A new Epsilon Filter for Efficient Composition of Weighted Finite-State Transducers

Frank Duckhorn, Matthias Wolff, Rüdiger Hoffmann

Institute of Acoustics and Speech Communication, Technical University Dresden, Germany
{frank.duckhorn,matthias.wolff,ruediger.hoffmann}@ias.et.tu-dresden.de

Abstract

In this paper we propose a new composition algorithm for weighted finite-states transducers that are more and more used for speech and pattern recognition applications. Composition joins multiple transducers into one. We have implemented an embedded speech based dialog system for steering applications. Therefore regular grammars are very useful, but they may enlarge strongly by determinization. Composition using the sequential or the matching epsilon-filter does not perform optimal without determinization. Our new algorithm combines the advantages of these two epsilon-filters for size reduction. So composition and decoding time can be saved. It can be applied to many current algorithms including on-the-fly ones.

Index Terms: Weighted finite-state transducer, WFST, Composition, Automatic speech recognition

1. Introduction

In the last years weighted finite-state transducers (WFSTs) have become very popular in the special case of speech recognition and in the general case of pattern recognition (see [1, 2]). The advantage of WFSTs is that different types of models used for recognition tasks can be mapped to only one data structure. Thus there is no need for model specific algorithms, because there are algorithms for combining (namely composition), optimizing (for example minimization) and decoding of WFSTs. For speech recognition WFSTs can offer a uniform representation of Hidden-Markov-Models ($\mathcal{H}$), phoneme context-dependency models ($\mathcal{C}$), phonetic lexicons ($\mathcal{L}$) and language models ($\mathcal{G}$). These models can be combined into one single recognition network ($\mathcal{R}$) through composition operation ($\circ$).

$$\mathcal{R} = \mathcal{H} \circ \mathcal{C} \circ \mathcal{L} \circ \mathcal{G} \tag{1}$$

In this paper we will investigate the composition algorithm in detail and extend it for efficiency reasons. The classical composition algorithm for WSTs in speech recognition was published by Mohri (see [1, 2, 3]). There were many enhancements, especially on-the-fly composition became more and more common in the last years (see [4, 5, 6, 7]). Here the task is to perform composition and decoding simultaneously. There is no need to generate a complete recognition network. Only the states used for decoding are generated. Also some on-the-fly approaches were generalized so that they can work with arbitrary transducers (see [8, 9, 10]). Many of those up to date composition algorithms use the either the matching ϵ-filter introduced by Mohri (for example in [2]) or they use the sequential one.

We will show that, in the case of transducer with empty self-loop transitions (without input or output symbols), the matching ϵ-filter on the one side does not always produce an optimal result. On the other side determinization may enlarge regular grammar transducers considerably (see section 6). Without determinization there may be empty transitions in both source transducers and the sequential ϵ-filter will cascade them because it is not able to join them.

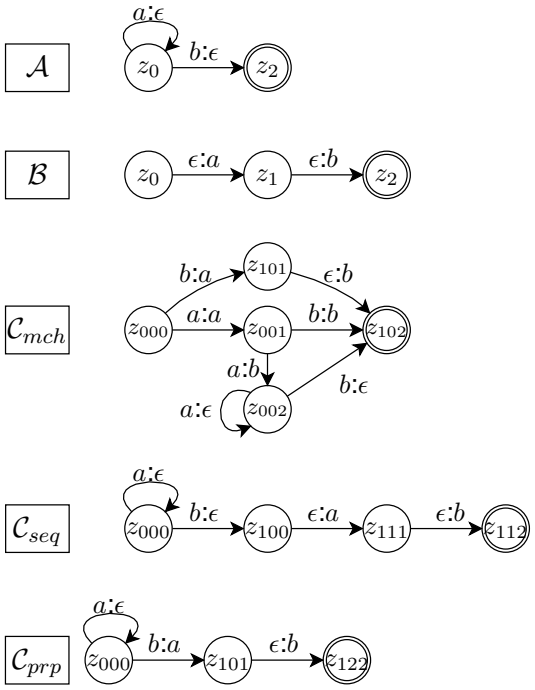

Figure 1: Composition example with matching, sequential and proposed ϵ-filter ($\mathcal{A} \circ \mathcal{B} = \mathcal{C}$). The transitions are marked with their input and output symbols: $x(e) : y(e)$. Generated with openfst.

For example in figure 1 the composition of transducers $\mathcal{A}$ and $\mathcal{B}$ would produce $\mathcal{C}_{mch}$ using matching ϵ-filter and $\mathcal{C}_{seq}$ using sequential one. The optimal result here would be transducer $\mathcal{C}_{prp}$. This is the result of our proposed algorithm. We present an optimized composition algorithm with an adapted ϵ-filter which works alike the matching one, but has a special treatment for empty self-loop transitions. Thus it can create a recognition network with fewer states and transitions leading to less composition and decoding time. The proposed algorithm can also be applied to many current, widely used composition algorithms including on-the-fly ones.

First of all in section 2 we define WFSTs. The composition algorithm in general is explained in section 3 followed by a detailed view on problems with existing ϵ-filters in section 4. Section 5 describes our composition algorithm which solves the empty self-loop problem. And finally section 6 presents practical results that can be obtained for recognition network size. Section 7 gives a short summary.

2. Weighted Finite-State Transducers

A weighted finite-state transducer $\mathcal{A} = (Z, I, F, X, Y, S, E)$ consists of:

- a finite set of states $Z^{\mathcal{A}} = \{z_0, z_1, \ldots\}$,
- a set of initial states $I^{\mathcal{A}} \subseteq Z$,
- a set of final states $F^{\mathcal{A}} \subseteq Z$,
- an input alphabet $X^{\mathcal{A}} = \{\epsilon, x_0, x_1, \ldots\}$,
- an output alphabet $Y^{\mathcal{A}} = \{\epsilon, y_0, y_1, \ldots\}$,
- a weight semiring $S = \{\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1}\}$ and
- a set of transitions $E^{\mathcal{A}} \subseteq Z^{\mathcal{A}} \times (X^{\mathcal{A}} \cup \{\epsilon\}) \times (Y^{\mathcal{A}} \cup \{\epsilon\}) \times \mathbb{K} \times Z^{\mathcal{A}}$.

A transition $e = (z(e), x(e), y(e), w(e), z'(e)) \in E^{\mathcal{A}}$ leads from state $z(e)$ to state $z'(e)$. Thereby the input symbol $x(e)$ is consumed and the output symbol $y(e)$ is emitted. $w(e)$ denotes the weight of the transition.

A path $U = (e_1, \ldots, e_K)$ is a sequence of transitions (with $z'(e_i) = z(e_{i+1})$). The weight of a path is the product of the transition weights in the semiring S: $w(U) = w(e_1) \otimes \ldots \otimes w(e_K)$. The other functions ($z$,$z'$,$x$,$y$) defined for transitions can be used for paths in an analog way.

Overall a WFST assigns to every pair of input and output strings a single weight. We say the transducer translates the string x in the string y with the weight $\mathcal{A}(x, y)$ if the weight is greater than $\overline{0}$.

$$\mathcal{A}(x, y) = \bigoplus_{\substack{U \in \mathcal{A}: \\ z(U) \in I^{\mathcal{A}} \wedge z'(U) \in F^{\mathcal{A}} \wedge \\ x(U) = x \wedge y(U) = y}} w(U) \qquad (2)$$

Algorithm 1 General composition algorithm ($\mathcal{C} = \mathcal{A} \circ \mathcal{B}$) with use of ϵ-filter $\mathcal{E}$ (will be replaced with matching ($\mathcal{E}_{mch}$), sequential ($\mathcal{E}_{seq}$) or proposed ($\mathcal{E}_{prp}$) one)

1. Add empty self-loops to every state in $\mathcal{A}$ and $\mathcal{B}$:
   $E^{\mathcal{A}} \leftarrow E^{\mathcal{A}} \cup \{(z_i, \epsilon, \epsilon_{cps}, \overline{1}, z_i) | z_i \in Z^{\mathcal{A}}\}$
   $E^{\mathcal{B}} \leftarrow E^{\mathcal{B}} \cup \{(z_i, \epsilon_{cps}, \epsilon, \overline{1}, z_i) | z_i \in Z^{\mathcal{B}}\}$
2. Create initial states and queue Q:
   $I^{\mathcal{C}} \leftarrow \{(z_a, z_e, z_b) | z_a \in Z^{\mathcal{A}}, z_e \in Z^{\mathcal{E}}, z_b \in Z^{\mathcal{B}}\}$
   $Z^{\mathcal{C}} \leftarrow I^{\mathcal{C}}$, $Q \leftarrow I^{\mathcal{C}}$
3. Terminate if queue empty: $Q = \emptyset$
4. Remove one state from queue:
   $Q \leftarrow Q \setminus \{z_c = (z_a, z_e, z_b) \in Q\}$
5. If $z_c \in (F^{\mathcal{A}} \times F^{\mathcal{E}} \times F^{\mathcal{B}})$ then $F^{\mathcal{C}} \leftarrow F^{\mathcal{C}} \cup \{z_c\}$
6. For all transitions e_a, e_e, e_b with $y(e_a) = x(e_e)$, $y(e_e) = x(e_b)$ and $(z(e_a), z(e_e), z(e_b)) = z_c$ do:
   (a) If state $z'_c = (z'(e_a), z'(e_e), z'(e_b)) \notin Z^{\mathcal{C}}$ then create it:
   $Z^{\mathcal{C}} \leftarrow Z^{\mathcal{C}} \cup \{z'_c\}$, $Q \leftarrow Q \cup \{z'_c\}$,
   (b) Create new transition:
   $E^{\mathcal{C}} \leftarrow E^{\mathcal{C}} \cup \{(z_c, x(e_a), y(e_b), w(e_a) \otimes w(e_b), z'_c)\}$
7. Goto step 3.

3. Composition in general

In the section 5 we will propose an extension of the classical composition algorithm, so for understanding reasons we will give a short introduction into the common algorithm. A more detailed presentation can be found in [1]. The intention of composition is to combine one transducer $\mathcal{A}$ which translates from strings in alphabet X to strings is alphabet Y with another transducer $\mathcal{B}$ which translates from strings in alphabet Y to Z so that the composed transducer $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$ translates directly from X to Z. Therefore the translation weights of $\mathcal{C}$ have to be specified in the following way:

$$\mathcal{C}(x, z) = \bigoplus_{y \in Y^*} \mathcal{A}(x, y) \otimes \mathcal{B}(y, z) \qquad (3)$$

An algorithm which generates the transducer $\mathcal{C}$ from $\mathcal{A}$ and $\mathcal{B}$ is given in algorithm 1. This algorithm uses an ϵ-filter to handle empty transitions (transitions with output symbol $y(e) = \epsilon$ in the left operand $\mathcal{A}$ or with input symbol $x(e) = \epsilon$ in the right operand $\mathcal{B}$). The main purpose of the ϵ-filter is to make sure that only one path is used through every transducer part consisting of empty transitions exclusively. Otherwise the composition result would be incorrect. Figure 2 shows the matching and the sequential ϵ-filter. Both filter use different ϵ-symbols to distinguish between an ϵ-symbol existing in the source transducers (ϵ) or a symbol inserted by the algorithm (ϵ_{cps}). Later we will use another symbol (ϵ_l) for existing empty self-loop transitions. In composition with the filter every ϵ-symbol (ϵ, ϵ_{cps} and ϵ_l) is treated as real symbol and compared with other symbols in that way. The only exception are transitions with symbols $X : X$ in ϵ-filter. They match in composition if the output of the left operand $y(e_a)$ is equal to the input of the right one $x(e_b)$ and is not an ϵ-symbol.

4. Problems with existing ϵ-filters

The algorithm outlined in section 3 always leads to a correct composition result, if you use either the matching or the sequential ϵ-filter. But in some cases it does not generate an optimal solution. In other words there may be transducers with fewer states or fewer transitions which are equivalent to the generated ones.

A case for a non optimal algorithm result for the matching ϵ-filter is when either one of the source transducers contains empty loops. An empty loop is a path U through the transducer with $z(U) = z'(U)$ and $y(U) = \epsilon$ for the left operand or $x(U) = \epsilon$ for the right one. A special case are empty transitions which originate at their own terminal state. We discuss only that special case because it occurs quite frequently in speech and pattern recognition. Especially the transducers of Hidden-Markov-Models have self-loops without an output symbol at ev-

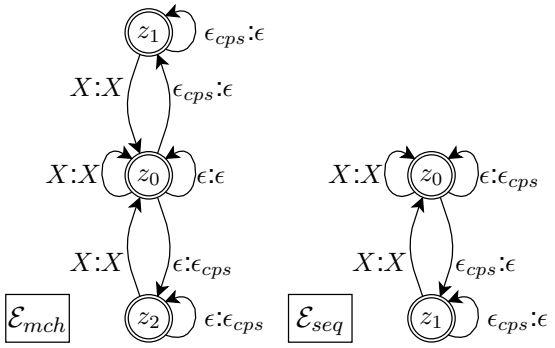

Figure 2: Matching and sequential ϵ-filter

Algorithm 2 First extension to composition algorithm (preprocessing)

0. Rename symbol of empty self-loop transitions

   (a) For all transitions $e_a \in E^{\mathcal{A}}$
   with $z(e_a) = z'(e_a)$ and $y(e_a) = \epsilon$
   do: $y(e_a) \leftarrow \epsilon_l$

   (b) For all transitions $e_b \in E^{\mathcal{B}}$
   with $z(e_b) = z'(e_b)$ and $x(e_b) = \epsilon$
   do: $x(e_b) \leftarrow \epsilon_l$

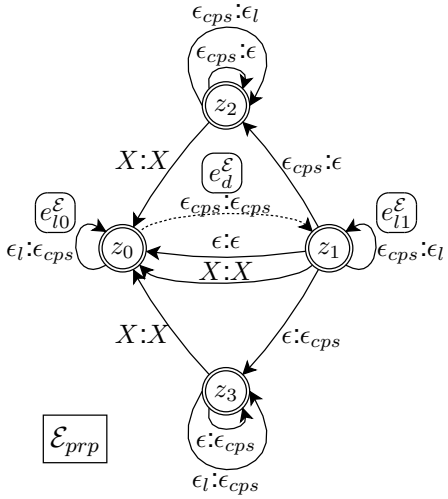

Figure 3: Proposed ϵ-filter (Symbols in round boxes are transition identifiers)

ery emitting state. Figure 1 shows the composition result $\mathcal{C}_{mch}$ for composition of $\mathcal{A}$ and $\mathcal{B}$ using matching filter. You see that the empty self-loop transition of state z_0 in $\mathcal{A}$ results in three distinct ones. The transducer $\mathcal{C}_{mch}$ can be transform to $\mathcal{C}_{prp}$ by minimization algorithm (see [11, 12, 3]). In many cases this is impracticable for real recognition networks, especially if we want to do on-the-fly composition in combination with decoding. We have observed that for bigger source transducers the effect increases strongly, in particular for a sequence of states with empty self-loop transitions (like in Hidden-Markov-Models).

Because of the empty self-loop problem the sequential ϵ-filter is used in many cases. Here the empty transitions of both source transducer are processed sequentially. So there is no matching between empty transitions in both sources. The created transducer is also not always optimal. In Figure 1 you see that the transition from z_{000} to z_{100} and the next to z_{111} in transducer $\mathcal{C}_{seq}$ may be combined to the transitions from z_{000} to z_{101} in $\mathcal{C}_{prp}$. Empty transitions in both source transducers occur if they have not been determinized. This is useful for many regular grammar recognizer where determinization may enlarge the transducer considerably (see table 1).

5. Proposed composition algorithm

The previous section shows that both matching and sequential ϵ-filter in composition cannot produce optimal results especially for Hidden-Markov-Model-like WFSTs. Figure 3 presents a solution for that problem. It is an ϵ-filter which does a matching on ϵ-transitions and can treat empty self-loop transitions separately. To use that filter we need to modify the composition algorithm given in algorithm 1. First of all we need to add a step 0. (see algorithm 2) which renames the ϵ-symbols of self-loop transitions in both operands to identify them with the ϵ-filter.

Algorithm 3 Second extension to composition algorithm (postprocessing)

Transition definitions (see figure 3):

$e_d^{\mathcal{E}} \leftarrow$ dotted transition in ϵ-filter $\mathcal{E}_{prp}$
$e_{l0}^{\mathcal{E}} \leftarrow$ self-loop transition at state z_0 in ϵ-filter $\mathcal{E}_{prp}$
$e_{l1}^{\mathcal{E}} \leftarrow$ self-loop transition at state z_1 in ϵ-filter $\mathcal{E}_{prp}$

8. For all transitions $e^{\mathcal{C}} \in E^{\mathcal{C}}$ which are derived from $e_d^{\mathcal{E}}$

   (a) If there is no transition $e_{l0}^{\mathcal{C}} \in E^{\mathcal{C}}$ which is derived from $e_{l0}^{\mathcal{E}}$ with $z(e_{l0}^{\mathcal{C}}) = z(e^{\mathcal{C}})$ then

   - For all transitions $e_{chg}^{\mathcal{C}} \in E^{\mathcal{C}}$ with $z'(e_{chg}^{\mathcal{C}}) = z(e^{\mathcal{C}})$ do $z'(e_{chg}^{\mathcal{C}}) \leftarrow z'(e^{\mathcal{C}})$
   - Remove state $z(e^{\mathcal{C}})$ and transition $e^{\mathcal{C}}$:
     $Z^{\mathcal{C}} \leftarrow Z^{\mathcal{C}} \setminus z(e^{\mathcal{C}})$
     $E^{\mathcal{C}} \leftarrow E^{\mathcal{C}} \setminus e^{\mathcal{C}}$

   (b) Else if there is no transition $e_{l1}^{\mathcal{C}} \in E^{\mathcal{C}}$ which is derived from $e_{l1}^{\mathcal{E}}$ with $z(e_{l1}^{\mathcal{C}}) = z'(e^{\mathcal{C}})$ then

   - For all transitions $e_{chg}^{\mathcal{C}} \in E^{\mathcal{C}}$ with $z(e_{chg}^{\mathcal{C}}) = z'(e^{\mathcal{C}})$ do $z(e_{chg}^{\mathcal{C}}) \leftarrow z(e^{\mathcal{C}})$
   - Remove state $z'(e^{\mathcal{C}})$ and transition $e^{\mathcal{C}}$:
     $Z^{\mathcal{C}} \leftarrow Z^{\mathcal{C}} \setminus z'(e^{\mathcal{C}})$
     $E^{\mathcal{C}} \leftarrow E^{\mathcal{C}} \setminus e^{\mathcal{C}}$

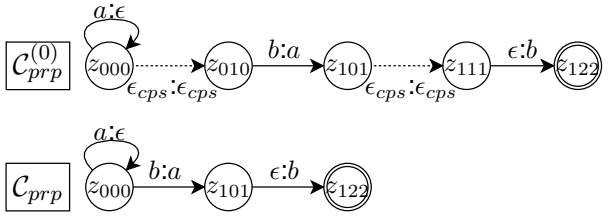

Figure 4: Composition result with proposed ϵ-filter

The composition result of the example transducers $\mathcal{A}$ and $\mathcal{B}$ by use of the proposed ϵ-filter $\mathcal{E}_{prp}$ and the preprocessing step 0. is the transducer $\mathcal{C}_{prp}^{(0)}$ in figure 4. It still contains two more transitions compared to the optimal result (transducer $\mathcal{C}_{prp}$). Those two transitions are generated through composing with the dotted transition ($e_d^{\mathcal{E}}$) in the proposed ϵ-filter $\mathcal{E}_{prp}$.

In general transitions generated through composing with the dotted one ($e_d^{\mathcal{E}}$) in $\mathcal{E}_{prp}$ are omissible if and only if the self-loop transition at state z_0 in $\mathcal{E}_{prp}$ and the self-loop transition at state z_1 are not used both before respectively after the dotted one. This is true in many cases. We add a postprocessing step (see algorithm 3) to the end of the composition algorithm which analyzes the composition result and removes the omissible transitions. Thereafter we get $\mathcal{C}_{prp}$ as composition result of transducer $\mathcal{A}$ and $\mathcal{B}$.

For on-the-fly processing the step of deleting omissible transitions has to be carried out while composition is done. A postprocessing step has no benefit. This is also possible with the proposed algorithm. After a state is processed, which refers to the state z_1 in the ϵ-filter $\mathcal{E}_{prp}$, an omissible transition can safely be removed because no more paths will reach the state z_1. We will not give a detailed algorithm. It can easily be derived from the postprocessing algorithm 3. Altogether the proposed composition is usable for on-the-fly and offline processing.

The composed transducer retrieved by the proposed algorithm is equivalent to that by the matching and sequential one. We have verified that by experiments. All transducers produce exactly the same recognition results.

	ϵ-filter	not determinized $\mathcal{G}$	determinized $\mathcal{G}$
#states	matching	1929	7038
	sequential	1894	5640
	proposed	**1485**	5640
#transitions	matching	3927	13077
	sequential	3818	10239
	proposed	**3007**	10239

Table 1: Recognition network sizes for spoken German number recognition

Lexicon size (words)		3k	9k	3k	9k
n-gram type		bi	bi	tri	tri
#states	matching	110k	359k	1524k	2467k
	sequential	89k	289k	1228k	1984k
	proposed	89k	289k	1228k	1984k
#transitions	matching	256k	733k	2938k	4701k
	sequential	213k	593k	2342k	3725k
	proposed	213k	593k	2342k	3725k

Table 2: Recognition network sizes for statistical language models

6. Results

To evaluate the proposed algorithm we have generated recognition networks using a regular grammar for German number recognition. It models spoken numbers with up to 9 digits before and an arbitrary number of digits after the decimal point (i.e. three thousand one hundred twenty dot five seven). The lexicon contains 40 words and is build upon 43 phoneme Hidden-Markov-Models. The phoneme models are context-independent, so not context-dependency model is used. The recognition network is obtained either by equation (4) or (5). For both composition operations the same ϵ-filter is used (either matching, sequential or proposed one).

$$\mathcal{R} = (\mathcal{H} \circ \det(\mathcal{L})) \circ \mathcal{G} \quad (4)$$

$$\mathcal{R} = (\mathcal{H} \circ \det(\mathcal{L})) \circ \det(\mathcal{G}) \quad (5)$$

In table 1 you see the recognition network sizes with and without determinization of the grammar transducer. Without determinization our algorithm reduces the network size by 20 to 25 percent. Determinization of the regular grammar transducer for German numbers removes empty transitions, so the result is the same as for the sequential ϵ-filter. But determinization prohibits the compact representation of the grammar using empty transitions. Therefore the network size rises approximately by a factor of 3.5, which wastes memory and decoding time especially on embedded devices.

Another experiment using statistical language models shows in table 2 that the proposed algorithm does produce at least as good results as the classical ones. For the lexicon and language model generation we use the German Verbmobil database [13]. We generate two lexicon models with different sizes, one with 2947 words (abbreviated as 3k lexicon model) and another with 9037 words (abbreviated as 9k lexicon model). We use a bigram and a trigram language model with backoff transitions. Symbol insertion as proposed in [14] is used to make the lexicon and the language model determinizable. Compared to the matching ϵ-filter we reduce the recognition network size by nearly 20 percent. For sequential ϵ-filter there is no change observable caused by the determinized grammar transducer.

All recognition experiments produce identical recognition rates. Therefore we claim that the generated transducers are equivalent.

7. Conclusion

We have presented special problems in composition algorithm using matching or sequential ϵ-filter. One is the existence of empty loop transition like in Hidden-Markov-Models. Another is the composition of non-determinized transducers. We have shown that for recognition tasks using regular grammars like speech dialog systems this is very useful. One solution to deal with that problems is to modify the matching ϵ-filter so that it has a special treatment for such transitions. Advantageously this solution can be used for all composition algorithms based on the classical ϵ-filters, or similar ones and it is also suitable for on-the-fly composition. The composed recognition network is smaller or equal in size to one generated with classical filters. For a spoken German number regular grammar the network size is reduced by nearly 25 percent. So it improves composition as well as decoding speed.

8. References

[1] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[2] ——, "Speech recognition with weighted finite-state transducers," in *Handbook of Speech Processing*, Y. J. Benesty and M. Sondhi, Eds. Springer, 2008, pp. 559–582.

[3] M. Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 23, no. 2, 1997.

[4] H. J. Dolfing and I. L. Hetherington, "Incremental language models for speech recognition using finite-state transducers," in *Automatic Speech Recognition and Understanding, 2001. ASRU '01. IEEE Workshop on*, 2001, pp. 194–197.

[5] D. Caseiro and I. Trancoso, "A specialized on-the-fly algorithm for lexicon and language model composition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 4, pp. 1281–1291, july 2006.

[6] T. Hori and A. Nakamura, "Generalized fast on-the-fly composition algorithm for wfst-based speech recogition," in *Interspeech 2005*, sep 2005, pp. 557–560.

[7] T. Oonishi, P. R. Dixon, K. Iwano, and S. Furui, "Implementation and evaluation of fast on-the-fly wfst composition algorithms," in *Interspeech 2008*, sep 2008, pp. 2110–2113.

[8] C. Allauzen, M. Riley, and J. Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," in *Interspeech 2009*, 2009.

[9] O. Cheng, J. Dines, and M. M. Doss, "A generalized dynamic composition algorithm of weighted finite state transducers for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, april 2007, pp. IV–345 – IV–348.

[10] T. Oonishi, P. R. Dixon, K. Iwano, and S. Furui, "Generalization of specialized on-the-fly composition," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, april 2009, pp. 4317–4320.

[11] A. V. Aho and J. E. Hopcroft, *The Design and Analysis of Computer Algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1974.

[12] D. Revuz, "Minimisation of acyclic deterministic automata in linear time," *Theoretical Computer Sience*, vol. 92, pp. 181–189, 1992.

[13] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal, "The karlsruhe-verbmobil speech recognition engine," in *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97) -Volume 1*. Washington, DC, USA: IEEE Computer Society, 1997, p. 83.

[14] C. Allauzen and M. Mohri, "Generalized optimization algorithm for speech recognition transducers," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, vol. 1, april 2003, pp. I–352 – I–355.