

NFA reduction algorithms by means of regular inequalities — correction

J.-M. Champarnaud and F. Coulon

Université de Rouen, LIFAR

F-76821 Mont Saint Aignan Cedex, France

{Jean-Marc.Champarnaud,Fabien.Coulon}@univ-rouen.fr

April 17, 2005

This note rectifies paper [1] which contains a flaw.

Let us first recall some definitions. Let $A = \langle Q, \Sigma, \delta, I, F \rangle$ be an NFA, that is, a nondeterministic finite automaton over states Q , alphabet Σ , whose initial states are I , final states are F and whose transition function is δ . The *left language* of a state q in A is defined as $\overleftarrow{\mathcal{L}}^A(q) = \{w \in \Sigma^* \mid q \in \delta(I, w)\}$. The *right language* of a state q in A is defined as $\overrightarrow{\mathcal{L}}^A(q) = \{w \in \Sigma^* \mid \delta(q, w) \cap F \neq \emptyset\}$. The language of A is noted $\mathcal{L}(A)$. The language $\mathcal{L}^A(q, p)$ is defined as the language of the automaton obtained from A by considering q as the unique initial state and p as the unique final state.

Merging two states q and p of A consists in adding all incoming (resp. outgoing) transitions of p to the set of incoming (resp. outgoing) transitions of q , and then deleting p . States q and p are said to be *mergeable*, which is noted $q \sim p$, if the language of the automaton is unchanged by the merging operation.

Paper [1] states the following proposition :

Proposition 5 (of [1]) *Let $q, p \in Q$, we have $q \sim p$ if and only if $\overleftarrow{\mathcal{L}}^A(q) \cdot \overrightarrow{\mathcal{L}}^A(p) \subseteq \mathcal{L}(A)$ and $\overleftarrow{\mathcal{L}}^A(p) \cdot \overrightarrow{\mathcal{L}}^A(q) \subseteq \mathcal{L}(A)$.*

In fact, when merging two states q and p , words are added to the language recognized by the automaton. Contrary to what Proposition 5 suggests, those words are not contained in $\overleftarrow{\mathcal{L}}^A(q) \cdot \overrightarrow{\mathcal{L}}^A(p) \cup \overleftarrow{\mathcal{L}}^A(p) \cdot \overrightarrow{\mathcal{L}}^A(q)$. Figure 1 gives a counter-example to Proposition 5: left and right languages of state 4 are equal to (a^+, a^+) , those of state 1 are equal to (a^+c^*, c^*a^+) . However, merging 1 and 4 adds some words to the language. But still, in this particular example, we could simply delete state 4 without changing the language. Figure 2 gives another counter-example showing that we cannot always delete a state whose left and right languages are respectively included in the left and right languages of another state. Here, left and right languages of state 1 are (a^+, a^+) . Left and

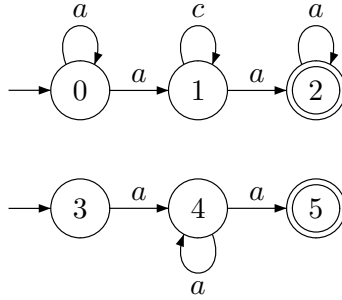


Figure 1: A counter-example where we cannot merge states 1 and 4.

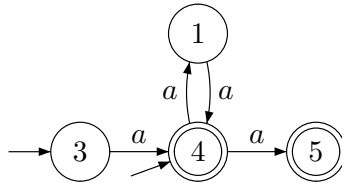


Figure 2: A counter-example where we cannot delete state 1.

right languages of state 4 are (a^*, a^*) . However, deleting state 1 would change the language of the automaton.

Determining which states can be merged is hard, though Proposition 5' stated below remains true. This new proposition replaces Proposition 5 on p. 243 of [1].

Proposition 5' *Two states having the same right language or having the same left language can be merged.*

Hence, some results of [1] should be corrected :

- In Proposition 8, p. 245, delete conditions 3 and 4,
- In Algorithm REDUCTION, p. 250, delete Lines 11 and 12.

Proof of Proposition 5'. Let q and p be two states having the same right language R in the automaton A . Let A' be the automaton obtained after merging q and p , that is, the incoming transitions of p are added to q and p is deleted.

The automaton A' recognizes all words in $\mathcal{L}(A)$. Indeed, let $w \in \mathcal{L}(A)$, there is a path π labelled by w in A . Consider the path π' in A' obtained from π by replacing each occurrence of p by q . Since the incoming transitions of p have been added to q , π' is a valid path in A , that is still an accepting path.

Let us prove the reverse inclusion. Let $w' \in \mathcal{L}(A')$. There is an accepting path π' in A' , labelled by w' . Let s be the length of π' . Either π' never goes through q and π' is an accepting path for A , so that $w' \in \mathcal{L}(A)$, or there

exist i and j , $1 \leq i \leq j \leq s$, such that $\pi'_i = q$, $\pi'_j = q$ and π' does not go through q before i nor after j . Since the path $\pi'[1..i]$ does not pass through q except at step i , it is labelled by a word in $\overleftarrow{\mathcal{L}}^A(q) \cup \overleftarrow{\mathcal{L}}^A(p)$. Since the path $\pi'[j..s]$ does not pass through q except at step j , it is labelled by a word in $R = \overrightarrow{\mathcal{L}}^A(q) \cup \overrightarrow{\mathcal{L}}^A(p)$. The path $\pi'[i..j]$ is labelled by a word of the language $(\mathcal{L}^A(q, p) \cup \mathcal{L}^A(p, q) \cup \mathcal{L}^A(p, p) \cup \mathcal{L}^A(q, q))^*$.

Let w be a word of the language

$$(\overleftarrow{\mathcal{L}}^A(q) \cup \overleftarrow{\mathcal{L}}^A(p)) \cdot (\mathcal{L}^A(q, p) \cup \mathcal{L}^A(p, q) \cup \mathcal{L}^A(p, p) \cup \mathcal{L}^A(q, q))^* \cdot R$$

We have to prove that w is in $\mathcal{L}(A)$. Indeed, we trivially have $\mathcal{L}^A(p, p) \cdot R \subseteq R$ and $\mathcal{L}^A(q, q) \cdot R \subseteq R$. Since R is the right language of both q and p , we also have $\mathcal{L}^A(p, q) \cdot R \subseteq R$ and $\mathcal{L}^A(q, p) \cdot R \subseteq R$. So, w is in $(\overleftarrow{\mathcal{L}}^A(q) \cup \overleftarrow{\mathcal{L}}^A(p)) \cdot R$, that is in $\mathcal{L}(A)$.

We have proved the result for right languages. The same is obtained for left languages by simply considering the reverse automaton. ■

Practical tests

The original paper shows some experimental results on randomly generated automata. Here we present new tests for the corrected algorithm.

We have observed the reduction achieved by detecting equalities between left or right languages using two different heuristics:

- Using preorders defined in Section 5 of our paper [1].
- Using equivalence relations as described in Section 3 of [1].

For both heuristics, the algorithm we applied is the same. Left and right languages are handled successively:

1. Reduce by detecting left language equalities,
2. reduce by detecting right language equalities,
3. and then, reduce once more by detecting left language equalities.

Former tests revealed that it was almost always useless to continue with one more step.

These methods have been applied on two sets of automata:

- Set 1: 10000 random automata on a 2-letter alphabet with 10 to 60 states.
- Set 2: 5000 random automata on a 3-letter alphabet with 10 to 60 states.

We obtain the following reduction ratios, that appear to be constant relatively to the number of states:

	Preorders	Equivalence relations
Set 1	2%	1%
Set 2	0.15%	0.075%

The reduction ratios are low, due to the fact that automata are randomly generated. Still, preorders appear approximately twice more efficient than equivalence relations.

Acknowledgements

We would like to thank Arnaldo Mandel for pointing out this error.

References

- [1] J.-M. Champarnaud and F. Coulon. NFA reduction algorithms by means of regular inequalities. *Theoret. Comput. Sci.* 327(2004) 241-253.