Minimal Unambiguous εNFA

Sebastian John

Technical University Berlin, IV, Computer Science
sebc@cs.tu-berlin.de

Abstract. A nondeterministic finite automaton with ε-transitions (εNFA) accepts a regular language. Among the εNFA accepting a certain language some are more compact than others. This essay treats the problem of how to compactify a given εNFA by reducing the number of transitions. Compared to the standard techniques to minimize deterministic complete finite automata (complete DFA) two novel features matter in compactifying εNFA: the principle of transition partition and the principle of transition union. An algorithm for compactifying εNFA is presented that exploits the union principle. This algorithm has the following property: if the algorithm returns an unambiguous automaton, then this automaton is the transition minimal εNFA.

1 Introduction

Minimization of finite automata is important for theoretical and practical reasons. In several application domains only small automata and their representations yield feasible solutions for practical problems. Basic algorithms for the problem of minimizing deterministic finite automata (DFA) have been published 50 years ago [10, 15, 17]. About succinctness, the feature of nondeterminism introduced in [21] allows to give much more compact automata. A nondeterministic finite automaton (NFA) may be smaller than a DFA by an exponential factor [16]. This even holds for unambiguous NFA [23, 26, 22]; ie NFA, which have at most one accepting computation for every word.

This essay treats the problem of how to compactify a given nondeterministic finite automaton with ε-transitions (εNFA). In general, the problem to compute a state minimal εNFA from a given εNFA is PSPACE-complete, by reduction to a problem given in [1, 6, p. 174], whereas in the deterministic case the problem to compute minimal complete DFA can be solved in time $\mathcal{O}(n \log n)$ [8] — a DFA is complete if each state has a transition for all inputs. Hereby, the minimal complete DFA is unique up to isomorphism [10, 17] and the number of states and the number of transitions are simultaneously minimized. A state minimal complete DFA is a transition minimal complete DFA, and vice versa. This does not hold for NFA and εNFA.

Essentially, there are three ways to define a minimization problem for NFA resp. εNFA : to minimize the number of states or the number of transitions or a sophisticated weighted sum of both. All of them can be solved by an exhaustive search algorithm in time $2^{\mathcal{O}(n^2)}$. In the formal language community, it is traditional to minimize the number of states. The approaches to this problem, that work by searching a minimal subautomaton, started with [5]. Many algorithms and heuristics have been proposed by [12, 13, 3, 4, 2, 14].

Another approach to reduce the number of states is due to [25, 20], where the minimization techniques for complete DFA are applied [9, 18, 19]. According to their reduction criteria, a NFA is most compact if there are no equivalent states. But in general, this does not mean state minimization, because there may be NFA accepting the same language using a smaller number of states.

Contrary, this essay investigates transition minimization. We do so mainly for three reasons : the size of an automaton is primarily influenced by the number of transitions, eg regarding an adjacency list, which is an efficient linear representation of an automaton. Secondly, the minimal unambiguous ε-complete εNFA is unique up to isomorphism. And finally, our algorithm to reduce εNFA with respect to the number of transitions performs better than the best known algorithms to compute state minimal NFA. The problem to compute a transition minimal εNFA from a given εNFA is PSPACE-complete by reduction to a problem described in [1, 6, p. 174]. The time complexity for our algorithm depends on the time needed to compute NFA equivalences or NFA complements. At the time, we therefore obtain a bound of $\mathcal{O}(2^n)$ for the running time of the algorithm.

The essay proceeds as follows. Section 2 introduces the formalism of εNFA, explains how the compactness of an εNFA is measured and defines the problem, to be solved. Section 3 elaborates the theory necessary to reduce the number of transitions. This theory and the proofs of the theorems are presented in more detail and with examples in [11]. Section 4 presents the algorithm, proves its correctness and determines its complexity. In Section 5 we conclude with final remarks.

In the sequel, $\mathbb{N}$ denotes the set of natural numbers including 0, Σ is the set of input symbols, Σ^* the set of finite words over Σ including the empty word λ. Variables $u, v, w \in \Sigma^*$ stand for words, $P, F \subseteq \Sigma^*$ for sets of words, A, B for automata, $s, t \in T$ for transitions, and $p \in T^*$ for finite paths with $p = p_1 ... p_n$ and $n \in \mathbb{N}$. By default, indices are omitted if it is clear from the context, what is meant.

2 Problem

Definition 1. *An εNFA $A = (Q, \Sigma, I, F, E, T)$ is given by*

- *a finite set of states Q*
- *a finite set of input symbols Σ*
- *a set of initial states $I \subseteq Q$*
- *a set of final states $F \subseteq Q$*
- *a set of ε-transitions $E \subseteq Q \times Q$*
- *a set of transitions $T \subseteq Q \times \Sigma \times Q$*

A word $w \in \Sigma^*$, including the empty word $\lambda \in \Sigma^*$, is accepted by the automaton, $w \in L(A)$, if there is a path from an initial state via the automaton's transitions to a final state yielding w. We come back to that point later. Two εNFA A and B are *equivalent* if and only if $L(A) = L(B)$. Among the εNFA accepting the same language there are some that are more compact than others. In our case, the measurement of compactness is based on the number of transitions. More precisely, on the number of transitions not counting ε-transitions. For εNFA A and B, the term “A is more compact than B”, for short $A < B$, is defined by:

Definition 2 (Compactness). $A < B \;:\Longleftrightarrow\; L(A) = L(B)$ *and* $|T_A| < |T_B|$

Problem : The problem is to find an algorithm, that computes a most compact automaton, ie: to compute from any given εNFA A an εNFA B with $L(A) = L(B)$ such that $C \not< B$ for all εNFA C.

Theorem 1 (Main Theorem). *There is an algorithm with running time $\mathcal{O}(2^n)$ that does always find an εNFA at least as compact as the input εNFA. The algorithm returns an εNFA, which is transition minimal if unambiguous.*

3 Theory

3.1 Acceptance Criterium

Words are accepted by an automaton along transition paths from initial to final states. We make this more precise by introducing a follow–relation and a label–path homomorphism in contrast to a direct definition as it is commonly used (eg [24, 9]).

The follow–relation $\longrightarrow$ expresses the connectivity of transitions within an εNFA A. For $s, t \in T$, the statement $s \longrightarrow t$ displays that s is connected via states and ε-transitions to t. For convenience, we add a new transition $t_0 \notin T$, set $T_0 := T \cup \{t_0\}$ and define the follow–relation $\longrightarrow$ on $T_0 \times T_0$. We denote the source state and the target state of a transition t as *source*(t) and *target*(t):

Definition 3 (follow–relation). *For $s, t \in T$:*

- $s \longrightarrow t \;:\Leftrightarrow\; target(s)\ E^*\ source(t)$
- $t_0 \longrightarrow t \;:\Leftrightarrow$ *There is an initial state $q \in I$ with $q\ E^*\ source(t)$*
- $s \longrightarrow t_0 \;:\Leftrightarrow$ *There is a final state $q \in F$ with $target(s)\ E^*\ q$*

Transitions t with $t_0 \longrightarrow t$ are called initial transitions, transitions s with $s \longrightarrow t_0$ are called final transitions.

A path $p \in T^*$ is a sequence $p = t_1 ... t_n$ with $n \in \mathbb{N}$ of transitions $t_i \in T$ connected by the follow–relation each labeled $l(t_i) \in \Sigma$. The word yielding from the path is given by the label–path homomorphism $l : T_0^* \to \Sigma^*$ with $l(p) = l(t_1) ... l(t_n)$. More precisely, t_0 transitions are allowed to be at the beginning or at the end or both for the purpose of accepting paths:

Definition 4 (label–path homomorphism). $l(stp) := l(s)l(tp) \;\Leftrightarrow\; s \longrightarrow t$ *with* $l(\lambda) := \lambda$ *and* $l(t_0) := \lambda$.

Note, that the definition incorporates the connectivity via the follow–relation, elsewhere l is partially undefined.

Proposition 1. $\forall p \in T^* : |l(p)| = |p|$

The accepted language $L(A)$ of an εNFA A, given in the notion of the follow–relation and the label–path homomorphism l, is defined on paths p of connected transitions from an initial to a final transition yielding words w according to l:

Definition 5. $L(A) = \{w \in \Sigma^* \mid \textit{there is a path } p \in T^* \textit{ with } l(t_0 p t_0) = w\}$
An εNFA A is unambiguous if and only if for each $w \in L(A)$ there is exactly one path p with $l(t_0 p t_0) = w$.

Regarding the definition of the unambiguousness of an εNFA, arbitrary many ε-transitions are not relevant with respect to this essay. That point would otherwise provoke to distinct strong and weak unambiguousness by the means of ε-transitions.

The information given by the follow–relation of an automaton A is sufficient to reconstruct an automaton that accepts the language $L(A)$. This reconstruction is not unique, but can be done canonically or small regarding the number of states and ε-transitions.

Fact : From any follow–relation an εNFA can be reconstructed.

For convenience, we assume in the following, without the loss of generality, that every εNFA considered has only productive transitions $t \in T$ — there is a linear time algorithm to eliminate every non-productive transition. A transition is productive if and only if it is connected to an initial and a final state over a path of other transitions possibly including ε-transitions, ie: it might be responsible for the acceptance of at least one word of the language. Thus, in the following, for all εNFA considered, and transitions $t \in T$ it holds: $t_0 \longrightarrow^+ t \longrightarrow^+ t_0$.

3.2 Future and Past

In this section, we will determine the semantics of the transitions by exploring what a transition $t \in T$ is responsible for — the future $\varphi(t)$ and the past $\pi(t)$. Similar ideas for states already appeared in [12, 6, 2].

Along a path to accept a word w, there is a transition t processing one of the letters of w. The letters before are part of the past $\pi(t)$. The letters, which are to be processed, belong to future $\varphi(t)$, whereby the present $l(t)$ is part of both $\pi(t)$ and $\varphi(t)$. This reflects the setting of the transition t among the others:

Definition 6 (future and past). $\varphi, \pi : T \to \Sigma^*$

- $\varphi(t) = \{w \in \Sigma^* \mid \textit{there is a path } p \textit{ with } l(p t_0) = w \textit{ and } p_1 = t\}$
- $\pi(t) = \{w \in \Sigma^* \mid \textit{there is a path } p \textit{ with } l(t_0 p) = w \textit{ and } p_{|w|} = t\}$

We define the language $M{:}1$ of words of $M \subseteq \Sigma^*$ without the last letter and $1{:}M$ without the first letter:

Definition 7. $M{:}1 := \{w \in \Sigma^* \mid wa \in M\}$ *and* $1{:}M := \{w \in \Sigma^* \mid aw \in M\}$

According to *future* $\varphi(t)$ and *past* $\pi(t)$ there are the *strict future* $1{:}\varphi(t)$ and the *strict past* $\pi(t){:}1$ each without the *present* $l(t)$. Because the present $l(t)$ of a transition t is unique, we observe obviously, that past is strict past combined with the present and analogously future is present combined with strict future:

Proposition 2. *$\pi(t) = \pi(t){:}1\, l(t)$ and $\varphi(t) = l(t)\, 1{:}\varphi(t)$*

By a little more algebraic investigation of the label–path homomorphism l, we get for all transitions t, all words $w, v \in \Sigma^*$ and all $a \in \Sigma$ that:

Lemma 1. *$vaw \in L(A) \;\Leftrightarrow\; \exists\, t \in T : va \in \pi(t)$ and $aw \in \varphi(t)$*
In case that the εNFA is unambiguous, the property holds for exactly one $t \in T$.

The accepted language of an εNFA A is completely determined by its future and past and from its strict future and strict past:

Proposition 3. $L(A) = \bigcup\limits_{s \,\rightarrow_A t} \pi(s)\varphi(t) = \bigcup\limits_{t \in T} \pi(t){:}1\; l(t)\, 1{:}\varphi(t)$

More central is the point that future φ and past π of an εNFA may be obtained by a *fixpoint* construction as they fulfill the equation system if we set $\varphi(t_0) := \{\lambda\} =: \pi(t_0)$:

Lemma 2 (fixpoint). $\varphi(s) = \bigcup\limits_{s \,\rightarrow_A t} l(s)\varphi(t)$ *and* $\pi(t) = \bigcup\limits_{s \,\rightarrow_A t} \pi(s)l(t)$

The Fixpoint Lemma implies directly a correspondence between the follow–relation and the future and past:

Corollary 1. *For transitions s and t with same label $l(s) = l(t)$ it holds:*
$s \longrightarrow = t \longrightarrow$ implies $\varphi(s) = \varphi(t)$ and $\longrightarrow s = \longrightarrow t$ implies $\pi(s) = \pi(t)$

3.3 Slicing

Central to the Minimizing Theorem 2 is the notion of slicing a regular language. Slices are unified to most compact slices, $S^{\mathcal{T}}$, which forms a compact εNFA.

Definition 8 (slice). *Given a regular language $L \subseteq \Sigma^*$. For all $P, F \subseteq \Sigma^*$, $a \in \Sigma$:*

$$(P, a, F) \text{ is a slice of } L \;:\Leftrightarrow\; P \neq \emptyset \text{ and } F \neq \emptyset \text{ and } PaF \subseteq L$$

A slicing of L is a set of slices of L. In particular, let S be the set of all slices of L. We define a partial order on S:

$$(P_1, a, F_1) \leq (P_2, a, F_2) \;:\Leftrightarrow\; P_1 \subseteq P_2 \text{ and } F_1 \subseteq F_2$$

We define $S^{\mathcal{T}} \subseteq S$, the set of maximal slices of L, by

$$S^{\mathcal{T}} := \{(P, a, F) \in S \mid \text{there is no } (P', a, F') \in S \text{ with } (P, a, F) < (P', a, F')\}$$

Lemma 3. *Given the set of all slices S of a regular language L, every linearly ordered set $X \subseteq S$ has a maximum in S, which is:*

$$\Big(\bigcup_{(P,a,F)\in X} P,\; a,\; \bigcup_{(P,a,F)\in X} F\Big)$$

Proof: Let us assume the settings $\cup P := \cup_{(P,a,F)\in X} P$ and $\cup F = \cup_{(P,a,F)\in X} F$. First, we show that we have defined a slice of L. Because future F and past P of a slice $(P, a, F) \in X$ are not empty, the unions $\cup P$ and $\cup F$ are not empty, either.

It remains to show $\cup Pa\cup F \subseteq L$. Every $w \in \cup Pa\cup F$ is split into subwords $w_1 \in \cup P$ and $w_2 \in \cup F$ with $w = w_1aw_2$, those come out of slices $x, y \in X$ with $w_1 \in P_x$ and $w_2 \in F_y$. These slices are ordered. We assume without the loss of generality that $x \le y$, which implies $w_1 \in P_x \subseteq P_y$ and $w_1aw_2 \in P_yaF_y \subseteq L$. So, $w = w_1aw_2 \in L$ and consequently $\cup Pa\cup F \subseteq L$. □

Due to the fact that every slice is part of a maximal linearly ordered set, we have:

Corollary 2. $\forall x \in S, \exists\, y \in S^{\mathcal{T}} : x \le y$

In the following, we want to read an automaton out of a slicing of L. In order to do so, we transform every slice into a transition. Usually, there are more than a finite number of slices. So, let us relax the finiteness of εNFA for the moment. We define the automaton A_S from the slicing S of L via the follow–relation. The other automata are subautomata of A_S, especially the ones corresponding to the finite slicings, which are subsets of S; eg $S^{\mathcal{T}}$ is a finite slicing of a regular language L. We come back to this point within the Minimizing Theorem 2.

Definition 9 **($A_S, A_{S^{\mathcal{T}}}, A_{\bar{F}}$).** *Assume $t_0 \notin S$ and $S_0 := S \cup \{t_0\}$. The follow–-relation $\longrightarrow \subseteq S_0 \times S_0$ is defined for all slices (P_1, a, F_1) and $(P_2, b, F_2) \in S$:*

$$\begin{array}{cccll} & (P_1,a,F_1) & \longrightarrow & (P_2,b,F_2) & :\Leftrightarrow P_1a \subseteq P_2 \text{ and } bF_2 \subseteq F_1 \\ & t_0 & \longrightarrow & (P_2,b,F_2) & :\Leftrightarrow \lambda \in P_2 \\ & (P_1,a,F_1) & \longrightarrow & t_0 & :\Leftrightarrow \lambda \in F_1 \\ [& t_0 & \longrightarrow & t_0 & :\Leftrightarrow \lambda \in L \quad] \end{array}$$

The last case fits if λ is not excluded from L from the beginning.

Lemma 4. *$L(A_{S'}) \subseteq L$ for each slicing $S' \subseteq S$.*

Sketch of proof: Let $w = a_1 ... a_n \in L(A_{S'})$. There is a path $p \in T^*$ with $l(t_0pt_0) = w$. Examining the path $p = p_1 ... p_n$, it sequences slices of the form $p_i = (P_i, a_i, F_i) \in S$ with the initial slice p_1 and final slice p_n due to $\lambda \in P_1$ and $\lambda \in F_n$. The follow–relation implies $a_1 \in P_1a_1 \subseteq P_2$ and by the induction principle $a_1 ... a_n \in P_na_n$. Together with $\lambda \in F_n$, we get $w = a_1 ... a_n \in P_na_nF_n \subseteq L$. □

Lemma 5. *The regular language L is accepted by A_S and $A_{S^{\mathcal{T}}}$:*

$$L(A_S) = L \text{ and } L(A_{S^{\mathcal{T}}}) = L$$

Sketch of proof: Let $w = a_1 ... a_n \in L$ we prove by induction:
$L \subseteq L(A_S)$: Let $p_k = (\{a_1 ... a_{k-1}\}, a_k, \{a_{k+1} ... a_n\}) = (P_k, a_k, F_k) \in S$. The sequence $p = p_1 ... p_n$ is an accepting path in A_S.
$L \subseteq L(A_{S^{\mathcal{T}}})$: By Corollary 2 there is, for each slice p_k, a slice $p_k^* = (P_k^*, a_k, F_k^*) \in S^{\mathcal{T}}$ with $p_k \le p_k^*$. That forms an accepting path $p^* = p_1^* ... p_n^*$. To prove that, we use intermediate slices $(\tilde{P}_{k+1}, a_{k+1}, \tilde{F}_{k+1}) = (P_k^*a_k, a_{k+1}, \{w \mid a_{k+1}w \in F_k^*\})$ to analyze $a_{k+1}F_{k+1}^* - F_k^*$ in order to prove $p_k^* \longrightarrow p_{k+1}^*$. □

Lemma 6. *Given a regular language L and a slicing $S' \subseteq S$ of L. Within the εNFA $A_{S'}$ we observe that for all slices $(P, a, F) \in S'$, which are transitions in $A_{S'}$, it holds:*

– *$\pi(P,a,F) \subseteq Pa$ and $\varphi(P,a,F) \subseteq aF$*

Moreover, if $S' = S$ or $S' = S^{\mathcal{T}}$:

– *$\pi(P,a,F) = Pa$ and $\varphi(P,a,F) = aF$*

Sketch of proof: By induction on the word length, we reason about the follow–relation. In case of π, consider the predecessors introduced by the fix point lemma, 2, and show that $\pi(P,a,F) = \bigcup_{(\tilde{P},b,\tilde{F})\to(P,a,F)} \pi(\tilde{P},b,\tilde{F})a$, and finally $\tilde{P}b \subseteq P$ if $(\tilde{P},b,\tilde{F}) \to (P,a,F)$. A proof works analogously for φ.

The second part uses, in addition to the idea in the part above, an idea similar to the one of the proof of Lemma 5: For each slice $(P,a,F) \in S$ analyse $(\{w|wb \in P\}, b, aF)$—which has also a maximum in $S^{\mathcal{T}}$ in case $S' = S^{\mathcal{T}}$—being connected to (P,a,F) by the follow–relation. □

3.4 Minimal Unambiguous εNFA

The following theorem is presented in the style of [9, 18, 19].

Theorem 2. *The three following statements are equivalent for languages $L \subseteq \Sigma^*$ if the slicing $S^{\mathcal{T}}$ of L induces an unambiguous εNFA $A_{S^{\mathcal{T}}}$:*

– *L is accepted by an εNFA*
– *$L = L(A_{\bar{F}})$ for some finite slicing $\bar{F} \subseteq S$*
– *$S^{\mathcal{T}}$ is finite*

Furthermore it holds:

– $|\,S^{\mathcal{T}}| \leq |\bar{F}| \leq |\,T_A|$

Corollary 3. *An unambiguous εNFA $A_{S^{\mathcal{T}}}$ is transition minimal.*

Proof: (1) → (2) : Let A be an εNFA with $L = L(A)$. For transitions $t \in T$ we define $\bar{t} := (\pi(t){:}1, l(t), 1{:}\varphi(t))$ to construct a finite slicing $\bar{F} = \{\bar{t}|t \in T\}$ — Proposition 3 includes $\pi(t){:}1\, l(t) 1{:}\varphi(t) \subseteq L$. And $\bar{F}$ is finite: $|\bar{F}| \leq |T|$.

It remains to show, that $L = L(A_{\bar{F}})$: We show for transitions s and $t \in T_{0A}$ that $s \longrightarrow_A t$ implies $\bar{s} \longrightarrow \bar{t}$ provided that $\bar{t}_0 := t_0$. Hence, every accepting path of A is an accepting path of $A_{\bar{F}}$, ie: $L = L(A) \subseteq L(A_{\bar{F}}) \subseteq L$, and by Corollary 4 then $L(A_{\bar{F}}) = L$.

Assume $s \longrightarrow_A t$. By the Fix Point Lemma 2, we get $\pi(s)l(t) \subseteq \pi(t)$. Therefore $\pi(s){:}1 l(s) \overset{P.2}{=} \pi(s) = \pi(s)l(t){:}1 \subseteq \pi(t){:}1$. That means $\pi(s){:}1 l(s) \subseteq \pi(t){:}1$ and similarly $l(t)1{:}\varphi(t) \subseteq 1{:}\varphi(s)$, which implies $\bar{s} \longrightarrow \bar{t}$ by the definition of the follow relation, 9. The cases of t_0 are easy to show.

(2) → (3) : Assume a finite slicing $\bar{F} \subseteq S$ with $L(A_{\bar{F}}) = L$. By the Lemma 2 we know $\forall x \in \bar{F} \subseteq S, \exists\, y \in S^{\mathcal{T}} : x \leq y$. Hence, we are allowed to assume a function $f : \bar{F} \to S^{\mathcal{T}}$ with $x \leq f(x) \in S^{\mathcal{T}}$. We show that f is surjective.

For all slices $(P,a,F) \in S^{\mathcal{T}}$ there are $w_1 \in P$ and $w_2 \in F$ with $w_1 a w_2 \in PaF \subseteq L = L(A_{\bar{F}})$. This implies by Lemma 1 and 6 that there is a slice $(\tilde{P},a,\tilde{F})$ in the finite slicing $\bar{F}$ with $w_1 a \in \pi_{\bar{F}}(\tilde{P},a,\tilde{F}) \subseteq \tilde{P}a \subseteq Pa = \pi_{S^{\mathcal{T}}}(P,a,F)$ and $aw_2 \in \varphi_{\bar{F}}(\tilde{P},a,\tilde{F}) \subseteq a\tilde{F} \subseteq aF = \varphi_{S^{\mathcal{T}}}(P,a,F)$.

Applying the function f to the slice $(\tilde{P}, a, \tilde{F})$ we will prove that we get the same slice, which we have started with. For that purpose, we assume a slice (P', a, F') with $f(\tilde{P}, a, \tilde{F}) = (P', a, F')$ and we are going to show that $(P, a, F) = (P', a, F')$. For that slice holds $(\tilde{P}, a, \tilde{F}) \leq (P', a, F')$ and therefore $w_1 a \in \tilde{P}a \subseteq P'a = \pi_{S^{\mathcal{T}}}(P', a, F')$ and $aw_2 \in a\tilde{F} \subseteq aF' = \varphi_{S^{\mathcal{T}}}(P', a, F')$. That means we have found to (P, a, F) another slice (P', a, F') responsible to accept the word wav. But for unambiguous $A_{S^{\mathcal{T}}}$ there can be only one slice, by Lemma 1. Thus it must be that $(P, a, F) = (P', a, F') = f(\tilde{P}, a, \tilde{F})$. We conclude that f is surjective, ie: $f(\bar{F}) = S^{\mathcal{T}}$, from which it follows that $|S^{\mathcal{T}}| \leq |\bar{F}|$ and therefore $S^{\mathcal{T}}$ is finite.

(3) $\rightarrow$ (1) : We take $A_{S^{\mathcal{T}}}$ for that automaton into account. Because of Lemma 5 it is $L(A_{S^{\mathcal{T}}}) = L$.

4 The Algorithm

In this section, an algorithm is outlined to minimize an εNFA A. It reduces the set of transitions $T_0 = \{t_0, t_1, \ldots, t_n\}$. The algorithm is given in pseudo code. Takes as input a boolean adjacency matrix $A : T_0 \times T_0 \rightarrow \{0, 1\}$ representing the follow–relation of the input automaton, ie: $A(s, t) = 1 \Leftrightarrow s \longrightarrow_A t$. Those prerequisites are fixed in the declaration (line 0-1).

The algorithm proceeds in two phases. The first pass (line 2-8), which we might name ε-completion, introduces ε-transitions without changing the original regular language. That the language doesn't change is assured by the test $L(A) = L(A_0)$ in line 6. The second pass (line 9-13) eliminates superfluous transitions in order to reduce the εNFA A. Afterwards, the algorithm returns a result in A, which is the automaton with the remaining transitions.

Algorithm

```
(0)  $T_0 = [0, 1, \ldots, n]$ ;          $\hat{=}$ $\{t_0, t_1, \ldots, t_n\}$
(1)  $A : T_0 \times T_0 \rightarrow \{0, 1\}$ ;  $\hat{=}$ $A(s, t) = 1 \Leftrightarrow s \longrightarrow_A t$
(2)  $A_0 = A$ ;
(3)  for $s = 0$ to $n$ do
(4)     for $t = 0$ to $n$ do
(5)        $A(s, t) = 1$ ;
(6)        if $L(A_0) \neq L(A)$ then $A(s, t) = 0$
(7)     end
(8)  end ;
(9)  for $s = 1$ to $n$ do
(10)    for $t = s + 1$ to $n$ do
(11)       if $l(s) = l(t)$ and $A(s, \_) = A(t, \_)$ then delete($s$)
(12)    end
(13) end
```

Theorem 3. *The algorithm on input of an εNFA A outputs an εNFA B, so that $L(A) = L(B)$ and $B \leq A$. Moreover, if B is unambiguous then B is transition minimal.*

We prove this theorem in the following subsection.

4.1 Correctness

We shall now be concerned with the correctness of the algorithm. Within the first pass we are adding ε-transition to the automaton. As a result there are more paths possible to accept the words of L. The sole chance to change the language is to accept more words, which is exactly ruled out by the test in line (6) of the algorithm. Moreover, we have

Proposition 4. *It is sufficient to check $L(A) \nsubseteq L(A_0)$ with regard to line (6).*

The second pass eliminates superfluous transitions if there is still an equivalent transition within the automaton covering all accepting paths of the transition. Hence, it still holds $L(A_0) = L(A)$ after the second pass. In fact, the only chance to fail the invariant $L(A_0) = L(A)$ is in line (5), that is instantly corrected in line (6) thereafter.

We are deleting transitions; none are added, ie: $A \leq A_0$. We have sketched that the algorithm is correct in the sense that it still returns an automaton, which accepts the same language compared to the input automaton, having a smaller or equal number of transitions.

Let us now investigate the algorithm's minimization property, ie the fact, that the returned automaton is transition minimal if it is unambiguous. For that purpose, we use the notations of the Theorem 2. Then each transition $t \in T_A$ has got assigned the strict future F_t and the strict past P_t, yielding slice $\bar{t} = (P_t, a, F_t)$ according to the automaton A's finite slicing $\bar{F}$, which is mapped via function f to $S^{\mathcal{T}}$ — recall the construction of the function f of the Theorem 2:

$$\bar{t} = (P_t, a, F_t) \in \bar{F} \quad \text{and} \quad f(\bar{t}) \in S^{\mathcal{T}}$$

Consider the first pass, the ε-completion, in which ε-transitions are placed if and only if the language is not changed. At least this is the case if and only if their correspondent transitions $f(\bar{s})$ and $f(\bar{t})$, with respect to the automaton according to $S^{\mathcal{T}}$, are connected $f(\bar{s}) \longrightarrow_{S^{\mathcal{T}}} f(\bar{t})$. We observe:

Lemma 7. *$f(\bar{s}) \longrightarrow_{S^{\mathcal{T}}} f(\bar{t})$ implies $s \longrightarrow_A t$*

Proof: By the setting of f in Theorem 2 it is $\bar{s} \leq f(\bar{s}) = (P_s, a, F_s)$ and $\bar{t} \leq f(\bar{t}) = (P_t, b, F_t)$, which implies $\pi(s) = (\pi(s){:}1)a \subseteq P_s a$ and $\varphi(t) = b(1{:}\varphi(t)) \subseteq bF_t \subseteq F_s$. We conclude $\pi(s)\varphi(t) \subseteq P_s a F_s \subseteq L$. In that case, the algorithm connects s and t by an ε-transition, which does not change the original language $L(A) = L$ we have started with.
Via the proof of Lemma 5 it also holds: $s \longrightarrow_A t \implies f(\bar{s}) \longrightarrow_{S^{\mathcal{T}}} f(\bar{t})$ in every stage of the algorithm. That is why we obtain after the first pass:

Corollary 4. *$s \longrightarrow_A t$ if and only if $f(\bar{s}) \longrightarrow_{S^{\mathcal{T}}} f(\bar{t})$*

The second pass of the algorithm deletes all except one of equivalent transitions. Two transitions s and t are equivalent if and only if there is a semantically more compact transition $x \in S^{\mathcal{T}}$ with $\bar{s} \leq x \geq \bar{t}$. Actually, this transition is $f(\bar{s}) = x = f(\bar{t})$ — see Lemma 8. We define:

Definition 10. $s \equiv t \;:\Longleftrightarrow\; f(\bar{s}) = f(\bar{t})$

Within the proof it turns out that after the first pass of the algorithm the semantics of the transitions in automaton A in terms of future and past are the same as in the automaton according to $S^{\mathcal{T}}$. The deletion of transitions cleans up by dropping the superfluous material. It results in an automaton with equal many transitions having equal semantics, compared to the one of $f(\bar{F}) = S^{\mathcal{T}}$, if $S^{\mathcal{T}}$ is unambiguous.

Lemma 8. *$f(\bar{s}) = f(\bar{t})$ if and only if $l(s) = l(t)$ and $s{\longrightarrow}_A = t{\longrightarrow}_A$*

Sketch of proof: Slices are equal iff future and past are equal — show $\pi(\bar{s}) = \pi(f(\bar{s}))$ and $\varphi(\bar{s}) = \varphi(f(\bar{s}))$ by maintaining Corollary 4. It is sufficient to check $s{\longrightarrow}_A = t{\longrightarrow}_A$ because of ϵ-completeness. □

This completes the correctness proof as the algorithm computes $f(\bar{F})$ from the input automaton. If $S^{\mathcal{T}}$ is unambiguous, it holds that $f(\bar{F}) = S^{\mathcal{T}}$. Thus the result is transition minimal. If $S^{\mathcal{T}}$ is not unambiguous, the algorithm returns a subautomaton $f(\bar{F}) \subseteq S^{\mathcal{T}}$. Considering all this, and Lemma 8, we conclude:

Corollary 5. *$S^{\mathcal{T}}$ forms a unambiguous εNFA:*

- *If there is a minimal unambiguous εNFA, then the algorithm returns it.*
- *The minimal unambiguous ϵ-complete εNFA is unique upto isomorphism.*

Corollary 6.

- *If the algorithm returns an ambiguous εNFA then a minimal εNFA is a subautomaton thereof.*

4.2 Complexity

We consider the deterministic worst case time complexity of the algorithm. Let $t(n)$ be the deterministic time complexity for the test whether $L(A) = L(B)$ or not for two εNFA A and B. Then the algorithm runs in deterministic time $\mathcal{O}(n^2 t(n) + n^3)$. At the time one only knows $t(n) \in \mathcal{O}(2^n)$. Therefore the running time of the algorithm is in

$$\mathcal{O}(n^2 2^n + n^3) \subseteq 2^{\mathcal{O}(n)}$$

The observation in Proposition 4 is interesting because it allows us to speed up the first pass. Instead of the equivalence test, we should better run an implication test:

$$L(A) \subseteq L(A_0) \Leftrightarrow L(A) \cap \neg L(A_0) = \emptyset$$

Here the most time consuming operation is the complement $\neg L(A_0) = L(\neg A_0)$ — the automaton has to be made complete and deterministic via the power set construction [21] in order to complement it — but this expression only depends on A_0 and can be precomputed. Hence, the implication test has to do the complement operation just once. In every test, to compute the intersection takes $\mathcal{O}(n^2)$ steps, combined with the test of emptiness, $\mathcal{O}(n)$. Let $t(n)$ the deterministic time complexity to compute the complement. It is $t(n) \in \mathcal{O}(2^n)$ [7]. Hence, the worst case time complexity with this optimization is

$$\mathcal{O}(t(n) + n^2(n^2 + n) + n^3) \subseteq \mathcal{O}(2^n + n^4 + 2n^3) = \mathcal{O}(2^n)$$

Moreover, there is a rich variety of possibilities to further optimize the running time of the algorithm. But this is out of the scope for this paper. Finally, it is worth to mention. that the test whether the returned automaton is unambiguous and therefore transition minimal is easy to check in deterministic time $\mathcal{O}(n^2)$, eg [24, p. 97].

5 Conclusion

Minimization of εNFA was investigated, in particular the theory and an algorithm was developed to reduce the number of transitions of a given εNFA. In general, this problem is PSPACE-complete. The algorithm presented reduces the problem of transition minimization polynomially to NFA equivalences, or alternatively to NFA complements; and runs in deterministic time $\mathcal{O}(2^n)$.

Generally observed, the union principle was exploited. By this means, two transitions t_1 and t_2 are equivalent if there is a semantically more compact transition t with $t_1 \leq t \geq t_2$. The algorithm unions the transitions such that $t_1 \cup t_2 \leq t$. Essentially, the reduction is based on partial orders; as opposed to state equivalences in the deterministic case of DFA. The fact that states resp. transitions in NFA are no objects of equivalence classes, is the main difference to other work.

The union principle is sufficient to reduce εNFA to minimal unambiguous εNFA. In other cases, the minimal εNFA it not unambiguous, a partition principle may be applicable, eg a transition t is superfluous if it can be partitioned into t_1 and t_2 with $t = t_1 \cup t_2$ such that there are more compact transitions t_1^c and t_2^c with $t_1 \leq t_1^c$ and $t_2 \leq t_2^c$.

It remains the main open question for further work whether the union and the partition principle together are sufficient to minimize the number of transitions of an εNFA, in general.

Acknowledgements. Stephan Weber and Arfst Nickelsen are gratefully thanked for their fruitful discussions on the subject. And the referees for their interesting remarks.

References

1. Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Anlalysis of Computer Algorithms.* Addison-Wesley, Reading, Massachusetts, 1974.
2. A. Arnold, A. Dicky, and M. Nivat. A note about minimal non-deterministic automata. *Bulletin of the European Association for Theoretical Computer Science*, 47:166–169, 1992.
3. Wilfried Brauer. *Automatentheorie.* Teuber, Stuttgart, 1984.
4. Wilfried Brauer. On minimizing finite automata. *Bulletin of the European Assciation for Teoretical Computer Science (EATCS)*, 35:113–116, 1988.
5. Janusz A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. In *Mathematical Theory of Automata*, pages 529–561. Polytechnic Press, Polytechnic Institute of Brooklyn, New York, 1962.

6. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman and Co, 1979.
7. Markus Holzer and Martin Kutrib. State complexity of basic operations on nondeterministic finite automata. In *Implementation and Application of Automata (CIAA '02), LNCS 2608*, pages 151–160, 2002.
8. John E. Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In *Proc. International Symposium on Theory of Machines and Computations, Technion, Haifa (IL)*, pages 189–196, 1971.
9. John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.
10. David A. Huffman. The synthesis of sequential switching circuits. *Journal of the Franklin Institute*, 257(3-4):161–190, 275–303, 1954.
11. Sebastian John. Minimal unambiguous eNFA. Technical report, TR-2003-22, Technical University Berlin.
12. Tsunehiko Kameda and Peter Weiner. On the state minimization of nondeterministic finite automata. In *IEEE Transactions on Computers*, volume C-19, pages 617–627, 1970.
13. J. Kim. State minimization of nondeterministic machines. Technical report, RC 4896, IBM Thomas J. Watson Research Center, 1974.
14. Oliver Matz and Andreas Potthoff. Computing small nondeterministic finite automata. In *Proc. Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pages 74–88, 1995.
15. Georg H. Mealy. Method for synthesizing sequential circuits. *Bell System Technical Journal*, 34:1045–1079, 1955.
16. A.R. Meyer and M.J. Fischer. Economy of description by automata, grammars, and formal systems. In *Proc. 12th Annual Symposium on Switching and Automata Theory*, pages 188–191, 1971.
17. Edward F. Moore. Gedanken-experiments on sequential machines. *Automata Studies, Annals of Mathematics Series*, 34:129–153, 1956.
18. J. Myhill. Finite automata and the representation of events. Technical report, WADC TR-57-624, Wright Patterson Air Force Base, Ohio, USA, 1957.
19. Anil Nerode. Linear automaton transformations. In *Proc. American Mathematical Society, 9*, pages 514–544, 1958.
20. Siegfried Neuber and Peter H. Starke. Über Homomorphie und Reduktion bei nicht-deterministischen Automaten. *EIK: Elektronische Informationsverarbeitung und Kybernetik*, 3:351–362, 1967.
21. Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 1959.
22. Bala Ravikumar and Oscar H. Ibarra. Relating the type of ambiguity of finite automata to the succinctness of their representation. *SIAM Journal on Computing*, 18(6):1263–1282, 1989.
23. Erik M. Schmidt. Succinctness of descriptions of context-free, regular, and finite languages. Technical Report DAIMI PB-84, Department of Computer Science, University of Aarhus, Denmark, 1978.
24. Seppo Sippu and Eljas Soisalon-Soininen. *Parsing Theory, Vol. I: Languages and Parsing.* EATCS Monographs on Theoretical Computer Science. Springer, 1988.
25. Peter H. Starke. Einige Bemerkungen über nicht-deterministische Automaten. *EIK: Elektronische Informationsverarbeitung und Kybernetik*, 2:61–82, 1966.
26. Richard E. Stearns and Harry B. Hunt, III. On the equivalence and containment problems for unambiguous regular expressions, grammars, and automata. In *IEEE: 22nd Annual Symposium on Foundations of Computer Science*, pages 74–81, 1981.