then
9:       /* splitting based on outgoing transitions */
10:      Choose an outgoing transition of $B$; let $a$ be its symbol, $D$ be its target
11:      $\pi_{i-1} \leftarrow (\pi_i \setminus B) \cup \{\delta^{-1}_B(D,a),\ B - \delta^{-1}_B(D,a)\}$
12:   else if $\exists B \in \pi_i,\ |B| \geq 2$ then
13:      /* other splitting */
14:      Choose a state $s$ of $B$
15:      $\pi_{i-1} \leftarrow (\pi_i \setminus B) \cup \{\{s\},\ B - \{s\}\}$
16:   end if
17:   $i \leftarrow i - 1$
18: end while
```

possesses at least one ingoing transition. The splitting of block B is realized with respect to this transition. $\delta_B(S,a)$ is the first block resulting from the splitting and contains target states of the transition in the MCA. The second block is $B - \delta_B(S,a)$ which contains all other states.

The two created blocks are trivially disjoint and their union is B. We will show that they are also empty :

- *$\delta_B(S,a)$ contains, by definition, target states of transitions of MCA that created the transition from S to B by a in the automaton MCA/π_i.*
- *$B - \delta_B(S,a)$ contains, thanks to the condition $ntrans^{-1}_{\pi}(B) \geq 2$, the states of MCA that created either ingoing transitions in B different from the transition going from S to B by a, or the states of B that are initial (B cannot be initial thanks to states of $\delta_B(S,a)$: actually they possess an ingoing transition, and all states of MCA that have an ingoing transition are not initial).*

Next, in the case where lines 5 through 7 are executed, if π_i is a partition, the set π_{i-1} is also a partition such that $\pi_{i-1} \prec \pi_i$.

The case of lines 9 through 11 is similar to the one of lines 5 through 7.

Whatever the lines executed at each iteration (5 through 7, or 9 through 11, or 13 through 15), if π_i is a partition, π_{i-1} is also a partition such that $\pi_{i-1} \prec \pi_i$. By hypothesis, π_n is a partition; therefore by induction on i from n to 1, $\pi_0, \ldots, \pi_n$ are partitions such that $\pi_0 \prec \pi_1 \prec \ldots \prec \pi_n$.

□

The lemma 3 shows that the algorithm computes a sequence of more and more specific partitions on the MCA. The only partition satisfying the conditon of the **while** loop (line 3) is π_0 (no other has a single state per bloc) and this partition will be reached because, at each step of the loop, a more specific partition is constructed. At the end of the algorithm, we therefore have $MCA/\pi_i = MCA/\pi_0 = MCA$.

D.3 Invariant : obtained automata are k-ambiguous

Theorem 12 *For all $i \in [1, n]$, if the automaton MCA/π_i is k-ambiguous, then the automaton MCA/π_{i-1} is k-ambiguous.*

The proof of theorem 12 is given in three parts (in sections D.3.1, D.3.2 and D.3.3) for the splitting computed at lines 7, 11 and 15 respectively.

D.3.1 Splitting computed on line 7

The blocks B and S, the letter a, and the partitions π_i and π_{i-1} that are used in this section correspond to the one that are line 6 and 7 of algorithm 3. To simplify notations, the two blocks $\delta_B(S, a)$ and $B - \delta_B(S, a)$ will be respectively denoted by B' and B''.

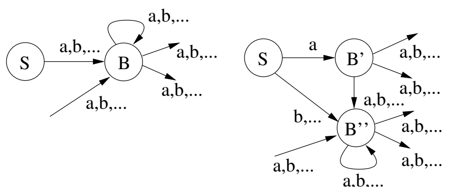
The goal of this section is to show that each partition π_{i-1} computed line 7 represents a k-ambiguous automata. To realize this, we consider each acceptance in MCA/π_i and show that it is the projection of at most one acceptance in MCA/π_{i-1}. Hence, we deduce that if MCA/π_i is k-ambiguous, then MCA/π_{i-1} is also k-ambiguous.

Each of the following proofs consider two cases of splitting, depending on the choice of the transition (S, a, B). Either (S, a, B) is a loop, i.e. we have $S = B$, or (S, a, B) is not a loop, i.e. $S \neq B$. Figure 19 represents - when considering these two cases - transitions that can exist between blocks B, S, B' and B'' before and after the splitting.

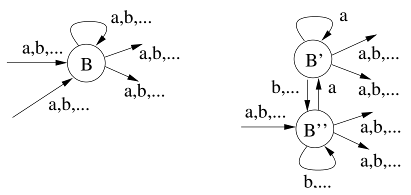
Next property will be used (without being explicity cited) in all proofs of the section:

Property 12 *Let $c = (B_1, \ldots, B_n)_w$ be a way for a word w in MCA/π_i. Any way $c' = (B'_1, \ldots, B'_n)_w$ in MCA/π_{i-1} that is projected in c in MCA/π_i is such that for all $i \in [1, n]$:*
- $B_i = B$ if and only if $B'_i = B'$ or $B'_i = B''$
- $B_i \neq B$ if and only if $B'_i = B_i$
Proof: *Intuitively, this property means that the modified blocks between c and c' are the ones equal to B in c. More formally, this is a direct restriction of the definition 16 of projection to the state-merging of states B' and B'' in a state B.*

We first show two properties on ways of size two (properties 13 and 14). Next, these properties will enable to prove properties on acceptances of MCA/π_{i-1} (property 15 and lemma 4) and finally property 15 and lemma 4 are used to show that MCA/π_{i-1} is k-ambiguous.



First case, $B \neq S$: the states before (left part) and after the splitting (right part). By construction the only ingoing transition of B' comes from S by a.



Second case, $B = S$: the states before (left part) and after the splitting (right part). By construction the only ingoing transitions of B' come from B' and B'' by a.

Figure 19: Illustrations of splittings computed on line 7.

Property 13 *Every way $c = (B_1, B_2)_{a'}$ for the word $a' \in \Sigma$ in MCA/π_i with $B_1 \neq B$ is such that there exists at most one way c' in MCA/π_{i-1} such that $\phi(c') = c$.* **Proof:** *If $B_2 \neq B$, property 12 implies that the only way c' of MCA/π_{i-1} that can be projected into $c = (B_1, B_2)_{a'}$ in MCA/π_i is itself (i.e. $c = c' = (B_1, B_2)_{a'}$).*

If $B_2 = B$, ways of MCA/π_{i-1} that can be projected in c are - if they exist in MCA/π_{i-1} - $(B_1, B')_{a'}$ and $(B_1, B'')_{a'}$. We show that at most one of these ways exists in MCA/π_{i-1}.

- *If $S = B_1$ and $a = a'$: by construction of B', there exists a transition by a' from B_1 to B', but not from B_1 to B''. Therefore, only the way $(B_1, B')_{a'}$ can exist in MCA/π_{i-1}.*
- *If $S = B_1$ and $a \neq a'$: by construction of B', there exists a transition by a' from B_1 to B'', but not from B_1 to B'. Therefore, only the way $(B_1, B'')_{a'}$ can exist in MCA/π_{i-1}.*
- *If $S \neq B_1$, $S \neq B$: by construction of B', the only ingoing transition of B' is the one comming from S by a. Therefore there does not exist any transition by a' from B_1 to B', and therefore only the way $(B_1, B'')_{a'}$ can exist in MCA/π_{i-1}.*
- *If $S \neq B_1$, $S = B$: by construction of B', there exists at most two ingoing transitions in B', the first comming from B'', the second, if it exists being a loop on B'. Therefore there does not exist a transition by a' from B_1 to B', and therefore only the way $(B_1, B'')_{a'}$ can exist in MCA/π_{i-1}.*

These four cases cover all possibilities, therefore the property is true..

Property 14 *Every way $c = (B, B_2)_{a'}$ of size two for a word a' in MCA/π_i is such that there exists at most two ways c' and c'' in MCA/π_{i-1} such that $\phi(c') = c$ and $\phi(c'') = c$. If c' exists in MCA/π_{i-1} it is of the form $c' = (B', B_2')_{a'}$ and if c'' exists in MCA/π_{i-1} it is*

of the form $c'' = (B'', B_2')_{a'}$
Proof: *If $B_2' \neq B$, this property can be directly deduced from property 12, in this case we have $B_2' = B_2$.*

If $B_2 = B$, property 12 let four possibilities: $(B', B')_{a'}$, $(B', B'')_{a'}$, $(B'', B')_{a'}$ and $(B'', B'')_{a'}$. We show that the splitting realised line 7 is such that at most two of them are ways in MCA/π_{i-1} (the first beginning by B' and the second by B'').

Ways beginning with B' :

- *if $a' = a$: the only possible transition starting from B' by a' and going to B' or B'' goes to B' if $S = B$ and to B'' if $S \neq B$. Therefore, if $S = B$, only the way $(B', B')_{a'}$ can exist in MCA/π_{i-1}, and if $S \neq B$, only the way $(B', B'')_{a'}$ can exist in MCA/π_{i-1}.*
- *if $a' \neq a$: the only possible transition starting from B' by a' and going to B' or B'' goes into B'' if $S = B$ and into B' if $S \neq B$. Therefore, if $S = B$, only the way $(B', B'')_{a'}$ can exist in MCA/π_{i-1}, and if $S \neq B$, only the way $(B', B')_{a'}$ can exist in MCA/π_{i-1}.*

If the way that is projected into c begins with B'':

- *If $a' = a$: the only possible transition from B'' by a' and going to B' or B'' goes to B' if $S = B$ and to B'' if $S \neq B$. Therefore, if $S = B$, only the way $(B'', B')_{a'}$ can exist, and if $S \neq B$, only the way $(B'', B'')_{a'}$ can exist.*
- *If $a' \neq a$: the only possible transition from B'' by a' and going to B' or B'' goes to B'' if $S = B$ and to B' if $S \neq B$. Therefore, if $S = B$, only the way $(B'', B'')_{a'}$ can exist, and if $S \neq B$, only the way $(B'', B')_{a'}$ can exist.*

These different cases cover all possibilities, therefore the property is true.

Thanks to these properties on ways of size two, we show the following property on ways of any size.

Property 15 *All ways $(B_1, \ldots, B_n)_w$ for a word w in MCA/π_i with B_1 initial is the projection of at most one way in MCA/π_{i-1} with a first state initial.*
Proof: *We first proof that the property is true for ways of length one. Then we show, by induction on the length of the ways, that the property is true for ways of any length.*

Ways of length 1: *Let c=$(B_0)_\epsilon$ be the considered way such that B_0 is initial. If $B_0 \neq B$, property 12 implies that the only block of MCA/π_{i-1} that can be projected in B_0 is B_0 itself, the property is therefore respected. If $B_0 = B$, property 12 implies that two ways can exist: $(B')_\epsilon$ and $(B'')_\epsilon$. We show that only B'' is initial.*

B' contains states of MCA that possess an ingoing transition by a from S. No states of MCA possess simultaneously an ingoing transition and is initial. Therefore B' is not initial. Because B_0 is initial, this property comes either from B', either from B''. We remarked that B' is not initial therefore B'' is initial.

Induction step: *Let $(B_0, \ldots, B_{n-1})_w$ be a way for a word w in MCA/π_i with B_0 initial. We suppose that $(B_0, \ldots, B_{n-1})_w$ is the projection of only one way of MCA/π_{i-1} with the* *first state initial. Let us consider a state B_n of MCA/π_i reachable with letter a' from B_{n-1}. $(B_0,\dots,B_{n-1},B_n)_{wa'}$ is therefore a way in MCA/π_i. We show that $(B_0,\dots,B_{n-1},B_n)_{wa'}$ is the projection of at most one way of MCA/π_i beginning with an initial state.*

- *If $B_{n-1} \neq B$: the way $(B_{n-1},B_n)_{a'}$ is the projection of at most one way of MCA/π_{i-1} (property 13). By hypothesis, the way $(B_0,\dots,B_{n-1})_w$ is the projection of at most one way of MCA/π_{i-1} with the first state initial. Therefore the way $(B_0,\dots,B_{n-1},B_n)_{wa'}$ is the projection of at most one way of MCA/π_{i-1} with a first state initial.*

- *If $B_{n-1} = B$: let us denote $(B'_1,\dots,B'_{n-1})_w$ the only way (by hypothesis) of MCA/π_{i-1} with B'_1 initial and such that $(B_0,\dots,B_{n-1})_w$ is its projection in MCA/π_{i-1}. We have either $B'_{n-1} = B'$, or $B'_{n-1} = B''$. From property 14 if $B'_{n-1} = B'$ there is at most one possible way $(B'_{n-1},B'_n)_{a'} = (B',B'_n)_{a'}$, respectively if $B'_{n-1} = B''$ there is at most one possible way $(B'',B'_n)_{a'}$. Therefore $(B_0,\dots,B_{n-1},B_n)_{wa'}$ is the projection of at most one way of MCA/π_{i-1} with a first state initial.*

By induction, the property is respected for all ways of length superior or equal to one. It is therefore respected for all ways.

We can deduce from the previous the following result linking acceptances of MCA/π_i to the one of MCA/π_{i-1}:

Lemma 4 *Any acceptance of MCA/π_i is the projection of at most an acceptance of MCA/π_{i-1}.*
Proof: *Acceptances being a subset of ways with a first state initial, property 15 is valuable for acceptances, which give the lemma.*

We deduce from this last lemma, that for a given word the automata MCA/π_{i-1} possess as many acceptances as, or less acceptances than the automata MCA/π_i. MCA/π_i being k-ambiguous, the automaton MCA/π_{i-1} is therefore k-ambiguous.

D.3.2 Splitting computed on line 11

The goal of this section is to show that all partitions π_{i-1} that are computed line 11 represent k-ambiguous automata. This proof is very similar to the one of previous section. We therefore only give the last lemma, adapted to the splitting computed on line 11.

Lemma 5 *All ways $(B_n,\dots,B_1)_w$ for a word w in MCA/π_i with B_1 final is the projection of at most one way in MCA/π_{i-1} with a last state final.*
Proof: *Similar to proof of lemma 4.*

Like in the previous section, this lemma can also be applied to acceptances. It therefore implies that for a given word the automaton MCA/π_{i-1} possess as many acceptances as, or less acceptances than the automaton MCA/π_i. MCA/π_i being k-ambiguous, the automaton MCA/π_{i-1} is also k-ambiguous.

D.3.3 Splitting computed on line 15

Lemma 6 Partition computed line 15 : *The partitions computed line 15 represent k-ambiguous automata.*
Proof: *For the line 15 to be executed, the conditions line 4 and 8 should not be respected. We therefore have : $\forall B \in \pi_i,\ ntrans_{\pi_i}^{-1}(B) = 1 \wedge ntrans_{\pi_i}(B) = 1$. This means that each state of MCA/π_i :*

- *either possess only one outgoing transition, or is final,*
- *either possess only one ingoing transition, or is initial.*

We therefore have a k-ambiguous automaton composed of branches that each classify only one word, this automaton is very similar to the MCA. However, it is not the MCA because it exists at least one block of the partition π_i that contains more than one state (this is true because of the condition of the **while** *loop on line 3).*

The block B to split is therefore used for the recognition of only one word w. This word possess strictly less than k acceptances in MCA/π_i otherwise MCA would not be k-ambiguous. By splitting this block, we add one acceptance for the word w. w possess therefore at most k acceptances MCA/π_{i-1} (for the other words the acceptance number is unchanged: one for each branch of the automaton). MCA/π_{i-1} is therefore k-ambiguous.

References

[Ang82] Angluin (D.). – Inference of reversible languages. *Communications of the ACM*, vol. 29, 1982, pp. 741–765.

[BV96] Bergadano (F.) et Varricchio (S.). – Learning behaviors of automata from multiplicity and equivalence queries. *SIAM Journal on Computing*, vol. 25, n6, 1996, pp. 1268 – 1280.

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

[CO94] Carrasco (R.) et Oncina (J.). – Learning stochastic regular grammars by means of a state merging method. *In : Grammatical Inference and Applications, ICGI'94*. pp. 139–150. – Springer Verlag, 1994.

[Cos99] Coste (F.). – *State merging inference of finite state classifiers.* – Rapport technique nINRIA/RR-3695, IRISA, septembre 1999.

[Cos00] Coste (F.). – *Apprentissage d'automates classifieurs en inférence grammaticale.* – Thèse de PhD, Science pour l'Ingénieur, Université de Rennes I, janvier 2000.

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

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

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

[DP90] Davey (B.) et Priesley (A.). – *Introducton to lattices and order.* – Cambridge mathematical textbooks, 1990.

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

[FB75] Fu (K. S.) et Booth (T. L.). – Grammatical inference: Introduction and survey — part I and II. *IEEE-Transactions on Systems, Man and Cybernetics*, vol. 5, 1975, pp. 95–111 and 409–423.

[GBE96] Goan (T.), Benson (N.) et Etzioni (O.). – A grammar inference algorithm for the world wide web. *In: AAAI spring Symp. on Machine Learning in Information Access.* – 1996.

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

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

[Hén95] Hénaff (M.). – Inférence de grammaires régulières à partir d'échantillons positifs et négatifs. – 1995. Rapport de DEA, encadrant L. Miclet, ENSSAT, Université de Rennes I.

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

[LPP98] Lang (K. J.), Pearlmutter (B. A.) et Price (R. A.). – Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. *Lecture Notes in Computer Science*, vol. 1433, 1998, pp. 1–12.

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

[PC78] Pao (T.) et Carr (J.). – A solution for the syntactic induction inference problem for regular languages. *Computer Language*, vol. 3, n53, 1978, pp. 53–64.

[Plo70] Plotkin (G.). – A note on inductive generalization. *Machine Intelligence*, vol. 5, 1970, pp. 153–163.

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

[Rey68] Reynolds (J. C.). – *Grammatical Covering.* – Rapport technique n 96, Applied Math. Div., Argonne National Lab., March 1968.

[Rot64] Rota (G.). – The number of partitions. *American Math.*, vol. 71, 1964, pp. 498–504.

[SH85] Stearns (R.) et Hunt (H.). – On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, vol. 14, n3, 1985.

[Tar75] Tarjan (R.E.). – Efficiency of a good but not linearset merging algorithm. *Journal of the ACM*, vol. 22, 1975, pp. 215–225.

[Tho01] Thollard (F.). – Improving probabilistic grammatical inference core algorithms with post-processing techniques. *In: Eighth Intl. Conf. on Machine Learning*. pp. 561–568. – Williams, July 2001.

[Yok94] Yokomori (T.). – Learning non-deterministic finite automata from queries and counterexamples. *Machine Intelligence*, vol. 13, 1994, pp. 169–189.

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399