Generalized Fast On-the-fly Composition Algorithm for WFST-Based Speech Recognition

Takaaki Hori, Atsushi Nakamura

NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan
{hori,ats}@cslab.kecl.ntt.co.jp

Abstract
This paper describes a Generalized Fast On-the-fly Composition (GFOC) algorithm for Weighted Finite-State Transducers (WFSTs) in speech recognition. We already proposed the original version of GFOC, which yields fast and memory-efficient decoding using two WFSTs. GFOC enables fast on-the-fly composition of three or more WFSTs during decoding. In many cases, it is actually difficult or impossible to organize an entire transduction process of speech recognition using only one or two WFSTs since some types of models considerably enlarge after written in WFST form. For example, a class-based language model, a combined language model, or a long-span language model often results in a much larger transducer than a simple word n-gram model for the same vocabulary. For this problem, on-the-fly composition of multiple WFSTs is efficient since such a huge model can be used after decomposing it into small WFSTs.

In this paper, we propose an extension of the GFOC algorithm: the Generalized Fast On-the-fly Composition (GFOC) algorithm. GFOC enables efficient on-the-fly composition of three or more WFSTs during decoding while only two WFSTs can be composed by the original algorithm. Thus GFOC simplifies incorporating different language models and complicated constraints in the framework of WFST since it can deal with composed multiple WFSTs.

We conducted experiments in a benchmark test of Corpus of Spontaneous Japanese [6] (CSJ). In addition to a word n-gram model, we introduced a class n-gram model and a word-class combined n-gram model. We evaluated the performance of a simple search method using a single transducer, the original FOC, and GFOC using different language models.

1. Introduction
Recently, the Weighted Finite-State Transducer (WFST) approach has become a promising alternative formulation to traditional decoding approaches in speech recognition. It offers a unified framework representing various knowledge sources and produces a full search network optimized up to the HMM states [1]. The optimization step minimizes search space and accelerates decoding. However, when using a very large vocabulary, lexicon, a detailed language model, or complicated transduction model, an enormous huge transducer is usually generated by composing all the components. Accordingly, both high computational cost and large memory usage are necessary for decoding even if the WFST is optimized. Especially for memory usage, this problem is serious because states and transitions will be made for all possible combinations of the input-output relation between each adjacent pair of the cascaded WFSTs. Thus, the size of such a composite transducer easily exceeds the limitations of standard personal computers. To solve this problem, we proposed a Fast On-the-fly Composition algorithm [4] which is a fast version of the standard on-the-fly composition [2][3].

On-the-fly composition is a practical alternative to using a fully composed transducer. The set of WFSTs are separated into two groups, and in each group, one WFST is composed and optimized before decoding. Composition between the two WFSTs is performed as necessary during decoding. A lot of memory is saved by using on-the-fly composition, but search efficiency was decreased due to composition overhead. The Fast On-the-fly Composition (FOC) algorithm accelerates decoding by performing a one-pass Viterbi search based only on the first transducer of the two. The second transducer is only used to rescore the hypotheses generated during the search with minimal overhead. In our evaluation experiments, FOC achieved real-time 1.8 million-word vocabulary continuous speech recognition.

However, there are actually many cases when it is difficult or impossible to organize an entire transduction process of speech recognition using only one or two WFSTs since some types of models considerably enlarge after written in WFST form. For example, a class-based language model, a combined language model, or a long-span language model often results in a much larger transducer than a simple word n-gram model for the same vocabulary. For this problem, on-the-fly composition of multiple WFSTs is efficient since such a huge model can be used after decomposing it into small WFSTs.

2. Weighted Finite-State Transducers in Speech Recognition
WFST is a finite state network associating input and output symbols on each arc that can be weighted with a log probability value. Speech recognition is a transduction process from speech input to the corresponding word sequence. The process can be represented as a cascade of several transductions, each of which can be written in WFST form. Those WFSTs can then be combined by using a composition operator, leading to the integration of the underlying knowledge sources into a single input-output relation. An integrated WFST for speech recognition can be composed as

\[ H \circ C \circ L \circ G, \quad (1) \]

where \( H, C, L, \) and \( G \) are WFSTs for a state network of triphone HMMs, a set of connection rules for triphones, a pronunciation lexicon, and a trigram language model, respectively; \( \circ \) represents the composition operator. As a result, decoding is a one-pass search problem in a single integrated network including cross-word triphones and a trigram language model. Once the network is further optimized by proceeding to weighted de-
terminization and minimization, search efficiency dramatically increases.

When a single WFST is composed of all knowledge sources for LVCSR, the number of states and transitions often becomes so large that an enormous amount of memory is required during decoding. To avoid this problem, on-the-fly composition is available. Generally, in on-the-fly composition, two WFSTs are prepared and composed during decoding.

In [3], the WFSTs are divided into two groups:

\[(H \circ C \circ L \circ G_1) \circ G_{3/1}, \]

where we have introduced an on-the-fly composition operator “\(\circ\)” to distinguish it from the general composition operator “\(\otimes\)”;

\(G_1\) and \(G_{3/1}\) are WFSTs for a unigram model and for a trigram model adjusted by dividing each trigram probability by the unigram probability of \(G_1\). By including \(G_1\) in the first group, search efficiency improves since \(G_1\) yields prospect to each hypothesis during the search.

By using on-the-fly composition, memory usage is usually greatly saved since only a part of the entire composite search space is organized as necessary during the search. However, the entire space itself is larger than the full composition method since it is difficult to optimize the search space during decoding. Furthermore, on-the-fly composition overhead also increases the amount of computation required for decoding.

### 3. GFOC for Multiple WFSTs

First we show the concept of the original FOC. We assume that the decoder searches for the minimal cost path, where cost means an accumulated weight defined under Tropical semiring. Thus the weight of each transition is based on minus log probability that derives \(W\).

Equation (3) can be rewritten as

\[W(X \rightarrow Y) = \min_Y \{W_A(X \rightarrow Y) + W_B(Y \rightarrow Z)\}. \]

This means that the algorithm for finding \(Y\) can be applied to obtain \(W(X \rightarrow Z)\), which is equal to \(W(X \rightarrow Y)\), where \(\min_Z W_B(Y \rightarrow Z)\) can be assumed to be the compensation weight.

A one-pass Viterbi search is performed based only on \(A\) but not based on \(A \circ B\). \(B\) is only used to compensate accumulated weights of hypotheses generated by \(A\) with minimal overhead. Hence FOC is fast and memory-efficient compared to standard on-the-fly composition.

In frame-synchronous processing, hypotheses are generated by \(A\), each of which represents an individual state transition process in \(A\). If a new hypothesis \(h\) is generated by adding a new transition \(e\) to an existing hypothesis, \(h\) can be compensated with

\[\min_B o[h] \rightarrow o[f] \]

by \(B\) only when \(e\) has a non-epsilon output, where \(f\) indicates a hypothesis generated by \(B\) accepting \(o[h]\) that denotes the output symbol sequence of \(h\). \(o[f]\) denotes the output symbol sequences of \(f\) as well. By associating each hypothesis \(h\) with a list of hypotheses \(g[h]\) produced by \(B\), the compensation process can be efficiently performed. Here, \(g[h]\) means a set of hypotheses generated by \(B\) when \(o[h]\) is given as an input symbol sequence for \(B\). We call the hypotheses in \(g[h]\) “cohypotheses” to distinguish them from the hypotheses produced by \(A\). GFOC is an extended version of FOC for composing multiple WFSTs. Before describing the extended algorithm, we show a basic procedure in cases using a single WFST. We introduce some symbols, quantities, and functions as follows.

**Notation related to WFST**

- \(F\) : set of final states
- \(\lambda\) : initial weight
- \(\rho(s)\) : final weight of state \(s\) \((s \in F)\)
- \(E(s, x)\) : set of transitions accepting label \(x\) from state \(s\)
- \(o[e]\) : output label of transition \(e\)
- \(w[e]\) : weight of transition \(e\)

**Variables in the search**

- \(h\) : hypothesis indicating a path
- \(H\) : list of hypotheses
- \(\alpha(h)\) : accumulated weight for \(h\).
- \(n[h]\) : state reached by \(h\).

**Basic functions for the search**

- \(ReadSym()\) : return the next input symbol
- \(NewHyp()\) : generate a hypothesis and initialize it as \(h\)
- \(Insert(H, h)\) : insert \(h\) into \(H\) (If there is a hypothesis \(h\) that has reached \(n[h]\) in \(H\), either \(h\) or \(h\) having a smaller weight is retained).

Figure 1 shows an example of the algorithm. At line 1, initial hypothesis \(h\) is generated. At line 2 and 3, \(h\) is weighted with the initial weight and stored in \(H\). In lines 4 - 13, the main search step is performed until the last input symbol is processed. The main step iterates the generation of hypothesis by \(NewHyp()\) and their recombination by \(Insert()\). Finally at line 14, the best hypothesis \(BestHyp\) is chosen among the hypotheses that have reached one of the final states. In the algorithm, however, transitions by \(e\) symbol are not assumed for simplification. Pruning of hypotheses is also omitted.

Next, we explain the GFOC algorithm illustrated in Fig. 2. Given a set of \(M\) WFSTs, the algorithm searches for the minimal cost path of the \(M\)-th WFST while applying on-the-fly

```
1 h = NewHyp(0)
2 \(\alpha(h) = \lambda\)
3 Insert(H, h)
4 while the next input exists do
5 \(x = ReadSym()\)
6 \(H' = \phi\)
7 for each pair of \(h \in H\) and \(e \in E(n[h], x)\) do
8 \(h' = NewHyp(h \cdot e)\)
9 \(\alpha(h') = \alpha(h) + w[e]\)
10 Insert(H', h')
11 end for
12 \(H = H'\)
13 end while
14 BestHyp = \(\min_{h \in H, n[h] \in F} \alpha(h) + \rho(n[h])\)
```
composition. The symbols and quantities in Fig. 2 are extended
by subscript composition. The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.

The symbols and quantities in Fig. 2 are extended.
In this case, each transition weight is derived by interpolating transitions than FULL. FOC(CLM), however, needs about 5 domain. D FOC(WLM) because a huge

We set the number of classes to 2,000.

class model. In this experiment, a set of classes is derived using used in decoding. WLM and CLM indicate a word model and a

position is derived by counting all states or transitions of WFSTs respectively. #WFST denotes the number of WFSTs composed ing only two WFSTs and when using three or more WFSTs,

FOC and GFOC indicate fast on-the-fly composition when us-

optimization step was performed after each composition step.

First we evaluated the size of the WFSTs. Table 1 shows the number of states and transitions of each WFST. In the table,

FULL (WLM)

FULL (CLM)

FOC (WLM)

FOC (CLM)

GFOC (CLM)

GFOC (WLM+CLM)

\( G \)

- 248,611 751,458

\( T \)

- 1 30,000

\( B \)

- 170,154 638,986

\( G^C \)

- 166,475 3,959,781

\( WLM+CLM \) indicates an interpolated model of word and class n-grams. The WFST can be composed as

\[
(H \circ C \circ L \circ G) \circ G_{3/1} \circ \tilde{\gamma}_1(T_0^{-1} \circ D_{3/1}).
\]

In this case, each transition weight is derived by interpolating the weights of \( G_{3/1} \) and \( G_{5/1} \). We multiplied each weight in \( G_{3/1} \) and \( G_{5/1} \) by 0.5. This means linear interpolation in log-domain.

In table 1, it is shown that FOC and GFOC need less transitions than FULL. FOC(CLM), however, needs about 5 times transitions of FOC(WLM) because \( T^{-1} \circ D \) results in a huge \( G^C \). In GFOC(CLM), the total transitions is less than FOC(WLM) because \( D \) and \( T \) are not composed beforehand.

Second, we performed speech recognition using each WFST set. Figure 4 shows the relationship between word accuracy (WACC) and real-time factor (RTF) in each method when changing the beam width parameter of the decoder. It is shown that CLM yielded higher accuracies than WLM in all decoding methods, and both FOC and GFOC outperformed FULL(WLM). We could not evaluate FULL(CLM) in speech recognition because of the limitations of memory. In CLM, GFOC worked 2 - 7% faster than FOC, and the process memory size was less than a half of FOC. Finally GFOC(WLM+CLM) using four separated transducers worked well with small memory usage, and marked the best performance.

5. Conclusions

In this paper, we proposed a Generalized Fast On-the-fly Composition (GFOC) algorithm for WFST-based speech recognition. In a 30k-word vocabulary spontaneous speech transcription task, GFOC yielded fast and memory efficient decoding especially when using class-based language models. GFOC is also very convenient when off-line composition is difficult due to time and/or memory problems. GFOC has the potential to be successfully applied to other speech-input language processing. In the future, we would like to apply this technique to speech translation, speech summarization and so on.

6. Acknowledgments

We thank MIT’s SLS group for providing the tools for composing and optimizing WFSTs.

7. References


