ELSEVIER
Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE @ DIRECT®
Theoretical Computer Science 313 (2004) 145–158
Theoretical Computer Science
www.elsevier.com/locate/tcs

Extraction and recoding of input-ε-cycles in finite state transducers

André Kempe

Xerox Research Centre Europe, Grenoble Laboratory, 6 chemin de Maupertuis, Meylan 38240, France

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 particular type of non-determinism, from an arbitrary finite-state transducer (FST). The algorithm 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. The article also proposes an alternative approach where each ε-cycle of T is replaced by a single transitions with a complex label that describes the output of the cycle. Since ε-cycles are an obstacle for some algorithms such as the decomposition of ambiguous FSTs, the proposed approaches allow us to by-pass this problem. ε-Cycles can be extracted or recoded before and re-inserted (by composition) after such algorithms.

MSC: 05C85; 05C38; 68Q45; 68W30

Keywords: Finite state transducer; Decomposition; Non-determinism; Epsilon cycle

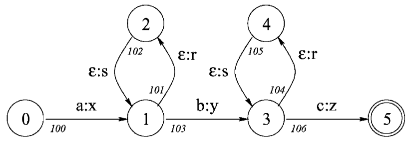
1. Introduction

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

This article describes an algorithm for extracting all ε-cycles, which represent a special type of non-determinism consisting of consecutive transitions with the empty string ε as input label, from an arbitrary finite-state transducer (FST). The algorithm 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* (that simulates composition), T_1 and T_2 describe the same relation as T.

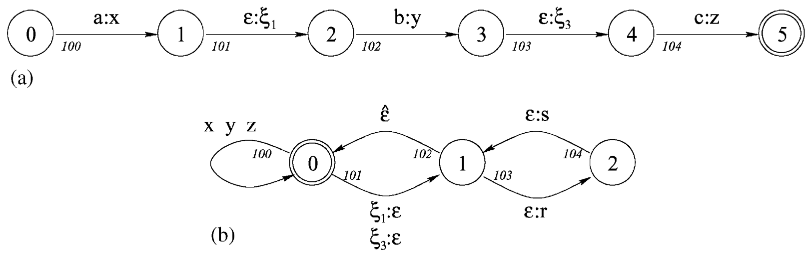
E-mail address: andre.kempe@xrce.xerox.com (A. Kempe).
URL: http://www.xrce.xerox.com/people/kempe/

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2003.10.012



$$\text{abc} \longrightarrow \text{x (rs)}^* \text{ y (rs)}^* \text{ z}$$

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



$$\text{abc} \longrightarrow \text{x } \xi_1 \text{ y } \xi_3 \text{ z} \longrightarrow \text{x (rs)}^* \text{ y (rs)}^* \text{ z}$$

Fig. 2. Decomposition 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).

Motivation: Some algorithms, such as the decomposition of ambiguous FSTs [3,7,8], can only be performed on *real-time* FSTs, where every transition has exactly one symbol on the input side. Transitions 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-transitions and concatenating their output labels with the output of adjacent non-ε-transitions. 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 decomposed 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 decomposed 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 Figs. 1–3.

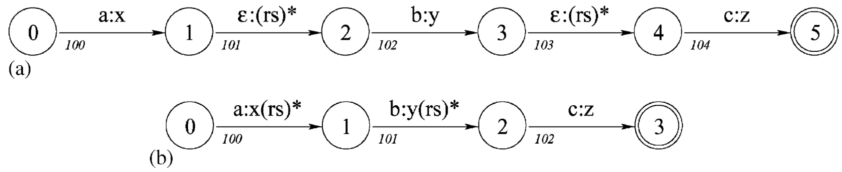


Fig. 3. Representation of ε-cycles by complex labels (a) with ε-transitions or (b) as a real-time transducer without ε-transitions (Example 1).

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 transition numbers have no meaning for the FST but are just used to reference a state or a transition. A transition with n labels designates a set of n transitions with one label each that all have the same source and destination. In a symbol pair occurring as a transition 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 $T_1 \diamond T_2 \diamond T_3 = T_3 \circ T_2 \circ T_1$, T_1 is applied first and T_3 last [2]. We prefer left-to-right notation (and application) and will therefore use the $\diamond$-operator. We find it also clearer in examples such as $(\mathtt{a{:}b}) \diamond (\mathtt{b{:}c}) \diamond (\mathtt{c{:}d}) = (\mathtt{a{:}d})$, compared to $(\mathtt{c{:}d}) \circ (\mathtt{b{:}c}) \circ (\mathtt{a{:}b}) = (\mathtt{a{:}d})$.

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 $\pi = e_1, \ldots, e_k$ is an element of E^* with consecutive transitions. To express that a transition e is on a path π, we write $e \,\bar{\in}\, \pi$. To refer to a particular path in a figure, we give the transition numbers in ceiling brackets; e.g., $\pi = \lceil 100, 101, 102, 103 \rceil$ is a path consisting of the four named transitions. 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 transitions with ε as input label:

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

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

$$C_\varepsilon(q) = \{\pi \in C(q) \,|\, \forall e \mathrel{\bar{\in}} \pi,\; i(e) = \varepsilon\} \tag{3}$$

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

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

$$\hat{C}_\varepsilon(q) = \{\pi \in C_\varepsilon(q) \,|\, \forall e, e' \mathrel{\bar{\in}} \pi,\; e \neq e' \Rightarrow n(e) \neq n(e')\} \tag{5}$$

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.

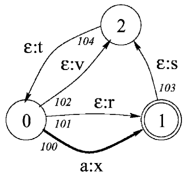
2. Basic idea

Any arbitrary FST, T, containing ε-cycles can be 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 transition 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:

$$C_\varepsilon(q_i) \rightarrow (\varepsilon : \xi_i) \diamond (\xi_i : o(C_\varepsilon(q_i))) \tag{6}$$

Fig. 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.

Fig. 2 shows the same example after the extraction of ε-cycles (decomposition). 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 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).



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

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

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

An alternative to decomposing T into T_1 and T_2 would be representing it by a single FST, $\hat{T}$, that is similar to T_1 but with 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 transition in $\hat{T}$ that maps ε to a complex label (Fig. 3a). $\hat{T}$ can be further converted into a real-time FST, without input-ε-transitions (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 decomposition.

3. Algorithm

The above Example 1 contains only ε-cycles that could be removed by physically removing their transitions (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 transitions.

Fig. 4 shows a more complex example.[1] None of the ε-transitions 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 transitions and can therefore be ignored by the algorithm (cf. all pseudo code).

[1] In all the following figures, thin arrows are used for ε-transitions and thick arrows for non-ε-transitions.

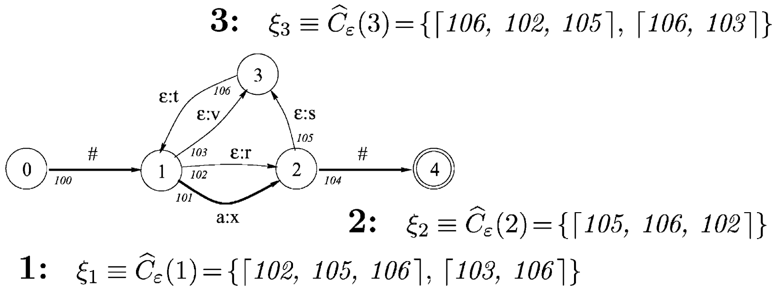


Fig. 5. Transducer T' with boundaries, auxiliary symbols, and ε-cycle information (Example 2).

```
T → T':
1    for ∀q ∈ Q do
2        Stack ← {}
3        Ĉε(q) ← {}
4        FollowEpsilonArcs(q, q)

FollowEpsilonArcs(p, q) :
5    for ∀e ∈ E(p) do
6        if i(e) = ε
7            then push(Stack, e)
8                if n(e) = q
9                    then Ĉε(q) ← Ĉε(q) ∪ {π | π = path(Stack)}
10                   else if ∀e' ∈⃗ path(Stack), n(e) ≠ n(e')
11                       then FollowEpsilonArcs(n(e), q)
12               pop(Stack)
```

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

There are different ways to compute the $\hat{C}_\varepsilon(q)$ 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 $\hat{C}_\varepsilon(q)$. All transitions e along a traversed path are put onto a stack (pseudo code, line 7: $PUSH(Stack,\ e)$) so that at any time we can describe the path by the content of the stack (line 9: $\pi = \mathrm{PATH}(Stack)$).

Although the $\hat{C}_\varepsilon(q)$ 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 $\hat{C}_\varepsilon(q')$

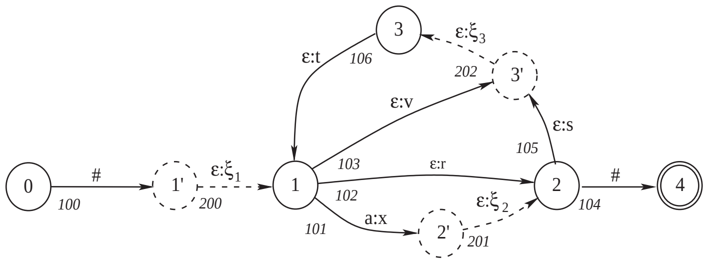


Fig. 6. Transducer T_1' with redirected ε-transitions (Example 2).

of q' which is sufficient for our final purpose. The reason for building $\hat{C}_\varepsilon(q)$ instead of $C_\varepsilon(q)$ is that $\hat{C}_\varepsilon(q)$ is easier to construct, to represent (by a transition sequence), and to "rotate" (Section 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 $\hat{C}_\varepsilon(q_i)$, a transition mapping ε to ξ_i must be inserted. Second, all ε-cycles must be removed without physically removing their transitions.

We insert for every state q_i with non-empty $\hat{C}_\varepsilon(q_i)$, an auxiliary state q_i' and an auxiliary transition e_i' leading from q_i' to q_i (Fig. 6, dashed states and transitions; pseudo code line 2–4). The transition 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 transition 200 labeled with $\varepsilon{:}\xi_1$ is inserted from state $1'$ to 1.

T′ → T′₁:

1   for $\forall q_i \in Q$ do
2   if $\hat{C}_\varepsilon(q_i) \neq \{\}$
3   then $Q \leftarrow Q \cup \{q_i'\}$
4          $E \leftarrow E \cup \{(q_i', \varepsilon, \xi_i, q_i)\}$
5   for $\forall e \in E^R(q_i)$ do
6   if $\hat{C}_\varepsilon(q_i) \not\subseteq \text{ROTATE}_e^{LR}(\hat{C}_\varepsilon(p(e)))$
7   then $n(e) \leftarrow q_i'$

Then, some incoming transitions 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 transition e requires no redirection if the set $\hat{C}_\varepsilon(q_i)$ of its destination state $n(e) = q_i$ is a "repetition", relative to e, of part of the $\hat{C}_\varepsilon(p(e))$ of its source state $p(e)$. This is the case if every ε-cycle in $\hat{C}_\varepsilon(q_i)$ can be obtained by "rotating" an ε-cycle in $\hat{C}_\varepsilon(p(e))$, 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 transition 106 requires no redirection from state 1 to $1'$ because every ε-cycle in $\hat{C}_\varepsilon(1)$ can be obtained by rotating an ε-cycle in $\hat{C}_\varepsilon(3)$ over the tran-

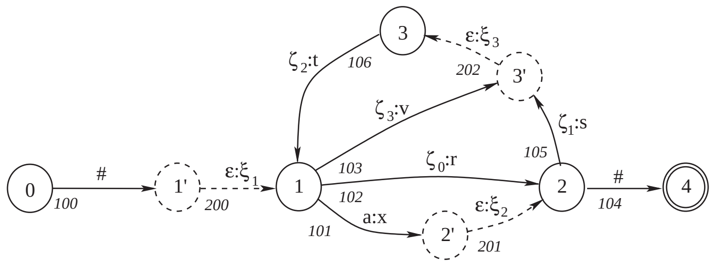


Fig. 7. Transducer T_1'' with redirected and overwritten ε-transitions (Example 2).

sition 106; namely the ε-cycle $\lceil 102, 105, \underline{106} \rceil$ in $\hat{C}_\varepsilon(1)$ by rotating $\lceil \underline{106}, 102, 105 \rceil$ in $\hat{C}_\varepsilon(3)$ over the transition $\underline{106}$, and the ε-cycle $\lceil 103, \underline{106} \rceil$ in $\hat{C}_\varepsilon(1)$ by rotating $\lceil \underline{106}, 103 \rceil$ in $\hat{C}_\varepsilon(3)$ over the same transition $\underline{106}$ (Figs. 5, 6). In other terms, since $\lceil (106, 102, 105)^*, 106 \rceil = \lceil 106, (102, 105, 106)^* \rceil$ and $\lceil (106, 103)^*, 106 \rceil = \lceil 106, (103, 106)^* \rceil$, which in both cases means $\lceil \xi_3, 106 \rceil = \lceil 106, \xi_1 \rceil$, the insertion of ξ_1 after the transition 106, which would result from a redirection of this transition, is unnecessary; ξ_1 would not express anything that has not been described yet by ξ_3.

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

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

$\quad j \leftarrow 0$
$\quad$ for $\forall q \in Q$ do
$\quad\quad$ for $\forall e \mathrel{\bar{\bar{\in}}} \pi \in \hat{C}_\varepsilon(q)$ do
$\quad\quad\quad$ if $i(e) = \varepsilon$
$\quad\quad\quad\quad$ then $i(e) \leftarrow \zeta_j$
$\quad\quad\quad\quad\quad j \leftarrow j + 1$

To prepare the removal of ε-cycles, the ε on the input side of every transition of every $\hat{C}_\varepsilon(q_i)$ is temporarily overwritten by an auxiliary symbol ζ_j (Figs. 6, 7). This auxiliary symbol is different for every concerned transition, e.g., it is ζ_0 for the transition 102 and ζ_1 for the transition 105. We call the result T_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 $\hat{C}_\varepsilon(1)$ is described by the sequence $\lceil \zeta_0, \zeta_1, \zeta_2 \rceil$ that consists of the new input symbols of this cycle (Figs. 5–7). Then, a constraint R_1 is formulated to disallow all ε-cycles in all sets $\hat{C}_\varepsilon(q_i)$, by disallowing the corresponding ζ_j-sequences:

$$R_1 = \neg \left(?^* \left(\bigcup_q i(\hat{C}_\varepsilon(q)) \right) ?^* \right) \tag{7}$$

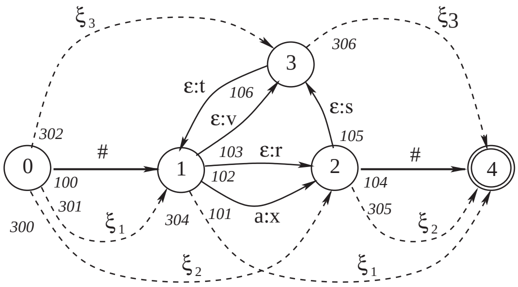


Fig. 8. Transducer T_2' (Example 2).

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

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

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 $\hat{C}_\varepsilon(q_i)$, of a state q_i because they appear in $\hat{C}_\varepsilon(q_k)$ of at least one other state q_k:

$$T_1''' = R_1 \diamond T_1'' \tag{9}$$

However, instances of the ζ_j-transitions 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. 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 transitions 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 $\hat{C}_\varepsilon(q_i)$. For every state q_i with non-empty $\hat{C}_\varepsilon(q_i)$, two auxiliary transitions, both labeled with the auxiliary symbol ξ_i, are created (Fig. 8); one transition 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'.

$\mathbf{T}' \rightarrow \mathbf{T}_2'$:

1 for $\forall q_i \in Q$ do
2   if $\hat{C}_\varepsilon(q_i) \neq \{\}$
3   then $E \leftarrow E \cup \{\langle i, \xi_i, \xi_i, q_i\rangle\}$
4             $E \leftarrow E \cup \{\langle q_i, \xi_i, \xi_i, f\rangle\}$

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:

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

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

$$\begin{aligned} T_2'' &= ((\xi_1\,\xi_1)\cup(\xi_2\,\xi_2)\cup(\xi_3\,\xi_3))\diamond T_2' \\ &\quad \mathit{example} \end{aligned} \tag{11}$$

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:

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

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:

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

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. 9a, b).

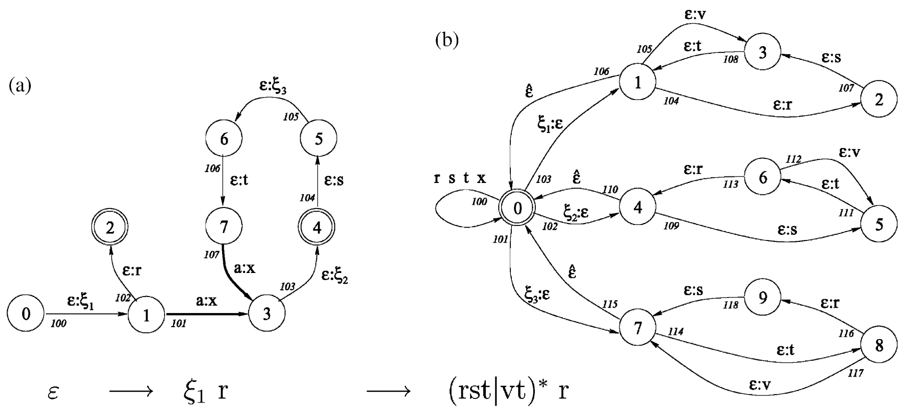
3.4. Proof

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

In $\mathbf{T}_1$: If an ε-cycle in $C_\varepsilon(q_i)$ contains a state q_k more that once, which means that this cycle has an "embedded" 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 embedded cycle on q_k. This also holds if there are several embedded cycles. In general:

$$C_\varepsilon(q_i)\neq\{\} \;\;\Rightarrow\;\; \hat{C}_\varepsilon(q_i)\neq\{\} \tag{14}$$

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 $\hat{C}_\varepsilon(q_i)\neq\{\}$.



$$\varepsilon \longrightarrow \xi_1\ \mathrm{r} \longrightarrow (\mathrm{rst}|\mathrm{vt})^*\ \mathrm{r}$$

$$\mathbf{a}\ \mathbf{a}^n \longrightarrow \xi_1\ \mathbf{x}\ \xi_2\ \Big(\ \mathrm{s}\ \xi_3\ \mathrm{t}\ \mathbf{x}\ \xi_2\ \Big)^n$$
$$\longrightarrow (\mathrm{rst}|\mathrm{vt})^*\ \mathbf{x}\ (\mathrm{s}(\mathrm{tv})^*\mathrm{tr})^*\ \Big(\ \mathrm{s}\ (\mathrm{tv}|\mathrm{trs})^*\ \mathrm{t}\ \mathbf{x}\ (\mathrm{s}(\mathrm{tv})^*\mathrm{tr})^*\ \Big)^n$$

Fig. 9. Decomposition 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).

All embedded ε-cycles, that are not in $\hat{C}_\varepsilon(q_i)$, are in $\hat{C}_\varepsilon(q_k)$, 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 $\hat{C}_\varepsilon(2) = \{\lceil 105, 106, 102\rceil\}$ and will therefore be assigned ξ_2. The embedded cycle $\lceil (106, 103)^* \rceil$ in $C_\varepsilon(2)$ will be removed from T_1, despite not being in $\hat{C}_\varepsilon(2)$, because it is in $\hat{C}_\varepsilon(1)$ and $\hat{C}_\varepsilon(3)$.

In 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 $\hat{C}_\varepsilon(q_i)$ that originally caused their introduction.

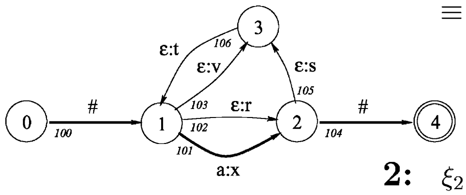
The initial limitation to $\hat{C}_\varepsilon(q_i)$ is not reflected in the final result, T_1 and T_2.

4. Alternative representation with complex labels

As previously shown for Example 1, instead of decomposing T into T_1 and T_2, one can represent it by a single FST, $\hat{T}$, that is similar to T_1 but with complex output labels that directly describe sets of ε-cycles $C_\varepsilon(q)$ (Fig. 3). Every $C_\varepsilon(q)$ in T would be reduced to a single transition in $\hat{T}$. Both representations are equivalent.

To build $\hat{T}$, we first create T' as described above (Section 3.1, Fig. 5). Each state q with non-empty $C_\varepsilon(q)$ is then additionally assigned a complex label $L(q)$ that describes $C_\varepsilon(q)$. The resulting FST is called T'' (Fig. 10, pseudo code line 1 to 3). For example, state 2 with $\hat{C}_\varepsilon(2) = \{\lceil 105, 106, 102\rceil\}$ is assigned $L(2) =$ "(s(tv) $*$ tr) $*$ ".

3: $\xi_3 \equiv \widehat{C}_\varepsilon(3) = \{\lceil 106,\ 102,\ 105 \rceil, \lceil 106,\ 103 \rceil\}$
$\equiv L(3) =$ "(trs|tv)∗"



2: $\xi_2 \equiv \widehat{C}_\varepsilon(2) = \{\lceil 105,\ 106,\ 102 \rceil\}$
$\equiv L(2) =$ "(s(tv) ∗ tr)∗"

1: $\xi_1 \equiv \widehat{C}_\varepsilon(1) = \{\lceil 102,\ 105,\ 106 \rceil, \lceil 103,\ 106 \rceil\}$
$\equiv L(1) =$ "(rst|vt)∗"

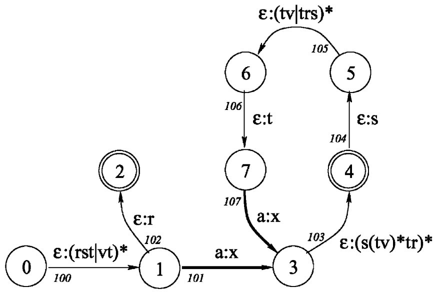
Fig. 10. Transducer T'' with boundaries, auxiliary symbols, ε-cycle information, and complex labels (Example 2).

A label $L(q)$ begins with "(" (line 8), ends with ")*" (line 8), and contains the labels of all paths $\pi \in \hat{C}_\varepsilon(q)$ separated by "|" (line 7). (Note, the "|" after the last path label is first appended and then removed (line 7, 8)). The label of a path π is created form the output symbols $o(e)$ of its transitions $e \,\overline{\in}\, \pi$ (line 12) and from the labels of embedded ε-cycles encountered along π.

T′ → T″:
1 for $\forall q \in Q$ do
2   if $\hat{C}_\varepsilon(q) \neq \{\}$
3   then $L(q) \leftarrow$ LabelOfCycles($\hat{C}_\varepsilon(q), q$)

LabelOfCycles(C, q):
4 $L_c \leftarrow$ "("
5 for $\forall \pi \in C$ do
6     $L_c \leftarrow L_c \cdot$ LabelOfPath(π, q)
7     $L_c \leftarrow L_c \cdot$ "|"
8 $L_c \leftarrow L_c \cdot$ "|"$^{-1}$ $\cdot$ ")∗"
9 return L_c

LabelOfPath(π, q):
10 $L_\pi \leftarrow \varepsilon$
11 for $\forall e \,\overline{\in}\, \pi$ do
12     $L_\pi \leftarrow L_\pi \cdot o(e)$
13     $C'_\varepsilon \leftarrow \{\pi' \in \hat{C}_\varepsilon(n(e)) \mid \text{ROTATE}^{RL}(\pi') \notin \hat{C}_\varepsilon(p(e)) \wedge \forall e' \,\overline{\in}\, \pi', n(e') \neq q\ \}$
14   if $C'_\varepsilon \neq \{\}$
15   then $L_\pi \leftarrow L_\pi \cdot$ LabelOfCycles(C'_ε, q)
16 return L_π



$$\begin{array}{lcl} \varepsilon & \longrightarrow & \underline{(\mathrm{rst}|\mathrm{vt})^*}\ \mathrm{r} \\ \mathbf{a}\ \mathbf{a}^n & \longrightarrow & \underline{(\mathrm{rst}|\mathrm{vt})^*}\ \mathbf{x}\ \underline{(\mathrm{s}(\mathrm{tv})^*\mathrm{tr})^*}\ \Big(\ \mathrm{s}\ \underline{(\mathrm{tv}|\mathrm{trs})^*}\ \mathrm{t}\ \mathbf{x}\ \underline{(\mathrm{s}(\mathrm{tv})^*\mathrm{tr})^*}\ \Big)^n \end{array}$$

Fig. 11. Representation of ε-cycles by complex labels (Example 2).

To account for those cycles, we create for each $e \bar{\in} \pi$ a set C'_ε of embedded ε-cycles that immediately follow e (line 13). This set C'_ε contains all those paths $\pi' \in C_\varepsilon(n(e))$ assigned to the transition's destination state $n(e)$ that meet the following two constrains: First, π' must not be a repetition (obtained by rotation) of any of the paths in the transition's source state $p(e)$ because then π' has already been taken into account with $p(e)$ or earlier. Second, π' must not traverse the state q for which we are creating the label $L(q)$ because then it is taken into account as an ordinary ε-cycle of q rather than as an embedded one. The label of the set of embedded ε-cycles C'_ε encountered after a transition $e \bar{\in} \pi$ is included (if not empty) into the label of the path π after the output symbol of e (line 15). Note that the paths of embedded cycles can as well contain embedded cycles.

For example, $L(2)$ of state 2 describes the only path $\pi = \lceil 105, 106, 102 \rceil$ of $\hat{C}_\varepsilon(2)$ (Fig. 10). After the output symbol "s" of transition 105, we append the label "(tv)*" of the embedded ε-cycle $\pi'_1 = \lceil 106, 103 \rceil \in \hat{C}_\varepsilon(3)$. The other path $\pi'_2 = \lceil 106, 102, 105 \rceil \in \hat{C}_\varepsilon(3)$ is ignored because it is a rotation of π. π'_1 has no embedded ε-cycles to be taken into account. The next symbol in $L(2)$ is "t", the output of transition 106. The ε-cycles of $\hat{C}_\varepsilon(1)$ of the following state 1 are all ignored because they are rotations of cycles in $\hat{C}_\varepsilon(3)$.

$\hat{T}$ is then built from T'' by the same algorithm as T_1 was built from T' (Section 3.2). Consequently, $\hat{T}$ has the same structure as T_1 (Figs. 9a, 11). However, instead of auxiliary symbols ξ that represent ε-cycles, $\hat{T}$ has complex labels $L(q)$ that directly describe them.

5. Conclusion and final remarks

The article has shown that an arbitrary FST, T, containing ε-cycles can be decomposed 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

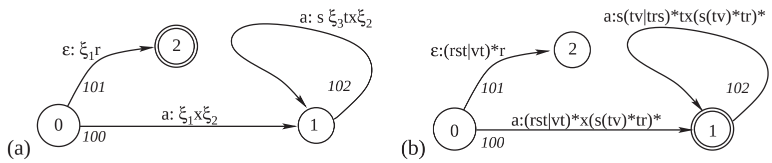


Fig. 12. (Almost) Real-time representation of ε-cycles (a) by auxiliary symbols and (b) by complex labels (Example 2).

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.

As an alternative to decomposition, T can be represented by a single FST, $\hat{T}$, that is similar to T_1 but with complex output labels directly describing sets of ε-cycles. Every such set in T is reduced to a single transition in $\hat{T}$ (Fig. 11). Both representations ($T_1 \diamond T_2$ and $\hat{T}$) are equivalent and can be constructed by similar algorithms.

Both T_1 and $\hat{T}$ (in Example 2) cannot be converted into real-time FSTs because they accept ε as input. To make them almost real-time they can be split into the union of an FST that accepts only ε (Figs. 9a and 11: transitions $\{100, 102\}$) and another FST that does not accept ε (transitions $\{100, 101, 103, 104, 105, 106, 107\}$). The second of these FSTs can be made real-time and then unioned again with the first (Fig. 12).

References

[1] A.V. Aho, R. Sethi, J.D. Ullman, Compilers—Principles, Techniques and Tools, Addison-Wesley, Reading, MA, USA, 1986.

[2] G. Birkhoff, T.C. Bartee, Modern Applied Algebra, McGraw-Hill, New York, USA, 1970.

[3] A. Kempe, Reduction of intermediate alphabets in finite-state transducer cascades, in: Proc. 7th Conf. on Automatic Natural Language Processing (TALN), Lausanne, Switzerland. ATALA, 2000, pp. 207–215.

[4] A. Kempe, Factorization of ambiguous finite-state transducers, in: S. Yu, A. Paun (Ed.), Proc. 5th Internat. Conf. on Implementation and Application of Automata (CIAA 2000), The University of Western Ontario, London, Ontario, Canada, July 24–25, 2000. Lecture Notes in Computer Science, Vol. 2088, Springer, Berlin, 2001, pp. 170–181.

[5] M. Mohri, Generic ε-removal algorithm for weighted automata, in: S. Yu, A. Paun (Eds.), Proc. 5th Internat. Conf. on Implementation and Application of Automata (CIAA 2000), The University of Western Ontario, London, Ontario, Canada, July 24–25, 2000. Lecture Notes in Computer Science, Vol. 2088 Springer, Berlin, 2001, pp. 230–242.

[6] G. van Noord, Treatment of ε-moves in subset construction, in: Proc. Internat. Workshop on Finite-State Methods in Natural Language Processing (FSMNLP), Bilkent University, Ankara, Turkey, June 29–July 1, 1998, pp. 1–12.

[7] J. Sakarovitch, A construction on finite automata that has remained hidden, Theoret. Comput. Sci. 204 (1998) 205–231.

[8] M.P. Schützenberger, Sur les relations rationnelles entre monoïdes libres, Theoret. Comput. Sci. 3 (1976) 243–259.