Algorithms for Weighted Finite Automata with Failure Transitions

Cyril Allauzen(✉) and Michael D. Riley

Google, New York, NY, USA
allauzen@google.com, riley@google.com

Abstract. In this paper we extend several weighted finite automata (WFA) algorithms to automata with failure transitions (φ-WFAs). Failure transitions, which are taken only when no immediate match is possible at a given state, are used to compactly represent automata and have many applications. Efficient algorithms to intersect two φ-WFAs, to remove failure transitions, to trim, and to compute (over $\mathbb{R}_+$) the shortest distance in a φ-WFA are presented.

1 Introduction

Weighted finite automata are used in many applications including speech recognition [19], speech synthesis [11], machine translation [13], computational biology [10], image processing [2], and optical character recognition [7]. Such applications often have strict time and memory requirements, so efficient representations and algorithms are paramount. We examine one useful technique, the use of *failure transitions*, to represent automata compactly. A failure transition is taken only when no immediate match to the input is possible at a given state. In this paper, we will present efficient algorithms to combine, optimize and search weighted automata with failure transitions.

Aho and Corasick [1] introduce failure transitions in the context of efficient string matching from a finite set of strings input. Mohri [16] shows how to use failure transitions in string matching from finite automata input. Several authors explore constructing deterministic failure automata from arbitrary deterministic finite automata (*DFA*) for space optimization [6,15,22].

Automata with failure transitions, initially introduced for string matching problems, have found wider use including compactly representing language, pronunciation, transliteration and semantic models [3,8,12,14,20,21,27].

Mohri [18] gives a concise presentation of various fundamental weighted automata algorithms, many generalizations of classical algorithms to the weighted case. These algorithms include intersection, epsilon removal and shortest distance. Our goal here is to present similar algorithms for weighted automata with failure transitions.

This paper is organized as follows. In Sect. 2 we introduce the automata classes and related notation used here. In Sect. 3 we present the algorithms for automata with failure transitions. In Sect. 4 offer discussion and include mention of a related open-source software library.

C. Câmpeanu (Ed.): CIAA 2018, LNCS 10977, pp. 46–58, 2018.
https://doi.org/10.1007/978-3-319-94812-6_5

2 Preliminaries

2.1 Semirings

A *semiring* $K = (\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ consists of a set $\mathbb{K}$ together with an associative and commutative operation $\oplus$ and an associative operation $\otimes$, with identities $\overline{0}$ and $\overline{1}$, respectively, such that $\otimes$ distributes over $\oplus$, and $\overline{0} \otimes x = x \otimes \overline{0} = \overline{0}$. A semiring is *commutative* if the $\otimes$ operation is also commutative.

Let K be a semiring equipped with a *metric* Δ.[1] A *family* $\{x_i\}_{i \in I}$ of elements in $\mathbb{K}$ is *summable* to $x \in \mathbb{K}$ if $\forall \eta > 0$ there is a $J_\eta \subset I$ such that

$$\Delta\Big(\bigoplus_{i \in L} x_i, x\Big) \leq \eta \tag{1}$$

for all finite L with $J_\eta \subseteq L \subset I$ [25].

2.2 Weighted Automata

A *weighted finite automaton* (*WFA*) $A = (\Sigma, Q, E, i, F, \rho)$ over a semiring K is given by a finite alphabet Σ, a finite set of states Q, a finite set of transitions $E \subseteq Q \times \Sigma \times \mathbb{K} \times Q$, an initial state $i \in Q$, a set of final states $F \subseteq Q$, and a *final weight function* $\rho\colon F \to \mathbb{K}$.

A transition $e = (p[e], \ell[e], w[e], n[e]) \in E$ represents a move from the *source* or *previous* state $p[e]$ to the *destination* or *next* state $n[e]$ with the *label* $\ell[e]$ and *weight* $w[e]$. The transitions with source state q are denoted by $E[q]$.

Transitions e_1 and e_2 are *consecutive* if $n[e_i] = p[e_{i+1}]$. A path $\pi = e_1 \cdots e_n \in E^*$ is a finite sequence of consecutive transitions. The source state of a path we denote by $p[\pi]$ and the destination state by $n[\pi]$. The label of a path is the concatenation of its transition labels: $\ell[\pi] = \ell[e_1] \cdots \ell[e_n]$. The weight of a path is obtained by $\otimes$-multiplying its transition weights: $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_n]$. For a non-empty path, the i-th transition is denoted by π_i.

$P(q, q')$ denotes the set of all paths in A from q to q'. We extend this to sets in the obvious way: $P(q, R)$ denotes the set of all paths q to $q' \in R$ and so forth.

A path π is successful if it is in $P(i, F)$ and in that case the automaton is said to accept the input string $\alpha = \ell[\pi]$. The weight of $\alpha \in \Sigma^*$ assigned by the automaton is:

$$A(\alpha) = \bigoplus_{\pi \in P(i,F)\colon \ell[\pi] = \alpha} w[\pi]\rho(n[\pi]). \tag{2}$$

2.3 Weighted Automata with ϵ or φ Transitions

A *weighted finite automaton with ϵ-transitions* (ϵ-*WFA*) $A_\epsilon = (\Sigma, Q, E_\epsilon, i, F, \rho)$ is a WFA extended to allow a transition to have an empty label denoted by ϵ: $E_\epsilon \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathbb{K} \times Q$. A *weighted finite automaton with failure transitions*

[1] A metric $\Delta : \mathbb{K} \times \mathbb{K} \to \mathbb{R}_+$ satisfies (1) $\Delta(x, y) = \Delta(y, x)$, (2) $\Delta(x, y) = 0$ iff $x = y$, and (3) $\Delta(x, y) \leq \Delta(x, z) + \Delta(y, z)$ for all $x, y, z \in \mathbb{K}$.

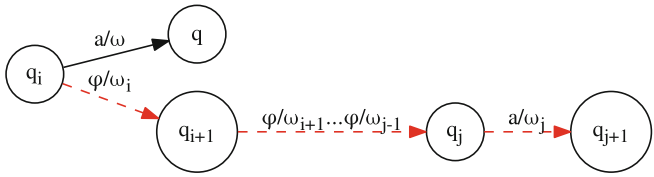

Fig. 1. The (dashed red) path $e_i = (q_i, \varphi, \omega_i, q_{i+1})$ to $e_j = (q_j, a, \omega_j, q_{j+1})$ is disallowed since a can be read already on $e = (q_i, a, \omega, q)$. (Color figure online)

(φ-*WFA*) $A_\varphi = (\Sigma, Q, E_\varphi, i, F, \rho)$ is a WFA extended to allow a transition to have a special *failure* label denoted by φ: $E_\varphi \subseteq Q \times (\Sigma \cup \{\varphi\}) \times \mathbb{K} \times Q$.

Neither ϵ and φ transitions add to a path label; they consume no input as their labels are identity elements of string concatenation for their respective automata. An ϵ-transition places no restriction on a path; it is a 'free' move. A failure transition, however, is followed only when the input can not be read immediately.

Specifically, a path $e_1 \cdots e_n$ in a φ-WFA is *disallowed* if it contains a subpath $e_i \cdots e_j$ such that $\ell[e_k] = \varphi$ for $i \leq k < j$ and there is another transition $e \in E$ such that $p[e_i] = p[e]$ and $\ell[e_j] = \ell[e] \in \Sigma$ (see Fig. 1). Since the label $a = l[e_j]$ can be read on e, we do not follow the failure transitions to read it on e_j as well.

We use $P^*(q, q') \subseteq P(q, q')$ to denote the set of (not dis-) allowed paths from q to q' in a φ-WFA. This again extends to sets in the obvious way. A path π is successful in a φ-WFA if $\pi \in P^*(i, F)$ and $\ell[\pi_{|\pi|}] \neq \varphi$ and only in that case is the input string $\alpha = \ell[\pi]$ accepted.[2]

The weight of $\alpha \in \Sigma^*$ assigned by the automaton is:

$$A_\varphi(\alpha) = \bigoplus_{\pi \in P^*(i,F):\, \ell[\pi]=\alpha, \ell[\pi_{|\pi|}] \neq \varphi} w[\pi]\rho(n[\pi]). \tag{3}$$

For these automata, we will assume there are no ϵ- or φ-labeled cycles. When there is at most one exiting failure transition per state we call the automaton φ-deterministic. We will also assume the φ-WFAs in this paper are φ-deterministic.

Two automata are *equivalent* if they accept the same strings with the same weights. Any weighted finite automaton is trivially also a φ-WFA. In the next section we describe how to remove the failure transitions from a φ-WFA to produce an equivalent φ-free WFA. As with ϵ-transitions, φ-transitions do not extend the set of weighted languages, rational power series, representable by WFAs [5].

[2] The condition that a successful path cannot end in a φ-labeled transition simplifies the presentation without loss of generality since there is an equivalent φ-WFA with the final weights propagated to the φ sources.

```
φ-Intersection(A_1, A_2)
 1  i ← (i_1, i_2)
 2  Q ← S ← {i}
 3  while S ≠ ∅ do
 4    (q_1, q_2) ← Head(S)
 5    Dequeue(S)
 6    for each e_1 ∈ E_1[q_1] do
 7      if ℓ[e_1] ≠ φ then
 8        for e_2 ∈ E_2^*(q_2) s.t. ℓ[e_1] = ℓ[e_2] do
 9          Add((q_1, q_2), ℓ[e_1], w[e_1] ⊗ w[e_2], (n[e_1], n[e_2]))
10      else
11        Add((q_1, q_2), φ, w[e_1], (n[e_1], q_2))
12    if q_1 ∈ F_1 and q_2 ∈ F_2 then
13      F ← F ∪ {(q_1, q_2)}
14      ρ(q_1, q_2) ← ρ_1(q_1) ⊗ ρ_2(q_2)
15  return A
```

```
Add(q, l, w, q')
 1  if q' ∉ Q then
 2    Q ← Q ∪ {q'}
 3    Enqueue(S, q')
 4  E ← E ∪ {(q, l, w, q')}
```

Fig. 2. Pseudocode of the intersection algorithm with failure transitions.

2.4 φ-Removed Automata

Given a φ-WFA $A = (\Sigma, Q, E, i, F, \rho)$, let the *$\varphi$-removed transitions* leaving q be defined as:

$$E^*[q] = \Big\{(q, a, \omega, q') : \quad \pi \in P^*(q, Q), a = \ell[\pi] = \ell[\pi_{|\pi|}] \in \Sigma, q' = n[\pi], \\ \omega = \bigoplus_{\pi' \in P^*(q,q'):\, a=\ell[\pi']=\ell[\pi'_{|\pi'|}],\, q'=n[\pi']} w[\pi']\Big\}$$

This is a set of (possibly new) transitions (q, a, ω, q'), one for each source state q and destination state q' of one or more a-labeled paths with optional leading failure transitions. The weight is the $\oplus$-sum of all such paths between those state pairs and with that label.

Then define the φ-removed WFA as $(\Sigma, Q, \cup_{q \in Q} E^*[q], i, F, \rho)$.

3 Algorithms

In this section we wish to extend some common WFA algorithms to the case where failure transitions are present.

3.1 Intersection

Computing the intersection of two WFAs is a fundamental operation and one of the most useful. For example, the application of an n-gram language model to an unweighted string (or more general unweighted automaton) is accomplished by intersection [3]. We extend intersection to φ-WFAs as follows:

Let K be a commutative semiring and let $A_1 = (\Sigma, Q_1, E_1, i_i, F_1, \rho_1)$ and $A_2 = (\Sigma, Q_2, E_2, i_2, F_2, \rho_2)$ be two φ-WFAs over the same alphabet. The intersection of A_1 and A_2 is a φ-WFA denoted by $A_1 \cap A_2$ and specified for all x by:

$$(A_1 \cap A_2)(x) = A_1(x) \otimes A_2(x). \tag{4}$$

Leaving aside φ-transitions, the following rule specifies how to compute a transition of $A_1 \cap A_2$ from transitions of A_1 and A_2 with matching labels: (q_1, a, ω_1, q_1') and (q_2, a, w_2, q_2') results in $((q_1, q_2), a, w_1 \otimes w_2, (q_1', q_2'))$. A simple algorithm to compute the intersection of two such automata, following the above rule, is given in [16].

The idea for extending the intersection algorithm when one or both automata have failure transitions is to output failure transitions where appropriate otherwise follow the failure transitions when matching. Figure 2 gives the pseudocode for computing $A = A_1 \cap A_2 = (\Sigma, Q, E, i, F, \rho)$ in this case.

E and F are assumed initialized to the empty set and grown as needed. The algorithm uses a queue S with arbitrary discipline to hold the state pairs yet to be examined. The state set Q is initially the pair of initial states (lines 1–2). Each time through the loop in lines 3–14, a new pair of states (q_1, q_2) is extracted from S (lines 4–5). Each non-φ-transition e_1 leaving q_1 is matched with φ-removed transitions e_2 leaving q_2 (lines 7–8). A transition is created with the matching label from (q_1, q_2) to $(n[e_1], n[e_2])$ with weight computed by $\otimes$-multiplying the weights of the matching transitions. A transition is also created for each φ-transition e_1 leaving q_1 from (q_1, q_2) to $(n[e_1], q_2)$ with weight $w[e_1]$ (lines 10–11). If q_1 and q_2 are final, the pair (q_1, q_2) is final with final weight computed by $\otimes$-multiplying the component final weights (lines 12–14).

If there are no failure transitions in A_1, this algorithm is simply the WFA intersection of A_1 with the φ-removed A_2 (lines 7–9). Failure transitions in A_1, however, appear in the output; they delay failure matching on that side (lines 10–11) similar to epsilon processing in ϵ-WFAs [16]. The choice of which automaton is used in line 6 can be generalized to be state-dependent. For example, one could use the automaton at line 6 for which $|E_i(q_i)|$ is less.

The worst-case time complexity of the algorithm is in $O(|E_1||Q_2|(m_2 + l_2 \log d_2))$, where d_i is the maximum out-degree, m_i is the maximum label multiplicity[3] and l_i the maximum length of a φ-labeled path at a state in A_i (assuming line 8 is implemented as a binary search).

3.2 φ-Removal

An algorithm to φ-remove a WFA A over alphabet Σ to produce an equivalent WFA is shown in Fig. 3. The pseudocode uses the algorithm of Fig. 2 to intersect the input with a φ-free WFA that accepts Σ^*. The intersection algorithm only outputs failure transitions from its first argument, as previously noted, and there are none. Intersection with Σ^* produces an equivalent result.

[3] Label multiplicity at state q is the maximum number of outgoing transitions in q sharing the same label.

```
φ-Removal(A)
1  E_0 ← {(i_0, a, 1̄, i_0) : a ∈ Σ}
2  ρ_0(i_0) ← 1̄
3  A_0 ← (Σ, {i_0}, E_0, i_0, {i_0}, ρ_0)
4  return φ-Intersection(A_0, A)
```

Fig. 3. Pseudocode of the φ-removal algorithm

```
φ-Trim(A)
1  Acc, Σ_⊥ ← φ-Accessible(A)
2  CoAcc ← φ-CoAccessible(A)
3  E ← {e ∈ E | ℓ[e] ∉ Σ_⊥[p[e]]}
4  Q' ← φ, E' ← φ
5  if i ∈ F then
6    Q' ← Q' ∪ {i}
7  for each e ∈ E do
8    if Useful(e) then
9      E' ← E' ∪ {e}
10     Q' ← Q' ∪ {p[e], n[e]}
11 return (Σ, Q', E', i, F ∩ Q', ρ)
```

```
Useful(e)
1  if Acc[p[e]] = undiscovered then
2    return false
3  if CoAcc[n[e]] ≠ undiscovered then
4    return true
5  if (p[e], φ, β, q') ∈ E then
6    for each e' ∈ E*[q'] s.t. ℓ[e'] = ℓ[e] do
7      if CoAcc[n[e']] ≠ undiscovered then
8        return true
9  return false
```

Fig. 4. Pseudocode to trim a φ-WFA.

3.3 Trimming

Trimming removes states and transitions from an automaton that are useless. These could arise, for example, as the by-product of an intersection algorithm. In a WFA, a state or transition is useless if it is not on a successful path or equivalently, it is not both *accessible* and *coaccessible*. A state is accessible if it can be discovered in a visitation (e.g., depth-first) from the initial state [9]. Similarly, coaccessibility can be determined in a visitation in the reverse direction from the final states [9].

For a φ-WFA we must keep each state and transition that is on a successful path or equivalently if it is both *φ-accessible* from the initial state and *φ-coaccessible* to a final state. A state q is φ-accessible if $P^*(i, q)$ is not empty. *φ-accessible* transitions and *φ-coaccessible* states and transitions are similarly defined. Unlike a WFA we may also need to retain a state or transition in order to disallow a path in a φ-WFA. For example in Fig. 7 the a-labeled transition leaving state q and its destination state are needed, regardless if on a successful path, to disallow reading φa from q to q''.

Figure 4 gives the pseudocode to φ-trim a φ-WFA A. First the *φ-accessible* and *φ-coaccessible* states are found (lines 1–2). The set of disallowed labels at a state $\Sigma_\perp$, i.e. those that are not the label of a transition leaving q that is on a φ-accessible path, is also computed at this time. See below for their implementations. The disallowed transitions are then filtered out (line 3).

```
φ-ACCESSIBLE(A)
 1  for each q ∈ Q do
 2    Acc[q] ← UNDISCOVERED
 3  Acc[i] ← DISCOVERED
 4  S ← {i}
 5  while S ≠ ∅ do
 6    q ← HEAD(S)
 7    DEQUEUE(S)
 8    N ← {e ∈ E[q] s.t. ℓ[e] ∉ Σ⊥[q] and Acc[n[e]] ∉ {DISCOVERED, VISITED}}
 9    for each e ∈ N do
10      if ℓ[e] ≠ φ then
11        Σ⊥[n[e]] ← φ
12        Acc[n[e]] ← DISCOVERED
13      else
14        φ-DISCOVER(n[e], q, N)
15        Acc[n[e]] ← φ-DISCOVERED
16      if n[e] ∉ S then
17        ENQUEUE(S, n[e])
18    if Acc[q] = DISCOVERED then
19      Acc[q] ← VISITED
20    else
21      Acc[q] ← φ-VISITED
22    return Acc, Σ⊥
```

```
φ-DISCOVER(n, q, N)
1  if Acc[n] = UNDISCOVERED then
2    Σ⊥[n] ← Σ⊥[q] ∪ {l[e] ∈ Σ | e ∈ N}
3  else
4    Σ⊥[n] ← Σ⊥[n] ∩ (Σ⊥[q] ∪ {l[e] ∈ Σ | e ∈ N})
```

Fig. 5. Pseudocode to determine which states are φ-accessible and which labels are disallowed at a state.

The automaton is then examined for useful transitions and any found are retained in the result (lines 4–11). A transition is useful if its source state is φ-accessible and its destination is φ-coaccessible (lines 1–4) or if it is needed to forbid a path (lines 5–8).

Figure 5 gives the pseudocode for computing φ-accessibility. The algorithm uses an arbitrary queue S containing the states to be processed. Each time through the loop in lines 5–21, a new state q is extracted from S (lines 6–7). Each transition e leaving q that is not already *discovered*, *visited* or has a disallowed label is considered in turn (lines 8–9). If it is a non-φ-transition, then its destination state is marked as discovered and cleared of any previously disallowed labels (lines 10–12). Otherwise, it is a φ-transition and its destination state is marked as φ-*discovered* (line 14–15).

In this φ-discovered case, the disallowed labels at q together with the transition labels leaving q become the disallowed labels at $n[q]$ if just discovered (lines 1–2). Otherwise, the existing set is filtered by the disallowed labels at q and any newly allowed transition labels leaving q (lines 3–4).

Once a transition is processed, its destination state $n[q]$ is enqueued if needed (lines 16–17). Once all the transitions leaving q are processed, state q is marked as visited or φ-*visited* if previously discovered or φ-*discovered* (lines 18–21).

φ-coaccessibility on the φ-accessible component of A can be computed with a standard coaccessibility computation if restricted to paths that are φ-accessible, as previously found, and that do not end in a φ-labeled transition. The time complexity of trimming is dominated by the computation of φ-accessibility, which is in $O(l(|E| + (C_S + \log d)|Q|))$ where d is the maximum out-degree, l is the maximum length of a φ-labeled path in A, and C_S is the maximum cost of a queue operation for S.

3.4 Shortest Distance

Shortest-distance algorithms play a central role in applications on weighted automata requiring searching, counting, normalization or approximation [3,20,23,27]. In these applications, the tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ and the positive real semiring $\mathcal{R}_+ = (\mathbb{R}_+, +, \times, 0, 1)$ are among the most widely used.

Shortest Distance on WFAs and ϵ-WFAs. The *shortest distance* from the initial state i to state q in a WFA A over semiring K is defined as:

$$\delta[q] = \bigoplus_{\pi \in P(i,q)} w[\pi] \tag{5}$$

when well-defined and in $\mathbb{K}$ [17]. Mohri presented an algorithm to compute this shortest distance that is often much more efficient than alternatives such as Floyd-Warshall [17]. The pseudocode is shown in Fig. 6. We show his extended version where $\mathbb{K}$ is equipped with a metric Δ and an ϵ threshold.

The algorithm is a generalization of classical shortest distance algorithms over the tropical semiring[4] to more general semirings [9]. The algorithm uses a queue S to extract states (line 7) whose transitions e are *relaxed* to update $d[n[e]]$ (lines 11–13), the estimate of the shortest distance to that state. Unlike the classical algorithms, a second array $r[q]$ is maintained (lines 9, 14) that ensures that the weights added to $d[q]$ from paths in $P(i, q)$ are applied only once, important for the non-idempotent case (i.e. when $a \oplus a \neq a$). See [17] for the detailed proofs.

Mohri proved exact computation of $\delta[q]$ (with $\Delta(x, y) = 1_{x \neq y}$ and $\epsilon = 1$ for any queue S) when the input is acyclic, is over a k-closed semiring, or is k-closed for A.[5] For other semirings such as $\mathcal{R}_+$, it is an approximation algorithm controlled by ϵ. We prove convergence and correctness for $\mathcal{R}_+$ below.

Theorem 1. *Let A be a WFA over $\mathcal{R}_+$ equipped with the usual metric $\Delta(x, y) = |x - y|$. Assume the family of path weights $\{w[\pi]\}_{\pi \in P(i,q)}$ defining $\delta[q]$ is summable for all $q \in Q$. Then* ShortestDistance*(A, S, ϵ) terminates for any queue S and any $\epsilon > 0$. Further for any $\eta > 0$, there is an $\epsilon > 0$ such that at termination $\Delta(d[q], \delta[q]) \leq \eta$ for any $q \in Q$.*

[4] Such as Dijkstra or Bellman-Ford with the appropriate queue disciplines on S.

[5] Semiring K is k-closed if for all a in $\mathbb{K}$, $\bigoplus_{i=0}^{k+1} a^i = \bigoplus_{i=0}^{k} a^i$. It is k-closed for A if the weight a of each cycle in A verifies $\bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$. The tropical semiring is 0-closed [17].

To prove this theorem, we first introduce a lemma. As in [17], define finite $D(q) \subseteq P(i, q)$, $q \in Q$ as the set of all paths whose weight have been added so far to $d[q]$ at some time in the execution of the program.

```
SHORTESTDISTANCE(A, S, ϵ)
 1  for each q ∈ Q do
 2    d[q] ← r[q] ← 0̄
 3  d[i] ← r[i] ← 1̄
 4  S ← {i}
 5  while S ≠ ∅ do
 6    q ← HEAD(S)
 7    DEQUEUE(S)
 8    r′ ← r[q]
 9    r[q] ← 0̄
10    for each e ∈ E[q] do
11      d′ ← d[n[e]] ⊕ (r′ ⊗ w[e])
12      if Δ(d[n[e]], d′) ≥ ϵ then
13        d[n[e]] ← d′
14        r[n[e]] ← r[n[e]] ⊕ (r′ ⊗ w[e])
15        if n[e] ∉ S then
16          ENQUEUE(S, n[e])
17    d[i] ← 1̄
18  return d

φ-SHORTESTDISTANCE(A, S, ϵ)
 1  return SHORTESTDISTANCE(A_ϵ, S_ϵ, ϵ)
```

```
ENQUEUE(S_ϵ, q)
 1  if q ∈ Q then
 2    ENQUEUE(S, q)
 3  else
 4    e ← (q − |Q|), φ, β, q′) ∈ E
 5    ENQUEUE(S_fifo[n[e]], q)

HEAD(S_ϵ)
 1  q ← HEAD(S)
 2  if S_fifo[q] ≠ φ then
 3    return HEAD(S_fifo[q])
 4  else
 5    return q

DEQUEUE(S_ϵ)
 1  q ← HEAD(S)
 2  if S_fifo[q] ≠ φ then
 3    DEQUEUE(S_fifo[q])
 4  else
 5    DEQUEUE(S)
```

Fig. 6. Shortest-distance algorithm without [17] and with failure transitions.

Lemma 1. *For any path $\pi \in P(i, q)$, there is a $\theta_\pi > 0$ such that π is in $D(n[\pi])$ after some point in the execution of the algorithm provided the algorithm is run with $\epsilon \leq \theta_\pi$ and it terminates.*

Proof. If $|\pi| = 0$, then $\pi \in D(i)$ with $\theta_\pi = 1$ by line 3. If $|\pi| > 0$, let $\pi = \tau e$, $e \in E$ and assume, by induction, $\tau \in D(n[\tau])$ for all $\epsilon \leq \theta_\tau$. Let $\theta_\pi = \min\{\theta_\tau, w[\pi]\}$. State $n[\tau]$ was enqueued since $\tau \in D(n[\tau])$. When it is dequeued, which must happen since the algorithm is assumed to terminate, line 11 will succeed for π. This follows since $\Delta(d[n[e]], d[n[e]] \oplus (r' \otimes w[e])) = |r'w[e]| \geq |(w[\tau] + x)\, w[e]| \geq w[\pi] \geq \theta_\pi$, where x represents the weight of any other paths added to r'. Thus π is added to $D(n[\pi])$ at line 12 for all $\epsilon \leq \theta_\pi$.

Proof (Theorem 1). Termination: The $d[q]$ form a monotone increasing sequence of partial sums $d_1, d_2, \ldots$ during the execution of the program and are bounded above by $\delta(q)$. This ensures the condition in line 11 succeeds only finitely many times.

Convergence to $\delta[q]$: Select finite $J_\eta(q)$ to satisfy Eq. 1 for the summable family $\{w[\pi]\}_{\pi \in P(i,q)}$ for $\eta > 0$. By Lemma 1 there is a $\theta_q = \min_{\pi \in J_\eta(q)} \theta_\pi$ such that $J_\eta(q) \subseteq D(q)$ and by Eq. 1

$$\Delta\Big(\bigoplus_{\pi \in D(q)} w[\pi], \delta[q] \Big) \leq \eta$$

provided $\epsilon \leq \theta_q$. But $d[q] = \bigoplus_{\pi \in D(q)} w[\pi]$ as shown in [17]. So if we select ϵ as $\min_{q \in Q} \theta_q$, we prove the theorem. □

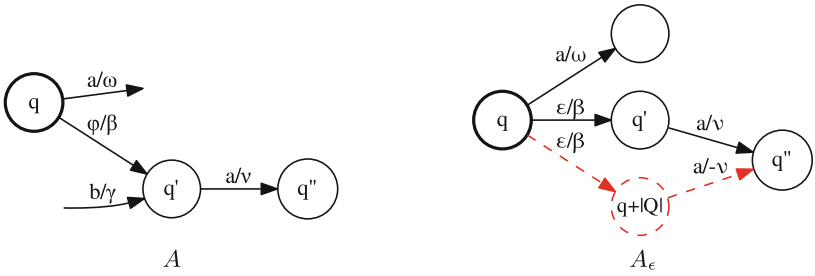

Fig. 7. Failure transitions in A are replaced by ϵ-transitions in A_ϵ. To compensate for the formerly disallowed paths, new (dashed red) negatively-weighted paths are added. (Color figure online)

The time complexity of the algorithm is in $O(N_\epsilon(|E| + C_S|Q|))$ where N_ϵ is the maximum number of times a state can be enqueued given the threshold ϵ and C_S is the maximum cost of a queue operation for S [17].

Since the transition labels play no role in the definitions and results of this section, they apply equally to ϵ-WFAs. In the next section, the labels matter.

Shortest Distance on φ-WFAs. We define the *shortest distance* from the initial state i to state q in a φ-WFA as:

$$\delta[q] = \bigoplus_{\pi \in P^*(i,q)} w[\pi] \tag{6}$$

when well-defined and in $\mathbb{K}$.

We present an algorithm for the positive real semiring $\mathcal{R}_+$, an important case in applications. To do so, we will transform φ-WFA A on $\mathcal{R}_+$ to an equivalent ϵ-WFA A_ϵ on $\mathcal{R} = (\mathbb{R}, +, *, 0, 1)$. We can then use the ShortestDistance algorithm in Fig. 6, suitably adapted.

Given a φ-WFA $A = (\Sigma, Q, E, i, F, \rho)$ with $Q = \{1, \ldots, |Q|\}$ define $Q_\epsilon = \{1, \ldots, 2|Q|\}$ and

$$\begin{aligned} E_\epsilon = &\{(q, a, \omega, q') \in E : a \in \Sigma\} \cup \\ &\{(q, \epsilon, \beta, q') : (q, \varphi, \beta, q') \in E\} \cup \{(q, \epsilon, \beta, q + |Q|) : (q, \varphi, \beta, q') \in E\} \cup \\ &\{(q + |Q|, a, -\nu, q'') : (q, \varphi, \beta, q')(q', a, \nu, q'') \in P(q, q''),\ a \in \Sigma\}. \end{aligned}$$

Then let the ϵ-WFA be $A_\epsilon = (\Sigma, Q_\epsilon, E_\epsilon, i, F, \rho)$. Each failure transition in A is relabeled with an ϵ in A_ϵ, allowing previously disallowed paths. To compensate for this, each disallowed consecutive φ- and a-labeled transition pair in A also has a corresponding negatively-weighted consecutive φ- and a-labeled transition pair in A_ϵ that cancels it (see Fig. 7) [20,23].

If A is acyclic, the family of path weights defining $\delta[q]$ is summable in A_ϵ since finite and ShortestDistance can be applied. The cyclic case requires more care; the presence of negative weights means that the partial sums may diverge or converge to something other than $\delta[q]$ depending on the ordering of the summands (cf. the Reimann Rearrangement Theorem [24]). We will select a queue discipline for the new states in A_ϵ to ensure the correct behavior. The idea is to have the negatively-weighted terms be immediately cancelled by their positively-weighted counterparts in the running of the algorithm (cf. [26]). Figure 6 shows the pseudocode for the queue and the general shortest distance algorithm with failure transitions.

The queue S_ϵ enqueues states in Q in the arbitrary subqueue S (lines 1–2). New states $q + |Q|$, $q \in Q$ are enqueued in a FIFO subqueue array $S_{\text{fifo}}[q']$ indexed by the φ-successor state q' of q (lines 3–5). S_ϵ ensures that any new state is dequeued from the subqueue array (lines 2–3) just before its index state (lines 4–5). In this way, the positive and negative weights of the disallowed a transitions will immediately cancel in $d[q'']$ at relaxation (see Fig. 7).[6]

The complexity is the same as in the φ-free case since $|Q_\epsilon| = 2|Q|$, $|E_\epsilon| < 2|E|$ and $C_{S_\epsilon} \in O(C_S)$.

4 Discussion

The intersection algorithm presented ensures φ-determinism in the result with φ-deterministic input. Mohri and Yang [20] describe an alternative algorithm that adds φ-non-determinism but avoids the φ-removal $E_2^*[q_2]$.

The proof of Theorem 1 takes advantage of rather specific (complete, monotonic, total) order properties of $\mathbb{R}$ rather than more general metric properties.[7] Can this shortest distance algorithm be extended to further semirings? It is easy to see that if the algorithm is correct with semirings K_1 and K_2 having metrics Δ_1 and Δ_2, then $K_1 \times K_2$ having the square metric $\Delta(x, y) = \max(\Delta_1(x, y), \Delta_2(x, y))$ is also correct.

These algorithms have been implemented as part of *SFST*, an open-source C++ library for normalizing, sampling, combining, and approximating *stochastic* finite-state transducers [4].

[6] We could also add logic so that when line 12 of the shortest distance algorithm is executed for a disallowed transition then it is also always executed for any negative compensating transition in case $|r[q + |Q|]\nu| < \epsilon < |r[q']\nu|$. This however is an unneeded precaution since with small enough ϵ any discrepancy is insignificant compared to the floating-point precision of $d[q'']$.

[7] The real numbers can be defined axiomatically as a field with a complete, monotonic total order [24].

References

1. Aho, A.V., Corasick, M.J.: Efficient string matching: an aid to bibliographic search. Commun. ACM **18**(6), 333–340 (1975)
2. Albert, J., Kari, J.: Digital image compression. In: Droste, M., Kuich, W., Vogler, H. (eds.) Handbook of Weighted Automata. Monographs in Theoretical Computer Science. An EATCS Series, pp. 453–479. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01492-5_11
3. Allauzen, C., Mohri, M., Roark, B.: Generalized algorithms for constructing language models. In: Proceedings of ACL, pp. 40–47 (2003)
4. Allauzen, C., Riley, M.: SFST: Stochastic FST Library (2017). http://sfst.opengrm.org
5. Berstel, J., Reutenauer, C.: Rational Series and Their Languages. Springer, Heidelberg (1988)
6. Björklund, H., Björklund, J., Zechner, N.: Compact representation of finite automata with failure transitions. Technical report, UMINF 13.11, Umeå University (2013)
7. Breuel, T.M.: The OCRopus open source OCR system. In: Proceedings of IS&T/SPIE 20th Annual Symposium (2008)
8. Chen, S., Goodman, J.: An empirical study of smoothing techniques for language modeling. Technical report, TR-10-98, Harvard University (1998)
9. Cormen, T., Leiserson, C., Rivest, R.: Introduction to Algorithms. MITP (1992)
10. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.J.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, London (1998)
11. Ebden, P., Sproat, R.: The Kestrel TTS text normalization system. Nat. Lang. Eng. **21**(3), 333–353 (2015)
12. Hellsten, L., Roark, B., Goyal, P., Allauzen, C., Beaufays, F., Ouyang, T., Riley, M., Rybach, D.: Transliterated mobile keyboard input via weighted finite-state transducers. In: FSMNLP 2017, pp. 10–19 (2017)
13. Iglesias, G., Allauzen, C., Byrne, W., de Gispert, A., Riley, M.: Hierarchical phrase-based translation representations. In: EMNLP 2011, pp. 1373–1383 (2011)
14. Katz, S.M.: Estimation of probabilities from sparse data for the language model component of a speech recogniser. IEEE Trans. Acoust. Speech Signal Process. **35**(3), 400–401 (1987)
15. Kourie, D.G., Watson, B.W., Cleophas, L.G., Venter, F.: Failure deterministic finite automata. In: Stringology, pp. 28–41 (2012)
16. Mohri, M.: String-matching with automata. Nord. J. Comput. **4**(2), 217–231 (1997)
17. Mohri, M.: Semiring frameworks and algorithms for shortest-distance problems. J. Automata Lang. Comb. **7**(3), 321–350 (2002)
18. Mohri, M.: Weighted automata algorithms. In: Droste, M., Kuich, W., Vogler, H. (eds.) Handbook of Weighted Automata. Monographs in Theoretical Computer Science. An EATCS Series, pp. 213–254. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01492-5_6
19. Mohri, M., Pereira, F., Riley, M.: Speech recognition with weighted finite-state transducers. In: Benesty, J., Sondhi, M.M., Huang, Y.A. (eds.) Springer Handbook of Speech Processing. SH, pp. 559–584. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-49127-9_28
20. Mohri, M., Yang, S.: Competing with automata-based expert sequences. In: Proceedings of AISTATS (2018)

21. Novak, J.R., Minematsu, N., Hirose, K.: Failure transitions for joint n-gram models and G2P conversion. In: INTERSPEECH, pp. 1821–1825 (2013)
22. Nxumalo, M., Kourie, D.G., Cleophas, L., Watson, B.W.: An assessment of algorithms for deriving failure deterministic finite automata. S. Afr. Comput. J. **29**(1), 43–68 (2017)
23. Roark, B., Allauzen, C., Riley, M.: Smoothed marginal distribution constraints for language modeling. In: ACL 2013, vol. 1, pp. 43–52 (2013)
24. Rudin, W.: Principles of Mathematical Analysis, vol. 3. McGraw-Hill, New York (1964)
25. Sakarovich, J.: Rational and recognizable power series. In: Droste, M., Kuich, W., Vogler, H. (eds.) Handbook of Weighted Automata. Monographs in Theoretical Computer Science. An EATCS Series, pp. 105–174. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01492-5_4
26. Schaefer, P.: Sum-preserving rearrangements of infinite series. Am. Math. Monthly **88**(1), 33–40 (1981)
27. Velikovich, L.: Semantic model for fast tagging of word lattices. In: 2016 IEEE Spoken Language Technology Workshop (SLT), pp. 398–405. IEEE (2016)