Regular Inference as a graph coloring problem

François Coste and **Jacques Nicolas**
IRISA, Campus de Beaulieu
35042 RENNES Cedex FRANCE
fcoste@irisa.fr, jnicolas@irisa.fr

Abstract

We consider the problem of learning the set of all most general DFA consistent with a labeled sample. The paper proposess a constraints based specification of the set of solutions and gives an efficient algorithm to build the system of constraints. The effective production of DFA may then be reduced to a graph coloring problem. We have implemented and tested our approach on a classical benchmark. First results are very encouraging and show that the production of all solution DFAs may be practically tractable.

Keywords

Regular inference, deterministic finite state automata, version space, graph coloring.

Introduction

The paper addresses the issue of inferring all the solutions of a regular inference problem. More precisely, the problem may be stated as follows:

Given
a set of positive sentences I_+
a set of negative sentences I_-

Find the set of all automata A verifying:

1. A is a deterministic finite state automata ;
2. I_+ is structurally complete for A ;
3. $L(A)$, the language accepted by A, does not contain any string of I_- ;
4. $L(A)$ is a most general language (i.e. no other solution A' is such that $L(A) \subseteq L(A')$).

Given these conditions, the search space is finite and may be partially ordered.

Most of proposed algorithms focus on the production of a single solution. They proceed either with a depth first strategy (Oncina & Garcia 1992), a beam search strategy (Miclet & de Gentille 1994), or with an optimization method (Dupont 1994). The problem has been proven NP-complete in the worst case (Angluin 1978; Kearns & Valiant 1989).

Our approach is related to the version space approach (Mitchell 1982) where all solutions of a learning problem are characterized using two sets: the set S of maximally specific solutions and the set G of maximally general solutions. In our case, S may be built straightforwardly, using the canonical automata. Building G (also called *Border Set* (Dupont, Miclet, & E.Vidal 1994)) is much more difficult, due to the size of the search space and, in some case, to the size of G itself.

Very few authors have studied the characterization of G. Miclet has proposed in (Miclet & de Gentille 1994) a heuristic algorithm. An incremental approach using membership queries has been proposed in (Parekh & Honavar 1996). Preliminary studies have also been tried in the context free case (Vanlehn & Ball 1987; Giordano 1993).

When the language of generalization consists of vectors of attribute-value pairs, several authors have proposed to manage an implicit representation of the version space (Hirsh 1992; Nicolas 1993; Sebag 1994).

In the same spirit, we propose a compact representation of G, based on the set of admissible merging of states (Coste & Nicolas 1997). We show that such a set may be represented with a *set of unequality constraints* and we give an efficient algorithm to build the elements of G, reducing the problem to the *coloring of the graph of constraints.*

The next section briefly introduces concepts and notations we need to present our results. The third section establishes a constraint characterization of the G set. We propose in the fourth section an algorithm for an effective production of automata elements of G. The results of a first experimentation on a classical benchmark(Dupont 1996) are given in the fifth section. It is the first time to our knowledge, that a characterization of the whole set of solutions is proposed for this benchmark. We conclude with a sketch of various tracks of improvement for our work.

Regular Inference: definitions, notations and theorems

Def 1 *(DFA) A finite state automata is a quintuplet* $(Q, \Sigma, \delta, q_0, F)$ *where* Q *is a finite set of states,* Σ *an alphabet,* δ *a transition function from* $Q \times \Sigma$ *to* 2^Q *(extended to* $Q \times \Sigma^* \rightarrow 2^Q$*),* q_0 *is the initial state and* $F \subseteq Q$ *is the set of accepting or final states.*
If $\forall q \in Q, \forall a \in \Sigma\, \delta(q,a)$ *has at most one element, the automaton is said deterministic and we denote DFA such an automaton.*
An Automaton A accepts a regular language $L(A)$.

Def 2 *(Derived automata* A/π*) Given an automata* $A = (Q, \Sigma, \delta, q_0, F)$ *and a partition* $\pi = (B_0, B_1, \ldots, B_r)$ *of* Q*, the derived or quotient automaton* $A/\pi = (\pi, \Sigma, \Delta, B_0, R)$ *is defined as follows:*

- $q_0 \in B_0$;
- $R = \{B_i \in \pi, \exists q \in B_i \ tq \ q \in F\}$;
- $B_j \in \Delta(B_i, a)$ *iff* $\exists q \in B_i, \exists q' \in B_j$ *such that* $q' \in \delta(q,a)$.

The set of all automata derived from A is a lattice $Lat(A)$.

Def 3 *(Structural completeness) A set of words I is structurally complete with respect to an automaton A if there exists an acceptance of I such that every transition of A is exercised and every final state of A is used as an accepting state.*

Def 4 *(*$A(L)$*,* MCA*,* PTA*) The canonical automaton of a language L,* $A(L)$ *is the DFA accepting L which has the minimal number of states.*
The maximal canonical automaton with respect to a set of words I, $MCA(I)$*, is the automaton A with the largest number of states such that* $L(A) = I$ *and I structurally complete with respect to A.*
The prefix tree acceptor of I, $PTA(I)$*, is obtained from* $MCA(I)$ *by merging states sharing the same prefixes.*

We can know characterize the search space of DFA's for an inference problem (Dupont, Miclet, & E.Vidal 1994).

Theorem 1 *Let* I_+ *be a positive sample. The set* Γ *of automata such that* I_+ *is structurally complete with any automaton in* Γ *is* $Lat(MCA(I_+))$.

Theorem 2 *Let* I_+ *be a positive sample. Let L be the target language. If* I_+ *is structurally complete with respect to* $A(L)$*, then* $A(L)$ *is an element of* $Lat(PTA(I_+))$.

Another way to look at this theorem is to state that every language, such that I_+ is structurally complete with respect to its canonical automaton, is learnable in the lattice:

Corollary 1 *Let* I_+ *be a positive sample.*
A regular language L is said admissible if I_+ *is structurally complete with respect to* $A(L)$*, the canonical automaton of L.*
The set of canonical automata of all admissible languages is included in $Lat(PTA(I_+))$.

A constraint characterization of the set of minimal DFA

Given a positive sample I_+, the set of solution automata may be derived from the null element $A_0 = PTA(I_+)$. The issue is then to find partitions π such that A_0/π verifies all conditions defined in introduction.

By construction, condition 2 is always satisfied. Conditions 1 and 3 generate some constraints on the set of admissible partitions. We are characterizing these constraints in this section. Condition 4 will be satisfied if one chooses to minimize the number of unequalities. So we just need to explicit conditions 1 and 3.

We introduce a preliminary notation:

Let Q be the set of states of an automata. For a given partition $\pi = (B_0, B_1, \ldots, B_r)$ of Q and a given state q, we denote q_π the set $B_i, i \in [0, r]$ to which q belongs.

Characterization of non admissible merging of states with respect to a negative sample I_-

We introduce here the concept of *augmented PTA* (APTA) following in this the data structure proposed in (Higuera (de la), Oncina, & Vidal 1996) in order to handle simultaneously the positive sample I_+ and

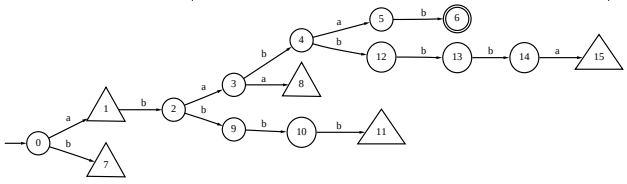
the negative sample I_-.

Intuitively, the APTA may be considered as the superposition of $PTA(I_+)$ and $PTA(I_-)$, each state being labeled into three classes : accepting state, rejecting state or intermediate state, accepting no positive and no negative instance.

Def 5 *An augmented PTA (APTA) with respect to a positive sample I_+ and a negative sample I_-, denoted $APTA(I_+, I_-)$, is a 6-tuple $(Q, \Sigma, \delta, q_0, F^+, F^-)$ where*

- $PTA(I_+ \cup I_-) = (Q, \Sigma, \delta, q_0, F^+ \cup F^-)$
- *F^+ and F^- are subsets of Q respectively identifying accepting states of I_+ and I_-*
- *F^+ is the set of final states of $APTA(I_+, I_-)$*

We give as an illustration, the APTA of sample 1 of language 2 in our benchmark (cf section Experimentation).



In this sample, $I_+ = \{ababab\}$, $I_- = \{a, b, abaa, abbbb, ababbbba\}$. States of F^+ are double circles and states of F^- are triangles.

By construction, we get the following properties :

$$L(APTA(I_+, I_-)) = I_+ \tag{1}$$

Moreover, for a partition π of states of the $PTA(I_+)$, we get:

$$L(APTA(I_+, I_-)/\pi) = L(PTA(I_+)/\pi) \tag{2}$$

Considering the APTA, instead of the PTA, has the great advantage of explicitly propagating the consequences of merging some states in the PTA on the acceptance of negative instances.

More precisely, an instance of I_- is accepted by an automata $APTA/\pi$ if the corresponding accepting state f^- in the APTA is merged with a final state of the APTA, i.e. $\exists f^+ \in F^+$, $f^-_\pi = f^+_\pi$.

Now, pairs of $F^+ \times F^-$ are not the only states that cannot be merged. Merging a set of states in an automaton A may lead to a non deterministic automaton A'. If the determinization of A' entails the creation of a new state merging a state of F^+ and a state of F^-, the same inconsistency appears. For instance, in the previous example, merging state $3, 4$ and 8 leads to a non deterministic automata. If this automata is determinized (Aho & Ullman 1972), a set of new states, $\{\{3, 4, 8\}, \{3, 4, 5, 8\}, \{3, 4, 8, 12\}, \{3, 4, 6, 8, 12\}\}$, corresponding to subsets of the previous set of states, are created. Since inconsistent pair $(6, 8)$ occurs in the new state $\{3, 4, 8, 6, 12\}$, merging state $3, 4$ and 8 leads to an inconsistent automata.

Since we are only interested in deterministic automata, the previous case may be easily handled in the more general framework of deterministic constraints. It is the subject of the next section.

Characterization of non admissible merging of states with respect to determinization

Algorithms such as RPNI (Oncina & Garcia 1992) use a procedure merging states until the corresponding automaton becomes deterministic. It allows the search to stay in the space of deterministic automata.

Such a procedure corresponds to the existence of a binary relation between pairs of states. Formally, if we denote $\rightarrow$ such a relation, we have

$$\begin{aligned} &(q_1, q_2) \rightarrow (q'_1, q'_2) \\ &iff\ \exists a \in \Sigma\ \delta(q_1, a) = q'_1 \wedge \delta(q_2, a) = q'_2 \end{aligned} \tag{3}$$

The transitive closure of $\rightarrow$ is denoted $\rightarrow^*$.
Although $\rightarrow$ induces only one kind of constraints, namely if q_1 and q_2 are merged then q'_1 and q'_2 must be merged also. One may refine them in two cases:

- consistency constraints (on pairs of states): q_1 cannot be merged with q_2 if $(q_1, q_2) \rightarrow^* (f^+, f^-)$, where $f^+ \in F^+$, $f^- \in F^-$, then clearly, q_1 and q_2 cannot be merged since there exists a same suffix accepted from q_1 and rejected from q_2.
- determinization constraints (on couple of pairs of states): if $(q_1, q_2) \rightarrow (q'_1, q'_2)$, then merging q_1 and q_2 implies that q'_1 and q'_2 must also be merged in order to produce a deterministic automaton.

In our example, $(5, 0) \rightarrow (6, 7)$ leads to a consistent constraint and $(0, 1) \rightarrow (7, 2)$ is a determinization constraint.

Finally we are able to give a purely constraint characterization of the set of solutions of the minimal DFA problem.

Inferring automata as the resolution of a set of unequality constraints

We characterize the set of non admissible merging of states in the APTA, denoted $NAM(APTA(I_+, I_-))$. Since the space of mergings is finite we might obtain dually the set AM of admissible mergings.

Def 6 *$NAM(APTA(I_+, I_-))$ is the set of constraints on admissible partitions π of Q, the set of states in $APTA(I_+, I_-)$.*

$$q_\pi^+ \neq q_\pi^- \qquad \forall q^+ \in F^+, \forall q^- \in F^- \quad (4)$$

$$q'_{1\pi} \neq q'_{2\pi} \Rightarrow q_{1\pi} \neq q_{2\pi} \quad \forall q'_1, q'_2 \in APTA(I_+, I_-)/(q_1, q_2) \rightarrow (q'_1, q'_2) \quad (5)$$

Note that constraints of type 5 may be reversed in form of equality constraints $q_{1\pi} = q_{2\pi} \Rightarrow q'_{1\pi} = q'_{2\pi}$. Moreover, the system of constraints may be simplified, applying modus ponens and discarding redundancy.

In our example, part of constraints are $(q_\pi \neq q'_\pi$ is simply denoted $(q, q'))$ $\{(1,6),(7,6),(8,6),(11,6),(15,6),(0,5),(5,10),(2,7) \Rightarrow (0,1),(7,9) \Rightarrow (0,2),(1,3) \Rightarrow (0,2),\ldots\}$

Since the APTA is a tree, there is at most one pair (q_1, q_2) such that $(q'_{1\pi} \neq q'_{2\pi}) \Rightarrow (q_{1\pi} \neq q_{2\pi})$

We give hereafter an algorithm for building the simplified set of constraints.

If n is the number of states in Q, NAM contains at most $n(n+1)/2$ constraints, since each pair of states appears at most once in the set of (left part of) constraints.

Thus the algorithm has a quadratic complexity in the worst case.

Building the set of admissible canonical automata

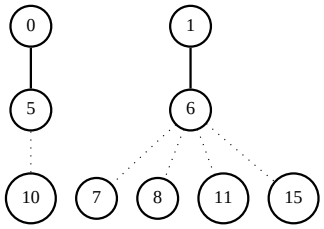
In this section, we propose an algorithm for an effective resolution of the previously defined system of constraints.

Two types of algorithms may be a priori designed for our problem. The first solution is to use a constraint solving algorithm, looking for a minimization of the number of unequalities. Such a minimization constraints renders difficult the formalization of the problem in this framework. We propose a second solution, considering the problem as a graph coloring issue.

Algorithm 1 Algorithm building NAM

```
NAM ← ∅
for all q+ ∈ F+ do
  for all q− ∈ F− do
    NAM ← NAM ∪ {(q+_π ≠ q−_π)}
  end for
end for
T ← ∅
for all q ∈ Q do
  for all q' ∈ Q do
    if (q' > q) ∧ (In_Symbol(q) = In_Symbol(q')) then
      T ← T ∪ {(q_π ≠ q'_π) ⇒ (parent(q)_π ≠ parent(q')_π)}
    end if
  end for
end for
for all (D1 ⇒ D2) ∈ T do
  if D1 ∈ NAM then
    NAM ← NAM ∪ {D2} {modus ponens}
  else
    if D2 ∉ NAM then
      NAM ← NAM ∪ {D1 ⇒ D2} {redundancy}
    end if
  end if
end for
```

Type 4 constraints may be represented with a graph on set Q, where each edge between vertices q and q' corresponds to a constrain $q_\pi \neq q'_\pi$. We give the figure of the corresponding graph in our example.



Type 5 constraints may be considered as dynamic constraints of type 4. Indeed, if during the building of π, some blocks q_π and q'_π are chosen for two states q and q', it is possible to propagate in the graph all the consequences of this choice. If $q_\pi \neq q'_\pi$ and $q_\pi \neq q'_\pi \Rightarrow D$, then constraint D may be added in the graph. In a dual way, equality constraints may be propagated with modus tollens.

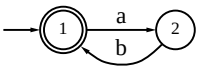
The relation between the set of solution automata and this graph may be formulated in terms of graph coloring:
Solution automata A_0/π, $\pi = (B_0, \ldots, B_r)$, are such that for every part $B_i, i \in [0, r]$, all its states are

compatible, that is, all its states may be colored with the same color.

Finding an admissible solution is equivalent to finding a k-coloring of the graph. Condition 4 states that we have to find a minimal coloring.

To be more precise, we do not have exactly one graph to color. Since some constraints are dynamic, we must consider a set of graphs, ranging from the graph where all left part of type 5 constraints are false to the graph where all left part of type 5 are true. Best k-coloring are obtained for k being the minimal chromatic number of this set of graphs.

The figure gives the sole canonical automaton solution in our example.



A sketch of the production algorithm follows:

Algorithm 2 Production of solution canonical automata

Build the APTA
Build NAM {algorithm 1}
Find all minimal colorings of the PTA, propagating dynamically the constraints, following the type 5 constraints (states with different color) and their reversed form (states with same color).
For each minimal coloring, product the corresponding solution automaton, derived from the PTA by merging the states with the same color.

Note that stating the problem as a graph coloring problem allows to implement a much more flexible resolution strategy than in classical approaches. Partitions of the states of the APTA may be searched for, following an ordering depending solely on the states themselves.

The coloring is NP-complete in the general case. The complexity of our algorithm remains unclear, due to the particular form of the graph. Our aim is to break the barrier of practically solvable problems, allowing more efficient heuristics to be tested. Next section shows a first experimental evidence in this respect. Remark also that test of consistency with respect to positive and negative instances are compiled in our algorithm and this is a factor of improvement with respect to traditional algorithms. We never have to consider automata themselves while building partitions of the states of the PTA.

Experimentation

We have used a benchmark described in (Dupont 1996) for a first validation of ideas developed in this paper.

We have implemented algorithm 2 using a version of DSATUR algorithm (Brélaz 1979) for the coloring step.

Results are summarized in the following table. #s/#t characterizes the size of target automata in terms of number of states and transitions, $\#G$ is the mean number of automata found in G. $APTA$ is the mean size of the augmented PTA. #c1 is the mean number of constraints of type 4, running time is given in seconds, with an UltraSparc 200MHz processor.

Language	#s/#t	$\#G$	APTA	#c1	time
L1	1/1	1	3.2	0	0.01
L2	2/2	1	25.3	43.0	0.03
L3	4/7	1.75	180.9	998.2	18.3
L4	3/5	8.5	102.2	192.2	7.08
L5	4/8	2.1	65.1	217.8	1.81
L6	3/6	2.45	40.7	108.5	0.51
L7	4/7	4.75	85.2	545	3.15
L8	2/2	1.25	19.0	43.0	0.02
L9	4/7	1.65	60.8	357.9	0.45
L10	5/6	2.4	81.2	431.7	0.59
L11	4/8	2.85	64.8	214.85	4.17
L12	3/3	1.5	37.6	120.2	0.09
L13	2/4	1.25	32.4	70.7	0.20
L14	3/4	1.4	37.6	100.5	0.16
L15	4/6	2.1	70.7	353.3	0.34

The production of solutions is on the whole very fast.

The benchmark needs of course to be completed with more difficult problems, increasing the size of the vocabulary and the size of the target automaton. The Abbadingo competition (see http://abbadingo.cs.unm.edu/) seems a good starting point in this respect.

However, our method already gives results that were beyond the reach of other methods. A particularly amazing fact is that the number of solutions remains generally low and loosely correlated to the recognition rate of RPNI algorithm.

For complex problems, stating them in terms of solving a system of constraints will allow to easily introduce auxiliary criteria in the selection process of relevant automata.

Conclusion

We have proposed a characterization of the set of minimal DFA that may be inferred from a positive and negative sample of words. We have also reduced the issue of generating explicitly the automata to a graph coloring problem.

First results are very encouraging, many tracks of improvement are possible in the coloring algorithm.

We have to study the determination of a coloring order on states, based on an a priori observation of most constrained states. No clever backtracking attempt has been made in our first implementation and this may lead to a great reduction in the overall search.

Our research will also look for the integration of this work in the version space framework. The next step in this direction will consist to define an incremental version of the algorithm.

References

Higuera (de la), C.; Oncina, J.; and Vidal, E. 1996. Identification of dfa : data-dependent versus data-independant algorithms. *Grammatical inference Learning Syntax from Sentences, ICGI'96* 311–325. Springer Verlag.

Aho, A., and Ullman, J. 1972. *The Theory of Parsing, Translation and compiling, Vol 1 : Parsing.* Englewood Cliffs, Prentice-Hall.

Angluin, D. 1978. On the complexity of minimum inference of regular sets. *Information and Control* 29(3):741 – 765.

Brélaz, D. 1979. New methods to color the vertices of a graph. *Communications of the ACM* 22:251–256.

Coste, F., and Nicolas, J. 1997. Inférence grammaticale régulière : caractérisation des solutions canoniques dans l'espace des fusions. In *Proceedings JFA97, to be published.*

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

Dupont, P. 1994. Regular grammatical inference from positive and negative samples by genetic search : the gig method. *ICGI'94, Grammatical inference and Applications* 236–245. Springer Verlag.

Dupont, P. 1996. *Utilisation et apprentissage de modèles de langages pour la reconnaissance de la parole continue.* Ph.D. Dissertation, Ecole Nationale Supérieure des Télécommunications.

Giordano, J. 1993. Espace des versions et inférence de grammaires algébriques. In *Journées Francophones d'Apprentissage.*

Hirsh, H. 1992. Polynomial-time learning with version spaces. In *National Conference on Artificial Intelligence*, number 18, 117–122.

Kearns, M., and Valiant, L. 1989. Cryptographic limitation on learning boolean formulae and finite automata. In *Proceedings of the Twenty First Annual Symposium on Theory of Computing*, 433 – 444.

Miclet, L., and de Gentille, C. 1994. Inférence grammaticale à partir d'exemples et de contre-exemples : deux algorithmes optimaux : (big et rig) et une version heuristique (brig). *JAVA94, Journées Acquisition, Validation, Apprentissage* F1–F13. Strasbourg France.

Mitchell, T. 1982. Generalization as search. *Artificial Intelligence* (18):203–226.

Nicolas, J. 1993. Une représentation efficace pour les espaces de versions. In *8ièmes JFA93.*

Oncina, J., and Garcia, P. 1992. Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis* 49 – 61.

Parekh, R., and Honavar, V. 1996. An incremental interactive algorithm for regular grammar inference. In Miclet, L., and de la Higueira, C., eds., *ICGI96, Grammatical Inference: Learning Syntax from Sentences*, volume 1147 of *Lecture Notes in Artificial Intelligence*, 238–249. Springer Verlag.

Sebag, M. 1994. Une approche par contraintes de l'espace des versions. In *RFIA'94*, 17–25.

Vanlehn, K., and Ball, W. 1987. A version space approach to learning context-free grammars. *Machine Learning* (2):39–74.