

ε -Removal by Loop Reduction for Finite-state Automata over Complete Semirings

Thomas Hanneforth¹ and Colin de la Higuera²

¹ Linguistics Dept., Potsdam University, Potsdam, Germany
`tom@ling.uni-potsdam.de`

² Laboratoire Hubert Curien, Saint-Etienne, France
`cdlh@univ-st-etienne.fr`

Abstract. We propose an incremental ε -removal algorithm for weighted finite-state automata. The key idea is that ε -cycles are stepwise reduced to ε -loops with same source and target state. The weight of these loops can be computed with a closure-operation which requires a complete semiring. Contrary to other approaches, this makes it unnecessary to approximate the contribution of ε -cycles to the weight of a path in certain (non k -closed) semirings.

1 Introduction

Many natural language processing tasks based on finite-state automata (FSA) create results which contain a lot of ε -transitions, that is, transitions which are labeled with the identity element with respect to concatenation. These ε -transitions have to be removed due to speed and memory efficiency reasons since they prevent further optimisations like determinisation and minimisation. This can be the case, for instance, in translation tasks [1]. Moreover, if the finite-state automaton is in addition *weighted* (see Section 2), standard ε -transition removal algorithms for unweighted automata³ are no longer applicable.

An approach not based on ε -closures may be called the *state-bypassing technique* (cf. [3]). This approach was generalised in [4] to weighted automata. The idea in [4] is to use a *distance algorithm* to compute the ε -distance from every source state q_p to every state q_r reachable only with ε -transitions⁴. After that, q_r is bypassed by adding transitions from q_p to all states q_s which are reachable from q_r with non- ε -transitions. Figure 1 shows this bypassing-technique schematically. A variation of the technique is to combine ε -paths from q_r to q_s with non- ε -transitions from q_p to q_r (forward vs. backward removal).

The approach in [4] has a problem with ε -cycles in certain semirings which are not *k-closed* (for example, the *probabilistic* semiring). In these cases the

³ For example, the construction of ε -closures (that is, sets of states which are only reachable by ε -transitions) which is a part of the algorithm for determinising unweighted automata [2].

⁴ All technical notions are defined in greater detail in the next section.

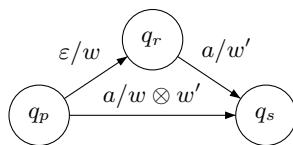


Fig. 1. ε -REMOVAL SCHEME: Whenever there is an ε -path from state q_p to state q_r with weight w , and there is a transition $q_r \xrightarrow{a/w'} q_s$, add a new transition $q_p \xrightarrow{a/w \otimes w'} q_s$. The weights of the ε -path and the a -transition are multiplicatively combined.

contribution of an ε -cycle can be computed only approximately (see also [5])⁵ We give an example in Fig. 2⁶:

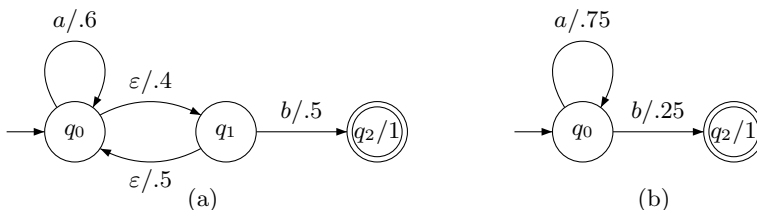


Fig. 2. PROBABILISTIC FINITE-STATE AUTOMATON WITH ε -CYCLE: (a) Before ε -removal, (b) After ε -removal.

The automaton in Fig. 2(a) accepts for example a single b . But what is the probability of b ? The simplest path is $q_0 \xrightarrow{\varepsilon/0.4} q_1 \xrightarrow{b/0.5} q_2$ with probability $0.4 \cdot 0.5 \cdot 1.0 = 0.2$. But there is a second path $q_0 \xrightarrow{\varepsilon/0.4} q_1 \xrightarrow{\varepsilon/0.5} q_0 \xrightarrow{\varepsilon/0.4} q_1 \xrightarrow{b/0.5} q_2$ with probability 0.04 . Since the probabilities of alternative paths are added (see Def. 6), we have a combined probability for the two paths of 0.24 . Since there is in fact an infinite number of paths to recognize b in Fig. 2, it is clear that every finite-number-of-steps algorithm which works in such a way that all paths are enumerated computes the probabilities only approximately. On the

⁵ Two reviewers pointed out that this misrepresents Mohri's ε -removal algorithm, since he also could have make use of an exact algorithm (for example, the Kleene/Floyd/Warshall-algorithm [8]) in the case of non k -closed semirings. While the latter is certainly true, we refer to the exposition in [4] where the exact algorithm is briefly mentioned (p. 236), but then abandoned in favour of the potentially approximate algorithm (p. 238). Of course, we appreciate Mehryar Mohri's seminal work very much.

⁶ We use the following conventions for drawing automata: the weights of transitions and final states are stated after a forward slash (/). Final states are depicted with double circles. The start state is indicated by an arrow.

other hand, if we would have prior knowledge about the cycle, it would be very easy to compute b 's joint probability. It is $\frac{1}{1-0.4 \cdot 0.5} \cdot 0.4 \cdot 0.5 = 0.25$.

The approach presented here avoids the computation of the weight of an ε -cycle with more than one intermediate state completely. Instead an ε -cycle is stepwise reduced to an ε -loop with same source and target state. The weight of this loop can be computed with a *closure*-operation which is based on the computation of the limit of a sequence.

After having settled the technical preliminaries in Section 2, we move in Section 3 to the exposition of an incremental ε -removal algorithm which is based on computing weight closures in an exact way.

2 Preliminaries

Before we move to the new algorithm, some definitions are in order. Readers familiar with the subject of semiring-weighted finite-state automata may skip this section.

Definition 1 (Semiring). *An algebraic structure $\mathcal{K} = \langle W, \oplus, \otimes, \bar{0}, \bar{1} \rangle$ is a semiring [6] if it fulfills the following conditions:*

1. $\langle W, \oplus, \bar{0} \rangle$ is a commutative monoid with $\bar{0}$ as the identity element for \oplus ,
2. $\langle W, \otimes, \bar{1} \rangle$ is a monoid with $\bar{1}$ as the identity element for \otimes ,
3. \otimes distributes over \oplus :
 $\forall x, y, z \in W : x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$ (left distributivity)
 $\forall x, y, z \in W : (y \oplus z) \otimes x = (y \otimes x) \oplus (z \otimes x)$ (right distributivity),
4. $\bar{0}$ is an annihilator for \otimes : $\forall x \in W, x \otimes \bar{0} = \bar{0} \otimes x = \bar{0}$.

In the following, a semiring \mathcal{K} is identified with its carrier set W .

Common semirings are for example the classical semiring for distance problems, the *tropical semiring* $\mathcal{T} = \langle \mathbb{R}_0^+ \cup \{+\infty\}, \min, +, +\infty, 0 \rangle$ for positive real numbers and the *Viterbi semiring* $\mathcal{V} = \langle [0 \dots 1], \max, \cdot, 0, 1 \rangle$ for probabilities.

Other semirings which are based on probabilities are the *real semiring* (also called *probabilistic semiring*) $\mathcal{R} = \langle \mathbb{R}, +, \cdot, 0, 1 \rangle$ and its isomorphic counterpart (under a $-\log$ -transformation), the *log semiring* $\mathcal{L} = \langle \mathbb{R}_\infty, +_{\log}, +, \infty, 0 \rangle$ where $x +_{\log} y = -\log(2^{-x} + 2^{-y})$. The log semiring is often used in statistical language processing because of its better numerical stability.

In addition to the semiring operations \oplus and \otimes , a derived operation called *closure* is necessary to define the weights of cycles in weighted finite-state automata.

Definition 2 (Closure). *Let $\mathcal{K} = \langle W, \oplus, \otimes, \bar{0}, \bar{1} \rangle$ be a semiring. The **closure** a^* of $a \in \mathcal{K}$ is defined as:*

$$a^* = \bigoplus_{n=0}^{\infty} a^n = \bar{1} \oplus a \oplus (a \otimes a) \oplus (a \otimes a \otimes a) \dots = \bar{1} \oplus a \oplus a^2 \oplus a^3 \dots$$

Table 1 contains equations for computing closures in different semirings.

Semiring	a^*
Real	$\sum_{n=0}^{\infty} a^n = \begin{cases} \frac{1}{1-a} & \text{if } 0 \leq a < 1 \\ \infty & \text{if } a \geq 1 \end{cases}$
Log	$\sum_{n=0}^{\infty} a^n = \begin{cases} \log(1-a) & \text{if } 0 \leq a < 1 \\ -\infty & \text{otherwise} \end{cases}$
Tropical	$\min_{n=0}^{\infty} a^n = \begin{cases} 0 & \text{if } a \geq 0 \\ -\infty & \text{otherwise} \end{cases}$

Table 1. Closure in different semirings.

Definition 3 (Semiring properties [5]). Let $\mathcal{K} = \langle W, \oplus, \otimes, \bar{0}, \bar{1} \rangle$ be a semiring.

- \mathcal{K} is **commutative** if $\forall x, y \in \mathcal{K} : x \otimes y = y \otimes x$,
- \mathcal{K} is **k -closed** (for a fixed integer k) if

$$\forall x \in \mathcal{K} : \bigoplus_{n=0}^k x^n = \bigoplus_{n=0}^{k+1} x^n,$$

- \mathcal{K} is **bounded** if $\forall x \in \mathcal{K} : x \oplus \bar{1} = \bar{1}$ (that is, $\bar{1}$ is an annihilator for \oplus).

Intuitively, k -closedness means that the closure of a weight in \mathcal{K} stabilises after k iterations and then does not change anymore. Examples for 0-closed semirings are the class of tropical semirings (over positive reals) and the Viterbi semiring. The real and log semirings are not bounded and not k -closed since there is no fixed integer k such that after k iterations the sequence converges. Note that on the other hand in bounded (that is, 0-closed) semirings $a^* = \bar{1}$, for all $a \in \mathcal{K}$.

The requirement that sums of an infinite number of elements are well-defined is expressed as *completeness* (e.g. [7]).

Definition 4 (Complete Semiring). A semiring \mathcal{K} is called *complete* if it is possible to define sums for all families $(a_i | i \in I)$ of elements in \mathcal{K} , where I is an arbitrary index set, such that the following conditions are satisfied:

- (i) $\bigoplus_{i \in \emptyset} a_i = \bar{0}$, $\bigoplus_{i \in \{j\}} a_i = a_j$, $\bigoplus_{i \in \{j, k\}} a_i = a_j \oplus a_k$ for $j \neq k$,
- (ii) $\bigoplus_{j \in J} (\bigoplus_{i \in I_j} a_i) = \bigoplus_{i \in I} a_i$, if $\bigcup_{j \in J} I_j = I$ and $I_j \cap I_{j'} = \emptyset$ for $j \neq j'$,
- (iii) $\bigoplus_{i \in I} (c \otimes a_i) = c \otimes (\bigoplus_{i \in I} a_i)$, $\bigoplus_{i \in I} (a_i \otimes c) = (\bigoplus_{i \in I} a_i) \otimes c$.

Intuitively, completeness guarantees that the weight of each cycle in a weighted finite-state automaton over semiring \mathcal{K} (see below) is in \mathcal{K} and thus well-defined. In the following, we restrict ourselves to complete semirings.

Weighted finite-state automata are an extension of the classical unweighted finite automata.

Definition 5 (Weighted finite-state automaton [5]).

A weighted finite-state automaton (WFSA) \mathcal{A} over a semiring \mathcal{K} is a 7-tuple $\langle \Sigma \cup \{\varepsilon\}, Q, q_0, F, E, \lambda, \rho \rangle$ with

1. Σ , the finite input alphabet,
2. Q , the finite set of states,
3. $q_0 \in Q$, the start state,
4. $F \subseteq Q$, the set of final states,
5. $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{K} \times Q$, the finite set of transitions,
6. $\lambda \in \mathcal{K}$, the initial weight and
7. $\rho : F \mapsto \mathcal{K}$, the final weight function mapping final states to elements in \mathcal{K} .

We call a WFSA \mathcal{A} ε -free if it does not contain ε -transitions. In that case, we may replace $\Sigma \cup \{\varepsilon\}$ in Def. 5 by Σ .

Definition 6 (Weight associated with a string [5]). Let a path $\pi = t_1 t_2 \dots t_k$ be a sequence of adjacent transitions⁷ in a WFSA \mathcal{A} . Let $\Pi(p, x, q)$ for $x \in \Sigma^*$ be the set of paths from state $p \in Q$ to state $q \in Q$ such that the concatenation of input symbols in each $\pi \in \Pi(p, x, q)$ equals x . Given a transition $t \in E$, let $w[t]$ denote the weight associated with t . Let $\omega(\pi)$ the \otimes -multiplication of all transition weights in a path $\pi: \omega(\pi = t_1 t_2 \dots t_k) = w[t_1] \otimes w[t_2] \otimes \dots \otimes w[t_k]$. The weight associated with an input string $x \in \Sigma^*$ wrt a WSFA \mathcal{A} – denoted by $\llbracket x \rrbracket_{\mathcal{A}}$ – is computed by the following equation:

$$\llbracket x \rrbracket_{\mathcal{A}} = \bigoplus_{\substack{\pi \in \Pi(q_0, x, q), \\ q \in F}} \lambda \otimes \omega(\pi) \otimes \rho(q) .$$

If $\pi = \emptyset$, $\llbracket x \rrbracket_{\mathcal{A}} = \bar{0}$.

Before we move to the algorithm, some further technical notions are required. By an ε -loop we mean an ε -transition with identical source and target state: $p \xrightarrow{\varepsilon / w} p$. In contrast, an ε -path from p to q is a non-empty sequence of adjacent ε -transitions originating in p and ending in q . If $p = q$, we call the ε -path an ε -cycle.

The ε -distance $\Delta_{\varepsilon}(p, q)$ is the \oplus -sum of all weights of ε -paths starting at state p and ending in state q :

$$\Delta_{\varepsilon}(p, q) = \bigoplus_{\pi \in \Pi(p, \varepsilon, q)} \omega(\pi) . \quad (1)$$

3 An incremental ε -removal algorithm

Algorithm 1 shows an ε -removal algorithm for arbitrary WFSA over complete semirings.

⁷ Two transitions $t_i = \langle q_i, a_i, w_i, p_i \rangle$ and $t_j = \langle q_j, a_j, w_j, p_j \rangle$ are called adjacent if $p_i = q_j$.

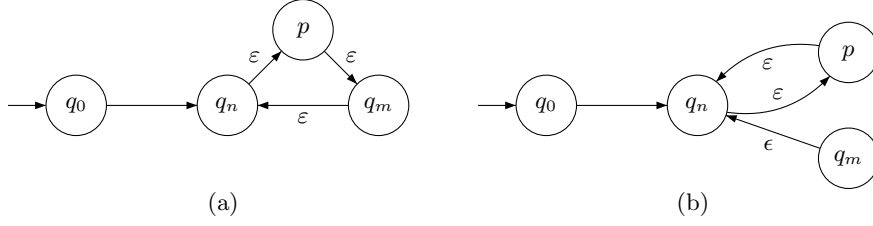


Fig. 3. ϵ -LOOPS: (a) ϵ -cycle going through q_m , (b) After the ϵ -cycle reduction.

Assuming an arbitrary order over state set Q , the algorithm maintains a queue of ϵ -transitions to be processed. It is a priority queue with an ordering relation $>$ defined as follows:

$$\langle p, w, q \rangle > \langle p', w', q' \rangle \text{ if } (p = q \wedge p' = q' \wedge q > q') \vee \quad (2)$$

$$(p \neq q \wedge p' \neq q' \wedge q > q') \vee \quad (3)$$

$$(p = q \wedge p' \neq q') \quad (4)$$

That is, ϵ -loops have always priority over non- ϵ -loops (4). If two ϵ -transitions not forming an ϵ -loop are compared, the one with the higher number of the destination state is processed first (3).

The use of a priority queue ensures the termination of the algorithm by eventually reducing a possible present ϵ -cycle going through q_m (perhaps after several steps if there are several ϵ -transitions entering q_m) to an ϵ -cycle not involving q_m and therefore only states q_n with $n < m$ with respect to the (arbitrary) state ordering. Fig. 3 shows this configuration in the simplest case.

The *foreach*-loop (lines 1–3) in Algorithm 1 inserts all ϵ -transitions of the input automaton \mathcal{A} into the queue which is processed in a while loop (lines 4–34).

In that loop, an ϵ -transition $t_\epsilon = p \xrightarrow{\epsilon/w_\epsilon} q_m$ is removed from the queue and also from the edge set E of \mathcal{A} (lines 5 and 6). If t_ϵ constitutes an ϵ -loop, all of p 's outgoing transitions t are modified by left-multiplying their weight $w[t]$ with w_ϵ^* , the closure of w_ϵ (lines 8–10).⁸ If p is a final state, p 's final weight $\rho(p)$ is modified accordingly.

If t_ϵ does not constitute an ϵ -loop (lines 14–33), two things happen:

1. Lines 15–23: for each outgoing transition $t_a = q_m \xrightarrow{a/w} q_n$ of q_m – whether labeled with ϵ or not – a transition $t = p \xrightarrow{a/w_\epsilon \otimes w} q_n$ is constructed which “bypasses” q_m by combining t_ϵ and t_a . If there exists already a similar transition t' between p and q_n , the weights of t and t' are additively combined. If t is on the other hand a transition not previously present, it is added to

⁸ Note that left-multiplication here and elsewhere is important since right multiplication would not be correct in non-commutative semirings (for example, the matrix semiring [8]).

Algorithm 1: ELIMINATING ε -TRANSITIONS.

Input: An WFSA $\mathcal{A} = \langle \Sigma \cup \{\varepsilon\}, Q, q_0, F, E, \lambda, \rho \rangle$
Output: An ε -free WFSA $\mathcal{A}' = \langle \Sigma, Q, q_0, F', E', \lambda, \rho' \rangle$

```
1 foreach  $\langle q, \varepsilon, w_\varepsilon, q' \rangle \in E$  do
2   | Enqueue(queue,  $\langle q, w_\varepsilon, q' \rangle$ )
3 end
4 while (queue  $\neq \emptyset$ ) do
5   |  $\langle p, w_\varepsilon, q_m \rangle = \text{Dequeue}(\text{queue});$ 
6   |  $E \leftarrow E \setminus \{\langle p, \varepsilon, w_\varepsilon, q_m \rangle\};$ 
7   | if ( $p = q_m$ ) then                                     /*  $\varepsilon$ -loop at state  $p$  */
8     | foreach  $\langle p, a, w, q \rangle \in E$  do                       /*  $a \in \Sigma \cup \{\varepsilon\}$  */
9       |  $E \leftarrow E \setminus \{\langle p, a, w, q \rangle\} \cup \{\langle p, a, w_\varepsilon^* \otimes w, q \rangle\}$ 
10      | end
11      | if ( $p \in F$ ) then
12        |  $\rho(p) \leftarrow w_\varepsilon^* \otimes \rho(p)$ 
13      | end
14    | else                                                 /* there are no  $\varepsilon$ -loops */
15      | foreach  $\langle q_m, a, w, q_n \rangle \in E$  do                 /*  $n < m, a \in \Sigma \cup \{\varepsilon\}$  */
16        | if ( $\exists w' : \langle p, a, w', q_n \rangle \in E$ ) then
17          |  $E \leftarrow E \setminus \{\langle p, a, w', q_n \rangle\} \cup \{\langle p, a, w' \oplus (w_\varepsilon \otimes w), q_n \rangle\}$ 
18        | else
19          |  $E \leftarrow E \cup \{\langle p, a, w_\varepsilon \otimes w, q_n \rangle\}$ 
20          | if ( $a = \varepsilon$ ) then
21            | Enqueue(queue,  $\langle p, w_\varepsilon \otimes w, q_n \rangle$ )
22          | end
23        | end
24      | end
25      | if ( $q_m \in F$ ) then
26        | if ( $p \in F$ ) then
27          |  $\rho(p) \leftarrow \rho(p) \oplus (w_\varepsilon \otimes \rho(q_m))$ 
28        | else
29          |  $F \leftarrow F \cup \{p\}$ 
30          |  $\rho(p) \leftarrow w_\varepsilon \otimes \rho(q_m)$ 
31        | end
32      | end
33    | end
34 end
35 return connect( $\langle \Sigma, Q, q_0, F, E, \lambda, \rho \rangle$ )
```

E . In case t is an ε -transition, it is also enqueued into the queue, because it can't be already there.

- Lines 25–32: If the target state q_m of the currently processed ε -transition t_ε is a final state, it is certain that p will also be a final state, since q_m is reachable by p with an ε -transition. If p already was a final state, $w_\varepsilon \otimes \rho(q_m)$ is abstractly added to $\rho(p)$.

Since some states in the original WFSA \mathcal{A} might have been only reachable by ε -transitions, a final connection step removes these states which are no longer connected after all ε -transitions have been removed.

3.1 An example

Figure 4 shows a run of the algorithm for a small example automaton over the real semiring, that is, a probabilistic WFSA (PFSA). The dashed transition is the ε -transition which is the current candidate for removal. The bold transitions (in the next figure) indicate the affected transitions after removal. The example shows how the ε -cycle between states 2 and 3 is reduced to an ε -loop at state 2 and then removed in the next step.

Notice that the example PFSA remains *stochastic* at each step of the algorithm, that is, the weights of transitions leaving a certain state q (plus $\rho(q)$, if q is a final state, add up to 1.

3.2 Correctness and complexity

Proposition 1. *For each input WFSA \mathcal{A} , Algorithm 1 terminates after a finite number of steps.*

Proof. To prove that in a finite number of steps all ε -transitions are removed, we first note that the outer while loop (lines 4–34) is executed as long as there remain ε -transitions in \mathcal{A} .

We first associate with any automaton \mathcal{A} the value $m(\mathcal{A})$ corresponding to the largest number of a state in which an ε -transition ends:

$$m(\mathcal{A}) = \max \{i \mid q_i \in Q \text{ and } \exists q' \in Q, w \in \mathcal{K} : \langle q', \varepsilon, w, q_i \rangle \in E\}. \quad (5)$$

Now let $deg_\varepsilon(\mathcal{A})$ be the number of ε -transitions ending in $q_{m(\mathcal{A})}$:

$$deg_\varepsilon(\mathcal{A}) = |\{\langle q, \varepsilon, w, q_{m(\mathcal{A})} \rangle \in E\}|. \quad (6)$$

Finally, let $n_\varepsilon(\mathcal{A})$ be the total number of ε -transitions in \mathcal{A} .

Associate with the i^{th} execution of the while loop (lines 4–34) a quantity $v(\mathcal{A}_i) = \langle m(\mathcal{A}_i), deg_\varepsilon(\mathcal{A}_i), n_\varepsilon(\mathcal{A}_i) \rangle$ reflecting the state of \mathcal{A} after the execution of the loop body.⁹

We show that each time the while loop is executed, the $v(\mathcal{A}_i)$'s will decrease with respect to the lexicographic ordering defined upon them.¹⁰

There are two cases to consider:

⁹ For example, $v(\mathcal{A}_0)$ in Fig. 4(a) is $\langle 3, 1, 4 \rangle$: $m(\mathcal{A}_0) = 3$, $deg_\varepsilon(\mathcal{A}_0) = 1$ and $n_\varepsilon(\mathcal{A}_0) = 4$.

¹⁰ Given two k -tuples $\langle x_1 \dots x_k \rangle$ and $\langle y_1 \dots y_k \rangle$, $\langle x_1 \dots x_k \rangle < \langle y_1 \dots y_k \rangle$ iff $i \geq 1$ and either $x_1 < y_1$ or $x_1 = y_1$ and $\langle x_2 \dots x_k \rangle < \langle y_2 \dots y_k \rangle$.

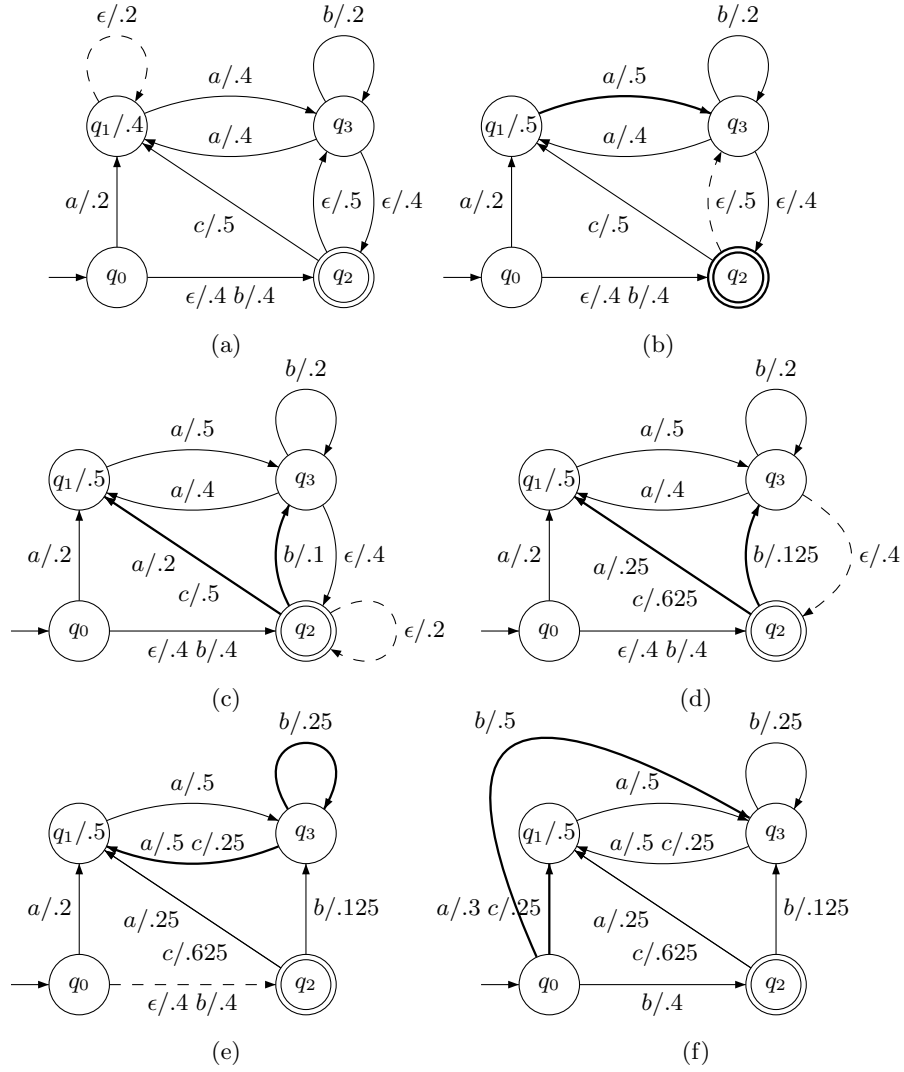


Fig. 4. A RUN OF THE ϵ -REMOVAL ALGORITHM: (a) (Stochastic) PFSA \mathcal{A} : Removal of ϵ -loop $q_1 \xrightarrow{0.2} q_1$, (b) Removal of $q_2 \xrightarrow{0.5} q_3$, (c) Removal of ϵ -loop $q_2 \xrightarrow{0.2} q_2$, (d) Removal of $q_3 \xrightarrow{0.4} q_2$, (e) Removal of $q_0 \xrightarrow{0.4} q_2$, (f) Resulting ϵ -free PFSA.

1. An ϵ -transition $p \xrightarrow{\epsilon/w_\epsilon} p$ (with same source and target state) is removed from the queue and processed in lines 7–14.
2. An ϵ -transition $p \xrightarrow{\epsilon/w_\epsilon} q_m$, with $p \neq q_m$ is removed and processed in lines 15–33.

For the first case $p \xrightarrow{\varepsilon/w_\varepsilon} p$, observe that the total number of ε -transitions $n_\varepsilon(\mathcal{A}_{i-1})$ strictly decreases since no new ε -transitions are introduced in lines 7–14. If on the one hand, $p \neq m(\mathcal{A}_{i-1})$ (for example, in Fig. 4(a) \rightarrow 4(b)), $v(\mathcal{A}_i) < v(\mathcal{A}_{i-1})$, since $n_\varepsilon(\mathcal{A}_i) < n_\varepsilon(\mathcal{A}_{i-1})$ (the first two components of $v(\mathcal{A}_{i-1})$ stay the same). On the other hand, if $p = m(\mathcal{A}_{i-1})$, $v(\mathcal{A}_i) < v(\mathcal{A}_{i-1})$ also holds, since the removal of $p \xrightarrow{\varepsilon/w_\varepsilon} p$ will decrease $\text{deg}_\varepsilon(\mathcal{A}_{i-1})$ or $m(\mathcal{A}_{i-1})$ (for example, in Fig. 4(c) \rightarrow 4(d)).

For the second case of removing a non-looping ε -transition $p \xrightarrow{\varepsilon/w_\varepsilon} q_m$, $v(\mathcal{A}_i) < v(\mathcal{A}_{i-1})$ will also hold, since by definition of the queue discipline, q_m is the highest state with ingoing ε -transitions with respect to the state ordering. If q_m has more than one incoming ε -transition (including the one to be removed), $\text{deg}_\varepsilon(\mathcal{A}_i) < \text{deg}_\varepsilon(\mathcal{A}_{i-1})$. If $p \xrightarrow{\varepsilon/w_\varepsilon} q_m$ is the only ε -transition entering q_m , $m(\mathcal{A}_i) < m(\mathcal{A}_{i-1})$. Note that $n_\varepsilon(\mathcal{A}_i) \geq n_\varepsilon(\mathcal{A}_{i-1})$ holds if q_m has at least one outgoing ε -transition. But this is irrelevant since we assumed a lexicographic ordering amongst the $v(\mathcal{A}_i)$'s.

Theorem 1. *Given an WFSA \mathcal{A} the result of Algorithm 1 is an ε -free WFSA \mathcal{B} such that $\forall x \in \Sigma^*, \llbracket x \rrbracket_{\mathcal{A}} = \llbracket x \rrbracket_{\mathcal{B}}$.*

Proof. We show that Algorithm 1 does not change the *weighted right language* of state p – the source state of the ε -transition which is going to be removed. Without loss of generality we assume that \mathcal{A} contains only a single final state f with $\rho(f) = \bar{1}$. Every WFSA \mathcal{A} can be transformed in such a way by creating a new trivially weighted final state f and adding new ε -transitions $q \xrightarrow{\varepsilon/\rho(q)} f$ from each original final state q to f and then setting $F = \{f\}$.

Given a WFSA $\mathcal{A} = \langle \Sigma \cup \{\varepsilon\}, Q, q_0, F, E, \lambda, \rho \rangle$ over a semiring \mathcal{K} , define the *weighted right language* of a state p as a function $\vec{L}_p: \Sigma^* \mapsto \mathcal{K}$ as follows:

$$\forall x \in \Sigma^*, \vec{L}_p(x) = \bigoplus_{\pi \in \Pi(p, x, \{f\})} \omega(\pi) \quad (7)$$

Depending on the existence of an ε -loop at p , there are two cases to consider:

1. There is an ε -loop at p with weight w_ε (lines 7–14): $\vec{L}_p(x)$ can be defined recursively as follows:

$$\begin{aligned} \forall x = ax' \in \Sigma^+, \vec{L}_p(x) &= \left(\bigoplus_{\langle p, \varepsilon, v, q \rangle \in E} v \otimes \vec{L}_q(x) \right) \\ &\oplus \left(\bigoplus_{\substack{\langle p, a, w, r \rangle \in E, \\ a \neq \varepsilon}} w \otimes \vec{L}_r(x') \right) \\ &\oplus (w_\varepsilon \otimes \vec{L}_p(x)) \end{aligned} \quad (8)$$

with

$$\vec{L}_p(\varepsilon) = \bigoplus_{q \in Q} \Delta_\varepsilon(p, q). \quad (9)$$

That is, $\overrightarrow{L_p}(x)$ is decomposed into three summands: (i) the right language $\overrightarrow{L_{q_i}}(x)$ of the states q_i reachable with ε and weight v_i , (ii) the right language $\overrightarrow{L_{r_i}}(x')$ of the states r_i reachable with a -transitions ($a \in \Sigma$, $x = ax'$) and weight w_i , and (iii) recursively $\overrightarrow{L_p}(x)$ itself, weighted with w_ε . Note that (i) or (ii) may be $\overline{0}$. By substituting k times Eq. (8) into its right-hand side and by using distributivity, we arrive at:

$$\begin{aligned} & \left((\overline{1} \oplus w_\varepsilon \oplus \dots \oplus w_\varepsilon^k) \otimes \left(\bigoplus_{\langle p, \varepsilon, v, q \rangle \in E} v \otimes \overrightarrow{L_q}(x) \oplus \bigoplus_{\substack{\langle p, a, w, r \rangle \in E, \\ a \neq \varepsilon}} w \otimes \overrightarrow{L_r}(x') \right) \right) \\ & \oplus (w_\varepsilon^k \otimes \overrightarrow{L_p}(x)) \end{aligned} \quad (10)$$

By letting k go to ∞ , we can replace the factor $(\overline{1} \oplus w_\varepsilon \oplus \dots \oplus w_\varepsilon^k)$ by w_ε^* (Definition 2) and $w_\varepsilon^k \otimes \overrightarrow{L_p}(x)$ by $\overline{0}$, since w_ε^k converges to $\overline{0}$ with increasing k :

$$w_\varepsilon^* \otimes \left(\left(\bigoplus_{\langle p, \varepsilon, v, q \rangle \in E} v \otimes \overrightarrow{L_q}(x) \right) \oplus \left(\bigoplus_{\substack{\langle p, a, w, r \rangle \in E \\ a \neq \varepsilon}} w \otimes \overrightarrow{L_r}(x') \right) \right) \quad (11)$$

After using distributivity again to move w_ε^* into the scope of the \oplus -operators, eq. (11) exactly states the effect of the *foreach*-loop at lines 8–10 in the algorithm.

2. The current ε -transition $p \xrightarrow{\varepsilon/w_\varepsilon} q_m$ does not form an ε -loop (lines 15–24). Since only the ε -transitions leaving p are affected by the algorithm – one transition, namely $p \xrightarrow{\varepsilon/w_\varepsilon} q_m$ was removed – it suffices to show that

$$\bigoplus_{\langle p, \varepsilon, v, q \rangle \in E} v \otimes \overrightarrow{L_q}(x) = \left(\bigoplus_{\substack{\langle p, \varepsilon, v, q' \rangle \\ \in E \setminus \{\langle p, \varepsilon, w_\varepsilon, q_m \rangle\}}} v \otimes \overrightarrow{L_{q'}}(x) \right) \oplus (w_\varepsilon \otimes \overrightarrow{L_{q_m}}(x)) \quad (12)$$

Eq. 12 holds immediately using distributivity.

Since $\overrightarrow{L_{q_0}}(x)$ is identical to $\llbracket x \rrbracket_{\mathcal{A}}$ for all $x \in \Sigma^*$, we conclude that Algorithm 1 does not change the weighted language of the input automaton \mathcal{A} . \square

Note that in bounded semirings w_ε^* is $\overline{1}$. That means that in these semirings lines 7–14 in Algorithm 1 need not to be executed.

Moving to the complexity analysis, we assume that the cost of the three semiring operations \otimes , \oplus and $*$ is c_\otimes , c_\oplus and c_* , respectively. Since in the worst case, every state $p \in Q$ is connected to every other state q by an ε -transition, the *while*-loop (4–34) in Algorithm 1 is executed $|Q|^2$ times. Removing an ε -transition from the priority queue in line 5 is in $\mathcal{O}(\log |Q|)$ time. The first *foreach*-loop (8–10) executes $|Q|$ times in the worst case, while the second *foreach*-loop

(15–24) is in $\mathcal{O}(|Q| \log |Q|)$. Assuming a Fibonacci heap for the priority queue – which takes amortised $\mathcal{O}(1)$ time for insertion (see [9]) – this can be reduced to $\mathcal{O}(|Q|)$. The other parts of the *while*-loop are in $\mathcal{O}(c_{\otimes} + c_{\oplus})$. The final connection step which removes non-accessible states can be performed in $\mathcal{O}(|Q| + |E|)$ by a depth-first search.

Putting it all together, we conclude that the running time of Algorithm 1 is in $\mathcal{O}(|Q|^2((\log |Q| + c_* + c_{\otimes}) + |Q|(c_{\otimes} + c_{\oplus})))$, irrespective of the semiring used. This upper bound is comparable to the one stated in [4] for the tropical semiring. If an adjacency list representation (see [9]) is used, the algorithm can take advantage of the possible sparsity of the automaton, since the inner *foreach*-loops have only to run over the outgoing transitions of p and q_m .

3.3 Enhancements

While the priority queue ensures termination of the *while*-loop in the presence of ε -cycles, it is clear that the ε -transitions can be removed in an arbitrary order if the underlying ε -subautomaton of \mathcal{A} – that is, the automaton $\mathcal{A}_{\varepsilon}$ which contains only the ε -transitions of \mathcal{A} – is acyclic. This can be established in time proportional to the size of $\mathcal{A}_{\varepsilon}$ using a depth-first search. In [4] it is proposed to process the transitions of the ε -subautomaton in reverse topological order when being acyclic. By always choosing an ε -transition $p \xrightarrow{\varepsilon/w_{\varepsilon}} q_m$ such that q_m does not have outgoing ε -transitions, we are able to process an ε -path of length l in l iterations of the *while*-loop.

However, as it was argued in [10], this strategy might be time and memory consuming for automata typically arising in natural language processing (NLP). Consider the stylised automaton in Fig. 5(a) which is following [10] a core pattern in applications of weighted automata to NLP.

When processing the ε -transitions in reverse topological order (see Fig. 5(b)), the state-bypassing technique attaches the two non- ε -transitions leaving q to state p . This is necessary since these two transitions carry the weight of $p \xrightarrow{\varepsilon} q$ and feed the next step. After the second ε -transition is processed, the connect-step in line 35 of Algorithm 1 will delete p (and also q) since p and q were only reachable by ε -transitions. If q has n outgoing transitions, this means to first create n transitions bypassing q and delete them afterwards.

If, on the other hand, we process the ε -transitions in forward topological order and maintain a set of reachable states (reachable with non- ε transitions), initialised with q_0 , we can prevent creating transitions for p . In Fig. 5(b), after removal of $q_0 \xrightarrow{\varepsilon} p$, we create a new transition $q_0 \xrightarrow{\varepsilon} q$. When now choosing between that one and $p \xrightarrow{\varepsilon} q$, the algorithm chooses the former, since p is only reachable by ε . Thus the creation of transitions at state p is entirely avoided.

A reachability constraint can be easily integrated into the priority queue mechanism. Again, when using Fibonacci heaps, the time complexity of Algorithm 1 remains the same, since the queue’s update operation – further states may become reachable after the state-bypassing operation – is also in $\mathcal{O}(1)$.

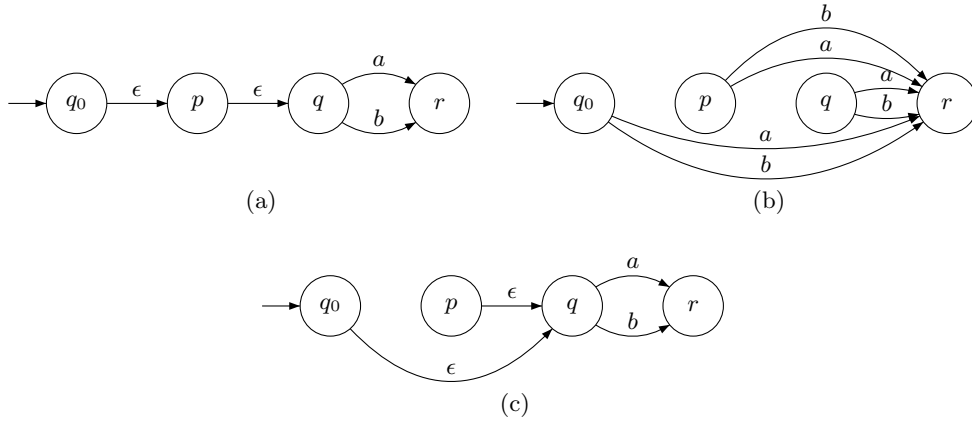


Fig. 5. ϵ -REMOVAL IN ACYCLIC \mathcal{A}_ϵ (a) WFS \mathcal{A} , (b) After ϵ -removal in reverse topological order $p \xrightarrow{\epsilon} q, q_0 \xrightarrow{\epsilon} p$, (c) After removal of $q_0 \xrightarrow{\epsilon} p$ in (forward) topological order.

4 Conclusion and further work

We presented an incremental ϵ -removal algorithm which is exact also for non- k -closed semirings. Future work will include the implementation of the algorithm and also its enhancements stated in Section 3.3.

References

1. Picó, D., Casacuberta, F.: Some statistical-estimation methods for stochastic finite-state transducers. *Machine Learning Journal* **44**(1) (2001) 121–141
2. Hopcroft, J.E., Ullman, J.D.: *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Company, Reading, MA (1979)
3. Sippu, S., Soisalon-Soininen, E.: *Parsing Theory. Volume I: Languages and Parsing*. Springer, Berlin (1988)
4. Mohri, M.: Generic Epsilon-Removal Algorithm for Weighted Automata. In Yu, S., Paun, A., eds.: *CIAA: International Conference on Implementation and Application of Automata, LNCS. Volume 2088 of Lecture Notes in Computer Science.*, Heidelberg, Springer (2001) 230–242
5. Mohri, M.: Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics* **7**(3) (2002) 321–350
6. Kuich, W., Salomaa, A.: *Semirings, Automata, Languages. Volume 5 of EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, Heidelberg (1986)
7. Ésik, Z., Kuich, W.: Equational Axioms for a Theory of Automata. In Vide, C.M., Mitran, V., Păun, G., eds.: *Formal Languages and Applications. Volume 148 of Studies in Fuzziness and Soft Computing*. Springer, Berlin, Heidelberg (2004) 183–196

8. Aho, A., Hopcroft, J., Ullman, J.: The Design and Analysis of Computer Algorithms. Addison-Wesley (1974)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. 2nd Edition. MIT Press, Cambridge, MA (2001)
10. Hanneforth, T.: A Memory-Efficient ε -Removal Algorithm for Weighted Finite-state Automata. In Piskorski, J., Watson, B., Yli-Jyrä, A., eds.: Finite-State Methods and Natural Language Processing. IOS Press, Amsterdam (2009) 72–81