Extraction of ε-Cycles from Finite-State Transducers

André Kempe

Xerox Research Centre Europe – Grenoble Laboratory
6 chemin de Maupertuis – 38240 Meylan – France
andre.kempe@xrce.xerox.com – http://www.xrce.xerox.com/research/mltt

Abstract. Much attention has been brought to determinization and ε-removal in previous work. This article describes an algorithm for extracting all ε-cycles, which are a special type of non-determinism, from an arbitrary finite-state transducer (FST). The algorithm factorizes (decomposes) the FST, T, into two FSTs, T_1 and T_2, such that T_1 contains no ε-cycles and T_2 contains all ε-cycles of T. Since ε-cycles are an obstacle for some algorithms such as the factorization of ambiguous FSTs, the proposed approach allows us to by-pass this problem. ε-Cycles can be extracted before and re-inserted (by composition) after such algorithms.

1 Introduction

Much attention has been brought to the problem of non-determinism. There has been work on both determinization in general and ε-removal [1, 6, 5, among many others].

This article describes an algorithm for extracting all ε-cycles, which represent a special type of non-determinism consisting of consecutive arcs with the empty string ε as input label, from an arbitrary finite-state transducer (FST). The algorithm factorizes (decomposes) the FST, T, into two FSTs, T_1 and T_2, such that T_1 contains no ε-cycles and T_2 contains all ε-cycles of T. Jointly in a cascade, T_1 and T_2 describe the same relation and perform the same mapping as T.

Motivation: Some algorithms, such as the factorization of ambiguous FSTs [8, 7, 4], can only be performed on *real-time FSTs*, where every arc has exactly one symbol on the input side. Arcs with ε as input label are an obstacle for such algorithms. In many cases, an FST can be made real-time by removing its ε-arcs and concatenating their output labels with the output of adjacent non-ε-arcs. This classical method, however, is not applicable to FSTs with ε-cycles. To by-pass the problem, the ε-cycles of an FST, T, can be extracted by the approach below, where T is factorized into T_1 and T_2. Then, the ε-cycle-free and (at most) finitely ambiguous T_1 can be made real-time and further factorized into a sequential $T_{1,1}$ and an ambiguous flower transducer $T_{1,2}$ that contains no failing paths for any output string of $T_{1,1}$ [4]. Finally, the ε-cycles can be re-inserted by composing $T_{1,2}$ with T_2.

1.1 Conventions

Input and output side: Although FSTs are inherently bidirectional, they are often intended to be used in a given direction. The proposed algorithm is performed relative to the direction of application. In this article, the two sides (or tapes or levels) of an FST are referred to as *input side* and *output side*.

Examples of finite-state networks: Every example is shown in one or more figures. The first figure usually shows the original network. Possible following figures show modified forms of the same example. For example, Example 1 is shown in Figure 1 to Figure 3.

Finite-state graphs: Every FST has one initial state, labeled with number 0, and one or more final states marked by double circles. The initial state can also be final. All other state numbers and all arc numbers have no meaning for the FST but are just used to reference a state or an arc. An arc with n labels designates a set of n arcs with one label each that all have the same source and destination. In a symbol pair occurring as an arc label, the first symbol is the input and the second the output symbol. For example, in the symbol pair `a:b`, `a` is the input and `b` the output symbol. Simple, i.e., unpaired labels represent identity pairs. For example, `a` means `a:a`.

Composition: In $T_1 \diamondsuit T_2 \diamondsuit T_3 = T_3 \circ T_2 \circ T_1$, the FST T_1 is applied first and T_3 last [2]. We will use the $\diamondsuit$-operator because we prefer left-to-right notation (and application) in general, and find it clearer in examples such as $(\texttt{a:b}) \diamondsuit (\texttt{b:c}) \diamondsuit (\texttt{c:d}) = (\texttt{a:d})$, compared to $(\texttt{c:d}) \circ (\texttt{b:c}) \circ (\texttt{a:b}) = (\texttt{a:d})$.

Special symbols: The "`?`" denotes any symbol (except ε or $\hat{\varepsilon}$) when it is used in a regular expression. Both ε and $\hat{\varepsilon}$ mean the empty string and have the same effect when the FST is applied to an input sequence, but $\hat{\varepsilon}$ should be preserved in minimization and determinization. Greek letters are used to denote auxiliary symbols. Those have a "special" meaning and are distinct from the ordinary input and output symbols.

1.2 Preliminaries

An FST can be described by the six-tuple $T = \langle \Sigma, \Delta, Q, i, F, E \rangle$ with an input alphabet Σ, an output alphabet Δ, a state set Q, an initial state $i \in Q$, a set of final states $F \subseteq Q$, and a set of transitions E.

Given a transition $e \in E$, we denote its input label by $i(e)$, its output label by $o(e)$, its source state by $p(e)$, and its destination state by $n(e)$. The transition can be described by the quadruple $e = \langle p(e), i(e), o(e), n(e) \rangle$. Given a state $q \in Q$, we denote the set of its outgoing transitions by $E(q)$ and the set of its incoming transitions by $E^R(q)$. A path $\pi = e_1 \cdots e_k$ is an element of E^* with consecutive transitions. To express that a transition e is on a path π, we write $e \,\overline{\in}\, \pi$. To refer to a particular path in a figure, we give the arc numbers in ceiling brackets; e.g., $\pi = \lceil$*100, 101, 102, 103*$\rceil$ is a path consisting of the four named arcs. We denote by $P(q, q')$ the set of all paths $\pi_i(q, q')$ from q to q', by $C(q)$ the set of all cycles on q (i.e., all paths from q to q), and by $C_\varepsilon(q)$ the set of all ε-cycles on q , i.e.,

those cycles consisting only of arcs with ε as input label:

$$P(q,q') = \bigcup_i \{\pi_i(q,q')\} \tag{1}$$

$$C(q) = P(q,q) \tag{2}$$

$$C_\varepsilon(q) = \{\pi \in C(q) \mid \forall e \,\overline{\overline{\in}}\, \pi,\ i(e) = \varepsilon\} \tag{3}$$

We are particularly interested in simple ε-cycles $\widehat{C}_\varepsilon(q)$ on a state q which do not traverse any state more than once:

$$\widehat{C}_\varepsilon(q) \subseteq C_\varepsilon(q) \tag{4}$$

$$\widehat{C}_\varepsilon(q) = \{\pi \in C_\varepsilon(q) \mid \forall e, e' \,\overline{\overline{\in}}\, \pi,\ e \neq e' \Rightarrow n(e) \neq n(e')\} \tag{5}$$

We extend the notion of input and output labels to paths and sets of paths, cycles, or ε-cycles, and denote their sequences of input and output labels by $i(\pi(q,q'))$, $o(\pi(q,q'))$, $i(C_\varepsilon(q))$, $o(C_\varepsilon(q))$, etc. Note that $i(\,)$, $o(\,)$, and their arguments can be single elements or sets.

2 Basic Idea

Any arbitrary FST, T, containing ε-cycles can be factorized (decomposed) into two FSTs, T_1 and T_2, such that T_1 contains no ε-cycles and is therefore at most finitely ambiguous, and T_2 contains all ε-cycles of T. The set of ε-cycles $C_\varepsilon(q_i)$ of every state q_i in T is represented by a single arc mapping ε to an auxiliary symbol ξ_i in T_1. Instead of (perhaps infinitely) traversing $C_\varepsilon(q_i)$, ξ_i is emitted. All ξ_i are then mapped to the corresponding original $C_\varepsilon(q_i)$ in T_2 :

$$C_\varepsilon(q_i) \quad \longrightarrow \quad (\varepsilon : \xi_i) \quad \diamondsuit \quad (\xi_i : o(C_\varepsilon(q_i))) \tag{6}$$

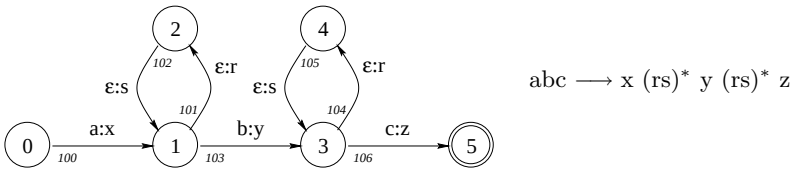


Fig. 1. Transducer T with ε-cycles (Example 1)

Figure 1 shows a simple example of an FST with two ε-cycles, $C_\varepsilon(1) = \{\lceil 101, 102 \rceil\}$ and $C_\varepsilon(3) = \{\lceil 104, 105 \rceil\}$. The FST maps the input string `abc` to the output string `xyz`, and inserts an arbitrary number of substrings `rs` inside.

Figure 2 shows the same example after the extraction of ε-cycles (factorization). T_1 maps the input string `abc` to the intermediate string $\mathtt{x}\xi_1\mathtt{y}\xi_3\mathtt{z}$ (Fig. 2a). T_2 maps the auxiliary symbols, ξ_1 and ξ_3, to ε-cycles, and every other symbol of the intermediate string to itself (Fig. 2b). Although the auxiliary symbols are single symbols, they describe (sets of) ε-cycles. Since actually ξ_1 and ξ_3 describe

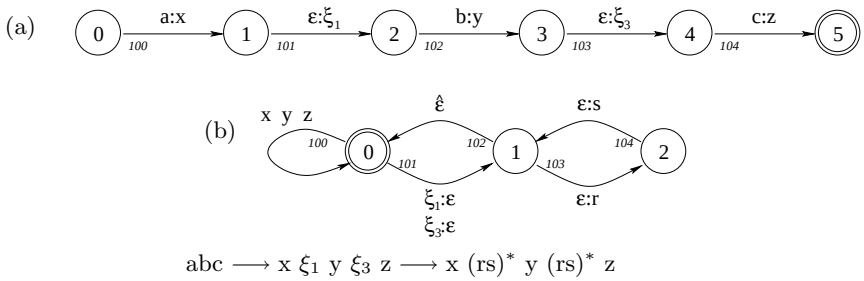


Fig. 2. Factorization of T into (a) an ε-cycle-free T_1 that emits auxiliary symbols, and (b) a T_2 that maps auxiliary symbols to ε-cycles (Example 1)

equal ε-cycles in this example, it would be sufficient to use two occurrences of the same auxiliary symbol, e.g. ξ_1, instead. In such cases, the number of auxiliary symbols can be reduced a posteriori [3]. The $\hat{\varepsilon}$ denotes the empty string, like ε, but it should be preserved in minimization and determinization. Otherwise T_2 would become larger (Example 1 Fig. 2b, and Example 2 Fig. 9b).

T_1 can be converted into a real-time FST, without ε-arcs, by removing the ε-arcs and concatenating their output symbols with the output of adjacent non-ε-arcs.

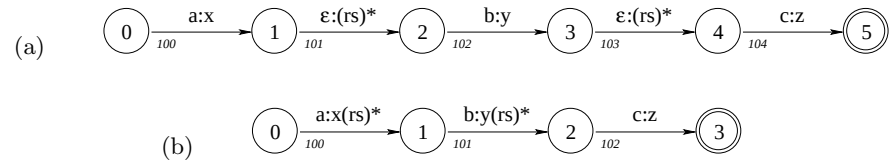
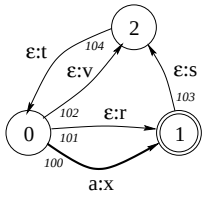


Fig. 3. Alternative representation of ε-cycles by complex labels (a) with ε-arcs or (b) as a real-time transducer without ε-arcs (Example 1)

An alternative to factorizing T into T_1 and T_2 would be representing T by a single FST, $\widehat{T}$, that is similar to T_1 but with more complex output labels that directly describe sets of ε-cycles $C_\varepsilon(q_i)$. Every $C_\varepsilon(q_i)$ in T would be reduced to a single ε-arc in $\widehat{T}$ (Fig. 3a). $\widehat{T}$ can be further converted into a real-time FST, without ε-arcs (Fig. 3b). This representation of ε-cycles is similarly to what can be seen, e.g., in [7, p. 221, Fig. 6], and is equivalent to our representation by two FSTs. In both cases one needs an algorithm (possibly very similar) for identifying the $C_\varepsilon(q)$, extracting them from T, and constructing one or the other representation.

3 Algorithm

The above Example 1 contains only ε-cycles that could be removed by physically removing their arcs (Fig. 1). However, ε-cycles can be more complex. They can overlap with each other, with non-ε-cycles, or with other (non-cyclic) paths. This means, ε-cycles must be removed without physically removing their arcs.



$$\varepsilon \longrightarrow (\text{rst}|\text{vt})^*\ \text{r}$$
$$\mathbf{a}\ \mathbf{a}^n \longrightarrow (\text{rst}|\text{vt})^*\ \mathbf{x}\ (\text{st}\ (\text{vt})^*\ \text{r})^*\ \Big(\ \text{s}\ (\text{tv}|\text{trs})^*\ \text{t}\ \mathbf{x}\ (\text{st}\ (\text{vt})^*\ \text{r})^*\ \Big)^n$$

Fig. 4. Transducer T with ε-cycles (Example 2)

Figure 4 shows a more complex example.[1] None of the ε-arcs *101*, *103*, and *104* can be physically removed because they are not only part of ε-cycles but among others also of the complete paths $\lceil$*101*$\rceil$ and $\lceil$*100, 103, 104, 100*$\rceil$ that accept the input strings ε and `aa` respectively.

3.1 Preparation

To extract all ε-cycles of an arbitrary FST, T, the algorithm proceeds as follows. First, T is concatenated on both ends with boundary symbols, # (Fig. 5). This operation causes that the properties of initiality and finality, so far only described by states, are now also described by arcs and can therefore be ignored by the algorithm (cf. all pseudo code).

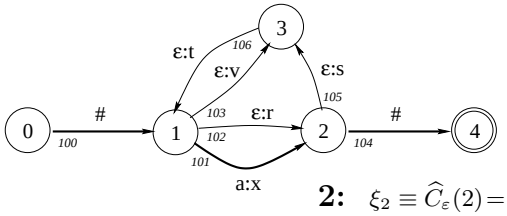


Fig. 5. Transducer T' with boundaries, auxiliary symbols, and ε-cycle information (Example 2)

Each state q_i in T is then assigned both information about its $\widehat{C}_\varepsilon(q_i)$ and an auxiliary symbol ξ_i that (at this stage) is considered as equivalent to $\widehat{C}_\varepsilon(q_i)$. The resulting FST is called T' (Fig. 5). For example, state 1 is assigned the set $\widehat{C}_\varepsilon(1) = \{\lceil 102, 105, 106\rceil, \lceil 103, 106\rceil\}$ and the auxiliary symbol ξ_1 which means that two ε-cycles consisting of the named arcs start at state 1 and are equivalent to ξ_1. These two ε-cycles generate the output substrings `(rst)`* and `(vt)`* respectively.

[1] In all figures of Example 2, thin arcs are used for ε-transitions and thick arcs for non-ε-transitions.

There are different ways to compute the $\widehat{C}_\varepsilon(q)$ of all q. For example, starting from a state q, we traverse every ε-path that does not encounter any state, except q, more than once. If the path ends at its start state q, it is an ε-cycle, and is inserted into $\widehat{C}_\varepsilon(q)$. All arcs e along a traversed path are put onto a stack (pseudo code, line 3: *push*(STACK, e)) so that at any time we can describe the path by the content of the stack (line 5: π=*path*(STACK)) :

```
T ⟶ T' :
1     for ∀q ∈ Q
2         do STACK := {}
3            Ĉ_ε(q) := {}
4            follow_epsilon_arcs(q)

follow_epsilon_arcs(p) :
1     for ∀e ∈ E(p)
2         do if i(e) = ε
3                then push(STACK, e)
4                     if n(e) = q
5                         then Ĉ_ε(q) := Ĉ_ε(q) ∪ {π | π=path(STACK)}
6                         else if ∀e' ∈ path(STACK), n(e)≠n(e')
7                                 then follow_epsilon_arcs(n(e))
8                     pop(STACK)
```

Although the $\widehat{C}_\varepsilon(q)$ do not contain all ε-cycles of a state q, the missing ε-cycles, that traverse a state q' more than once, do not escape our attention. They are in the $\widehat{C}_\varepsilon(q')$ of q' which is sufficient for our final purpose. The reason for building $\widehat{C}_\varepsilon(q)$ instead of $C_\varepsilon(q)$ is that $\widehat{C}_\varepsilon(q)$ is easier to construct, to represent (by an arc sequence), and to "rotate" (Sec 3.2).

3.2 Construction of T_1

Two steps are required to build T_1 from T' (Fig. 5) : First, at every state q_i with a non-empty set $\widehat{C}_\varepsilon(q_i)$, an arc mapping ε to ξ_i must be inserted. Second, all ε-cycles must be removed without physically removing their arcs.

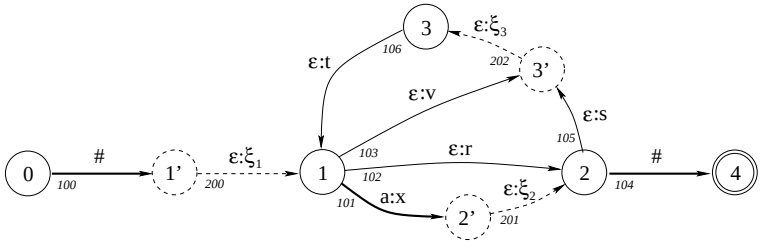


Fig. 6. Transducer T_1' with redirected ε-arcs (Example 2)

$\mathbf{T}' \longrightarrow \mathbf{T}'_1$:

```
1     for ∀q_i ∈ Q
2         do if Ĉ_ε(q_i) ≠ {}
3                then Q := Q ∪ {q'_i}
4                     E := E ∪ {(q'_i, ε, ξ_i, q_i)}
5                     for ∀e ∈ E^R(q_i)
6                         do if Ĉ_ε(q_i) ⊈ rotate_e^LR(Ĉ_ε(p(e)))
7                                then n(e) := q'_i
```

We insert for every state q_i with non-empty $\widehat{C}_\varepsilon(q_i)$, an auxiliary state q'_i and an auxiliary arc e'_i leading from q'_i to q_i (Fig. 6, dashed states and arcs). The arc e'_i is labeled with $\varepsilon{:}\xi_i$, i.e., it emits the auxiliary symbol ξ_i when it is traversed. For example, the auxiliary state $1'$ in created for state 1, and the auxiliary arc *200* labeled with $\varepsilon{:}\xi_1$ is inserted from state $1'$ to 1.

Then, some incoming arcs of every state q_i are redirected to the corresponding auxiliary state q'_i so that ξ_i is emitted before q_i is reached. An incoming arc e requires no redirection if the set $\widehat{C}_\varepsilon(q_i)$ of its destination state $n(e) = q_i$ is a "repetition", relative to e, of part of the $\widehat{C}_\varepsilon(p(e))$ of its source state $p(e)$. This is the case if every ε-cycle in $\widehat{C}_\varepsilon(q_i)$ can be obtained by "rotating" an ε-cycle in $\widehat{C}_\varepsilon(p(e))$, left to right, over e (pseudo code, line 6). In this case a redirection of e would not be wrong but it is redundant and can lead to a larger T_1 and T_2.

For example, the arc *106* requires no redirection from state 1 to $1'$ because every ε-cycle in $\widehat{C}_\varepsilon(1)$ can be obtained by rotating an ε-cycle in $\widehat{C}_\varepsilon(3)$ over the arc *106*; namely the ε-cycle ⌈*102, 105,* ***106***⌉ in $\widehat{C}_\varepsilon(1)$ by rotating ⌈***106***, *102, 105*⌉ in $\widehat{C}_\varepsilon(3)$ over the arc ***106***, and the ε-cycle ⌈*103,* ***106***⌉ in $\widehat{C}_\varepsilon(1)$ by rotating ⌈***106***, *103*⌉ in $\widehat{C}_\varepsilon(3)$ over the same arc ***106*** (Fig. 5, 6). In other terms, since ⌈*(106, 102, 105)*, 106*⌉=⌈*106, (102, 105, 106)** ⌉ and ⌈*(106, 103)*, 106*⌉=⌈*106, (103, 106)** ⌉, which in both cases means ⌈ξ_3*, 106*⌉=⌈*106,* ξ_1 ⌉, the insertion of ξ_1 after the arc *106*, which would result from a redirection of this arc, is unnecessary; ξ_1 would not express anything that has not been described yet by ξ_3.

The arc *103* must be redirected from state 3 to $3'$ because the ε-cycle ⌈*106, 102, 105*⌉ in $\widehat{C}_\varepsilon(3)$ cannot be obtained by rotating any of the ε-cycles in $\widehat{C}_\varepsilon(1)$ over the arc *103*. The arc *101* must be redirected from state 2 to $2'$ because it is not an ε-arc which means that no ε-cycles can be rotated over it.

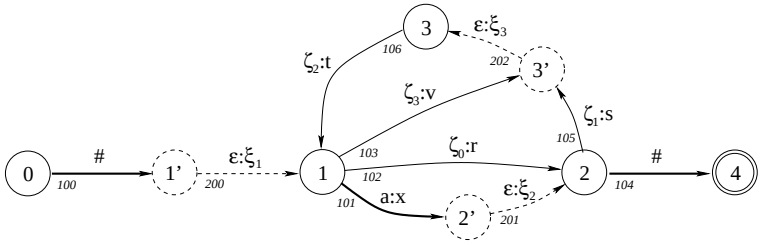


Fig. 7. Transducer T''_1 with redirected and overwritten ε-arcs (Example 2)

To prepare the removal of ε-cycles, the ε on the input side of every arc of every $\widehat{C}_\varepsilon(q_i)$ is temporarily overwritten by an auxiliary symbol ζ_j (Fig. 6, 7). This auxiliary symbol is different for every concerned arc, e.g., it is ζ_0 for the arc *102* and ζ_1 for the arc *105*. We call the result T_1''.

$\mathbf{T}_1' \longrightarrow \mathbf{T}_1''$ **:**
1 j := 0
2 for $\forall q \in Q$
3 do for $\forall e \,\overline{\in}\, \pi \in \widehat{C}_\varepsilon(q)$
4 do if $i(e) = eps$
5 then $i(e)$:= ζ_j
6 j := $j+1$

Every ε-cycle in T_1'' is then described by a sequence of ζ_j. For example, the ε-cycle $\lceil$*102, 105, 106*$\rceil$ in $\widehat{C}_\varepsilon(1)$ is described by the sequence $\lceil \zeta_0, \zeta_1, \zeta_2 \rceil$ that consists of the new input symbols of this cycle (Fig. 5, 6, 7). Then, a constraint R_1 is formulated to disallow all ε-cycles in all sets $\widehat{C}_\varepsilon(q_i)$, by disallowing the corresponding ζ_j-sequences:

$$R_1 \;=\; \neg\Big(\; ?^* \;\Big(\bigcup_q i(\widehat{C}_\varepsilon(q))\Big)\; ?^* \;\Big) \tag{7}$$

In Example 2, this constraint is (Fig. 7) :

$$R_1 \;=\; \neg\Big(\; ?^* \;\Big(\,(\zeta_0\,\zeta_1\,\zeta_2) \cup (\zeta_3\,\zeta_2) \cup (\zeta_1\,\zeta_2\,\zeta_0) \cup (\zeta_2\,\zeta_0\,\zeta_1) \cup (\zeta_2\,\zeta_3)\,\Big)\; ?^* \;\Big) \tag{8}$$
example

When R_1 is composed on the input side of T_1'', all ε-cycles disappear; even those that are in $C_\varepsilon(q_i)$, but not in $\widehat{C}_\varepsilon(q_i)$, of a state q_i because they appear in $\widehat{C}_\varepsilon(q_k)$ of at least one other state q_k:

$$T_1''' \;=\; R_1 \diamondsuit T_1'' \tag{9}$$

However, instances of the ζ_j-arcs remain in T_1''' if they are also part of another path that is not an ε-cycle. Finally, every remaining ζ_j, which stands for ε, and every boundary symbol, #, which has to be removed, is replaced with ε, and T_1''' is minimized (Fig. 9a). We call the result T_1. Note that an initially introduced auxiliary symbol ξ_i does no appear in T_1 if none of the incoming arcs of the state q_i have been redirected.

3.3 Construction of T_2

T_2 is built from T' (as was the case with T_1) (Fig. 5). T_2 must map any auxiliary symbol ξ_i to the corresponding set of ε-cycles $C_\varepsilon(q_i)$ rather than $\widehat{C}_\varepsilon(q_i)$. For every state q_i with non-empty $\widehat{C}_\varepsilon(q_i)$, two auxiliary arcs, both labeled with the auxiliary symbol ξ_i, are created (Fig. 8); one arc leading from the initial state i to q_i, the other from q_i to the only final state f (pseudo code, lines 3 and 4). The resulting FST will be referred to as T_2'.

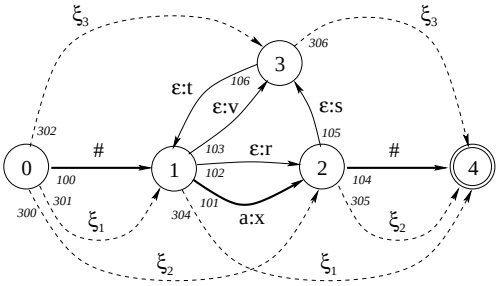


Fig. 8. Transducer T_2' (Example 2)

$\mathbf{T}' \longrightarrow \mathbf{T}_2'$ **:**
`1 for` $\forall q_i \in Q$
`2  do if` $\widehat{C}_\varepsilon(q_i) \neq \{\}$
`3  then` E `:=` $E \cup \{\langle i, \xi_i, \xi_i, q_i\rangle\}$
`4  ` E `:=` $E \cup \{\langle q_i, \xi_i, \xi_i, f\rangle\}$

All paths in T_2' that contain only full (and no partial) ε-cycles of a state q_i must be kept and all others removed. For example, the set of paths $\lceil$*301, (102, 105, 106)*, 304*$\rceil$ containing all ε-cycles of $C_\varepsilon(1)$ must be kept and $\lceil$*301, (102, 105, 106)*, 102, 305*$\rceil$ must be removed (Fig. 8). The paths to be kept, consist of twice the same auxiliary symbol, $\xi_i\,\xi_i$, on the input side. To allow only them, T_2' is composed with a constraint:

$$T_2'' \;=\; \left(\bigcup_i \{\xi_i\;\xi_i\} \right) \diamondsuit\, T_2' \tag{10}$$

This removes all undesired paths. In Example 2, the composition is (Fig. 8) :

$$T_2'' \;=\; \Big((\xi_1\;\xi_1) \,\cup\, (\xi_2\;\xi_2) \,\cup\, (\xi_3\;\xi_3) \Big) \diamondsuit\, T_2' \tag{11}$$

example

The resulting T_2'' maps any sequence of two identical auxiliary symbols $\xi_i\xi_i$ to itself, and inserts the corresponding set of ε-cycles $C_\varepsilon(q_i)$ in between. The second occurrence of every ξ_i is actually unwanted. The following composition removes this second occurrence on the input and output side, and the first occurrence of ξ_i on the output side only:

$$T_2''' \;=\; (\;?\;\;\hat{\varepsilon}{:}?\;)\;\diamondsuit\;T_2''\;\diamondsuit\;(\;?{:}\varepsilon\;\;?^*\;\;?{:}\hat{\varepsilon}\;) \tag{12}$$

The resulting T_2''' maps any single auxiliary symbol ξ_i to the corresponding set $C_\varepsilon(q_i)$. The $\hat{\varepsilon}$ denotes the (ordinary) empty string, like ε. It is, however, preserved in minimization and determinization which prevents T_2 from becoming larger. If the size is of no concern, ε can be used instead.

T_2 must accept any sequence of output symbols of T_1, i.e., any sequence in $\Delta_{T_1}^*$. It must map every auxiliary symbol ξ_i to the corresponding set of ε-cycles $C_\varepsilon(q_i)$, and every other symbol to itself. T_2 is built by:

$$T_2 \;=\; \left(\Delta_{T_1} \diamondsuit \left(T_2''' \;\cup\; \neg \bigcup_i \xi_i \right) \right)^* \tag{13}$$

This operation has the side effect that all initially introduced auxiliary symbols ξ_i that later disappeared from T_1, are now also removed from T_2. Finally, T_2 is minimized (Fig. 9b).

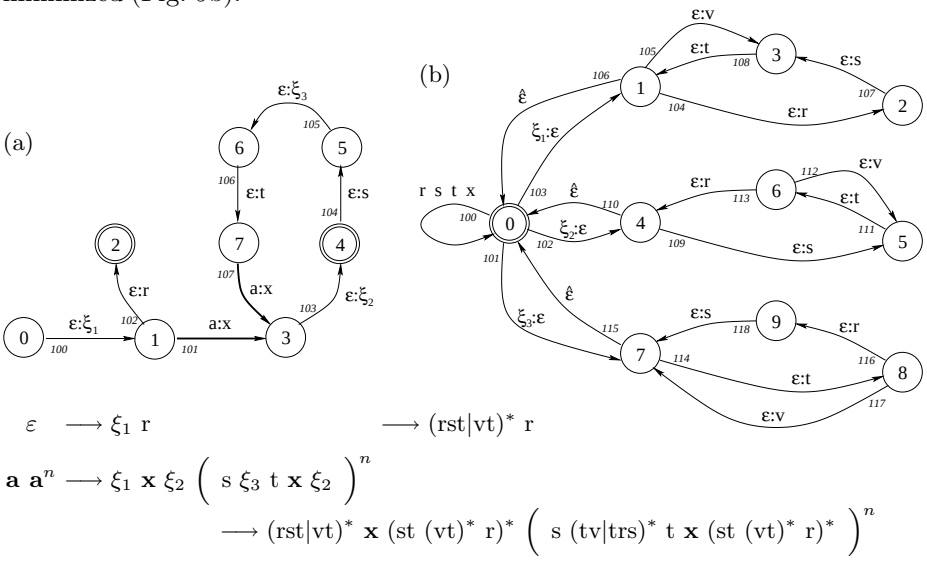


Fig. 9. Factorization of T with ε-cycles into (a) T_1 that emits auxiliary symbols, and (b) T_2 that maps auxiliary symbols to ε-cycles (Example 2)

3.4 Proof

For the following reason, the algorithm always leads to the described result.

In T_1: If an ε-cycle in $C_\varepsilon(q_i)$ contains a state q_k more that once, which means that this cycle has an "inner" ε-cycle on q_k, than $C_\varepsilon(q_i)$ also contains an ε-cycle where no q_k is encountered more that once, which can be obtained by not traversing the inner cycle on q_k. This also holds if there are several inner cycles. In general:

$$C_\varepsilon(q_i) \neq \{\} \quad \Rightarrow \quad \widehat{C}_\varepsilon(q_i) \neq \{\} \tag{14}$$

This means, every q_i with $C_\varepsilon(q_i) \neq \{\}$ will be assigned an auxiliary symbol ξ_i, although this action is triggered by $\widehat{C}_\varepsilon(q_i) \neq \{\}$.

All inner ε-cycles, that are not in $\widehat{C}_\varepsilon(q_i)$, are in $\widehat{C}_\varepsilon(q_k)$, of some other state q_k. Consequently, they will be removed as well, i.e., all ε-cycles of T_1 will be removed.

Example 2 (Fig. 5) : Since state 2 has a non-empty $C_\varepsilon(2) = \{\lceil 105,\ (106,\ 103)^*,\ 106,\ 102 \rceil\}$, it also has a non-empty $\widehat{C}_\varepsilon(2) = \{\lceil 105,\ 106,\ 102 \rceil\}$ and will therefore be assigned ξ_2. The inner cycle $\lceil (106,\ 103)^* \rceil$ in $C_\varepsilon(2)$ will be removed from T_1, despite not being in $\widehat{C}_\varepsilon(2)$, because it is in $\widehat{C}_\varepsilon(1)$ and $\widehat{C}_\varepsilon(3)$.

In $\mathbf{T_2}$: The $C_\varepsilon(q_i)$ of every state q_i that has been assigned an auxiliary symbol ξ_i are preserved whereas every other path is removed. This means, the ξ_i are mapped to $C_\varepsilon(q_i)$ rather than to $\widehat{C}_\varepsilon(q_i)$ that originally caused their introduction.

The initial limitation to $\widehat{C}_\varepsilon(q_i)$ is not reflected in the final result, T_1 and T_2.

4 Final Remarks

Although T_1 (Fig. 9a) cannot be converted into a single real-time FST, it can be split into the union of two FSTs, one containing the transitions {100, 102}, the other the transitions {100, 101, 103, 104, 105, 106, 107}. The second of these FSTs can be made real-time.

Jointly in a cascade, T_1 and T_2 describe the same relation and perform the same mapping as the original FST T (Fig. 9). When T_1 and T_2 are composed with each other, T is obtained. The size increase of T_2, compared to T, is not necessarily a concern. T_2 could be an intermediate result that is further processed.

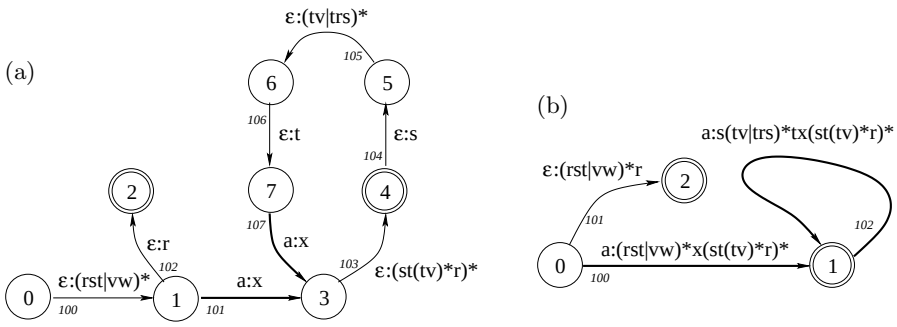


Fig. 10. Alternative representation of ε-cycles by complex labels (a) with ε-arcs or (b) more compact with less ε-arcs (Example 2)

As previously show for Example 1, instead of factorizing T into T_1 and T_2, one can represent T by a single FST, $\widehat{T}$, that is similar to T_1 but with output labels directly describing sets of ε-cycles $C_\varepsilon(q)$. Every $C_\varepsilon(q)$ in T would be reduced to a single arc in $\widehat{T}$ (Fig. 10). Both representations are equivalent and require, as mentioned, an algorithm (possibly very similar) to construct them.

References

1. A. V. Aho, R. Sethi, and J. D. Ullman. 1986. *Compilers - Principles, Techniques and Tools*. Addison-Wesley, Reading, MA, USA.
2. G. Birkhoff and T. C. Bartee. 1970. *Modern Applied Algebra*. McGraw-Hill, New York, USA.
3. A. Kempe. 2000. Reduction of intermediate alphabets in finite-state transducer cascades. In *Proceedings of the 7th Conference on Automatic Natural Language Processing (TALN)*, pages 207–215, Lausanne, Switzerland. ATALA.

4. A. Kempe. 2001. Factorization of ambiguous finite-state transducers. In S. Yu, A. Paun, editors, *Proceedings of the 5th International Conference on Implementation and Application of Automata (CIAA 2000)*, The University of Western Ontario, London, Ontario, Canada, July 24-25, 2000. Volume 2088 of *Lecture Notes in Computer Science*, pages 170–181, Springer-Verlag.
5. M. Mohri. 2001. Generic ε-removal algorithm for weighted automata. In S. Yu, A. Paun, editors, *Proceedings of the 5th International Conference on Implementation and Application of Automata (CIAA 2000)*, The University of Western Ontario, London, Ontario, Canada, July 24-25, 2000. Volume 2088 of *Lecture Notes in Computer Science*, pages 230–242, Springer-Verlag.
6. G. van Noord. 1998. Treatment of ε-moves in subset construction. In *Proceedings of the International Workshop on Finite-State Methods in Natural Language Processing (FSMNLP)*, pages 1–12, Ankara, Turkey, June 29 - July 1. Bilkent University.
7. J. Sakarovitch. 1998. A construction on finite automata that has remained hidden. *Theoretical Computer Science*, 204:205–231.
8. M. P. Schützenberger. 1976. Sur les relations rationnelles entre monoïdes libres. *Theoretical Computer Science*, 3:243–259.