# Modeling Polyphone Context with Weighted Finite-State Transducefrs

**Student Research Project/Studienarbeit**

Carnegie Mellon University
Language Technologies Institute

Universität Karlsruhe (TH)
Institut für Theoretische Informatik

submitted by

**Emilian Stoimenov**

in

January 2007

This work was supervised by

Dr. J. McDonough
Universität Karlsruhe (TH)

Prof. T. Schultz
Carnegie Mellon University

# Table of Contents

## Summary

Weighted finite-state transducers and the associated algorithms have become a popular means of constructing speech recognition decoding networks. It has been shown that building and optimizing the network offline contributes to a speech decoder's accuracy and speed [1]. Another advantage weighted finite-state transducers provide is a straightforward represnetation of all information sources involved in compiling the search space.

However, constructing the final recognition transducer can prove to be very memory demanding and time consuming. The explicit expansion of the transducer mapping from context-dependent subword units to phones contributes a lot to the inefficiency of the offline recognition network construction. In this paper we describe an algorithm for directly constructing a finite-state transducer mapping directly from sequences of gaussian mixture models to context-independent phones. Based on the 5-step procedure descirbed in [2], the method involves parsing a decision tree to derive an efficient representation of its leaves. It then enumerates and stores all possible $k$-long cluster combinations that could form a valid Hidden Markov Model according to the decision tree. An algorithm is then proposed, which interconnects these sequences to form the final transducer.

An important advantage of the connection algorithm is that it is local, allowing for a more efficient on-the-fly implementation and combination with other local weighted finite-state algorithms. We used an on-demand expansion and immediate determinization to construct a transducer using a decision tree with 16,000 leaves, which would be impossible otherwise – either using the standard techniques explained in [3] or the static expansion in [4].

Finally a proof of the correctness of the connection algorithm is proposed, which compares the edges set of the constructed transducer with the explicit composition of a hidden Markov model transducer and a context-dependency transducer. We show that both transducers describe the same string-to-string mapping.

# 1 Introduction

Using phone context in speech recognition has proven to be very beneficial in the past [5]. This reflects the fact that each subword unit produced by our articulatory mechanism has been affected by a combination of the previously pronounced and the anticipated upcoming subword units. Therefore we can hardly expect to see a pure realization of a phone in real speech.

A more accurate speech recognizer can be built, if an acoustic model for every separate phone variant in each different context is trained. However, the number of possible context-dependent phones is very large, even after restraining oneself to the phones occuring in the dictionary and not crossing word boundaries. Therefore, training each subword unit becomes a problem, because the limited amount of training data is unable to cover the whole set of context-dependent models with sufficient number of examples.

In order to compensate for the great variance introduced by context-dependent subword units, some form of clustering must be used. Young and Woodland [6] suggest a data-driven iterative state clustering procedure, where all available states are put into equivalence classes according to a form of distance measure between the different classes. A furthest neighbour hierarchical clustering algorithm is proposed, whereby initially each state is assigned an individual cluster of its own, and then during the subsequent iterations these clusters are merged together, making up for insignificant differences between them as dictated by the employed distance function.

A more useful approach is to grow a phonetic decision tree [7], which chooses the suitable clusters for each subword unit model. Using expert linguistic knowledge, this solution has the benefit of providing clusters for *unseen* in the training data context-dependent subword units. Decision-tree state clustering also facilitates the introduction of higher span context dependency, by simply extending the stretch of the questions attached to the nodes in the tree. These and other advantages make the use of decision trees very popular in contemporary speech recognition systems.

The use of $N$-gram language models, large dictionaries and tied-state acoustic models can lead to a lot of redundancy in the recognition network, expanded by the decoder during recognition. This increases the memory requirements and the runtime of a recognizer. Weighted finite-state transducers and the associated composition, determinization and minimization algorithms provide an efficient and elegant framework to compile the recognition network offline and to eliminate the redundancy.

## 1.1 Weighted Finite-State Transducers

Weighted finite-state transducers (WFSTs) are an extension to finite-state automata, in that, on addition to the input symbol, each arc can have a weight and an output symbol. Each transition in a WFST consumes an input

symbol and outputs a symbol and a weight. A successful path in a WFST accepts a word from its input alphabet, outputs a word from its output alphabet, and assigns a weight to the occuring transduction. The weights on the arcs are members of a *semiring* $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ and are combined using the $\oplus$ semiring operation. The $\otimes$ and $\oplus$ operations are used in conjunction with other WFST algorithms [8], [3], [9].

Having an output alphabet enables WFSTs to combine, forming the composition of the modeled transductions. The composition of the transducers $A$ and $B$ is denoted by $A \circ B$ and is formed over the intersection of all possible output strings of $A$ and all possible input strings of $B$, thus emulating the immediate application of $B$ after $A$. Through composition many information sources can be integrated into a single weighted finite-state network, which can be searched to produce the 'best' mapping from the input alphabet of the last source, to the output alphabet of the first. However, this network can become enormous, having millions of states and transitions and exhibiting a lot of redundancy.

## 1.2 Speech Recognition with WFSTs

As described in [3],[8], weighted finite-state transducers can provide a very natural environment for speech recognition.

We start with a language model WFST $G$, which assigns probabilities to sequences of words. The language model is an acceptor, which is converted to a transducer by duplicating the input alphabet on the output side. As outlined in [8], a way to build a WFST from a set of $N$-gram counts is to proceed as in a de Bruijn graph construction, where the nodes are labeled by $n$-long word sequences. For each word sequence $ww'$ there is a transition labeled $w'$ leading to a state $w'w''$, where $w'w''$ is a sequence with a nonzero $N$-gram count. The weight on this arc is computed from the corresponding probability of occurrence and is represented in a negative log form.

The construction of the dictionary transducer $L$ is very straightforward, since a word has a finite number of pronunciations, each of which can be modeled by a simple string of arcs labeled with the phones in the transcriptions. The word is assigned to one of these arcs as an output label, typically the starting or the ending arc of a string. All these arcs share a begin node, and each final node of a sequence connects back to it.

The language model transducer is composed with the dictionary transducer to produce a transducer $L \circ G$, each path in which maps from a sequence of phones to a sequence of words with the right probability given by the language model. The composition $L \circ G$ is *determinized* to facilitate subsequent compositions.

As described in the introduction, we want to profit from having cross-word context-dependency phones. Therefore we need a transducer which maps from sequences of context-dependent phones to sequences of context-

independent phones. A *context-dependency* transducer $C$ can be constructed in the same way as the language model, and the needed probabilities estimated from a training corpus.

Finally, each context-dependent phone has an associated tied-state acoustic model, consisting of a sequence of cluster symbols. A *hidden Markov model* transducer $H$ is needed to map from sequences of cluster symbols to sequences of context-dependent phones. Similarly to a word, a context-dependent phone has a finite number of cluster sequence expansions, so we can construct $H$ the same way we construct $L$.

As mentioned, the final composed transducer $H \circ C \circ det(L \circ G)$ is inefficient in terms of size, since for example many context-dependent phones can share common cluster subsequences, but the composition algorithm creates multiple identical arc strings for them. Mohri showed [9] that a subsequent determinization, followed by a minimization eliminates the ambiguity and produces the minimal transducer.

The constructed recognition network describes a mapping from Gaussian mixture models to words, and can be synchronously searched [3], [10] to find the best possible sequence of observation symbols corresponding to an input utterance. The transducer maps this sequence to a sequence of words, which is in turn the best match output by the recognizer.

While very elegant and straightforward, this technique has certain limitations. The intermediate transducers can become very large, even after the aforementioned optimizations. For example, a fourgram language model with 2,260,500 fourgrams, 3,559,905 trigrams, 4,714,631 bigrams and 48,282 words has 9,057,717 states and 18,917,331 arcs.

The context-dependency transducer alone can be tremendous in size for large contexts, having $\mathcal{O}(n^{k-1})$ states and $\mathcal{O}(n^k)$ arcs for a context length $n$ and a phone set of size $k$. Even when many of the context-dependent phones are impossible, or would be filtered by the following composition with a hidden Markov model transducer, the fact that the expansion must be kept in memory, limits the size of the recognition networks, which can be built by composing WFSTs.

## 1.3 Related Work

There are approaches to directly construct the composition $H \circ C$ and so avoid the complete expansion of $C$. Chen [11], proposed a technique, where each question in a decision tree is encoded as a finite-state transducer (FST). A simple one-state transducer containing self loops for all phones in the phone set is iteratively composed with these FSTs to produce a final three-state cluster expansion of a phone. The one state transducer is first extended with placeholders for the decision tree leaves. These placeholders are iteratively expanded by the compositions, as each following FST rewrites the output of the previous one until a final leaf FST is reached. A degree of

nondeterminism in the resulting graphs can be expected, since the final expansion of a phone depends on the left and right phone contexts. This is overcome by first applying the questions for the left-hand-side contexts, determinizing and minimizing, then reversing the graph and applying the right-hand-side questions.

The work of Chen has its roots in a paper by Sproat and Riley [12]. In it they use the fact that the questions in a phonetic decision tree can be represented as regular expressions and a leaf is an intersection of all the regular expressions traversed on the way to the leaf. Each leaf's regular expression can be compiled to a WFST and the intersection of all such WFSTs implements the mapping from the input alphabet of context-dependent phones, to the output alphabet of allophonic classes (clusters), which the decision tree represents.

Similarly to [11] and [12], Schuster and Hori [2] presented a simple 5-step procedure for constructing a WFST directly from decision trees to circumvent the static expansion of $C$. The procedure avoids expanding all polyphones explicitly, by first parsing the decision tree, and forming all Gaussian mixture combinations admitted by it. Thereafter it interconnects these Gaussian mixture sequences appropriately to form a transducer which maps from Gaussian mixture models to context-independent phones.

## 2 Direct Construction of the Composition

We describe an algorithm, which corrects a mistake in [2]. Our solution is based on the 5-step-procedure, but we show that a modified version of the fourth step generates the right in-between phone connections and the constructed transducer provides the right mapping from cluster sequences to phone sequences, as allowed by the decision trees. We also prove the correctness of the algorithm and the fact that it indeed emulates the explicit composition of $H$ and $C$.

### 2.1 Parsing the Decision Tree

It is important to clarify how the cluster symbols in the HMM transducer $H$ are derived. Since the number of polyphones for a general large-vocabulary speech recognition system is tremendous, it is the case that there is not enough amount of training data to train each separate parameter of a given HMM. Therefore, typically the states of all HMM are *clustered*, thereafter computing only the Gaussian components of each cluster, having $k$ clusters per HMM. This follows the assumption that a specific output distribution of an HMM will be common among different realizations of a phone as a polyphone [5].

The state clustering procedure is realized by $k$ decision trees, each of which is a binary tree with a question attached to each node and two subtrees

corresponding to the 'yes' and 'no' answer of this question. When a cluster is needed for the $i$-th state of an HMM, the $i$-th decision tree is parsed from the root, thereby answering the questions for the HMM's context-dependent phone. Each question refers to the phone at position one of the $n$ left or right contexts being in a predefined *phone class*. Thus, a leaf in the decision tree is a set of phones for each context position, determined by the answers to the questions along the path from the root to this leaf. The answers for a specific context position $i$ must not contradict each other in the different clusters assigned to a polyphone HMM.

Since each leaf in the decision tree is a set of $k$ permutations of a finite set of phones, it can be efficiently modeled by a matrix of bits. Each column of the matrix represents a context position and each row is assigned to a phone. When a specific phone is allowed at a particular context position of a leaf in the decision tree, the corresponding bit in the matrix is set to 1, otherwise it is set to 0. For example, in Figure 1 the phone 'B' is allowed at context position +1 and the phone '{Z:WB}' at context position -2. The phone 'Z' is not allowed anywhere.

| | polyphone position | | | | |
|---|---|---|---|---|---|
| phone | 0 | 1 | 2 | 3 | 4 |
| AH | 0 | 0 | 1 | 1 | 0 |
| {AH:WB} | 0 | 0 | 1 | 0 | 0 |
| B | 0 | 0 | 0 | 1 | 0 |
| {B:WB} | 0 | 1 | 0 | 0 | 1 |
| ⋮ | | | ⋮ | | |
| Z | 0 | 0 | 0 | 0 | 0 |
| {Z:WB} | 1 | 0 | 0 | 0 | 0 |

Figure 1: A bit matrix corresponding to the cluster modelling the context dependent phone 'AH'

Depending on the decision tree, a bit matrix can be largely sparce, containing many zeroes and few ones. Therefore it might be beneficial to compress this representation in the interest of saving memory. However, another representation could compromise the runtime efficiency we gain from the use of the very fast bitwise operations. Furthermore, an implementation packing the bits into long integers consumes a relatively small amount of memory. The results of an experiment replacing the bit matrices with hash values (Section 4.4) indicates that memory saving efforts should be targeted elsewhere.

Without loss of generality, let us assume that the left branch of a node in the decision tree corresponds to a 'no' answer and the right branch to a 'yes' answer of this node's question. The walk up the decision tree starts

at the root with a bit matrix, all of whose bits are set to 1. A modified copy of the matrix is passed to the *right* branch of the root, marking with zeros phones that do not belong to the question's phone set and are in its context scope[1]. Similarly, the bits that do belong to the question and are in its scope are zeroed out and the modified copy is passed to the *left* branch. The process continues until a leaf is reached.

A modification of this procedure must be made to accommodate a decision tree with compound questions, i.e. questions, which refer to many phone contexts at once. A 'yes' answer to such question would only require resetting the corresponding bits in the different columns of the bit matrix. However, handling a 'no' answer is not entirely as straightforward.

Consider a compound question $(A \wedge B)$, consisting of the simple questions 'A' and 'B', each of which is of the type "Is the phone at position -1 diphthong?". A 'no' answer to this question can be given by a negative answer of either one of the questions, i.e. $\neg(A \wedge B) = \neg A \vee \neg B$. Therefore, when parsing the decision tree we have to propagate two versions of the modified bit matrix – one for each negative answer. The leaf collects all the bit matrices from all the paths to it in a set we call a bit matrix list.

## 2.2 Metastate Enumeration

Our goal is to build a transducer, which maps from sequences of Gaussian mixture models to sequences of corresponding phones. This involves enumerating and storing all possible $k$-long cluster combinations that could form a valid HMM according to the decision tree. Each cluster is a description of all context-dependent phones, in whose expansion it can participate. Therefore, a Gaussian mixture cluster sequence is valid, if every context position of the intersection of the included clusters is non-empty.

A bitwise & operation of two equally sized bit matrices is defined as the bit matrix obtained by a bitwise & of the individual bits at the same positions in the initial bit matrices.

Let $\mathbf{L}' = \{\mathbf{B}'_i \mid i = 1 \dots l\}$ and $\mathbf{L}'' = \{\mathbf{B}''_i \mid i = 1 \dots p\}$ denote two bit matrix lists. We redefine the bitwise & operation for bit matrix lists:

$$\mathbf{L}' \& \mathbf{L}'' = \{\mathbf{B}'_i \& \mathbf{B}''_j | i = 1 \dots l, j = 1 \dots p\}.$$

$k$ clusters $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$ can form a valid Gaussian mixture sequence if the bitwise & product of their bit matrix lists

$$\mathbf{L} = \mathbf{L}_{\mathbf{c}_1} \& \mathbf{L}_{\mathbf{c}_2} \& \dots \& \mathbf{L}_{\mathbf{c}_k}$$

contains at least one bit matrix, each of whose columns contain at least one set bit. This prerequisite reflects the before mentioned requirement on the

[1] Assuming that the question refers to a single context position, the affected bits will occupy a single column.

cluster intersection. A bit matrix, which fulfills this requirement is called *valid.*

Since the number of Gaussian mixture permutations is very large even for only three-state sequences, we can calculate them in multiple passes, first forming the possible two-long cluster sequences, then the possible three-long cluster sequences, and so on until all possible $k$-long cluster sequences are enumerated.

We store the cluster permutations in a *metastate* structure

$$\mathbf{M} = (\mathbf{p}, \mathbf{c}_1, \ldots, \mathbf{c}_k, \mathbf{L}), \text{ where}$$

- $\mathbf{p}$ signifies a context independent subword unit (phone),
- $\mathbf{c}_i$ are the cluster symbols [2] , and
- $\mathbf{L}$ is a valid bit matrix list, obtained from the bitwise & product of the clusters' bit matrix lists.

The phone symbol $\mathbf{p}$ is stored for convenience, but is not needed since it can be inferred from the center column of all bit matrices in $\mathbf{L}$. During the construction of the decision tree, a separate decision tree is grown for each phone; hence the phone identity is known implicitly for each leaf in the decision tree. Checking the center columns of the bit matrices is a good sanity check during the bit matrix and cluster enumeration phases.

By forming all possible leaf permutations and examining the corresponding bit matrix lists we make sure that each context-dependent phone will be covered by one Gaussian mixture model in the final combined $HC$ transducer. What needs to be done next is to set the connections between the enumerated cluster sequences.

## 2.3 Connecting the Metastates

After computing all metastates, we need to interconnect them so that the combined $HC$ transducer maps successive cluster sequences to successive context-independent phones. We determine the metastates able to connect to each other, by doing the bitwise & operation on their bit matrix lists, but shifting the bit matrices in the starting metastate's bit matrix list by one column to the left beforehand. This is necessary, because any phones at context position $i$ in the starting metastate will be seen at context position $i - 1$ from the target metastate, as each following metastate moves us by one phone ahead on the output side of the transducer.

During the metastate connection step, we maintain a list $\mathbf{Q}$, which contains all metastates, which have to be further expanded. A queue $\mathbf{T}$ holds all

[2] During this step, we create an individual node for each cluster symbol $\mathbf{c}_i$. In the pseudocode below, we refer to the states by the names of their corresponding clusters.

previously visited metastates and is searched whenever new metastates are created, so to prevent adding redundancy to the transducer and to ensure that the algorithm will eventually finish.

Assuming that all metastates from the previous enumeration step are stored in a set **S**, the algorithm in Listing 1 interconnects the metastates in the *HC* transducer. Since our speech recognition system only allows utter-

**Listing 1** Metastate connection.

```
00  def connectMetastates(SIL, S):
01    push SIL on Q
02    add SIL to T
03    connect INITIAL to SIL
04    while ||Q|| > 0:
05      pop q from Q
06      if q.p == SIL:
07        connect q to FINAL
08      foreach s ∈ S:
09        L ← (q.L ≫) & s.L
10        if ||L|| > 0:
11          t ← (s.p, s.s1, s.s2, s.s3, L)
12          if t ∉ T:
13            add t to T
14            push t on Q
15          e ← (q.s3, t.s1, t.s1, t.p)
16          add e to E
17    return (T, E)
```

ances starting and ending with silence, we begin by pushing the metastate for 'SIL' (silence) on **Q**. Later we only allow 'SIL' metastates to connect to the end node of the transducer (Lines 06-07). This can of course be modified to allow any or all phones to start or end an utterance. The loop at line 04 pops metastates from the list **Q** and tries to connect them with metastates from the enumeration step in list **S** (Line 09). If a new metastate **t** can be created (Line 10), we check if a metastate with the same cluster symbols, output phone and bit matrix list has been created before(Line 12), and if not, we create new transducer states for its cluster sequence and add it to the **Q** list for further expansion. We also push it on **T**, so that if later another metastate with the same cluster sequence and bit matrix list is created, its cluster symbols will correspond to **t**'s nodes. If **t** is found in **T**, it is not pushed on **Q** and instead the existing metastate is used as **t**. Finally, an edge is added in Lines 15-16 from the end node of **q** to the begin node of **t**, using the cluster symbol $\mathbf{t}.s_1$ as an input symbol and the metastate's phone $\mathbf{t}.p$ as an output symbol.

# 3 Differences with Schuster and Hori

We base our development on the work of Schuster and Hori [2]. However, the metastate connection algorithm described here addresses a mistake in the fourth step "Generate between-phone connections" of their procedure. After a thorough examination, it may become apparent that this step leaves connections between three-state sequences, which should not in fact exist. It's important to mention that Schuster and Hori's implementation is *correct* for context lengths of three. The discrepancy appears when using larger span contexts.

A one-pass multiplication of the binary shifted bitmaps with all non-shifted phone bitmaps might allow invalid connections in the resulting $HC$ transducer. The reason for this lies in the fact that the context history reduces the number of future connections a metastate can have, but is not considered when using a one-pass multiplication. Instead of the newly created bit matrix list **L** (line 9), the old bit matrix list **s.L** is assigned to the new metastate, thus discarding the information that the **q** metastate has been connected to the **s** metastate. Therefore the effect, which this connection has on the future connections of **t** is neglected.

This property does not present itself in the case of triphones, where a one-pass multiplication is enough. Triphonic bitmap lists restrict only the immediate connections of their metastates. On the contrary, pentaphone and larger span context metastates can restrict the prospective connections of the metastates they connect with, i.e. the $n^{\text{th}}$ context of one metastate affects the $n-1^{\text{st}}$ context of next one, which in turn restricts its $n-2^{\text{nd}}$ context and so on. The same is valid for the negative (left) context positions.

As an example of this, consider the metastates corresponding to the phones **A**, **B**, and **C** and their bit matrices in Figure 2. After a one-pass

| | **A** | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| **A** | 1 | 1 | 1 | 1 | 1 |
| **B** | 1 | 1 | 0 | 1 | 0 |
| **C** | 0 | 1 | 0 | 0 | 0 |

| | **B** | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| **A** | 0 | 1 | 0 | 1 | 1 |
| **B** | 1 | 0 | 1 | 0 | 0 |
| **C** | 0 | 0 | 0 | 1 | 1 |

| | **C** | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| **A** | 1 | 0 | 0 | 1 | 1 |
| **B** | 1 | 1 | 0 | 0 | 1 |
| **C** | 0 | 0 | 1 | 0 | 1 |

Figure 2: Pentaphone metastate bit matrices

multiplication the connection pairs **A** → **B** and **B** → **C** can be formed, as the bit matrices in Fig. 3 show. This means that the transition **A** → **B** → **C** will appear in the combined $HC$ transducer and will be *incorrect*, since the influence the metastate **A** has on the connection **B** → **C** has been disregarded [3]. Figure 4 shows the invalid bit matrix produced with the

[3] Namely, **A** doesn't allow **C** to stay at +2 position

correct metastate connection algorithm.

| | $(\mathbf{A} \gg)\&\mathbf{B}$ | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| **A** | 0 | 1 | 0 | 1 | 1 |
| **B** | 1 | 0 | 1 | 0 | 0 |
| **C** | 0 | 0 | 0 | 0 | 1 |

| | $(\mathbf{B} \gg)\&\mathbf{C}$ | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| **A** | 1 | 0 | 0 | 1 | 1 |
| **B** | 0 | 1 | 0 | 0 | 1 |
| **C** | 0 | 0 | 1 | 0 | 1 |

Figure 3: The connection pairs can be formed.

| $(((\mathbf{A} \gg)\&\mathbf{B}) \gg)\&\mathbf{C}$ | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| **A** | 1 | 0 | 0 | 1 | 1 |
| **B** | 0 | 1 | 0 | 0 | 1 |
| **C** | 0 | 0 | 0 | 0 | 1 |

Figure 4: The invalid bit matrix for the transition $\mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathbf{C}$.

On the other hand, if we consider the same metastates, but for triphones, we see that the one-pass multiplication and the correct algorithm produce equal *valid* bit matrices, i.e. $(\mathbf{B} \gg)\&\mathbf{C} = (((\mathbf{A} \gg)\&\mathbf{B}) \gg)\&\mathbf{C}$ (Figure 6).

| | **A** | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| **A** | 1 | 1 | 1 |
| **B** | 1 | 0 | 1 |
| **C** | 1 | 0 | 0 |

| | **B** | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| **A** | 1 | 0 | 1 |
| **B** | 0 | 1 | 0 |
| **C** | 0 | 0 | 1 |

| | **C** | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| **A** | 0 | 0 | 1 |
| **B** | 1 | 0 | 0 |
| **C** | 0 | 1 | 0 |

Figure 5: Triphone metastates for the phones $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$.

| | $(\mathbf{B} \gg)\&\mathbf{C}$ | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| **A** | 0 | 0 | 1 |
| **B** | 1 | 0 | 0 |
| **C** | 0 | 1 | 0 |

| $(((\mathbf{A} \gg)\&\mathbf{B}) \gg)\&\mathbf{C}$ | | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| **A** | 0 | 0 | 1 |
| **B** | 1 | 0 | 0 |
| **C** | 0 | 1 | 0 |

Figure 6: Valid bit matrices produced with the one-pass multiplication and the correct algorithm.

# 4 Efficient Construction of the Combined *HC* Transducer

The algorithm in Listing 1 for connecting the metastates is correct, but slow and memory demanding for any distribution tree of a realistic size. The reason for this is the high number of metastates and the transducer size itself during construction. We address these problems in the following sections.

## 4.1 Metastate Connection Speed Up

A major acceleration of the metastate connection algorithm can be achieved if the search in line 08 is restricted to only a subset $\mathbf{C}(\mathbf{M}) \subset \mathbf{S}$, which contains all metastates that can be linked to the given metastate $\mathbf{M}$. We can determine $\mathbf{C}(\mathbf{M})$ by looking at the right column next to the center column of each bit matrix in the metastate's bit matrix list. This column represents the possible phones that the decision tree places at the $+1$ context position, i.e. the immediate neighbouring phones, and so the search can be restricted to only these phones' metastates.

During this precalculation step, we can also simultaneously compute a *bit mask* $\mathbf{B}$ for each metastate $\mathbf{M}$. The purpose of the bit mask is to accumulate the context information of all metastates, which can be paired with the current one:

$$\mathbf{B_M} = \bigvee_{s \in \mathbf{C}(\mathbf{M})} \bigvee_{i=1}^{|s.\mathbf{L}|} \mathbf{B}_i, \mathbf{B}_i \in s.\mathbf{L}^4$$

A bitwise & operation of a right shifted version of bit mask and $\mathbf{M.L}$ represents an intersection of the context information of $\mathbf{M}$ with the cumulative contextual information of all metastates in $\mathbf{C}(\mathbf{M})$. This intersection happens anyway during regular metastate connection, i.e. for any three bit matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$

$$\mathbf{A}\&(\mathbf{B} \vee \mathbf{C}) = (\mathbf{A}\&\mathbf{B}) \vee (\mathbf{A}\&\mathbf{C}).$$

The effect of applying this operation in advance is a reduction of the metastate's bit matrix list size by disregarding invalid bit matrices. It also prevents the forming of new metastates in the list $\mathbf{T}$, which differ only in insignificant set bits in their bit matrix lists. Such metastates will be unified in the subsequent determinization and minimization applied to the combined *HC* transducer.

[4] $\bigvee$ is to be understood as a bitwise 'or' operation of the bit matrices

## 4.2 On-demand Expansion of $HC$

Once the metastates are enumerated, the connection algorithm can be applied *locally* for a given node. The information to connect a given metastate in the transducer is contained in its bit matrix list, and therefore we can dispense with storing the contents of the entire transducer in main memory, while only keeping the identity of the already expanded metastates and the **S** list. An additional advantage this method gives is that another local finite-state transducer algorithm, such as weighted composition or weighted determinization, can be applied to the transducer simultaneously on-the-fly.

During an expansion of the transducer, an access frequency count is assigned to each node and a cleanup memory function is regularly called to free memory occupied by less frequently used nodes's adjacency lists. Should a node's edges list be needed again, it can be calculated using the bitmatrix list of the metastate, to which this node belongs.

The adjacency list expansion algorithm in Listing 2 implements an incremental transducer construction. The function edges(**q**) is called whenever the adgacency list of the end node of a metastate **q** is needed by a graph traversal technique, such as breadth first or depth first search.

**Listing 2** Adjacency list expansion.

```
00  def edges(q):
01    if q.E ≠ ∅:
02      return q.E
03    if q.p == SIL:
04      connect q to FINAL
05    foreach s ∈ C(s):
06      L ← (q.L ≫) & s.L & B_S
07      if ||L|| > 0:
08        t ← (s.p, s.s1, s.s2, s.s3, L)
09        if t ∉ T:
10          add t to T
11        e ← (q.s3, t.s1, t.s1.g, t.p)
12        add e to q.E
13    return q.E
```

The adjacency list of a node does not have to be regenerated, if it already exists (Line 01), otherwise a modified version of the metastate connection algorithm is used to regenerate it (Line 05 - 15). There is no need to keep the **Q** list anymore, because it is indirectly replaced by the internal structure of the graph expansion algorithm keeping the yet to be expanded nodes.

## 4.3 Directly Determinized Construction

An application of the local property will become apparent after examining Table 3. The size of the determinized combined $HC$ transducer is significantly smaller than the initial statically expanded transducer. This is due to the fact that many metastates share cluster subsequences, but unique states and arcs are created for each one. Therefore we may want to try to construct the determinized transducer directly, instead of first expanding the large redundant version of it, storing it to disk, and then determinizing it.

Indeed, as the determinization algorithm is a local one [9], we can immediately determinize the transducer while incrementally building it. This gives a significant superiority over the static expansion in terms of memory usage, as seen from Table 6. The memory footprint decreases drastically, as opposed to a mild construction runtime increase.

## 4.4 Eliminating the Bit Matrix Lists

The metastate connection algorithm does not need the explicit contents of the bit matrices of the metastates in the **T** list, since it only compares if two different lists are the same. This justifies a replacement of the bit matrix list with a unique hash code as follows, especially as the **T** list gets extremely large during construction (see Table 5 ).

We calculate a sequence of *run lengths* $\{r_n\}$, each represeting the number of zeroes between two consequtive ones in a bit matrix. The following checksum can be used to replace a bit matrix list:

$$h(k) = \sum_n k^n r_n,$$

where $k$ is a prime number. Replacing the bit matrices this way introduces a possibility of collisions, i.e. a single checksum corresponding to two different bit matrix lists. However, we can make the probability of this event very small if we store and compare different hash keys, each one corresponding to a different prime number.

A modified version of Listing 2 can be defined when using checksums to replace the bit matrix lists:

**Listing 3** Adjacency list expansion with removal of the bit matrix lists.

```
00  def edges(q):
01    if q.E ≠ ∅:
02      return q.E
03    if q.p == SIL:
04      connect q to FINAL
05    foreach s ∈ S:
06      L ← (q.L ≫) & s.L & B_s
07      if ||L|| > 0:
08        t ← (s.p, s.s_1, s.s_2, s.s_3, L)
09        if t ∉ T:
10          add t to T
11        elif t.L == ∅ and q.E == ∅:
12          t.L = L
13        e ← (q.s_3, t.s_1, t.s_1.g, t.p)
14        add e to q.E
15    q.h ← hash(q.L)
17    q.L ← ∅
18    return q.E
```

As we are now deleting the bit matrix lists, we must be careful when expanding a node that it either contains an expanded adjacency list, or the bit matrix list of its parent metastate is not deleted. Lines 11 and 12 make sure that whenever a metastate is found in the **T** list, a bit matrix list for it is preserved for later expansion of this metastate's nodes. At lines 15 and 16 we replace the bit matrix list with the calculated hash code and delete the bit matrix list. This does not geopardize the further expansion of q's nodes, because **q** can later be only accessed through the **T** list and thus undergoes the described bit matrix restoration mechanism.

## 5 Experimental Results

In this section we present results concerning the application of the combined *HC* transducer in speech recognition tasks, *HC* construction statistics and empirical verification of the metastate connection algorithm.

The HC constructions and recognition network compilations were done with the *Enigma* finite-state transducer library, maintained by the the first advisor and the author at the University of Karlsruhe, Karlsruhe, Germany. The speech recognition experiments were conducted with the *Millenium* speech recognition system, work of the first advisor and his scientific assistants.

## 5.1 Comparison with $H \circ C$

We performed experiments regarding the correctness of the $HC$ construction on our whole training set consisting of 791 speakers and totaling 176,355 utterances. A word string corresponding to a single utterance transcription is converted to a transducer $W$ then composed with the dictionary transducer. The result is determinized and then composed with a combined $HC$ transducer, which is in turn determinized and minimized to obtain a final transducer $\min(\det(HC \circ \det(L \circ W)))$. Independently, $\det(L \circ W)$ is expanded by composition with a context-dependency and a hidden Markov model transducer, and then determinized and minimized. We found no differences comparing the input sides of $\min(\det(HC \circ \det(L \circ W)))$ and $\min(\det(H \circ C \circ \det(L \circ W)))$, which means that both the conventional composition and our algorithm provided the same Gaussian mixture model expansion for each utterance in the training set.

## 5.2 Recognition Experiments

We used a fully-continuous acoustic model with 3,500 clusters for the recognition experiments to test both Schuster and Hori's and the correct $HC$ transducer.[5] The signal processing front-end of the speech recognizer extracts 13 cepstral features and then concatenates 15 consecutive frames together. It then reduces the dimensionality of the final observations to 42 by means of *Linear Discriminative Analysis* [14]. The observations were furthered processed by a global STC transform [15] and global mean substraction.

The training data totalled 100 hours, comprising of ICSI, NIST and CMU *meeting corpora*, as well as the *Transenglish Database* corpus. The training used both *vocal tract length normalization* (VTLN) [16] and *constrained maximum likelihood linear regression* (CMLLR) [17]. Maximum Likelihood speaker-adapted training (MMI-SAT) was conducted using the approximations described in [18] to conserve disk space.

For the purpose of MMI-SATraining, we decoded each utterance in the training set. A *word-trace decoder* [19] was used and its output projected on the output side to discard any information other than the word sequences as suggested by [20]. The correct transcription was then inserted into the lattice and the result was epsilon-removed, determinized and minimized. Through a following composition with a lexicon, determinization and composition with the minimized correct $HC$ transducer, we obtained a lattice mapping sequences of cluster symbols to the words in the recognized utterance as in [21]. This transducer was determinized, its weights pushed towards the initial state as discussed in [22] and minimized. The generation

[5] All acoustic models used in the experiments, described in this work, were trained by the first advisor.

was relatively efficient, with all steps completed in approximately 5× real time on a 3GHz Intel workstation.

To create the denominator lattices explained in [23], we converted each utterance to a simple word-string FST, composed it with a *unigram* language model, then with a lexicon and finally with the correct HC. This network was eventually optimized through determinization, weight-pushing and minimization. A *state-trace* decoder was now used to produce the exact time aligments between the feature vectors and the individual HMM states. Based on these Viterbi alignments, a new lattice was generated, with the time boundaries and log-likelihood scores for each HMM state. This information was then used to derive the posterior probabilities needed for discriminative training. As each lattice transducer was optimized by determinization, weight pushing and minimization, the final constrained decoding and MMI statistics accumulation for each iteration ran in approximately real time. In addition to estimating new means, covariances and mixture weights, we also performed *maximum mutual information semi-tied covariance*(MMI-STC) estimation [24].

The test set used for the experiments in Table 1 was the *lecture meeting* portion of the NIST RT-05s evaluation set. It consists of 22,258 words, a total of 3,5 hours of data. The lecture speakers spoke English, but often with German or other accents. The vocabulary contains largely technical terms, mostly about topics related to automatic speech recognition. This data was collected as part of the European Union integrated project CHIL, *Computers in the Human Interaction Loop* at the University of Karlsruhe, Karlsruhe, Germany.

Table 1 shows the word error rates from a set of speech recognition experiments. We built two recognition networks with both Schuster and Hori's and the correct HC, using a pentaphone decision tree with 3,500 clusters. The transducer sizes can be seen Table 3 in the next section.

We composed a bigram language model containing 113,705 bigrams with a dictionary of 50,886 words and variants. We then determinized and minimized the composition and then composed the resulting transducer with both $HC$ transducers to produce two recognition networks, which we then determinized, weight-pushed and minimized as advised by [3]. The decoding passes in Table 1 were run with the same beam-widths, which were so chosen, that the recognition would be performed in *approximately* real time.

The recognition passes with Schuster and Hori's $HC$ transducer paradoxically yields consistently lower word error rates than the one built with the correct $HC$ transducer. An explanation for this behaviour was found when examining the recognition statistics. When decoding with Schuster and Hori's network, significantly more active hypotheses were retained for each time step as compared to decoding with the correct recognition network. To further investigate, we conducted recognition experiments with the adapted model from the final MMI-SAT-STC pass and plotted word

| | % Word Error Rate | |
|---|---|---|
| | S&H | Correct |
| Unadapted Pass | 46.6 | 48.4 |
| VTLN, MLLR, FSA | 39.9 | 39.7 |
| ML-SAT | 36.9 | 36.7 |
| MMI-SAT | 35.1 | 35.3 |
| MMI-SAT-STC | 34.2 | 34.7 |

Table 1: WER from a set of experiments with HC transducers, generated with both Schuster and Hori's algorithm and the correct one

error rates obtained with both networks.

Better recognition rates can be obtained, when using a *semi-continuous acoustic model* (SCAM) [25]. In contrast to a fully-continuous acoustic model (FCAM), we can train a set of Gaussians, or *codebooks*, which can be shared between several models. Thus, only the mixture weights for a single output distribution need to be estimated, making a better usage of the available observations per state.

We undertook a final set of experiments to compare the performance of a SCAM to that of a FCAM under identical training condition and testing conditions. For these experiments, we began with a pentaphone FCAM with 4,000 codebooks. We first performed conventional Viterbi training, then ML-SAT, and finally MMI-SAT as described above. The SCAM systems were obtained by performing additional divisive clustering [5] beginning from the decision tree used for the FCAM systems to produce a final decision tree with 16,000 Gaussian mixture models sharing the same 4,000 codebooks contained in the FCAM system.

| | % Word Error Rate | |
|---|---|---|
| | FCAM | SCAM |
| Unadapted Conventional | 45.2 | |
| Adapted Conventional | 37.0 | |
| Adapted ML-SAT | 34.1 | 32.9 |
| Adapted ML-SAT, 4-gram LM | | 31.9 |

Table 2: WER obtained with FCAM and SCAM systems

Table 2 shows the word error rates obtained on the same NIST-RT05 test as used for the comparison between the two recognition networks above. For each decoding pass, speaker adaptation parameters were estimated using word lattices from the previous pass. As we were unable to compile a full recognition network with the 16,000 codebook $HC$ transducer, we performed

rescoring experiments over the adapted FCAM ML-SAT system.

In each pass, we constructed constrained recognition spaces by projecting the lattices output by the word-trace decoder on the output side, composing with the appropriate language model (bigram for the initial experiments, fourgram for the final pass), then with the lexicon and finally with either the 4,000 codebook or 16,000 codebook *HC* transducer. We applied the optimization techniques inbetween the compositions and performed a final determinization, weight-pushing and minimization. The constructed constrained recognition networks were used for both speech recognition and MLLR parameter estimations [26]. Table 2 shows a significant reduction of WER, which the SCAM system brings over the FCAM system. Rescoring the word lattices with the SCAM system brought down the WER from 34.1% to 32.9%, which was further decreased to 31.9% by the application of the fourgram language model.

The construction statistics for the *HC* transdcers are given in the following section.

## 5.3 *HC* Construction Statistics

We constructed a combined *HC* transducer with Schuster and Hori's and with the correct connection algorithm. Table 3 shows the sizes of the expanded *HC* transducer and it's size after determinization and minimization. We used a pentaphone decision tree with 3,500 leaves.

| | Correct | | Schuster & Hori | |
|---|---|---|---|---|
| | Nodes | Edges | Nodes | Edges |
| HC | 975,838 | 63,178,405 | 188,961 | 10,459,979 |
| det(HC) | 406,173 | 8,199,840 | 91,842 | 1,621,785 |
| min(det(HC)) | 81,499 | 968,078 | 43,263 | 451,254 |

Table 3: Pentaphone combined *HC* transducer sizes

As is evident from the table, the correct *HC* is twice as large as Schuster and Hori's after determinization and minimization. This reiterates the conclusion that many metastates and the connections between them are left out by the one-pass multiplication algorithm.

For the recognition experiments we constructed two *HC* transducers, one with a pentaphone decision tree containing 4,000 leaves, and the other containing 16,000 leaves as described in the previous section. Table 4 summarizes the sizes of the intermediate $det(HC)$ and $min(det(HC))$ transducers. We used the incremental expansion and immediate determinization to build both transducers, whereby the incremental expansion was essential for the larger one.

| | 4,000 leaves | | 16,000 leaves | |
|---|---|---|---|---|
| | Nodes | Edges | Nodes | Edges |
| det(HC) | 609,433 | 12,362,584 | 3,535,569 | 20,745,716 |
| min(det(HC)) | 154,665 | 2,069,737 | 1,662,704 | 11,184,683 |

Table 4: Pentaphone combined $HC$ transducer sizes for two decision trees

The construction times, memory usage, number of metastates during construction, and the total number of enumerated cluster sequences are given in Table 5. We see that a 4 times increase of the number of leaves

| | 4,000 leaves | 16,000 leaves |
|---|---|---|
| Memory usage | N/A | 11.7 GB |
| Cluster sequences | 60,683 | 379,156 |
| Metastates | 356,702 | 1,746,894 |
| Construction time | 1hr. 15 mins. | 82hrs. 26 mins. |

Table 5: Construction statistics

does not correspond linearly to a 4 times increase in memory usage, nor in construction time. This gives rise to the question of how the runtime and memory footprint of the metastate connection algorithm depend on the number leaves in the decision tree. The number of metastates during construction, as well as the amount of cluster sequences, is influenced by the type of questions in the decision tree and by the phone classes. The type of questions also conditions the tendency of the metastates to connect to other metastates and thus form new ones, which in turn determines the construction time.

Finally, we give a comparison of the static expansion and the incremental on-demand expansion.

| | Memory Usage | Runtime |
|---|---|---|
| Static | 7,70 GB | 50 min. |
| Hashing | 7,69 GB | 103 min. |
| Dynamic | 1,42 GB | 56 min. |

Table 6: Algorithm comparison between static expansion, static expansion with bit-matrix hashing and dynamic expansion

It is clear from Table 6 that a dynamic expansion is to be preferred when compiling large distribution trees, because the memory usage is decreased by a factor of 7. This decrease can be expected, because the memory oc-

cupied by infrequently used portions of the network is now being recovered. However, no significant increase in runtime is observed, which can again be explained by the decision tree structure - a relatively small number of metastates, which tend to connect to other metastates, will be kept in memory and further expanded. The larger number of metastates are inactive and their nodes' adjacency lists will be deleted from memory.

# 6 Algorithm Proof

In this section we provide a proof of the correctness of the $HC$ construction procedure. We derive an explicit definition of the edges list of the transducer $H \circ C$ and compare it to that of $HC$, thereafter showing the equivalence of both.

In the following discussion, we use these notations:

- $P$ is the set of phones;
- $n$ is the context-dependency depth;
- $k$ is the number of states in an HMM model;
- $\Gamma = P^{2n+1} \cup \{\epsilon\}$ is the alphabet of context-dependent phones;
- $\Sigma = \{d_\gamma^i | \gamma \in \Gamma, i \in 1 \dots k\}$ is the alphabet of cluster symbols (leaves), obtained from a decision tree clustering procedure, where $d_\gamma^i$ is the i-th cluster for the context-dependent phone $\gamma$.

As the metastates are the building blocks of $HC$, the following lemma serves to make sure that each context-dependent polyphone is covered by exactly one metastate in $\mathbf{S}$ as dictated by the decision trees.

**Lemma 1.** *Assume that the $k$ decision trees expand each polyphone to a unique cluster sequence.* [6] *There is a surjective mapping $f : \Gamma \to \mathbf{S}$.*

*Proof.* Let $\gamma = p_1 \dots p_{2n+1} \in \Gamma$. As explained in Section 2.1, a cluster in the $i$-th decision tree is a set of phones for each context position:

$$d_\gamma^i = \{\mathcal{A}_j^i | \mathcal{A}_j^i \in 2^P, j = 1 \dots 2n+1\}, i = 1 \dots k.$$

Observe that the decision tree growing procedure partitions the initial pool of context-dependent phones in a set of nonintersecting clusters. This justifies the notation $d_\gamma^i$, because there will be only one cluster corresponding to a polyphone $\gamma$. Therefore it holds that

$$p_j \in \bigcap_{\substack{i=1 \\ \mathcal{A}_j^i \in d_\gamma^i}}^{k} \mathcal{A}_j^i, \ j = 1 \dots 2n+1.$$

[6] The lemma assumption is valid anyway, as that is one of the main advantages of a decision tree, see [5].

Since the metastate enumeration step in Section 2.2 performs all possible intersections of the above form, it follows that $\gamma$ corresponds to at least one $\mathbf{s} \in \mathbf{S}$. Furthermore, $\mathbf{s}$ is unique, since each $d^i_\gamma$ in the above intersection is unique.

The surjectivity of $f$ can be easily seen, as there are no metastates, which do not correspond to any polyphones. During the enumeration, we require the validity of each bit matrix, thus imposing the condition

$$\forall \gamma \in \Gamma \bigcap_{\substack{i=1 \\ \mathcal{A}^i_j \in d^i_\gamma}}^{k} \mathcal{A}^i_j \neq \emptyset, \; j = 1 \ldots 2n+1.$$

However, because of the clustering, $f$ is not an injection. □

## 6.1 The $H$ Transducer

The role of the hidden Markov Model transducer $H$ in the composition chain descirbed in Section 1.2 is to assign to each hidden Markov model (HMM) the symbol of the context-dependent phone it models. As Hidden Markov models are represented naturally with directed state graphs, the HMM transducer can be easilly constructed by placing all such state graphs in a loop, connecting them with a common initial and final state.

For each HMM in the $H$, there is a transition from the common initial/end state of the HMM transducer to the begin node of each of the HMMs, which is labeled with the cluster symbol corresponding to the begin node of the HMM. The output of this transition is the context-dependent phone symbol, which this HMM models. There is also a transition from the end node(s) of the HMM back to $H$s common initial/end state.

A hidden Markov model transducer $H$ with a left-to-right $k$-state HMM topology is a 6-tuple $(\Sigma \cup \epsilon, \Gamma, Q_H, \eta_H, F_H, E_H)$, where

- $\Sigma \cup \{\epsilon\}$ is the input alphabet of cluster symbols;
- $\Gamma \cup \{\epsilon\}$ is the output alphabet of context-dependent phones
- $Q_H = \Sigma \cup \eta_H$ is the set of states,
- $\eta_H \in Q_H$ is the single initial state,
- $F_H = \{\eta_H\}$ is the single final state,
- $E_H = Q_H \times \Sigma \times \Gamma \times Q_H$ is the set of edges.

The edges set of $H$ can be explicitly defined as follows:

$$\begin{aligned} Q_H &= \{(\eta_H, d_\gamma^1, \gamma_i, d_\gamma^1)\} \cup \\ &\quad \{(d_\gamma^i, d_\gamma^{i+1}, \epsilon, d_\gamma^{i+1}) | i = 1 \ldots k-2\} \cup \\ &\quad \{(d_\gamma^{k-1}, d_\gamma^{k-1}, \epsilon, \eta_H)\}, \end{aligned}$$

where $d_\gamma^i$, $i = 1..k$ is the cluster expansion of the context-dependent phones $\gamma_i$.

The so constructed hidden Markov model transducer is simple but non-sequential, because each polyphone $\gamma_i$ is assigned its own cluster expansion. Another important property is that it doesn't admit null transitions, other than the implicit epsilon self-loops at each state.

## 6.2 The $C$ Transducer Structure

The purpose of the context-dependency transducer is to model the mapping of sequences of context-dependent phones to the correct sequences of context-independent phones. It can be formally described as a 6-tuple $(\Gamma, P, Q_C, \eta_C, F_C, E_C)$, where

- $\Gamma$ is the input alphabet of context dependent phones,
- $P$ is the output alphabet of context-independent phones,
- $Q_C = P^{2n} \cup \{\epsilon\}$ is the set of states,
- $\eta_C \in Q$ is the initial state,
- $F_C \subset Q_C$ is the set of final states,
- $E_C = Q_C \times \Gamma \times P \times Q_C$ is the edges set.

Let $\gamma = p_1 \ldots p_{2n+1} = uwv$ be a context-dependent polyphone, where $u, v \in P, w \in P^{2n-1}$. For a state $q = uw$

$$(uw, \gamma, p_n, wv) \in E_C.$$

The start state of an arc, labelled by a context-dependent phone is the state named after its first $2n$ phones, and the end state is named after its last $2n$ phones. We should note that, similarly to the $H$ transducer, the context dependency transducer is epsilon-free.

## 6.3 Proof of Correctness

Let $A = (\Sigma, \Delta, Q_A, \eta_A, F_A, E_A)$ and $B = (\Delta, \Gamma, Q_B, \eta_B, F_B, E_B)$ be two weighted finite-state transducers. As described in [13], their composition realizes the mapping from the input alphabet of $A$ to the output alphabet of $B$. This mapping simulates the application of transducer $B$ on the output of transducer $A$, thus requiring that the intersection of the set of input strings of $B$ and the set of output strings of $A$ is non-empty.

The composition of $A$ and $B$ is the finite-state transducer, defined as

$$A \circ B = (\Sigma, \Gamma, Q_A \times Q_B, (\eta_A, \eta_B), F_A \times F_B, E_{A \circ B}),$$

whose edges set is

$$\begin{aligned} E_{A \circ B} \quad = \quad & \{((q_A, q_B), \sigma, \gamma, (q'_A, q'_B))| \\ & (q_A, \sigma, \delta, q'_A) \in E_A, \\ & (q_B, \delta, \gamma, q'_B) \in E_B, \delta \in \Delta\} \end{aligned}$$

We are interested in an explicit definition of the set of edges of $H \circ C$. Using the composition formula from above, such definition can be given as follows.

Let $E_{H \circ C}$ be the edges set of the composition of $H$ and $C$, and $d^i_\gamma, i = 1 \ldots k$ be the cluster expansion of $\gamma = p_1 \ldots p_{2n+1} = uwv$ as in Lemma 1.

$$\begin{aligned} E_{H \circ C} \quad = \quad & \{((q_H, q_C), \sigma, p, (q'_H, q'_C))| && (1) \\ & (q_H, \sigma, \gamma, q'_H) \in E_H, && (2) \\ & (q_C, \gamma, p, q'_C) \in E_C, \sigma \in \Sigma, \gamma \in \Gamma\, p \in P\} && (3) \\ = \quad & \{(\eta_H, uw), d^1_\gamma, p_n, (d^1_\gamma, wv))\} \cup && (4) \\ & \{((d^i_\gamma, uw), d^{i+1}_\gamma, \epsilon, (d^{i+1}_\gamma, uw))|i = 1 \ldots k-2\} \cup && (5) \\ & \{((d^{k-1}_\gamma, uw), d^{k-1}_\gamma, \epsilon, (\eta_H, uw)\} && (6) \end{aligned}$$

The inverse of $H \circ C$, $(H \circ C)^{-1}$ is defined simply by interchanging the input and output labels on the arcs of $H \circ C$:

$$\begin{aligned} E_{(H \circ C)^{-1}} \quad = \quad & \{(\eta_H, uw), p_n, d^1_\gamma, (d^1_\gamma, wv))\} \cup && (7) \\ & \{((d^i_\gamma, uw), \epsilon, d^{i+1}_\gamma, (d^{i+1}_\gamma, uw))|i = 1 \ldots k-2\} \cup && (8) \\ & \{((d^{k-1}_\gamma, uw), \epsilon, d^{k-1}_\gamma, (\eta_H, uw)\} && (9) \end{aligned}$$

For simplicity, the proof of the following lemma and Theorem 1 considers the inverses of $H \circ C$ and $HC$. The lemma provides a neccessary and sufficient condition for the acceptance of a phone string by the inverse transducer $(H \circ C)^{-1}$.

**Lemma 2.** *Let $s = p_1 ... p_l$ be a phone string of length $l \geq 2n+1$. Also, let $\gamma_1, \gamma_2 \ldots \gamma_m$[7] be the consecutive polyphones contained in $s$. $s$ is accepted by $(H \circ C)^{-1}$ iff for $j = 1 \ldots 2n+1$*

$$p_{m+j-1} \in \bigcap_{q=j}^{2n+1} \Big( \bigcap_{\substack{i=1 \\ \mathcal{A}_q^i \in d^i_{\gamma_r}}}^{k} \mathcal{A}_q^i \Big), \; r = m - q + j \tag{10}$$

*Proof.* We prove the lemma by induction on the string length $l$. For $l = 2n+1$, $m = 1$ and

$$p_j \in \bigcap_{q=j}^{2n+1} \Big( \bigcap_{\substack{i=1 \\ \mathcal{A}_q^i \in d^i_{\gamma_r}}}^{k} \mathcal{A}_q^i \Big), \; r = 1 - q + j \tag{11}$$

Since $r$ is the poylphone index, $r > 0$. But $1 - q + j > 0 \Leftrightarrow q < j + 1$. The starting value of $q$ is equal to $j$, hence, for each context position $j$ this inequality is true for only one value of $q$, namely $q = j$. Thus, (11) reduces to

$$p_j \in \bigcap_{\substack{i=1 \\ \mathcal{A}_j^i \in d^i_{\gamma_1}}}^{k} \mathcal{A}_j^i, \text{ for } j = 1 \ldots 2n+1, \tag{12}$$

which is implicitly true, since $d^i_{\gamma_1}$ are the symbols in the cluster expansion of $\gamma_1$ (see the proof of Lemma 1).

Let $(H \circ C)^{-1}$ accept a polyphone of length $l \geq 2n+1$ and

$$p_{m+j-1} \in \bigcap_{q=j}^{2n+1} \Big( \bigcap_{\substack{i=1 \\ \mathcal{A}_q^i \in d^i_{\gamma_r}}}^{k} \mathcal{A}_q^i \Big), \; r = m - q + j, \text{ for } j = 1 \ldots 2n+1 \tag{13}$$

Trivially, this implies

$$p_{m+j-1} \in \bigcap_{q=j}^{2n+1} \Big( \bigcap_{\substack{i=1 \\ \mathcal{A}_q^i \in d^i_{\gamma_r}}}^{k} \mathcal{A}_q^i \Big), \; r = m - q + j, \text{ for } j = 2 \ldots 2n+1,$$

or

$$p_{m+j} \in \bigcap_{q=j+1}^{2n+1} \Big( \bigcap_{\substack{i=1 \\ \mathcal{A}_q^i \in d^i_{\gamma_r}}}^{k} \mathcal{A}_q^i \Big), \; r = m - q + j + 1, \text{ for } j = 1 \ldots 2n \tag{14}$$

[7] $m = l - 2n$

Now consider the phone string $s = p_1 ... p_l p_l + 1$ of length $l+1$. Equations (7), (8) and (9) show that $(H \circ C)^{-1}$ assigns a cluster symbol sequence to this phone string iff

$$p_{m+j} \in \bigcap_{\substack{i=1 \\ \mathcal{A}_j^i \in d^i_{\gamma_{m+1}}}}^{k} \mathcal{A}_j^i, \; j = 1 \ldots 2n+1, \tag{15}$$

where $\gamma_{m+1} = p_m \ldots p_{l+1}$. From (14) and (15) follows that

$$p_{m+j} \in \bigcap_{q=j+1}^{2n+1} \Big( \bigcap_{\substack{i=1 \\ \mathcal{A}_q^i \in d^i_{\gamma_r}}}^{k} \mathcal{A}_q^i \Big) \bigcap \Big( \bigcap_{\substack{i=1 \\ \mathcal{A}_j^i \in d^i_{\gamma_{m+1}}}}^{k} \mathcal{A}_j^i \Big), r = m+1-q+j, \text{ for } j = 1 \ldots 2n \tag{16}$$

which is equivalent to

$$p_{m+j} \in \bigcap_{q=j}^{2n+1} \Big( \bigcap_{\substack{i=1 \\ \mathcal{A}_q^i \in d^i_{\gamma_r}}}^{k} \mathcal{A}_q^i \Big), r = m+1-q+j, \text{ for } j = 1 \ldots 2n+1. \tag{17}$$

□

**Theorem 1.** *The string-to-string transducer HC implements the composition of H and C.*

*Proof.* We use induction on the phone string $l$ and follow closely the proof of Lemma 2.

Let $l = 1$. Since each utterance must start and end with silence, a phone string of length 1 consists of the silence phone only. The $HC$ construction pushes the silence metastate on the expansion queue $\mathbf{Q}$ and connects it to the start node of the transducer (Lines 02 - 03). Only silence phones are allowed to be connected to the final node as assured by the condition at Line 06.

Let $(H \circ C)^{-1}$ and $HC^{-1}$ both provide the same unique cluster symbol expansion for a phone string $p_1 \ldots p_l$ of length $l$. We can assume without loss of generality that $l \geq 2n$. Indeed, if $l < 2n$, we can pad it with enough $\epsilon$ symbols at the beginning and the end. By Lemma 2, $(H \circ C)^{-1}$ assigns a cluster symbol sequence to this phone string, iff property (10) holds.

The set intersection (10) describes exactly a metastate $\mathbf{q} \in \mathbf{Q}$. The equivalence is obvious when we consider that $j$ denotes the context-position and $\mathcal{A}_q^i$ is modeled by a column at context position $q$ in a bit matrix. The successive intersections are performed at Line 09 in each loop iteration.

Now consider a string $p_1 \ldots p_l p_{l+1}$. Property (10) holds for $(H \circ C)^{-1}$. We need to make sure that a new metastate $\mathbf{t}$ is formed as in (16) and (17). Lemma 1 provides that a metastate $\mathbf{s} \in \mathbf{S}$, satisfying property (15), will always exist for the new polyphone $\gamma_{m+1}$. Therefore, the condition at Line

10 is fulfilled and indeed a new metastate $\mathbf{t}$ is created (Line 11) and $\mathbf{q}$ and $\mathbf{t}$ are connected (Line 15). Pushing $\mathbf{t}$ on $\mathbf{Q}$ for further expansion ensures that the metastate connection algorithm emulates the inductive proof of Lemma 2. □

# 7 Conclusions

We presented an algorithm for directly constructing a transducer mapping from gaussian mixture model sequences to phone sequences. We corrected and extended the work of Schuster and Hori [2], by introducing a correct connection algorithm and providing it with an on-the fly implementation. The direct construction avoids explicitly expanding all possible context-depedent phones by deciding which decision tree clusters can be connected in the final $H \circ C$ composition, allowing for fast compilation of large decision trees into transducers.

Another property of the metastate connection algorithm is that it is local, which allows for an on-demand implementation. In addition, this property enables the on-the-fly application of another weighted finite-state transducer algorithm. We used weighted determinization to construct and immediately determinize an $HC$ transducer, containing as many as 16,000 codebooks. A set of recognition experiments performed with it showed the clear advantage of a semi-continuous acoustic model over a fully continuous acoustic model.

We also proved the correctness of the algorithm, by comparing it with the explicit composition of $H$ and $C$. We showed that the two transducers implement the same string-to-string transduction.

# 8 Future Work

As seen in Table 3, it is neccessary to store many metastates during the expansion. Future work may concentrate on compressing the memory demand of a single metastate, or reducing the size of the $\mathbf{T}$ list during expansion.

Another unexplored property is the on-demand composition of the combined $HC$ transducer with the composition of the language model and the dictionary transducer. Using an $HC$ transducer for a larger decision tree, we were unable to expand a full network with even a bigram language model, which on the other hand might be possible if using a combination of on-the-fly composition and determinization.

## References

[1] S. Kanthak, H. Ney, M. Riley, and M. Mohri, *A Comparison of Two LVR Search Optimization Techniques*, in Proc. ICSLP '02, September 2002.

[2] M. Schuster and T. Hori, *Efficient generation of high-order context-dependent weighted finite-state transducers for speech recognition*, in Proc. ICASSP '05, pp. 201-204, 2005

[3] M. Mohri, F. Pereira, M. Riley, *Weighted Finite State Transducers in Speech Recognition*, in Computer Speech & Language, Volume 16, Issue 1, pp 69-88, January 2002

[4] E. Stoimenov, J. McDonough *Modeling polyphone context with weighted finite-state transducers*, in Proc. ICASSP, 2006

[5] J. Odell, *The Use of Context in Large Vocabulary Speech Recognition*, PhD thesis, Cambridge University Engineering Department, March 1995

[6] S. J. Young, P. C. Woodland, *The Use of State Tying in Continuous Speech Recognition*, in Proc. Eurospeech '93, pp. 2203-2206, 1993

[7] S. J. Young, J. J. Odell, and P.C. Woodland, *Tree-based State Tying for High Accuracy Acoustic Modelling*, In Proc. ARPA Workshop on Human Language Technology, pages 307-312, 1994

[8] M. Mohri, *Statistical Natural Language Processing*, in M. Lothaire, editor, Applied Combinatorics on Words, Cambridge University Press, 2005.

[9] M. Mohri, *Finite-State Transducers in Language and Speech Processing*, Computational Linguistics, 23:2, 1997

[10] S.J. Young, N.H. Russell, J.H.S Thornton, *Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems*, Technical Report CUED/F-INFENG/TR38, Cambridge University Engineering Dept, 1989

[11] S. F. Chen, *Compiling Large-Context Phonetic Decision Trees into Finite-State Transducers*, in Proc. Eurospeech '03, pp. 1169-1172, 2003

[12] R. Sproat, M. Riley, *Compilation of Weighted Finite-State Transducers from Decision Trees*, in Proc. ACL, Santa Cruz, California, June 1996

[13] F. Pereira and M. Riley, *Speech recognition by composition of weighted finite automata*, in Finite-State Language Processing, (E. Roche and Y. Schabes, eds.), pp. 431-453, Cambridge, MA: MIT Press, 1997

[14] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, New York: Academic Press, 1990

[15] M. J. F. Gales, *Semi-tied covariance matrices for hidden Markov models*, IEEE Transactions Speech and Audio Processing, vol. 7, pp. 272281, 1999

[16] E. Eide and H. Gish, *A parametric approach to vocal tract length normalization*, in Proc. ICASSP, 1996

[17] M. J. F. Gales, *Maximum likelihood linear transformations for HMM-based speech recognition*, Computer Speech and Language, vol. 12, 1998

[18] J. McDonough, T. Schaaf, and A. Waibel, *On maximum mutual information speaker-adapted training*, in Proc. ICASSP, 2002

[19] G. Saon, D. Povey, and G. Zweig, *Anatomy of an extremely fast LVCSR decoder*, in Proc. Interspeech, Lisbon, Portugal, 2005

[20] A. Ljolje, F. Pereira, and M. Riley, *Efficient general lattice generation and rescoring*, in Proc. Eurospeech, Budapest, Hungary, 1999

[21] M. Mohri and M. Riley, *Network optimizations for large vocabulary speech recognition*, Speech Communication, vol. 25, no. 3, 1998.

[22] M. Mohri and M. Riley, *A Weight Pushing Algorithm for Large Vocabulary Speech Recognition*, in European Conf. on Speech Communication and Technology, Aalborg, Denmark, pp. 1603 – 1606, 2001

[23] V. Valtchev, P. C. Woodland, and S. J. Young, *MMIE training of large vocabulary speech recognition systems*, Speech Communication, vol. 22, pp. 303314, 1997

[24] J. McDonough, M. Woelfel, and E. Stoimenov *On maximum mutual information speaker-adapted training*, Computer Speech and Language, submitted, September, 2006

[25] J. Duchateau, K. Demuynck, D. V. Campernolle, and P. Wambacq, *Improved parameter tying for efficient acoustic model evaluation in large vocabulary continuous speech recognition*, in Proc. ICSLP, 1998

[26] L. Uebel, P. Woodland, *Improvements in linear transform based speaker adaptation*, in Proc. ICASSP, 2001