TR2007-902

N-Way Composition of Weighted Finite-State Transducers

Cyril Allauzen[1] and Mehryar Mohri[1,2]

[1] Courant Institute of Mathematical Sciences,
251 Mercer Street, New York, NY 10012.
[2] Google Research,
76 Ninth Avenue, New York, NY 10011.

Abstract. Composition of weighted transducers is a fundamental algorithm used in many applications, including for computing complex edit-distances between automata, or string kernels in machine learning, or to combine different components of a speech recognition, speech synthesis, or information extraction system. We present a generalization of the composition of weighted transducers, *n-way composition*, which is dramatically faster in practice than the standard composition algorithm when combining more than two transducers. The expected worst-case complexity of our algorithm for composing three transducers T_1, T_2, and T_3 is $O(\min(|T_1|_E|T_2|_Q|T_3|_E, |T_1|_Q|T_2|_E|T_3|_Q) + |T|)$, where T is the result of that composition and $|T_i| = |T_i|_Q + |T_i|_E$ with $|T_i|_Q$ the number of states and $|T_i|_E$ the number of transitions of T_i, $i = 1, 2, 3$. In many cases, this significantly improves on the complexity of standard composition. Our algorithm also leads to a dramatically faster composition in practice. Furthermore, standard composition can be obtained as a special case of our algorithm. We report the results of several experiments demonstrating this improvement. These theoretical and empirical improvements significantly enhance performance in the applications already mentioned.

1 Introduction

Weighted finite-state transducers are widely used in text, speech, and image processing applications and other related areas such as information extraction [8, 10, 12, 11, 4]. They are finite automata in which each transition is augmented with an output label and some weight, in addition to the familiar (input) label [13, 5, 7]. The weights may represent probabilities, log-likelihoods, or they may be some other costs used to rank alternatives. They are, more generally, elements of a semiring [7].

Weighted transducers are used to represent models derived from large data sets using various statistical learning techniques such as pronunciation dictionaries, statistical grammars, string kernels, or complex edit-distance models [11, 6, 2, 3]. These models can be combined to create complex systems such as a speech

recognition or information extraction system using a fundamental transducer algorithm, *composition of weighted transducers* [12, 11]. Weighted composition is a generalization of the composition algorithm for unweighted finite-state transducers which consists of matching the output label of the transitions of one transducer with the input label of the transitions of another transducer. The weighted case is however more complex and requires the introduction of an ϵ-filter to avoid the creation of redundant ϵ-paths and preserve the correct path multiplicity [12, 11]. The result is a new weighted transducer representing the relational composition of the two transducers.

Composition is widely used in computational biology, text and speech, and machine learning applications. In many of these applications, the transducers used are quite large, they may have as many as several hundred million states or transitions. A critical problem is thus to devise efficient algorithms for combining them.

This paper presents a generalization of the composition of weighted transducer, *n-way composition*, that is dramatically faster than the standard composition algorithm when combining more than two transducers. The worst-case complexity of composing three transducer T_1, T_2, and T_3, with the standard composition algorithm is $O(|T_1||T_2||T_3|)$ [12, 11]. Using perfect hashing, the expected worst-case complexity is $O(\min(|T_1 \circ T_2|_Q|T_3|_E, |T_1 \circ T_2|_E|T_3|_Q) + |T_1 \circ T_2 \circ T_3| + \min(|T_1|_Q|T_2|_E, |T_1|_E|T_2|_Q) + |T_1 \circ T_2|)$, which may be prohibitive in some cases, even when the size of the resulting transducer T is not large. Instead, the expected worst-case complexity of our algorithm is

$$O(\min(|T_1|_E|T_2|_Q|T_3|_E, |T_1|_Q|T_2|_E|T_3|_Q) + |T|). \tag{1}$$

Furthermore, as we shall see later, standard composition can be obtained as a special case of n-way composition.

Our algorithm also leads to a dramatically faster computation of the result of composition in practice. We report the results of several experiments demonstrating this improvement. These theoretical and empirical improvements significantly enhance performance in a series of applications: string kernel-based algorithms in machine learning, the computation of complex edit-distances between automata, speech recognition and speech synthesis, and information extraction.

The remainder of the paper is structured as follows. Some preliminary definitions and terminology are introduced in the next section (Section 2). Section 3 will describe in detail our n-way algorithm, including the different options for construction of ϵ-filters. Section 4 reports the results of experiments using the n-way algorithm and compares them with the standard composition.

2 Preliminaries

This section gives the standard definition and specifies the notation used for weighted transducers.

Finite-state transducers are finite automata in which each transition is augmented with an output label in addition to the familiar input label [1, 5]. Output

labels are concatenated along a path to form an output sequence and similarly with input labels. *Weighted transducers* are finite-state transducers in which each transition carries some weight in addition to the input and output labels [13, 7].

The weights are elements of a semiring, that is a ring that may lack negation [7]. Some familiar semirings are the Boolean semiring, the tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ related to classical shortest-paths problems and algorithms, and the probability semiring $(\mathbb{R}, +, \cdot, 0, 1)$. A semiring is *idempotent* if for all $a \in \mathbb{K}$, $a \oplus a = a$. It is *commutative* when $\otimes$ is commutative. We will assume in this paper that the semiring used is commutative, which is a necessary condition for composition to be an efficient algorithm [10].

The following gives a formal definition of weighted transducers.

Definition 1. *A* weighted finite-state transducer T *over* $(\mathbb{K}, \oplus, \cdot, 0, 1)$ *is an 8-tuple* $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ *where* Σ *is the finite input alphabet of the transducer,* Δ *is the finite output alphabet,* Q *is a finite set of states,* $I \subseteq Q$ *the set of initial states,* $F \subseteq Q$ *the set of final states,* $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$ *a finite set of transitions,* $\lambda : I \to \mathbb{K}$ *the initial weight function, and* $\rho : F \to \mathbb{K}$ *the final weight function mapping* F *to* $\mathbb{K}$.

The weight of a path π is obtained by multiplying the weights of its constituent transitions using the multiplication rule of the semiring and is denoted by $w[\pi]$. The weight of a pair of input and output strings (x, y) is obtained by $\oplus$-summing the weights of the paths labeled with (x, y) from an initial state to a final state.

For a path π, we denote by $p[\pi]$ its origin state and by $n[\pi]$ its destination state. We also denote by $P(I, x, y, F)$ the set of paths from the initial states I to the final states F labeled with input string x and output string y. A transducer T is *regulated* if the output weight associated by T to any pair of strings (x, y):

$$T(x, y) = \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \cdot w[\pi] \cdot \rho[n[\pi]] \tag{2}$$

is well-defined and in $\mathbb{K}$. $T(x, y) = \overline{0}$ when $P(I, x, y, F) = \emptyset$. If for all $q \in Q$ $\bigoplus_{\pi \in P(q,\epsilon,\epsilon,q)} w[\pi] \in \mathbb{K}$, then T is regulated. In particular, when T does not admit any ϵ-cycle, it is regulated. The weighted transducers we will be considering in this paper will be regulated. Figure 1(a) shows an example.

The *composition* of two weighted transducers T_1 and T_2 with matching input and output alphabets Σ, is a weighted transducer denoted by $T_1 \circ T_2$ when the sum:

$$(T_1 \circ T_2)(x, y) = \bigoplus_{z \in \Sigma^*} T_1(x, z) \otimes T_2(z, y) \tag{3}$$

is well-defined and in $\mathbb{K}$ for all $x, y \in \Sigma^*$ [13, 7]. *Weighted automata* can be defined as weighted transducers A with identical input and output labels, for any transition. Thus, only pairs of the form (x, x) can have a non-zero weight by A, which is why the weight associated by A to (x, x) is abusively denoted by $A(x)$ and identified with the *weight associated by* A *to* x. Similarly, in the graph representation of weighted automata, the output (or input) label is omitted.

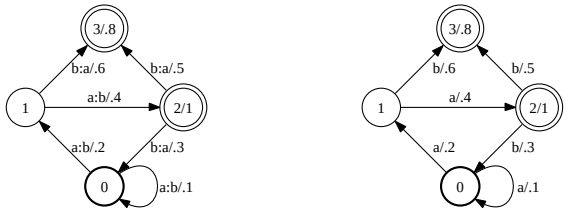

Fig. 1. (a) Example of a weighted transducer T. (b) Example of a weighted automaton A. $[\![T]\!](aab, bba) = [\![A]\!](aab) = .1 \times .2 \times .6 \times .8 + .2 \times .4 \times .5 \times .8$. A bold circle indicates an initial state and a double-circle a final state. The final weight $\rho[q]$ of a final state q is indicated after the slash symbol representing q.

3 Algorithm

3.1 Standard Composition Algorithm

Let us start with a brief description of the standard composition algorithm for weighted transducers [12, 11]. States in the composition $T_1 \circ T_2$ of two weighted transducers T_1 and T_2 are identified with pairs of a state of T_1 and a state of T_2. Leaving aside transitions with ϵ inputs or outputs, the following rule specifies how to compute a transition of $T_1 \circ T_2$ from appropriate transitions of T_1 and T_2:

$$(q_1, a, b, w_1, q_2) \text{ and } (q_1', b, c, w_2, q_2') \Longrightarrow ((q_1, q_1'), a, c, w_1 \otimes w_2, (q_2, q_2')). \quad (4)$$

Figure 2 illustrates the algorithm. In the worst case, all transitions of T_1 leaving a state q_1 match all those of T_2 leaving state q_1', thus the space and time worst-case complexity of composition is quadratic: $O(|T_1||T_2|)$. However, using perfect hashing on the input transducer with the highest number of transitions leads to an expected worst-case complexity of $O(\min(|T_1|_Q|T_2|_E, |T_1|_E|T_2|_Q) + |T_1 \circ T_2|)$.

The main problem with the standard composition algorithm is the following. Assume that one wishes to compute $T_1 \circ T_2 \circ T_3$, say for example by proceeding left to right. Thus, first T_1 and T_2 are composed to compute $T_1 \circ T_2$ and then the result is composed with T_3. The expected worst-case complexity of that computation is:

$$\begin{aligned} O(\, &\min(|T_1 \circ T_2|_Q|T_3|_E, |T_1 \circ T_2|_E|T_3|_Q) + |T_1 \circ T_2 \circ T_3| + \\ &\min(|T_1|_Q|T_2|_E, |T_1|_E|T_2|_Q) + |T_1 \circ T_2|). \end{aligned} \quad (5)$$

But, in many cases, computing $T_1 \circ T_2$ creates a very large number of transitions that may never match any transition of T_3. For example, T_2 may represent a complex edit-distance transducer, allowing all possible insertions, deletions, substitutions and perhaps other operations such as transpositions or more complex edits in T_1 all with different costs. Even when T_1 is a simple non-deterministic

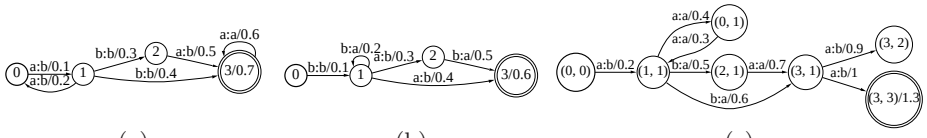

Fig. 2. Example of transducer composition. (a) Weighted transducer T_1 and (b) Weighted transducer T_2 over the probability semiring $(\mathbb{R}, +, \cdot, 0, 1)$. (c) Result of the composition of T_1 and T_2.

finite automaton with ϵ-transitions, which is often the case in the applications already mentioned, $T_1 \circ T_2$ will then have a very large number paths, most of which will not match those of the non-deterministic automaton T_3. In other applications in speech recognition, or for the computation of kernels in machine learning, the central transducer T_2 could be far more complex and the set of transitions or paths of $T_1 \circ T_2$ not matching those of T_3 could be even larger.

3.2 N-Way Composition Algorithm

The key idea behind our algorithm is precisely to avoid creating these unnecessary transitions by directly constructing $T_1 \circ T_2 \circ T_3$, which we refer to as a 3-*way composition*. Thus, our algorithm does not include the intermediate step of creating $T_1 \circ T_2$ or $T_2 \circ T_3$. To do so, we can proceed following a *lateral* or *sideways strategy*: for each transition e_1 in T_1 and e_3 in T_3, we search for matching transitions in T_2. The following is the pseudocode of the algorithm.

N-WAY-COMPOSITION(T_1, T_2, T_3)

```
1  Q ← I_1 × I_2 × I_3
2  S ← I_1 × I_2 × I_3
3  while S ≠ ∅ do
4       (q_1, q_2, q_3) ← HEAD(S)
5       DEQUEUE(S)
6       if (q_1, q_2, q_3) ∈ I_1 × I_2 × I_3 then
7            I ← I ∪ {(q_1, q_2, q_3)}
8            λ(q_1, q_2, q_3) ← λ_1(q_1) ⊗ λ_2(q_2) ⊗ λ_3(q_3)
9       if (q_1, q_2, q_3) ∈ F_1 × F_2 × F_3 then
10           F ← F ∪ {(q_1, q_2, q_3)}
11           ρ(q_1, q_2, q_3) ← ρ_1(q_1) ⊗ ρ_2(q_2) ⊗ ρ_3(q_3)
12      for each (e_1, e_3) ∈ E[q_1] × E[q_3] do
13           G ← {e ∈ E[q_2] : i[e] = o[e_1] ∧ o[e] = i[e_3]}
14           for each e_2 ∈ G do
15                if (n[e_1], n[e_2], n[e_3]) ∉ Q then
16                     Q ← Q ∪ {(n[e_1], n[e_2], n[e_3])}
17                     ENQUEUE(S, (n[e_1], n[e_2], n[e_3]))
18           E ← E ∪ {((q_1, q_2, q_3), i[e_1], o[e_3], w[e_1] ⊗ w[e_2] ⊗ w[e_3], (n[e_1], n[e_2], n[e_3]))}
19 return T
```

The pseudocode is given in the simple case where no ϵ transition is present. The algorithm computes T, the result of the composition $T_1 \circ T_2 \circ T_3$. It uses a queue S containing the set of pairs of states yet to be examined. The queue discipline of S can be arbitrarily chosen and does not affect the termination of the algorithm. Using a FIFO or LIFO discipline, the queue operations can be performed in constant time. We can pre-process the transducers T_1, T_2, and T_3 in linear time and use perfect hashing so that the transitions G (line 13) can be found in expected worst-case linear time $O(|G|)$. Thus, the expected worst-case running time complexity of the n-way composition algorithm is in $O(|T_1|_E|T_2|_Q|T_3|_E + |T|)$, where T is transducer returned by the algorithm.

Alternatively, depending on the size of the three transducers, it may be advantageous to direct the 3-way composition from the center, i.e., ask for each transition e_2 in T_2 if there are matching transitions e_1 in T_1 and e_3 in T_3. We refer to this as the *central strategy* for our n-way composition algorithm. Pre-processing the three transducers and creating hash tables for the transitions leaving each state, this strategy leads to a expected worst-case running time complexity of $O(|T_1|_Q|T_2|_E|T_3|_Q + |T|)$. The lateral and central strategies can be combined by using, at a state (q_1, q_2, q_3), the lateral strategy if $|E[q_1]| \cdot |E[q_3]| \leq |E[q_2]$ and the central strategy otherwise. The algorithm leads to a natural lazy or on-demand implementation in which the transitions of the resulting transducer T are generated only as needed by other operations on T. The standard composition coincides with the n-way algorithm when using the central strategy with either T_1 or T_2 equal to the identity transducer.

3.3 Epsilon filtering

The algorithm described thus far cannot be readily used in most cases found in practice. In general, a transducer T_1 may have transitions with output label ϵ and T_2 transitions with input ϵ. A straightforward generalization of the ϵ-free case would generate redundant ϵ-paths and, in the case of non-idempotent semirings, would lead to an incorrect result, even just for composing two transducers. The weight of two matching ϵ-paths of the original transducers would be counted as many times as the number of redundant ϵ-paths generated in the result, instead of one. Thus, a crucial component of our algorithm consists of coping with this problem.

Figure 3(a) illustrates the problem just mentioned in the simpler case of two transducers. To match ϵ-paths leaving q_1 and those leaving q_2, a generalization of the ϵ-free composition can make the following moves: (1) first move forward on a transition of q_1 with output ϵ, or even a path with output ϵ, and stay at the same state q_2 in T_2, with the hope of later finding a transition whose output label is some label $a \neq \epsilon$ matching a transition of q_2 with the same input label; (2) proceed similarly by following a transition or path leaving q_2 with input label ϵ while staying at the same state q_1 in T_1; or, (3) match a transition of q_1 with output label ϵ with a transition of q_2 with input label ϵ.

Let us rename existing output ϵ-labels of T_1 as ϵ_2, and existing input ϵ-labels of T_2 ϵ_1, and let us augment T_1 with a self-loop labeled with ϵ_1 at all states and

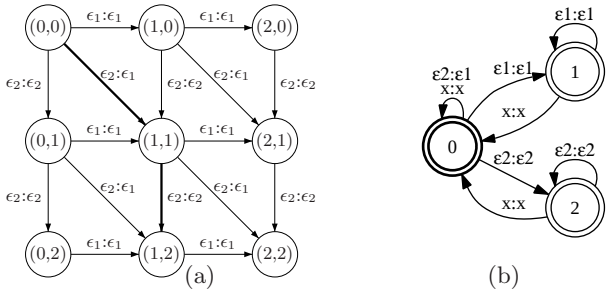

Fig. 3. (a) Redundant ϵ-paths. A straightforward generalization of the ϵ-free case could generate all the paths from $(0, 0)$ to $(2, 2)$ for example, even when composing just two simple transducers. (b) Filter transducer M allowing a unique ϵ-path.

similarly, augment T_2 with a self-loop labeled with ϵ_2 at all states, as illustrated by Figures 5(a) and (c). These self-loops correspond to staying at the same state in that machine while consuming an ϵ-label of the other transition. The three moves just described now correspond to the matches (1) $(\epsilon_2 : \epsilon_2)$, (2) $(\epsilon_1 : \epsilon_1)$, and (3) $(\epsilon_2 : \epsilon_1)$. The grid of Figure 3(a) shows all the possible ϵ-paths between composition states. We will denote by $\tilde{T}_1$ and $\tilde{T}_2$ the transducers obtained after application of these changes.

For the result of composition to be correct, between any two of these states, all but one path must be disallowed. There are many possible ways of selecting that path. One natural way is to opt for the shortest path with the diagonal transitions (ϵ-matching transitions) taken first. Figure 3(a) illustrates in boldface the path just described from state $(0, 0)$ to state $(1, 2)$. Remarkably, this filtering mechanism itself can be encoded as a finite-state transducer such as the transducer M of Figure 3(b). We denote by $(p, q) \preceq (r, s)$ to indicate that (r, s) can be reached from (p, q) in the grid.

Proposition 1. *Let M be the transducer of Figure 3(b). M allows a unique path between any two states (p, q) and (r, s), with $(p, q) \preceq (r, s)$.*

Proof. Let a denote $(\epsilon_1 : \epsilon_1)$, b denote $(\epsilon_2 : \epsilon_2)$, c denote $(\epsilon_2 : \epsilon_1)$, and let x stand for any $(x : x)$, with $x \in \Sigma$. The following sequences must be disallowed by a shortest-path filter with matching transitions first: ab, ba, ac, bc. This is because, from any state, instead of the moves ab or ba, the matching or diagonal transition c can be taken. Similarly, instead of ac or bc, ca and cb can be taken for an earlier match. Conversely, it is clear from the grid or an immediate recursion that a filter disallowing these sequences accepts a unique path between two connected states of the grid.

Let L be the set of sequences over $\sigma = \{a, b, c, x\}$ that contain one of the disallowed sequence just mentioned as a substring that is $L = \sigma^*(ab + ba + ac + bc)\sigma^*$. Then $\overline{L}$ represents exactly the set of paths allowed by that filter and is

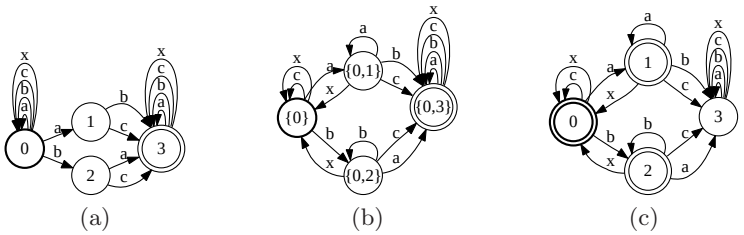

Fig. 4. (a) Finite automaton A representing the set of disallowed sequences. (b) Automaton B, result of the determinization of A. Subsets are indicated at each state. (c) Automaton C obtained from B by complementation, state 3 is not coaccessible.

thus a regular language. Let A be an automaton representing L (Figure 4(a)). An automaton representing $\overline{L}$ can be constructed from A by determinization and complementation (Figures 4(a)-(c)). The resulting automaton C is equivalent to the transducer M after removal of the state 3, which does not admit a path to a final state. □

Thus, to compose two transducers T_1 and T_2 with ϵ-transitions, it suffices to compute $\tilde{T_1} \circ M \circ \tilde{T_2}$, using the rules of composition in the ϵ-free case.

The problem of avoiding the creation of redundant ϵ-paths is more complex in 3-way composition since the ϵ-transitions of all three transducers must be taken into account. We describe two solutions for this problem, one based on two filters, another based on a single filter.

2-way ϵ-Filters. One way to deal with this problem is to use the 2-way filter M, by first dealing with matching ϵ-paths in $U = (T_1 \circ T_2)$, and then $U \circ T_3$. However, in 3-way composition, it is possible to remain at the same state of T_1 and the same state of T_2, and move on an ϵ-transition of T_3, which previously was not an option. This corresponds to staying at the same state of U, while moving on a transition of T_3 with input ϵ. To account for this move, we introduce a new symbol ϵ_0 matching ϵ_1 in T_3. But, we must also ensure the existence of a self-loop with output label ϵ_0 at all states of U. To do so, we augment the filter M with self-loops $(\epsilon_1{:}\epsilon_0)$ and the transducer T_2 with self-loops $(\epsilon_0{:}\epsilon_1)$ (see Figure 5(b)). Figure 5(d) shows the resulting filter transducer M_1. From Figures 5(a)-(c), it is clear that $\tilde{T_1} \circ M_1 \circ \tilde{T_2}$ will have precisely a self-loop labeled with $(\epsilon_1{:}\epsilon_1)$ at all states.

In the same way, we must allow for moving forward on a transition of T_1 with output ϵ, that is consuming ϵ_2, while remaining at the same states of T_2 and T_3. To do so, we introduce again a new symbol ϵ_0 this time only relevant for matching T_2 with T_3, add self-loops $(\epsilon_2{:}\epsilon_0)$ to T_2, and augment the filter M by adding a transition labeled with $(\epsilon_0{:}\epsilon_2)$ (resp. $(\epsilon_0{:}\epsilon_1)$) wherever there used to be one labeled with $(\epsilon_2{:}\epsilon_2)$ (resp. $(\epsilon_2{:}\epsilon_1)$). Figure 5(e) shows the resulting filter transducer M_2.

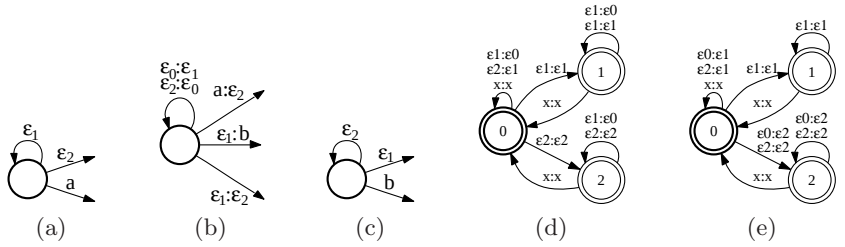

Fig. 5. Marking of transducers and 2-way filters. (a) $\tilde{T}_1$. Self-loop labeled with ϵ_1 added at all states of T_1, regular output ϵs renamed to ϵ_2. (b) $\tilde{T}_2$. Self-loops with labels (ϵ_0:ϵ_1) and (ϵ_2:ϵ_0) added at all states of T_2. Input ϵs are replaced by ϵ_1, output ϵs by ϵ_2. (c) $\tilde{T}_3$. Self-loop labeled with ϵ_2 added at all states of T_3, regular input ϵs renamed to ϵ_1. (d) Left-to-right filter M_1. (e) Left-to-right filter M_2.

Thus, the composition $\tilde{T}_1 \circ M_1 \circ \tilde{T}_2 \circ M_2 \circ \tilde{T}_3$ ensures the uniqueness of matching ϵ-paths. In practice, the modifications of the transducers T_1, T_2, and T_3 to generate $\tilde{T}_1$, $\tilde{T}_2$, and $\tilde{T}_3$, as well as the filters M_1 and M_2 can be directly simulated or encoded in the 3-way composition algorithm for greater efficiency. The states in T become quintuples $(q_1, q_2, q_3, f_1, f_2)$ with f_1 and f_2 are states of the filters M_1 and M_2. The introduction of self-loops and marking of ϵs can be simulated (line 12-13) and the filter states f_1 and f_2 taken into account to compute the set G of the transition matches allowed (line 13).

Note that while N-way composition is symmetric, the analysis of ϵ-paths just presented is left-to-right and the filters M_1 and M_2 are not symmetric. In fact, we could similarly define right-to-left filters M_1' and M_2'. The advantage of the filters presented in this section is however that they can help modify easily an existing implementation of composition into 3-way composition. The filters needed for the 3-way case are also straightforward generalizations of the ϵ-filter used in standard composition.

3-way ϵ-Filter. There exists however a direct and symmetric method for dealing with ϵ-paths in 3-way composition. Remarkably, this can be done using a single filter automaton whose labels are 3-dimensional vectors. Figure 6 shows a filter W that can be used for that purpose. Each transition is labeled with a triplet. The ith element of the triplet corresponding to the move on the ith transducer. 0 indicates staying at the same state or not moving, 1 that a move is made reading an ϵ-transition, and x a move along a matching transition with a non-empty symbol (i.e., non-ϵ output in T_1, non-ϵ input or output in T_2 and non-ϵ input in T_3).

Matching ϵ-paths now correspond to a three-dimensional grid. As in the two-dimensional case, $(p, q, r) \preceq (s, t, u)$ indicates that (s, t, u) can be reached from (p, q, r) in the grid. Several filters are possible, here we will again favor the matching of ϵ-transitions (i.e. the diagonals on the grid).

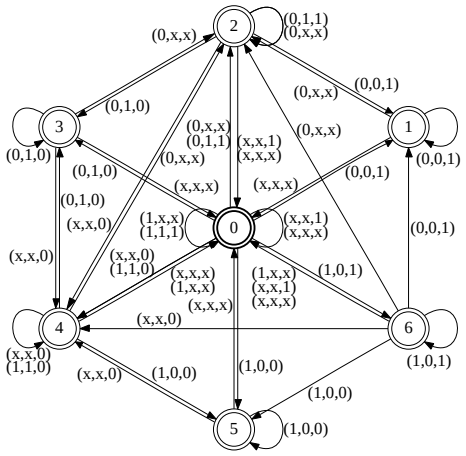

Fig. 6. 3-way matching ϵ-filter W.

Proposition 2. *The filter automaton W allows a unique path between any two states (p, q, r) and (s, t, u) of a three-dimensional grid, with $(p, q, r) \preceq (s, t, u)$.*

Proof. We give a sketch of the proof, which is similar to that of Proposition 1. As in that proof, we can enumerate disallowed sequences of triplets. The triplet (0, 0, 0) is always forbidden since it corresponds to remaining at the same state in all three transducers. Observe that in two consecutive triplets, for $i \in [1, 3]$, 0 in the ith machine of the first triplet cannot be followed by 1 in the second. Indeed, as in the 2-way case, if we stay at a state, then we must remain at that state until a match with a non-empty symbol is made. Also, two 0s in adjacent transducers (T_1 and T_2, or T_2 and T_3), cannot become both xs unless all components become xs. For example, the sequence $(0, 0, 1)(x, x, 1)$ is disallowed since instead $(x, x, 1)(0, 0, 1)$ with an earlier match can be followed. Similarly, the sequence $(0, 0, 1)(x, x, 0)$ is disallowed since instead the single and shorter move $(x, x, 1)$ can be taken. Conversely, it is not hard to see that a filter disallowing these sequences accepts a unique path between two connected states of the grid.

Thus, a filter can be obtained by taking the complement of the automaton accepting the sequences admitting such forbidden substrings. The resulting deterministic and minimal automaton is exactly the filter W shown in Figure 6. □

4 Experiments

This section reports the results of experiments carried out in two different applications: the computation of a complex edit-distance between two automata, as motivated by applications in text and speech processing [9], and the computation of kernels between automata needed in spoken-dialog classification and other machine learning tasks.

Table 1. Comparison of N-way composition with standard composition. The computation times are reported in seconds, the size of T_2 in number of transitions. These experiments were performed on a dual-core AMD Opteron 2.2GHz with 16GB of memory, using the same software library and basic infrastructure.

	n-gram Kernel						Edit distance	
	≤ 2	≤ 3	≤ 4	≤ 5	≤ 6	≤ 7	standard	+transpositions
Standard	65.3	68.3	71.0	73.5	76.3	78.3	586.1	913.5
N-way	8.0	8.1	8.2	8.2	8.2	8.2	3.8	5.9
Size of T_2	70K	100K	130K	160K	190K	220K	25M	75M

In the edit-distance case, the standard transducer T_2 used was one based on all insertions, deletions, and substitutions with different costs [9]. A more realistic transducer T_2 was one augmented with all transpositions, e.g., $ab \rightarrow ba$, with different costs. In the kernel case, n-gram kernels with varying n-gram order were used [3].

Table 4 shows the results of these experiments. The finite automata T_1 and T_3 used were extracted from real text and speech processing tasks. The results show that in all cases, N-way composition is orders of magnitude faster than standard composition.

5 Conclusion

We presented a general algorithm for the composition of weighted finite-state transducers. In many instances, N-way composition benefits from a significantly better time and space complexity. Our experiments with both complex edit-distance computations arising in a number of applications in text and speech processing, and with kernel computations, crucial to many machine learning algorithms applied to sequence prediction, show that our algorithm is also substantially faster than standard composition in practice. We expect N-way composition to further improve efficiency in a variety of other areas and applications in which weighted composition of transducers is used.

Acknowledgments

The research of Cyril Allauzen and Mehryar Mohri was partially supported by the New York State Office of Science Technology and Academic Research (NYSTAR). This project was also sponsored in part by the Department of the Army Award Number W81XWH-04-1-0307. The U.S. Army Medical Research Acquisition Activity, 820 Chandler Street, Fort Detrick MD 21702-5014 is the awarding and administering acquisition office. The content of this material does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

References

1. Jean Berstel. *Transductions and Context-Free Languages*. Teubner, 1979.

2. Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report, TR-10-98, Harvard University, 1998.
3. Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels: Theory and Algorithms. *Journal of Machine Learning Research*, 5:1035–1062, 2004.
4. Karel Culik II and Jarkko Kari. Digital Images and Formal Languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 599–616. Springer, 1997.
5. Samuel Eilenberg. *Automata, Languages and Machines*. Academic Press, 1974–76.
6. Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustic, Speech, and Signal Processing*, 35(3):400–401, 1987.
7. Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1986.
8. Mehryar Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23(2), 1997.
9. Mehryar Mohri. Edit-Distance of Weighted Automata: General Definitions and Algorithms. *International Journal of Foundations of Computer Science*, 14(6):957–982, 2003.
10. Mehryar Mohri. Statistical Natural Language Processing. In M. Lothaire, editor, *Applied Combinatorics on Words*. Cambridge University Press, 2005.
11. Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted Automata in Text and Speech Processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96)*. John Wiley and Sons, 1996.
12. Fernando Pereira and Michael Riley. *Finite State Language Processing*, chapter Speech Recognition by Composition of Weighted Finite Automata. The MIT Press, 1997.
13. Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag, 1978.