Finite-State Transducers in Language and Speech Processing

Mehryar Mohri *
AT&T Labs-Research

Finite-state machines have been used in various domains of natural language processing. We consider here the use of a type of transducers that supports very efficient programs: sequential transducers. We recall classical theorems and give new ones characterizing sequential string-to-string transducers. Transducers that output weights also play an important role in language and speech processing. We give a specific study of string-to-weight transducers, including algorithms for determinizing and minimizing these transducers very efficiently, and characterizations of the transducers admitting determinization and the corresponding algorithms. Some applications of these algorithms in speech recognition are described and illustrated.

1. Introduction

Finite-state machines have been used in many areas of computational linguistics. Their use can be justified by both linguistic and computational arguments. Linguistically, finite automata are convenient since they allow one to describe easily most of the relevant local phenomena encountered in the empirical study of language. They often lead to a compact representation of lexical rules, or idioms and clichés, that appears as natural to linguists (Gross, 1989). Graphic tools also allow one to visualize and modify automata. This helps in correcting and completing a grammar. Other more general phenomena such as parsing context-free grammars can also be dealt with using finite-state machines such as RTN's (Woods, 1970). Moreover, the underlying mechanisms in most of the methods used in parsing are related to automata.

From the computational point of view, the use of finite-state machines is mainly motivated by considerations of time and space efficiency. Time efficiency is usually achieved by using deterministic automata. The output of deterministic machines depends, in general linearly, only on the input size and can therefore be considered as optimal from this point of view. Space efficiency is achieved with classical minimization algorithms (Aho, Hopcroft, and Ullman, 1974) for deterministic automata. Applications such as compiler construction have shown deterministic finite automata to be very efficient in practice (Aho, Sethi, and Ullman, 1986). Finite automata now also constitute a rich chapter of theoretical computer science (Perrin, 1990).

Their recent applications in natural language processing which range from the construction of lexical analyzers (Silberztein, 1993) and the compilation of morphological and phonological rules (Kaplan and Kay, 1994; Karttunen, Kaplan, and Zaenen, 1992) to speech processing (Mohri, Pereira, and Riley, 1996) show the usefulness of finite-state machines in many areas. We provide here theoretical and algorithmic bases for the use and application of the devices that support very efficient programs: sequential trans-

* 600 Mountain Avenue, Murray Hill, NJ 07974, USA. This electronic version includes several corrections that did not appear in the printed version.

ducers.

We extend the idea of deterministic automata to transducers with deterministic input, that is machines that produce output strings or weights in addition to accepting (deterministically) input. Thus, we describe methods consistent with the initial reasons for using finite-state machines, in particular the time efficiency of deterministic machines, and the space efficiency achievable with new minimization algorithms for sequential transducers.

Both time and space concerns are important when dealing with language. Indeed, one of the trends which clearly come out of the new studies of language is a large increase in the size of data. Lexical approaches have shown to be the most appropriate in many areas of computational linguistics ranging from large-scale dictionaries in morphology to large lexical grammars in syntax. The effect of the size increase on time and space efficiency is probably the main computational problem one needs to face in language processing.

The use of finite-state machines in natural language processing is certainly not new. Limitations of the corresponding techniques, however, are very often pointed out more than their advantages. The reason for that is probably that recent work in this field are not yet described in computer science textbooks. Sequential finite-state transducers are now used in all areas of computational linguistics.

In the following we give an extended description of these devices. We first consider the case of string-to-string transducers. These transducers have been successfully used in the representation of large-scale dictionaries, computational morphology, and local grammars and syntax. We describe the theoretical bases for the use of these transducers. In particular, we recall classical theorems and give new ones characterizing these transducers.

We then consider the case of sequential string-to-weight transducers. These transducers appear as very interesting in speech processing. Language models, phone lattices and word lattices are among the objects that can be represented by these transducers. We give new theorems extending the characterizations known for usual transducers to these transducers. We define an algorithm for determinizing string-to-weight transducers. We both characterize the unambiguous transducers admitting determinization and describe an algorithm to test determinizability. We also give an algorithm to minimize sequential transducers which has a complexity equivalent to that of classical automata minimization and which is very efficient in practice. Under certain restrictions, the minimization of sequential string-to-weight transducers can also be performed using the determinization algorithm. We describe the corresponding algorithm and give the proof of its correctness in the appendix.

We have used most of these algorithms in speech processing. In the last section we describe some applications of determinization and minimization of string-to-weight transducers in speech recognition. We illustrate them by several results which show them to be very efficient. Our implementation of the determinization is such that it can be used *on-the-fly*: only the necessary part of the transducer needs to be expanded. This plays an important role in the space and time efficiency of speech recognition. The reduction of the word lattices these algorithms provide sheds new light on the complexity of the networks involved in speech processing.

2. Sequential string-to-string transducers

Sequential string-to-string transducers are used in various areas of natural language processing. Both determinization (Mohri, 1994c) and minimization algorithms (Mohri, 1994b) have been defined for the class of p-subsequential transducers which includes sequential string-to-string transducers. In this section the theoretical basis of the use of sequential transducers is described. Classical and new theorems help to indicate the usefulness of these devices as well as their characterization.

2.1 Sequential transducers

We consider here *sequential* transducers, namely transducers with a deterministic input. At any state of such transducers, at most one outgoing arc is labeled with a given element of the alphabet. Figure 1 gives an example of a sequential transducer. Notice that output labels might be strings, including the empty string ϵ. The empty string is not allowed on input however. The output of a sequential transducer is not necessarily deterministic. The one in figure 1 is not since two distinct arcs with output labels b for instance leave the state 0.

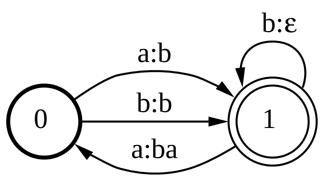

Figure 1
Example of a sequential transducer.

Sequential transducers are computationally very interesting because their use with a given input does not depend on the size of the transducer but only on that of the input. Since that use consists of following the only path corresponding to the input string and in writing consecutive output labels along this path, the total computational time is linear in the size of the input, if we consider that the cost of copying out each output label does not depend on its length.

Definition

More formally, a sequential string-to-string transducer T is a 7-tuple $(Q, i, F, \Sigma, \Delta, \delta, \sigma)$, where:

- Q is the set of states,
- $i \in Q$ is the initial state,
- $F \subseteq Q$, the set of final states,
- Σ and Δ, finite sets corresponding respectively to the input and output alphabets of the transducer,
- δ, the state transition function which maps $Q \times \Sigma$ to Q,
- σ, the output function which maps $Q \times \Sigma$ to Δ^*.

The functions δ and σ are generally partial functions: a state $q \in Q$ does not necessarily admit outgoing transitions labeled on the input side with all elements of the alphabet. These functions can be extended to mappings from $Q \times \Sigma^*$ by the following classical recurrence relations:

$$\forall s \in Q, \forall w \in \Sigma^*, \forall a \in \Sigma, \quad \begin{array}{l} \delta(s,\epsilon) = s, \delta(s,wa) = \delta(\delta(s,w),a); \\ \sigma(s,\epsilon) = \epsilon, \sigma(s,wa) = \sigma(s,w)\sigma(\delta(s,w),a). \end{array}$$

Thus, a string $w \in \Sigma^*$ is accepted by T iff $\delta(i,w) \in F$, and in that case the output of the transducer is $\sigma(i,w)$.

2.2 Subsequential and p-Subsequential transducers

Sequential transducers can be generalized by introducing the possibility of generating an additional output string at final states (Schützenberger, 1977). The application of the transducer to a string can then possibly finish with the concatenation of such an output string to the usual output. Such transducers are called *subsequential* transducers. Language processing often requires a more general extension. Indeed, the ambiguities

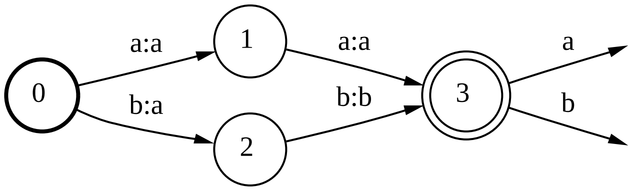

Figure 2
Example of a 2-subsequential transducer τ_1.

encountered in language – ambiguity of grammars, of morphological analyzers, or that of pronunciation dictionaries, for instance – cannot be taken into account when using sequential or subsequential transducers. These devices associate at most a single output to a given input. In order to deal with ambiguities one can introduce p-subsequential transducers (Mohri, 1994a), namely transducers provided with at most p final output strings at each final state. Figure 2 gives an example of a 2-subsequential transducer. Here, the input string $w = aa$ gives two distinct outputs aaa and aab. Since one cannot find any reasonable case in language in which the number of ambiguities would be infinite, p-subsequential transducers seem to be sufficient for describing linguistic ambiguities. However, the number of ambiguities could be very large in some cases. Notice that 1-subsequential transducers are exactly the subsequential transducers.

Transducers can be considered as representing mappings from strings to strings. So, they admit the composition operation defined for mappings. This operation is very useful since it allows one to construct more complex transducers from simpler ones. The result of the application of $\tau_2 \circ \tau_1$ to a string s can be computed by first considering all output strings associated with the input s in the transducer τ_1, then applying τ_2 to all these strings. The output strings obtained after this application represent the result $(\tau_2 \circ \tau_1)(s)$. In fact, instead of waiting for the result of the application of τ_1 to be completely given, one can gradually apply τ_2 to the output strings of τ_1 yet to be completed. This is the basic idea of the composition algorithm which allows one to directly construct the transducer $\tau_2 \circ \tau_1$ given τ_1 and τ_2.

We define *sequential* (resp. *p-subsequential*) *functions* to be those functions that can be represented by sequential (resp. p-subsequential) transducers. We noticed previously that the result of the composition of two transducers is a transducer which can be directly constructed. There exists an efficient algorithm for the general case of the composition of transducers (transducers subsequential or not, having ϵ-transitions or not, and with outputs in Σ^*, or in $\Sigma^* \cup \{\infty\} \times \mathcal{R}_+ \cup \{\infty\}$) (Mohri, Pereira, and Riley, 1996).

The following theorem gives a more specific result for the case of subsequential and p-subsequential functions which expresses their closure under composition. We use the expression p-subsequential in two ways here. One means that a finite number of ambiguities is admitted (the closure under composition matches this case), the second indicates that this number equals exactly p.

Theorem 1
Let $f : \Sigma^* \to \Delta^*$ be a sequential (resp. p-subsequential) and $g : \Delta^* \to \Omega^*$ be a sequential (resp. q-subsequential) function, then $g \circ f$ is sequential (resp. pq-subsequential).

Proof
We prove the theorem in the general case of p-subsequential transducers. The case of sequential transducers, first proved by Choffrut (1978), can be derived from the general case in a trivial way. Let τ_1 be a p-subsequential transducer representing f, $\tau_1 = (Q_1, i_1, F_1, \Sigma, \Delta, \delta_1, \sigma_1, \rho_1)$, and $\tau_2 = (Q_2, i_2, F_2, \Delta, \Omega, \delta_2, \sigma_2, \rho_2)$ a q-subsequential transducer representing g. ρ_1 and ρ_2 denote the final output functions of τ_1 and τ_2 which map respectively F_1 to $(\Delta^*)^p$ and F_2 to $(\Omega^*)^q$. $\rho_1(r)$ represents for instance the set of final output strings at a final state r. Define the pq-subsequential transducer $\tau = (Q, i, F, \Sigma, \Omega, \delta, \sigma, \rho)$ by $Q = Q_1 \times Q_2$, $i = (i_1, i_2)$, $F = \{(q_1, q_2) \in Q : q_1 \in F_1, \delta_2(q_2, \rho_1(q_1)) \cap F_2 \neq \emptyset\}$, with the following transition and output functions:

$$\begin{aligned} \forall a \in \Sigma,\ \forall (q_1, q_2) \in Q, \delta((q_1, q_2), a) &= (\delta_1(q_1, a), \delta_2(q_2, \sigma_1(q_1, a))) \\ \sigma((q_1, q_2), a) &= \sigma_2(q_2, \sigma_1(q_1, a)) \end{aligned}$$

and with the final output function defined by:

$$\forall (q_1, q_2) \in F, \rho((q_1, q_2)) = \sigma_2(q_2, \rho_1(q_1))\rho_2(\delta(q_2, \rho_1(q_1)))$$

Clearly, according to the definition of composition, the transducer τ realizes $g \circ f$. The definition of ρ shows that it admits at most pq distinct output strings for a given input one. This ends the proof of the theorem. ■
Figure 3 gives an example of a 1-subsequential or subsequential transducer τ_2. The

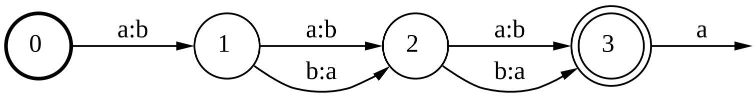

Figure 3
Example of a subsequential transducer τ_2.

result of the composition of the transducers τ_1 and τ_2 is shown in figure 4. States in the transducer τ_3 correspond to pairs of states of τ_1 and τ_2. The composition consists essentially of making the intersection of the outputs of τ_1 with the inputs of τ_2.

Transducers admit another useful operation: *union*. Given an input string w, a transducer union of τ_1 and τ_2 gives the set union of the strings obtained by application of τ_1 to w and τ_2 to w. We denote by $\tau_1 + \tau_2$ the union of τ_1 and τ_2. The following theorem specifies the type of the transducer $\tau_1 + \tau_2$, implying in particular the closure under union of acyclic p-subsequential transducers. It can be proved in a way similar to the composition theorem.

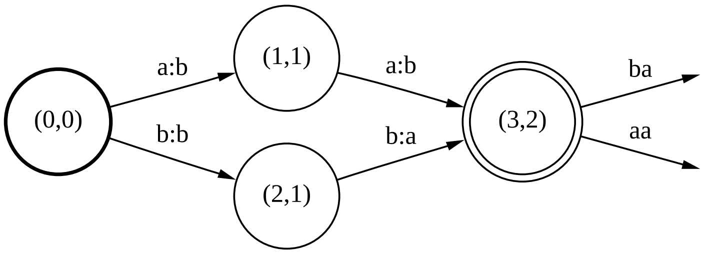

Figure 4
2-Subsequential transducer τ_3, obtained by composition of τ_1 and τ_2.

Theorem 2
Let $f : \Sigma^* \rightarrow \Delta^*$ be a sequential (resp. p-subsequential) and $g : \Sigma^* \rightarrow \Delta^*$ be a sequential (resp. q-subsequential) function represented by acyclic transducers, then $g + f$ is 2-subsequential (resp. $(p+q)$-subsequential).

The union transducer $\tau_1 + \tau_2$ can be constructed from τ_1 and τ_2 in a way close to the union of automata. One can indeed introduce a new initial state connected to the old initial states of τ_1 and τ_2 by transitions labeled with the empty string both on input and output. But, the transducer obtained using this construction is not sequential since it contains ϵ-transitions on the input side. There exists however an algorithm to construct the union of p-subsequential and q-subsequential transducers directly as a $(p+q)$-subsequential transducer.

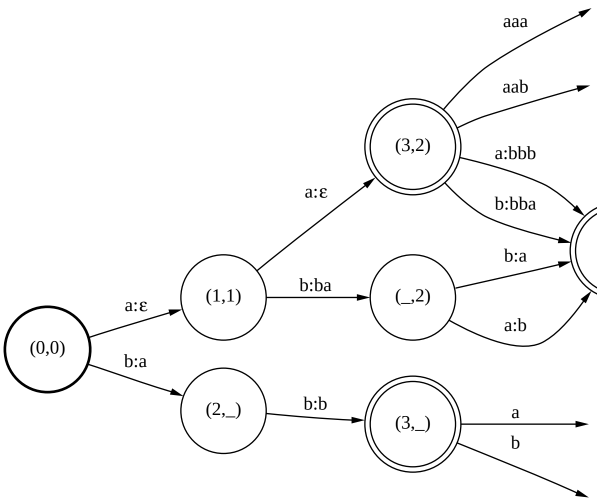

Figure 5
2-Subsequential transducer τ_4, union of τ_1 and τ_2.

The direct construction consists of considering pairs of states (q_1, q_2), q_1 being a state of τ_1 or an additional state that we denote by an underscore, q_2 a state of τ_2 or an additional state that we denote by an underscore. The transitions leaving (q_1, q_2) are obtained by taking the union of the transitions leaving q_1 and q_2, or by keeping only those of q_1 if q_2 is the underscore state, similarly by keeping only those of q_2 if q_1 is the underscore state. The union of the transitions is performed in such a way that if q_1 and

q_2 both have transitions labeled with the same input label a, then only one transition labeled with a is associated to (q_1, q_2). The output label of that transition is the longest common prefix of the output transitions labeled with a leaving q_1 and q_2. See (Mohri, 1996b) for a full description of this algorithm.

Figure 5 shows the 2-subsequential transducer obtained by constructing the union of the transducers τ_1 and τ_2 this way. Notice that according to the theorem the result could be *a priori* 3-subsequential, but these two transducers share no common accepted string. In such cases, the resulting transducer is $\max(p, q)$-subsequential.

2.3 Characterization and extensions

The linear complexity of their use makes sequential or p-subsequential transducers both mathematically and computationally of particular interest. However, not all transducers, even when they realize functions (*rational functions*), admit an equivalent sequential or subsequential transducer.

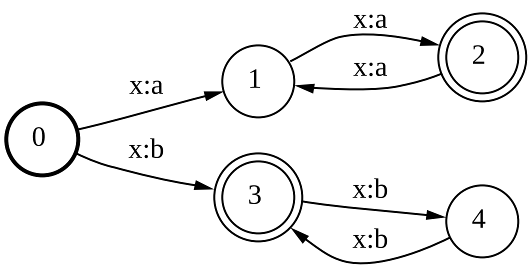

Figure 6
Transducer T with no equivalent sequential representation.

Consider for instance the function f associated with the classical transducer represented in figure 6. f can be defined by[1]:

$$\forall w \in \{x\}^+, \;\; f(w) = \begin{array}{ll} a^{|w|} & if\ |w|\ is\ even, \\ = \;\; b^{|w|} & otherwise \end{array} \qquad (1)$$

This function is not sequential, that is, it cannot be realized by any sequential transducer. Indeed, in order to start writing the output associated to an input string $w = x^n$, a or b according to whether n is even or odd, one needs to finish reading the whole input string w, which can be arbitrarily long. Sequential functions, namely functions which can be represented by sequential transducers do not allow such unbounded delays. More generally, sequential functions can be characterized among rational functions by the following theorem.

Theorem 3 (Ginsburg and Rose, 1966)
Let f be a rational function mapping Σ^* to Δ^*. f is sequential iff there exists a positive integer K such that:

$$\forall u \in \Sigma^*, \forall a \in \Sigma, \exists w \in \Delta^*, \;\; |w| \leq K : \;\; f(ua) = f(u)w \qquad (2)$$

1 We denote by $|w|$ the length of a string w.

In other words, for any string u and any element of the alphabet a, $f(ua)$ is equal to $f(u)$ concatenated with some bounded string. Notice that this implies that $f(u)$ is always a prefix of $f(ua)$, and more generally that if f is sequential then it preserves prefixes.

The fact that not all rational functions are sequential could reduce the interest of sequential transducers. The following theorem due to Elgot and Mezei (1965) shows however that transducers are exactly compositions of left and right sequential transducers.

Theorem 4 *(Elgot and Mezei, 1965)*
Let f be a partial function mapping Σ^* to Δ^*. f is rational iff there exist a left sequential function $l : \Sigma^* \rightarrow \Omega^*$ and a right sequential function $r : \Omega^* \rightarrow \Delta^*$ such that $f = r \circ l$.

Left sequential functions or transducers are those we previously defined. Their application to a string proceeds from left to right. Right sequential functions apply to strings from right to left. According to the theorem, considering a new sufficiently large alphabet Ω allows one to define two sequential functions l and r decomposing a rational function f. This result considerably increases the importance of sequential functions in the theory of finite-state machines as well as in the practical use of transducers.

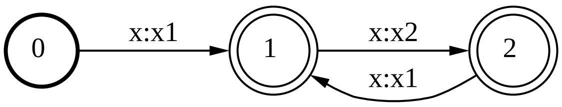

Figure 7
Left to right sequential transducer L.

Berstel (1979) gives a constructive proof of this theorem. Given a finite-state transducer T, one can easily construct a left sequential transducer L and a right sequential transducer R such that $R \circ L = T$. Intuitively, the extended alphabet Ω keeps track of the local ambiguities encountered when applying the transducer from left to right. A distinct element of the alphabet is assigned to each of these ambiguities. The right sequential transducer can be constructed in such a way that these ambiguities can then be resolved from right to left. Figures 7 and 8 give a decomposition of the non sequential

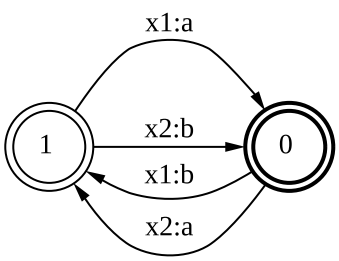

Figure 8
Right to left sequential transducer R

transducer T of Figure 6. The symbols of the alphabet $\Omega = \{x1, x2\}$ store information about the size of the input string w. The output of L ends with $x1$ iff $|w|$ is odd. The right sequential function R is then easy to construct.

Sequential transducers offer other theoretical advantages. In particular, while several important tests such as the equivalence are undecidable with general transducers, sequential transducers have the following decidability property.

Theorem 5
Let T be a transducer mapping Σ^* to Δ^*. It is decidable whether T is sequential.

A constructive proof of this theorem was given by Choffrut (1978). An efficient polynomial algorithm for testing the sequentiability of transducers based on this proof was given by Weber and Klemm (1995).

Choffrut also gave a characterization of subsequential functions based on the definition of a metric on Σ^*. Denote by $u \wedge v$ the longest common prefix of two strings u and v in Σ^*. It is easy to verify that the following defines a metric on Σ^*:

$$d(u,v) = |u| + |v| - 2|u \wedge v| \tag{3}$$

The following theorem describes this characterization of subsequential functions.

Theorem 6
Let f be a partial function mapping Σ^* to Δ^*. f is subsequential iff:

1. f has bounded variation (according to the metric defined above).
2. for any rational subset Y of Δ^*, $f^{-1}(Y)$ is rational.

The notion of bounded variation can be roughly understood here in the following way: if $d(x,y)$ is small enough, namely if the prefix x and y share is sufficiently long compared to their lengths, then the same is true of their images by f, $f(x)$ and $f(y)$. This theorem can be extended to describe the case of p-subsequential functions by defining a metric d_∞ on $(\Delta^*)^p$. For any $u = (u_1, ..., u_p)$ and $v = (v_1, ..., v_p) \in (\Delta^*)^p$, we define:

$$d_\infty(u,v) = \max_{1 \le i \le p} d(u_i, v_i) \tag{4}$$

Theorem 7
Let $f = (f_1, ..., f_p)$ be a partial function mapping $Dom(f) \subseteq \Sigma^*$ to $(\Delta^*)^p$. f is p-subsequential iff:

1. f has bounded variation (using the metric d on Σ^* and d_∞ on $(\Delta^*)^p$).
2. for all i ($1 \le i \le p$) and any rational subset Y of Δ^*, $f_i^{-1}(Y)$ is rational.

Proof
Assume f p-subsequential, and let T be a p-subsequential transducer realizing f. A transducer T_i, $1 \le i \le p$, realizing a component f_i of f can be obtained from T simply by keeping only one of the p outputs at each final state of T. T_i is subsequential by construction, hence the component f_i is subsequential. Then the previous theorem implies that each component f_i has bounded variation, and by definition of d_∞, f has also bounded variation.

Conversely, if the first condition holds, a fortiori each f_i has bounded variation. This combined with the second condition implies that each f_i is subsequential. A transducer

T realizing f can be obtained by taking the union of p subsequential transducers realizing each component f_i. Thus, in view of the theorem 2, f is p-subsequential. ■

One can also give a characterization of p-subsequential transducers irrespective of the choice of their components. Let d'_p be the semi-metric defined by:

$$\forall (u,v) \in [(\Delta^*)^p]^2,\ d'_p(u,v) = \max_{1 \leq i,j \leq p} d(u_i, v_j) \tag{5}$$

Then the following theorem gives that characterization.

Theorem 8
Let f be a rational function mapping Σ^* to $(\Delta^*)^p$. f is p-subsequential iff it has bounded variation (using the semi-metric d'_p on $(\Delta^*)^p$).

Proof
According to the previous theorem the condition is sufficient since:

$$\forall (u,v) \in (\Sigma^*)^2, d_\infty(u,v) \leq d'_p(u,v)$$

Conversely if f is p-subsequential, let $T = (Q, i, F, \Sigma, \Delta, \delta, \sigma, \rho)$ be a p-subsequential transducer representing f, where $\rho = (\rho_1, \ldots, \rho_p)$ is the output function mapping Q to $(\Delta^*)^p$. Let N and M be defined by:

$$N = \max_{q \in F, 1 \leq i,j \leq p} |\rho_i(q)| \ \ and \ \ M = \max_{a \in \Sigma, q \in Q} |\sigma(q,a)| \tag{6}$$

We denote by $Dom(T)$ the set of strings accepted by T. Let $k \geq 0$ and $(u_1, u_2) \in [Dom(T)]^2$ such that $d(u_1, u_2) \leq k$. Then, there exists $u \in \Sigma^*$ such that:

$$u_1 = uv_1, \ \ u_2 = uv_2, \ \ and \ \ |v_1| + |v_2| \leq k \tag{7}$$

Hence,

$$\begin{aligned} f(u_1) &= \{\sigma(i,u)\sigma(\delta(i,u),v_1)\rho_j(\delta(i,u_1)) : 1 \leq j \leq p\} \\ f(u_2) &= \{\sigma(i,u)\sigma(\delta(i,u),v_2)\rho_j(\delta(i,u_2)) : 1 \leq j \leq p\} \end{aligned} \tag{8}$$

Let $K = kM + 2N$. We have:

$$\begin{aligned} d'_p(f(u_1), f(u_2)) &\leq M(|v_1| + |v_2|) + d'_p(\rho(\delta(i,u_1)), \rho(\delta(i,u_2))) \\ &\leq kM + 2N = K \end{aligned}$$

Thus, f has bounded variation using d'_p. This ends the proof of the theorem. ■

2.4 Application to language processing

We briefly mentioned several theoretical and computational properties of sequential and p-subsequential transducers. These devices are used in many areas of computational linguistics. In all those areas, the determinization algorithm can be used to obtain a p-subsequential transducer (Mohri, 1996b), and the minimization algorithm to reduce the size of the p-subsequential transducer used (Mohri, 1994b). The composition, union, and equivalence algorithms for subsequential transducers are also very useful in many applications.

2.4.1 Representation of very large dictionaries. Very large-scale dictionaries can be represented by p-subsequential dictionaries because the number of entries and that of the ambiguities they contain are finite. The corresponding representation offers very fast look-up since then the recognition does not depend on the size of the dictionary but only on that of the input string considered. The minimization algorithm for sequential and p-subsequential transducers allows one to reduce to the minimum the size of these devices. Experiments have shown that one can obtain in a very efficient way these compact and fast look-up representations for large natural language dictionaries. As an example, a French morphological dictionary of about 21.2 Mb can be compiled into a p-subsequential transducer of size 1.3 Mb, in a few minutes (Mohri, 1996b).

2.4.2 Compilation of morphological and phonological rules. Similarly, context-dependent phonological and morphological rules can be represented by finite-state transducers (Kaplan and Kay, 1994). Most phonological and morphological rules correspond to p-subsequential functions. The result of the computation described by Kaplan and Kay (1994) is not necessarily a p-subsequential transducer. But, it can often be determinized using the determinization algorithm for p-subsequentiable transducers. This increases considerably the time efficiency of the transducer. It can be further minimized to reduce its size. These observations can be extended to the case of weighted rewrite rules (Mohri and Sproat, 1996).

2.4.3 Syntax. Finite-state machines are also currently used to represent local syntactic constraints (Silberztein, 1993; Roche, 1993; Karlsson et al., 1995; Mohri, 1994d). Linguists can conveniently introduce local grammar transducers that can be used to disambiguate sentences. The number of local grammars for a given language and even for a specific domain can be large. The local grammar transducers are mostly p-subsequential. The determinization and minimization can then be used to make their use more time efficient and to reduce their size. Since p-subsequential transducers are closed under composition, the result of the composition of all local grammar transducers is a p-subsequential transducer. The equivalence of local grammars can also be tested using the equivalence algorithm for sequential transducers.

For a more detailed overview of the applications of sequential string to string transducers to language processing see (Mohri, 1996a).

Their time and space efficiency suggest that the use of sequential transducers will be more and more extended in natural language processing as well as in other connected areas. In the following, we consider the case of string-to-weight transducers which are also used in many areas of computational linguistics.

3. Power series and subsequential string-to-weight transducers

We consider string-to-weight transducers, namely transducers with input strings and output weights. These transducers are used in various domains such as language modeling, representation of word or phonetic lattices, etc. They are used in the following way: one reads and follows a path corresponding to a given input string and outputs a number obtained by combining the weights along this path. In most applications to natural language processing, the weights are simply added along the path, since they are interpreted as (negative) logarithms of probabilities. In case the transducer is not sequential, that is when it does not have a deterministic input, one proceeds in the same way for all the paths corresponding to the input string. In natural language processing, specifically in speech processing, one keeps the minimum of the weights associated to these paths. This corresponds to the *Viterbi approximation* in speech recognition or in other related areas for which hidden Markov models (HMM's) are used. In all such applications, one looks for the best path, i.e. the path with the minimum weight.

3.1 Definitions

In this section we give the definition of string-to-weight transducers as well as others useful for the presentation of the theorems of the following sections.

In addition to the output weights of the transitions, string-to-weight transducers are provided with initial and output weights. For instance, when used with the input string ab, the transducer of figure 9 outputs: $5+1+2+3=11$, 5 being the initial and 3 the final weight.

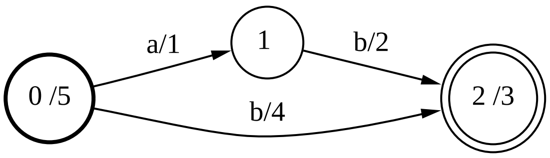

Figure 9
Example of a string-to-weight transducer.

Definition

More formally, a string-to-weight transducer T is defined by $T = (Q, \Sigma, I, F, E, \lambda, \rho)$ with:

- Q a finite set of states,
- Σ the input alphabet,
- $I \subseteq Q$ the set of initial states,
- $F \subseteq Q$ the set of final states,
- $E \subseteq Q \times \Sigma \times \mathcal{R}_+ \times Q$ a finite set of transitions,
- λ the initial weight function mapping I to $\mathcal{R}_+$,
- ρ the final weight function mapping F to $\mathcal{R}_+$.

One can define for T a transition (partial) function δ mapping $Q \times \Sigma$ to 2^Q by:

$$\forall (q,a) \in Q \times \Sigma, \delta(q,a) = \{q' | \exists x \in \mathcal{R}_+ : (q,a,x,q') \in E\}$$

and an output function σ mapping E to $\mathcal{R}_+$ by:

$$\forall t = (p,a,x,q) \in E, \sigma(t) = x$$

A *path* π in T from $q \in Q$ to $q' \in Q$ is a set of successive transitions from q to q': $\pi = ((q_0, a_0, x_0, q_1), \ldots, (q_{m-1}, a_{m-1}, x_{m-1}, q_m))$, with $\forall i \in [0, m-1], q_{i+1} \in \delta(q_i, a_i)$. We can extend the definition of σ to paths by: $\sigma(\pi) = x_0 x_1 \cdots x_{m-1}$.

We denote by $\pi \in q \overset{w}{\leadsto} q'$ the set of paths from q to q' labeled with the input string w. The definition of δ can be extended to $Q \times \Sigma^*$ by:

$$\forall (q,w) \in Q \times \Sigma^*, \delta(q,w) = \{q' : \exists \text{ path } \pi \text{ in } T, \pi \in q \overset{w}{\leadsto} q'\}$$

and to $2^Q \times \Sigma^*$, by:

$$\forall R \subseteq Q, \forall w \in \Sigma^*, \delta(R, w) = \bigcup_{q \in R} \delta(q, w)$$

For $(q, w, q') \in Q \times \Sigma \times Q$ such that there exists a path from q to q' labeled with w, we define $\theta(q, w, q')$ as the minimum of the outputs of all paths from q to q' with input w:

$$\theta(q, w, q') = \min_{\pi \in q \overset{w}{\rightsquigarrow} q'} \sigma(\pi)$$

A *successful path* in T is a path from an initial state to a final state. A string $w \in \Sigma^*$ is accepted by T iff there exists a successful path labeled with w: $w \in \delta(I, w) \cap F$. The output corresponding to an accepted string w is then obtained by taking the minimum of the outputs of all successful paths with input label w:

$$\min_{(i,f) \in I \times F : f \in \delta(i,w)} (\lambda(i) + \theta(i, w, f) + \rho(f))$$

A transducer T is said to be *trim* if all states of T belong to a successful path. String-to-weight transducers clearly realize functions mapping Σ^* to $\mathcal{R}_+$. Since the operations we need to consider are addition and $\min$, and since $(\mathcal{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ is a semiring[2], we call these functions *formal power series*. We adopt the terminology and notation used in formal language theory (Berstel and Reutenauer, 1988; Kuich and Salomaa, 1986; Salomaa and Soittola, 1978):

- the image by a formal power series S of a string w is denoted by (S, w) and called the *coefficient* of w in S,
- the notation $S = \sum_{w \in \Sigma^*} (S, w) w$ is then used to define a power series by its coefficients,
- The *support* of S is the language defined by:

$$supp(S) = \{w \in \Sigma^* : (S, w) \neq \infty\}$$

The fundamental theorem of Schützenberger (1961), analogous of the Kleene's theorem for formal languages, states that a formal power series S is *rational* iff it is recognizable, that is realizable by a string-to-weight transducer. The semiring $(\mathcal{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ used in many optimization problems is called the *tropical semiring*[3]. So, the functions we consider here are more precisely rational power series over the tropical semiring.

A string-to-weight transducer T is said to be *unambiguous* if for any given string w there exists at most one successful path labeled with w.

In the following, we examine, more specifically, efficient string-to-weight transducers: *subsequential transducers*. A transducer is said to be *subsequential* if its input is deterministic, that is if at any state there exists at most one outgoing transition labeled with a given element of the input alphabet Σ. Subsequential string-to-weight transducers are

2 Recall that a semiring is essentially a ring that may lack *negation*, namely in which the first operation does not necessarily admit inverse. $(\mathcal{R}, +, \cdot, 0, 1)$, where 0 and 1 are respectively the identity elements for $+$ and $\cdot$, or for any non empty set E, $(E, \cup, \cap, \emptyset, E)$, where $\emptyset$ and E are respectively the identity elements for $\cup$ and $\cap$, are other examples of semirings.

3 This terminology is often used more specifically when the set is restricted to natural integers $(\mathcal{N} \cup \{\infty\}, \min, +, \infty, 0)$.

sometimes called *weighted automata*, or *weighted acceptors*, or *probabilistic automata*, or *distance automata*. Our terminology is meant to favor the functional view of these devices, which is the view that we consider here. Not all string-to-weight transducers are subsequential but we define an algorithm to determinize non subsequential transducers when possible.

Definition
More formally a string-to-weight subsequential transducer $\tau = (Q, i, F, \Sigma, \delta, \sigma, \lambda, \rho)$ is an 8-tuple, where:

- Q is the set of its states,
- $i \in Q$ its initial state,
- $F \subseteq Q$ the set of final states,
- Σ the input alphabet,
- δ the transition function mapping $Q \times \Sigma$ to Q, δ can be extended as in the string case to map $Q \times \Sigma^*$ to Q,
- σ the output function which maps $Q \times \Sigma$ to $\mathcal{R}_+$, σ can also be extended to $Q \times \Sigma^*$,
- $\lambda \in \mathcal{R}_+$ the initial weight,
- ρ the final weight function mapping F to $\mathcal{R}_+$.

A string $w \in \Sigma^*$ is accepted by a subsequential transducer T if there exists $f \in F$ such that $\delta(i, w) = f$. The output associated to w is then: $\lambda + \sigma(i, w) + \rho(f)$.

We will use the following definition for characterizing the transducers that admit determinization.

Definition
Two states q and q' of a string-to-weight transducer $T = (Q, I, F, \Sigma, \delta, \sigma, \lambda, \rho)$, not necessarily subsequential, are said to be *twins* if:

$$\forall (u, v) \in (\Sigma^*)^2, (\{q, q'\} \subset \delta(I, u), q \in \delta(q, v), q' \in \delta(q', v)) \Rightarrow \theta(q, v, q) = \theta(q', v, q') \tag{9}$$

In other words, q and q' are twins if, when they can be reached from the initial state by the same string u, the minimum outputs of loops at q and q' labeled with any string v are identical. We say that T has the *twins property* when any two states q and q' of T are twins. Notice that according to the definition, two states that do not have cycles with the same string v are twins. In particular, two states that do not belong to any cycle are necessarily twins. Thus, an acyclic transducer has the twins property.

In the following section we consider subsequential power series in the tropical semiring, that is functions that can be realized by subsequential string-to-weight transducers. Many rational power series defined on the tropical semiring considered in practice are subsequential, in particular acyclic transducers represent subsequential power series.

We introduce a theorem giving an intrinsic characterization of subsequential power series irrespective of the transducer realizing them. We then present an algorithm that allows one to determinize some string-to-weight transducers. We give a very general presentation of the algorithm since it can be used with many other semirings. In particular, the algorithm can also be used with string-to-string transducers and with transducers whose output labels are pairs of strings and weights.

We then use the twins property to define a set of transducers to which the determinization algorithm applies. We give a characterization of unambiguous transducers admitting determinization. We then use this characterization to define an algorithm to test if a given transducer can be determinized.

We also present a minimization algorithm which applies to subsequential string-to-weight transducers. The algorithm is very efficient. The determinization algorithm can also be used in many cases to minimize a subsequential transducer. We describe that algorithm and give the related proofs in the appendix.

3.2 Characterization of subsequential power series

Recall that one can define a metric on Σ^* by:

$$d(u, v) = |u| + |v| - 2|u \wedge v| \tag{10}$$

where we denote by $u \wedge v$ the longest common prefix of two strings u and v in Σ^*. The definition we gave for subsequential power series depends on the transducers representing them. The following theorem gives an intrinsic characterization of unambiguous subsequential power series[4].

Theorem 9

Let S be a rational power series defined on the tropical semiring. If S is subsequential then S has bounded variation. When S can be represented by an unambiguous weighted automaton, the converse holds as well.

Proof

Assume that S is subsequential. Let $\tau = (Q, i, F, \Sigma, \delta, \sigma, \lambda, \rho)$ be a subsequential transducer. δ denotes the transition function associated with τ, σ its output function, and λ and ρ the initial and final weight functions. Let L be the maximum of the lengths of all output labels of T:

$$L = \max_{(q,a) \in Q \times \Sigma} |\sigma(q, a)| \tag{11}$$

and R the upper bound of all output differences at final states:

$$R = \max_{(q,q') \in F^2} |\rho(q) - \rho(q')| \tag{12}$$

and define M as $M = L + R$. Let (u_1, u_2) be in $(\Sigma^*)^2$. By definition of d, there exists $u \in \Sigma^*$ such that:

$$u_1 = uv_1, \;\; u_2 = uv_2, \; and \; |v_1| + |v_2| = d(u_1, u_2) \tag{13}$$

Hence,

$$\begin{aligned} \sigma(i, u_1) &= \sigma(i, u) + \sigma(\delta(i, u), v_1) \\ \sigma(i, u_2) &= \sigma(i, u) + \sigma(\delta(i, u), v_2) \end{aligned}$$

4 This is an extension of the characterization theorem of Choffrut (1978) for string-to-string functions. The extension is not straightforward because the length of an output string is a natural integer. Here we deal with real numbers.

Since

$$|\sigma(\delta(i,u),v_1)-\sigma(\delta(i,u),v_2)| \quad \leq \quad L\cdot(|v_1|+|v_2|)=L\cdot d(u_1,u_2)$$

and

$$|\rho(\delta(i,u_1))-\rho(\delta(i,u_2))| \quad \leq \quad R$$

we have

$$|\lambda+\sigma(i,u_1)+\rho(\delta(i,u_1))-\lambda+\sigma(i,u_2)+\rho(\delta(i,u_2))| \quad \leq \quad L\cdot d(u_1,u_2)+R$$

Notice that if $u_1 \neq u_2$, $R \leq R\cdot d(u_1,u_2)$. Thus

$$|\lambda+\sigma(i,u_1)+\rho(\delta(i,u_1))-\lambda+\sigma(i,u_2)+\rho(\delta(i,u_2))| \quad \leq \quad (L+R)\cdot d(u_1,u_2)$$

Therefore:

$$\forall(u_1,u_2)\in(\Sigma^*)^2,\ |S(u_1)-S(u_2)| \quad \leq \quad M\cdot d(u_1,u_2) \tag{14}$$

This proves that S is M-Lipschitzian[5] and *a fortiori* that it has bounded variation.

Conversely, suppose that S has bounded variation. Since S is rational, according to the theorem of Schützenberger (1961) it is recognizable and therefore there exists a string-to-weight transducer $\tau = (Q, I, F, \Sigma, \delta, \sigma, \lambda, \rho)$ realizing S. Assume that τ is a trim unambiguous transducer.

We describe in the next sections a determinization algorithm. We show that this algorithm applies to any transducer that has the twins property. Thus, in order to show that S is subsequentiable, it is sufficient to show that τ has the twins property.

Consider two states q and q' of τ and let $(u,v)\in(\Sigma^*)^2$ be such that:

$$\{q,q'\}\subset\delta(I,u), q\in\delta(q,v), q'\in\delta(q',v)$$

Since τ is trim there exists $(w,w')\in(\Sigma^*)^2$ such that $\delta(q,w)\cap F\neq\emptyset$ and $\delta(q,w')\cap F\neq\emptyset$. Notice that

$$\forall k\geq 0, d(uv^kw, uv^kw') = d(w,w')$$

Thus, since S has bounded variation

$$\exists K\geq 0, \forall k\geq 0, \quad |S(uv^kw)-S(uv^kw')|\leq K$$

Since τ is unambiguous, there is only one path from I to F corresponding to uv^kw (resp. uv^kw'). We have:

$$\begin{aligned} S(uv^kw) &= \theta(I,uw,F)+k\theta(q,v,q) \\ S(uv^kw') &= \theta(I,uw',F)+k\theta(q',v,q') \end{aligned}$$

Hence

$$\begin{array}{ll} & \exists K\geq 0, \forall k\geq 0, \quad |(\theta(I,uw,F)-\theta(I,uw',F))+k(\theta(q,v,q)-\theta(q',v,q'))|\leq K \\ \Longrightarrow & \theta(q,v,q)-\theta(q',v,q')=0 \end{array}$$

Thus τ has the twins property. This ends the proof of the theorem. ■

5 This implies in particular that the subsequential power series over the tropical semiring define continuous functions for the topology induced by the metric d on Σ^*. Also this shows that in the theorem one can replace *has bounded variation* by *is Lipschitzian*.

3.3 General determinization algorithm for power series

We describe in this section an algorithm for constructing a subsequential transducer $\tau_2 = (Q_2, i_2, F_2, \Sigma, \delta_2, \sigma_2, \lambda_2, \rho_2)$ equivalent to a given non-subsequential one $\tau_1 = (Q_1, \Sigma, I_1, F_1, E_1, \lambda_1, \rho_1)$. The algorithm extends to the case of transducers outputting weights our determinization algorithm for string-to-string transducers representing p-subsequential functions (Mohri, 1994c).

Figure 10 gives the pseudocode of the algorithm. We present the algorithm in the general case of a semiring $(S, \oplus, \odot, \bar{0}, \bar{1})$ on which the transducer τ_1 is defined. Indeed, the algorithm we are describing here applies as well to transducers representing power series defined on many other semirings[6]. We describe the algorithm in the case of the tropical semiring. For the tropical semiring, one can replace $\oplus$ by min and $\odot$ by $+$ in the pseudocode[7] of figure 10.

PowerSeriesDeterminization(τ_1, τ_2)
1 $F_2 \leftarrow \emptyset$
2 $\lambda_2 \leftarrow \bigoplus_{i \in I_1} \lambda_1(i)$
3 $i_2 \leftarrow \bigcup_{i \in I_1} \{(i, \lambda_2^{-1} \odot \lambda_1(i))\}$
4 $Q \leftarrow \{i_2\}$
5 **while** $Q \neq \emptyset$
6 **do** $q_2 \leftarrow head[Q]$
7 **if** (there exists $(q, x) \in q_2$ such that $q \in F_1$)
8 **then** $F_2 \leftarrow F_2 \cup \{q_2\}$
9 $\rho_2(q_2) \leftarrow \bigoplus_{q \in F_1, (q,x) \in q_2} x \odot \rho_1(q)$
10 **for** each a such that $\Gamma(q_2, a) \neq \emptyset$
11 **do** $\sigma_2(q_2, a) \leftarrow \bigoplus_{(q,x) \in \Gamma(q_2,a)} [x \odot \bigoplus_{t=(q,a,\sigma_1(t),n_1(t)) \in E_1} \sigma_1(t)]$
12 $\delta_2(q_2, a) \leftarrow \bigcup_{q' \in \nu(q_2,a)} \{(q', \bigoplus_{(q,x,t) \in \gamma(q_2,a), n_1(t)=q'} [\sigma_2(q_2, a)]^{-1} \odot x \odot \sigma_1(t)\}$
13 **if** ($\delta_2(q_2, a)$ is a new state)
14 **then** ENQUEUE($Q, \delta_2(q_2, a)$)
15 DEQUEUE(Q)

Figure 10
Algorithm for the determinization of a transducer τ_1 representing a power series defined on the semiring $(S, \oplus, \odot, \bar{0}, \bar{1})$.

The algorithm is similar to the powerset construction used for the determinization of automata. However, since the outputs of two transitions bearing the same input label might differ, one can only output the minimum of these outputs in the resulting transducer, therefore one needs to keep track of the residual weights. Hence, the subsets q_2 that we consider here are made of pairs (q, x) of states and weights.

The initial weight λ_2 of τ_2 is the minimum of all the initial weights of τ_1 (line 2). The initial state i_2 is a subset made of pairs (i, x), where i is an initial state of τ_1, and $x = \lambda_1(i) - \lambda_2$ (line 3). We use a queue Q to maintain the set of subsets q_2 yet to be

6 In particular, the algorithm also applies to string to string subsequentiable transducers and to transducers that output pairs of strings and weights. We will come back to this point later.
7 Similarly, λ_2^{-1} should be interpreted as $-\lambda$, and $[\sigma_2(q_2, a)]^{-1}$ as $-\sigma_2(q_2, a)$.

examined, as in the classical powerset construction[8]. Initially, Q contains only the subset i_2. The subsets q_2 are the states of the resulting transducer. q_2 is a final state of τ_2 iff it contains at least one pair (q, x), with q a final state of τ_1 (lines 7-8). The final output associated to q_2 is then the minimum of the final outputs of all the final states in q_2 combined with their respective residual weight (line 9).

For each input label a such that there exists at least one state q of the subset q_2 admitting an outgoing transition labeled with a, one outgoing transition leaving q_2 with the input label a is constructed (lines 10-14). The output $\sigma_2(q_2, a)$ of this transition is the minimum of the outputs of all the transitions with input label a that leave a state in the subset q_2, when combined with the residual weight associated to that state (line 11).

The destination state $\delta_2(q_2, a)$ of the transition leaving q_2 is a subset made of pairs (q', x'), where q' is a state of τ_1 that can be reached by a transition labeled with a, and x' the corresponding residual weight (line 12). x' is computed by taking the minimum of all the transitions with input label a that leave a state q of q_2 and reach q', when combined with the residual weight of q minus the output weight $\sigma_2(q_2, a)$. Finally, $\delta_2(q_2, a)$ is enqueued in Q iff it is a new subset.

We denote by $n_1(t)$ the destination state of a transition $t \in E_1$. Hence $n_1(t) = q'$, if $t = (q, a, x, q') \in E_1$. The sets $\Gamma(q_2, a)$, $\gamma(q_2, a)$, and $\nu(q_2, a)$ used in the algorithm are defined by:

- $\Gamma(q_2, a) = \{(q, x) \in q_2 : \exists t = (q, a, \sigma_1(t), n_1(t)) \in E_1\}$
- $\gamma(q_2, a) = \{(q, x, t) \in q_2 \times E_1 : t = (q, a, \sigma_1(t), n_1(t)) \in E_1\}$
- $\nu(q_2, a) = \{q' : \exists (q, x) \in q_2, \exists t = (q, a, \sigma_1(t), q') \in E_1\}$

$\Gamma(q_2, a)$ denotes the set of pairs (q, x), elements of the subset q_2, having transitions labeled with the input a. $\gamma(q_2, a)$ denotes the set of triples (q, x, t) where (q, x) is a pair in q_2 such that q admits a transition with input label a. $\nu(q_2, a)$ is the set of states q' that can be reached by transitions labeled with a from the states of the subset q_2.

The algorithm is illustrated in figures 11 and 12. Notice that the input ab admits several outputs in μ_1: $\{1 + 1 = 2, 1 + 3 = 4, 3 + 3 = 6, 3 + 5 = 8\}$. Only one of these outputs (2, the smallest) is kept in the determinized transducer μ_2 since in the tropical semiring one is only interested in the minimum outputs for any given string.

Notice that several transitions might reach the same state with *a priori* different residual weights. Since one is only interested in the best path, namely the path corresponding to the minimum weight, one can keep the minimum of these weights for a given state element of a subset (line 11 of the algorithm of figure 10). We give in the next section a set of transducers τ_1 for which the determinization algorithm terminates. The following theorem shows the correctness of the algorithm when it terminates.

Theorem 10
Assume that the determinization algorithm terminates, then the resulting transducer τ_2 is equivalent to τ_1.

Proof
We denote by $\theta_1(q, w, q')$ the minimum of the outputs of all paths from q to q'. By construction we have:

$$\lambda_2 = \min_{i_1 \in I_1} \lambda_1(i_1)$$

8 The algorithm works with any queue discipline chosen for Q.

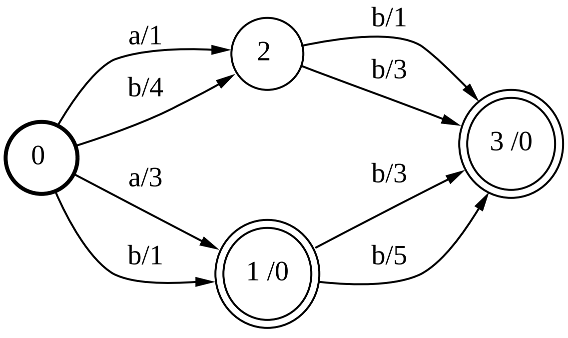

Figure 11
Transducer μ_1 representing a power series defined on $(\mathcal{R}_+ \cup \{\infty\}, \min, +)$.

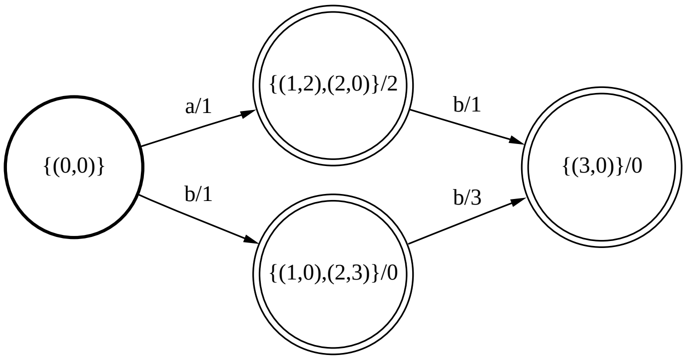

Figure 12
Transducer μ_2 obtained by power series determinization of μ_1.

We define the *residual output* associated to q in the subset $\delta_2(i_2, w)$ as the weight $c(q, w)$ associated to the pair containing q in $\delta_2(i_2, w)$. It is not hard to show by induction on $|w|$ that the subsets constructed by the algorithm are the sets $\delta_2(i_2, w)$, $w \in \Sigma^*$, such that:

$$\begin{aligned}
\forall w \in \Sigma^*,\ \delta_2(i_2, w) &= \bigcup_{q \in \delta_1(I_1, w)} \{(q, c(q, w)\} \qquad (15)\\
c(q, w) &= \min_{i_1 \in I_1} (\lambda_1(i_1) + \theta_1(i_1, w, q)) - \sigma_2(i_2, w) - \lambda_2\\
\sigma_2(i_2, w) &= \min_{q \in \delta_1(I_1, w)} (\lambda_1(i_1) + \theta_1(i_1, w, q)) - \lambda_2
\end{aligned}$$

Notice that the size of a subset never exceeds $|Q_1|$: card$(\delta_2(i_2, w)) \leq |Q_1|$. A state q belongs at most to one pair of a subset since for all paths reaching q, only the minimum of the residual outputs is kept. Notice also that, by definition of $\min$, in any subset there exists at least one state q with a residual output $c(q, w)$ equal to 0.

A string w is accepted by τ_1 iff there exists $q \in F_1$ such that $q \in \delta_1(I_1, w)$. Using the equations 15, it is accepted iff $\delta_2(i_2, w)$ contains a pair $(q, c(q, w))$ with $q \in F_1$. This is exactly the definition of the final states F_2 (line 7). So τ_1 and τ_2 accept the same set of strings.

Let $w \in \Sigma^*$ be a string accepted by τ_1 and τ_2. The definition of ρ_2 in the algorithm of figure 10, line 9, gives:

$$\rho_2(\delta_2(i_2,w)) = \min_{q\in\delta_1(I_1,w)\cap F_1} \rho_1(q) + \min_{i_1\in I_1}(\lambda_1(i_1)+\theta_1(i_1,w,q)) - \sigma_2(i_2,w) - \lambda_2 \quad (16)$$

Thus, if we denote by S the power series realized by τ_1, we have:

$$\rho_2(\delta_2(i_2,w)) = (S,w) - \sigma_2(i_2,w) - \lambda_2 \quad (17)$$

Hence: $\lambda_2 + \sigma_2(i_2,w) + \rho_2(\delta_2(i_2,w)) = (S,w)$. ■

The power series determinization algorithm is equivalent to the usual determinization of automata when the initial weight, the final weights, and all output labels are equal to 0. The subsets considered in the algorithm are then exactly those obtained in the powerset determinization of automata, all residual outputs $c(q,w)$ being equal to 0.

Both space and time complexity of the determinization algorithm for automata are exponential. There are minimal deterministic automata with exponential size with respect to an equivalent non-deterministic one. *A fortiori* the complexity of the determinization algorithm in the weighted case we just described is also exponential. However, in some cases in which the degree of nondeterminism of the initial transducer is high, the determinization algorithm turns out to be fast and the resulting transducer has less states than the initial one. We present such cases which appear in speech recognition in the last section. We also present a minimization algorithm which allows one to reduce the size of subsequential transducers representing power series.

The complexity of the application of subsequential transducers is linear in the size of the string to which it applies. This property makes it worthwhile to use the power series determinization in order to speed up the application of transducers. Not all transducers can be determinized using the power series determinization. In the following section we define a set of transducers that admit determinization, and characterize unambiguous transducers that admit the application of the algorithm.

Determinization does not apply to all transducers. It is therefore important to be able to test the determinizability of a transducer. We will present in the next section an algorithm to test this property in the case of unambiguous trim transducers.

The proofs of some of the theorems in the next two sections are complex, they can be skipped on first reading.

3.4 Determinizable transducers

There are transducers with which determinization does not halt. It then generates an infinite number of subsets. We define *determinizable* transducers as those transducers with which the algorithm terminates. We first show that a large set of transducers admit determinization. We then give a characterization of unambiguous transducers admitting determinization. In the following, the states of the transducers considered will be assumed to be accessible from the initial one.

The following lemma will be useful in the proof of the theorems.

Lemma 1

Let $T=(Q,\Sigma,I,F,E,\lambda,\rho)$ be a string-to-weight transducer, $\pi \in p \overset{w}{\leadsto} q$ a path in T from the state $p\in Q$ to $q \in Q$, and $\pi' \in p' \overset{w}{\leadsto} q'$ a path from $p'\in Q$ to $q'\in Q$ both labeled with the input string $w\in\Sigma^*$. Assume that the lengths of π and π' are greater than $|Q|^2-1$, then there exist strings u_1, u_2, u_3 in Σ^*, and states p_1, p_2, p_1', and p_2' such that $|u_2|>0$, $u_1u_2u_3=w$ and such that π and π' be factored in the following way:

$$\pi \in p \overset{u_1}{\leadsto} p_1 \overset{u_2}{\leadsto} p_1 \overset{u_3}{\leadsto} q$$

$$\pi' \in p' \overset{u_1}{\leadsto} p'_1 \overset{u_2}{\leadsto} p'_1 \overset{u_3}{\leadsto} q'$$

Proof
The proof is based on the use of a *cross product* of two transducers. Given two transducers $T_1 = (Q_1, \Sigma, I_1, F_1, E_1, \lambda_1, \rho_1)$ and $T_2 = (Q_2, \Sigma, I_2, F_2, E_2, \lambda_2, \rho_2)$, we define the cross product of T_1 and T_2 as the transducer:

$$T_1 \times T_2 = (Q_1 \times Q_2, \Sigma, I_1 \times I_2, F_1 \times F_2, E, \lambda, \rho)$$

with outputs in $\mathcal{R}_+ \times \mathcal{R}_+$ such that $t = ((q_1, q_2), a, (x_1, x_2), (q'_1, q'_2)) \in Q_1 \times \Sigma \times \mathcal{R}_+ \times \mathcal{R}_+ \times Q_2$ is a transition of $T_1 \times T_2$, namely $t \in E$, iff $(q_1, a, x_1, q'_1) \in E_1$ and $(q_2, a, x_2, q'_2) \in E_2$. We also define λ and ρ by: $\forall (i_1, i_2) \in I_1 \times I_2, \lambda(i_1, i_2) = (\lambda_1(i_1), \lambda_2(i_2))$, $\forall (f_1, f_2) \in F_1 \times F_2, \rho(f_1, f_2) = (\rho_1(f_1), \rho_2(f_2))$.

Consider the cross product of T with itself, $T \times T$. Let π and π' be two paths in T with lengths greater than $|Q|^2 - 1$, ($m > |Q|^2 - 1$):

$$\pi = ((p = q_0, a_0, x_0, q_1), \ldots, (q_{m-1}, a_{m-1}, x_{m-1}, q_m = q))$$

$$\pi' = ((p' = q'_0, a_0, x'_0, q'_1), \ldots, (q'_{m-1}, a_{m-1}, x'_{m-1}, q'_m = q'))$$

then:

$$\Pi = (((q_0, q'_0), a_0, (x_0, x'_0), (q_1, q'_1)), \ldots, ((q_{m-1}, q'_{m-1}), a_{m-1}, (x_{m-1}, x'_{m-1}), (q_m, q'_m)))$$

is a path in $T \times T$ with length greater than $|Q|^2 - 1$. Since $T \times T$ has exactly $|Q|^2$ states, Π admits at least one cycle at some state (p_1, p'_1) labeled with a non empty input string u_2. This shows the existence of the factorization above and proves the lemma. ■

Theorem 11
Let $\tau_1 = (Q_1, \Sigma, I_1, F_1, E_1, \lambda_1, \rho_1)$ be a string-to-weight transducer defined on the tropical semiring. If τ_1 has the twins property then it is determinizable.

Proof
Assume that τ has the twins property. If the determinization algorithm does not halt, there exists at least one subset of 2^Q, $\{q_0, \ldots, q_m\}$, such that the algorithm generates an infinite number of distinct weighted subsets $\{(q_0, c_0), \ldots, (q_m, c_m)\}$.

Then we have necessarily $m > 1$. Indeed, we mentioned previously that in any subset there exists at least one state q_i with a residual output $c_i = 0$. If the subset contains only one state q_0, then $c_0 = 0$. So there cannot be an infinite number of distinct subsets $\{(q_0, c_0)\}$.

Let $A \subseteq \Sigma^*$ be the set of strings w such that the states of $\delta_2(i_2, w)$ be $\{q_0, \ldots, q_m\}$. We have: $\forall w \in A, \delta_2(i_2, w) = \{(q_0, c(q_0, w)), \ldots, (q_m, c(q_m, w))\}$. Since A is infinite, and since in each weighted subset there exists a null residual output, there exist i_0, $0 \leq i_0 \leq m$, such that $c(q_{i_0}, w) = 0$ for an infinite number of strings $w \in A$. Without loss of generality we can assume that $i_0 = 0$.

Let $B \subseteq A$ be the infinite set of strings w for which $c(q_0, w) = 0$. Since the number of subsets $\{(q_0, c(q_0, w)), \ldots, (q_m, c(q_m, w))\}$, $w \in B$, is infinite, there exists j, $0 < j \leq m$, such that $c(q_j, w)$ be distinct for an infinite number of strings $w \in B$. Without loss of generality we can assume $j = 1$.

Let $C \subseteq B$ be an infinite set of strings w with $c(q_1, w)$ all distinct. Define $R(q_0, q_1)$ to be the finite set of differences of the weights of paths leading to q_0 and q_1 labeled with the same string w, $|w| \leq |Q_1|^2 - 1$:

$$R(q_0,q_1)=\{(\lambda(i_1)+\sigma(\pi_1))-(\lambda(i_0)+\sigma(\pi_0)) : \pi_0 \in i_0 \overset{w}{\leadsto} q_0, \pi_1 \in i_1 \overset{w}{\leadsto} q_1, \\ i_0 \in I, i_1 \in I, |w| \leq |Q_1|^2 - 1\}$$

We will show that $\{c(q_1, w) : w \in C\} \subseteq R(q_0, q_1)$. This will yield a contradiction with the infinity of C, and will therefore prove that the algorithm terminates.

Let $w \in C$, and consider a shortest path π_0 from a state $i_0 \in I$ to q_0 labeled with the input string w and with total cost $\sigma(\pi_0)$. Similarly, consider a shortest path π_1 from $i_1 \in I$ to q_1 labeled with the input string w and with total cost $\sigma(\pi_1)$. By definition of the subset construction we have: $(\lambda(i_1) + \sigma(\pi_1)) - (\lambda(i_0) + \sigma(\pi_0)) = c(q_1, w)$. Assume that $|w| > |Q_1|^2 - 1$. Using the lemma 1, there exists a factorization of π_0 and π_1 of the type:

$$\pi_0 \in i_0 \overset{u_1}{\leadsto} p_0 \overset{u_2}{\leadsto} p_0 \overset{u_3}{\leadsto} q_0$$

$$\pi_1 \in i_1 \overset{u_1}{\leadsto} p_1 \overset{u_2}{\leadsto} p_1 \overset{u_3}{\leadsto} q_1$$

with $|u_2| > 0$. Since p_0 and p_1 are twins, $\theta_1(p_0, u_2, p_0) = \theta_1(p_1, u_2, p_1)$. Define π'_0 and π'_1 by:

$$\pi'_0 \in i_0 \overset{u_1}{\leadsto} p_0 \overset{u_3}{\leadsto} q_0$$

$$\pi'_1 \in i_1 \overset{u_1}{\leadsto} p_1 \overset{u_3}{\leadsto} q_1$$

Since the cycles at p_0 and p_1 in the factorization are sub-paths of the shortest paths π_0 and π_1, they are shortest cycles labeled with u_2. Thus, we have: $\sigma(\pi_0) = \sigma(\pi'_0) + \theta_1(p_0, u_2, p_0)$ and $\sigma(\pi_1) = \sigma(\pi'_1) + \theta_1(p_1, u_2, p_1)$, and: $(\lambda(i_1) + \sigma(\pi'_1)) - (\lambda(i_0) + \sigma(\pi'_0)) = c(q_1, w)$. By induction on $|w|$, we can therefore find paths Π_0 and Π_1 from i_0 to q_0 (resp. i_1 to q_1) extracted from π_0 and π_1 by deletion of some cycles, with length less or equal to $|Q_1|^2 - 1$, and such that $(\lambda(i_1) + \sigma(\Pi_1)) - (\lambda(i_0) + \sigma(\Pi_0)) = c(q_1, w)$. Since $\sigma(\Pi_1) - \sigma(\Pi_0) \in R(q_0, q_1)$, $c(q_1, w) \in R(q_0, q_1)$ and C is finite. This ends the proof of the theorem. ■

There are transducers that do not have the twins property and that are still determinizable. To characterize such transducers one needs more complex conditions. We do not describe those conditions here. However, in the case of trim unambiguous transducers, the twins property provides a characterization of determinizable transducers.

Theorem 12
Let $\tau_1 = (Q_1, \Sigma, I_1, F_1, E_1, \lambda_1, \rho_1)$ be a trim unambiguous string-to-weight transducer defined on the tropical semiring. Then τ_1 is determinizable iff it has the twins property.

Proof
According to the previous theorem, if τ_1 has the twins property, then it is determinizable. Assume now that τ does not have the twins property, then there exist at least two states q and q' in Q that are not twins. There exists $(u, v) \in \Sigma^*$ such that: $(\{q, q'\} \subset \delta_1(I, u), q \in \delta_1(q, v), q' \in \delta_1(q', v))$ and $\theta_1(q, v, q) \neq \theta_1(q', v, q')$. Consider the weighted subsets $\delta_2(i_2, uv^k)$, with $k \in \mathcal{N}$, constructed by the determinization algorithm. A subset $\delta_2(i_2, uv^k)$ contains the pairs $(q, c(q, uv^k))$ and $(q', c(q', uv^k))$. We will show that these subsets are all distinct. This will prove that the determinization algorithm does not terminate if τ_1 does not have the twins property.

Since τ_1 is a trim unambiguous transducer, there exits only one path in τ_1 from I to q or to q' with input string u. Similarly, the cycles at q and q' labeled with v are unique. Thus, there exist $i \in I$ and $i' \in I$ such that:

$$\begin{aligned} \forall k \in \mathcal{N}, \quad c(q, uv^k) = &\quad \lambda_1(i) + \theta_1(i, u, q) + k\theta_1(q, v, q) - \sigma_2(i_2, uv^k) - \lambda_2 \qquad (18) \\ \forall k \in \mathcal{N}, \quad c(q', uv^k) = &\quad \lambda_1(i') + \theta_1(i', u, q') + k\theta_1(q', v, q') - \sigma_2(i_2, uv^k) - \lambda_2 \end{aligned}$$

Let λ and θ be defined by:

$$\begin{array}{rcl} \lambda & = & (\lambda_1(i') - \lambda_1(i)) + (\theta_1(i', u, q') - \theta_1(i, u, q)) \\ \theta & = & \theta_1(q', v, q') - \theta_1(q, v, q) \end{array} \quad (19)$$

We have:

$$\forall k \in \mathcal{N}, c(q', uv^k) - c(q, uv^k) = \lambda + k\theta \quad (20)$$

Since $\theta \neq 0$, the equations 20 show that the subsets $\delta_2(i_2, uv^k)$ are all distinct. ■

3.5 Test of determinizability

The characterization of determinizable transducers provided by the theorem 12 leads to the definition of an algorithm for testing the determinizability of trim unambiguous transducers. Before describing the algorithm, we introduce a lemma which shows that it suffices to examine a finite number of paths to test the twins property.

Lemma 2

Let $\tau_1 = (Q_1, \Sigma, I_1, F_1, E_1, \lambda_1, \rho_1)$ be a trim unambiguous string-to-weight transducer defined on the tropical semiring. τ_1 has the twins property iff $\forall (u, v) \in (\Sigma^*)^2, |uv| \leq 2|Q_1|^2 - 1$,

$$(\{q, q'\} \subset \delta_1(I, u), q \in \delta_1(q, v), q' \in \delta_1(q', v)) \Rightarrow \theta_1(q, v, q) = \theta_1(q', v, q') \quad (21)$$

Proof

Clearly if τ_1 has the twins property, then 21 holds. Conversely, we prove that if 21 holds, then it also holds for any $(u, v) \in (\Sigma^*)^2$, by induction on $|uv|$. Our proof is similar to that of Berstel (1979) for string-to-string transducers. Consider $(u, v) \in (\Sigma^*)^2$ and $(q, q') \in |Q_1|^2$ such that: $\{q, q'\} \subset \delta_1(I, u), q \in \delta_1(q, v), q' \in \delta_1(q', v)$. Assume that $|uv| > 2|Q_1|^2 - 1$ with $|v| > 0$. Then either $|u| > |Q_1|^2 - 1$ or $|v| > |Q_1|^2 - 1$.

Assume that $|u| > |Q_1|^2 - 1$. Since τ_1 is a trim unambiguous transducer there exists a unique path π in τ_1 from $i \in I$ to q labeled with the input string u, and a unique path π' from $i' \in I$ to q'. In view of lemma 2, there exist strings u_1, u_2, u_3 in Σ^*, and states p_1, p_2, p_1', and p_2' such that $|u_2| > 0$, $u_1 u_2 u_3 = u$ and such that π and π' be factored in the following way:

$$\pi \in i \overset{u_1}{\leadsto} p_1 \overset{u_2}{\leadsto} p_1 \overset{u_3}{\leadsto} q$$

$$\pi' \in i' \overset{u_1}{\leadsto} p_1' \overset{u_2}{\leadsto} p_1' \overset{u_3}{\leadsto} q'$$

Since $|u_1 u_3 v| < |uv|$, by induction $\theta_1(q, v, q) = \theta_1(q', v, q')$.

Next, assume that $|v| > |Q_1|^2 - 1$. Then according to lemma 1, there exist strings v_1, v_2, v_3 in Σ^*, and states q_1, q_2, q_1', and q_2' such that $|v_2| > 0$, $v_1 v_2 v_3 = v$ and such that π and π' be factored in the following way:

$$\pi \in q \overset{v_1}{\leadsto} q_1 \overset{v_2}{\leadsto} q_1 \overset{v_3}{\leadsto} q$$

$$\pi' \in q' \overset{v_1}{\leadsto} q_1' \overset{v_2}{\leadsto} q_1' \overset{v_3}{\leadsto} q'$$

Since $|uv_1 v_3| < |uv|$, by induction, $\theta_1(q, v_1 v_3, q) = \theta_1(q', v_1 v_3, q')$. Similarly, since $|uv_1 v_2| < |uv|$, $\theta_1(q_1, v_2, q_1) = \theta_1(q_1', v_2, q_1')$. τ_1 is a trim unambiguous transducer, so:

$$\theta_1(q, v, q) = \theta_1(q, v_1 v_3, q) + \theta_1(q_1, v_2, q_1)$$

$$\theta_1(q', v, q') = \theta_1(q', v_1 v_3, q') + \theta_1(q_1', v_2, q_1')$$

Thus, $\theta_1(q, v, q) = \theta_1(q', v, q')$. This completes the proof of the lemma. ■

Theorem 13
Let $\tau_1 = (Q_1, \Sigma, I_1, F_1, E_1, \lambda_1, \rho_1)$ be a trim unambiguous string-to-weight transducer defined on the tropical semiring. There exists an algorithm to test the determinizability of τ_1.

Proof
According to the theorem 12, testing the determinizability of τ_1 is equivalent to testing for the twins property. We define an algorithm to test this property. Our algorithm is close to that of Weber and Klemm (1995) for testing the sequentiability of string-to-string transducers. It is based on the construction of an automaton $A = (Q, I, F, E)$ similar to the cross product of τ_1 with itself.

Let $K \subset \mathcal{R}$ be the finite set of real numbers defined by:

$$K = \{\sum_{i=1}^{k}(\sigma(t'_i) - \sigma(t_i)) : 1 \leq k \leq 2|Q_1|^2 - 1, \forall i \leq k \; (t_i, t'_i) \in E_1^2\}$$

We define A by the following:

- The set of states of A is defined by $Q = Q_1 \times Q_1 \times K$,
- The set of initial states by $I = I_1 \times I_1 \times \{0\}$,
- The set of final states by $F = F_1 \times F_1 \times K$,
- The set of transitions by:
$E = \{((q_1, q_2, c), a, (q'_1, q'_2, c')) \in Q \times \Sigma \times Q :$
$\exists \; (q_1, a, x, q_2) \in E_1, (q'_1, a, x', q'_2) \in E_1, c' - c = x' - x\}$.

By construction, two states q_1 and q_2 of Q can be reached by the same string u, $|u| \leq 2|Q_1|^2 - 1$, iff there exists $c \in K$ such that (q_1, q_2, c) can be reached from I in A. The set of such (q_1, q_2, c) is exactly the transitive closure of I in A. The transitive closure of I can be determined in time linear in the size of A, $O(|Q| + |E|)$.

Two such states q_1 and q_2 are not twins iff there exists a path in A from $(q_1, q_2, 0)$ to (q_1, q_2, c), with $c \neq 0$. Indeed, this is exactly equivalent to the existence of cycles at q_1 and q_2 with the same input label and distinct output weights. According to the lemma 2, it suffices to test the twins property for strings of length less than $2|Q_1|^2 - 1$. So the following gives an algorithm to test the twins property of a transducer τ_1:

1. Compute the transitive closure of I: $\mathcal{T}(I)$.
2. Determine the set of pairs (q_1, q_2) of $\mathcal{T}(I)$ with distinct states $q_1 \neq q_2$.
3. For each such $\{q_1, q_2\}$ compute the transitive closure of $(q_1, q_2, 0)$ in A. If it contains (q_1, q_2, c) with $c \neq 0$, then τ_1 does not have the twins property.

The operations used in the algorithm (computation of the transitive closure, determination of the set of states) can all be done in polynomial time with respect to the size of A, using classical algorithms (Aho, Hopcroft, and Ullman, 1974). ■

This provides an algorithm for testing the twins property of an unambiguous trim transducer T. It is very useful when one knows T to be unambiguous.

In many practical cases, the transducer one wishes to determinize is ambiguous. It is always possible to construct an unambiguous transducer T' from T (Eilenberg, 1974-1976) when T is unweighted or acyclic. The complexity of such a construction is exponential in the worst case. Thus the overall complexity of the test of determinizability is also exponential in the worst case.

Notice that if one wishes to construct the result of the determinization of T for a given input string w, one does not need to expand the whole result of the determinization, but only the necessary part of the determinized transducer. When restricted to a finite set the function realized by any transducer is subsequentiable since it has bounded variation[9]. Acyclic transducers have the twins property, so they are determinizable. Therefore, it is always possible to expand the result of the determinization algorithm for a finite set of input strings, even if T is not determinizable.

3.6 Determinization in other semirings

The determinization algorithm that we previously presented applies as well to transducers mapping strings to other semirings. We gave the pseudocode of the algorithm in the general case. The algorithm applies for instance to the real semiring $(\mathcal{R}, +, \cdot, 0, 1)$.

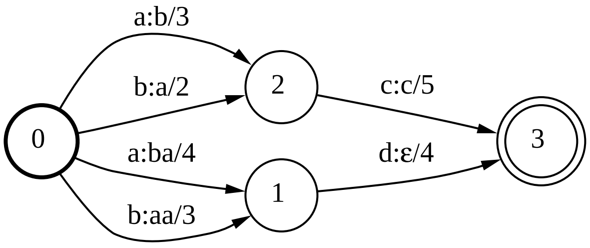

Figure 13
Transducer τ_1 with outputs in $\Sigma^* \times \mathcal{R}$.

One can also verify that $(\Sigma^* \cup \{\infty\}, \wedge, \cdot, \infty, \epsilon)$, where $\wedge$ denotes the longest common prefix operation and $\cdot$ concatenation, ∞ a new element such that for any string $w \in (\Sigma^* \cup \{\infty\})$, $w \wedge \infty = \infty \wedge w = w$ and $w \cdot \infty = \infty \cdot w = \infty$, defines a left semiring[10]. We call this semiring the *string semiring*. The algorithm of figure 10 used with the string semiring is exactly the determinization algorithm for subsequentiable string-to-string transducers as defined by Mohri (1994c). The cross product of two semirings defines a semiring. The algorithm also applies when the semiring is the cross product of $(\Sigma^* \cup \{\infty\}, \wedge, \cdot, \infty, \epsilon)$ and $(\mathcal{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$. This allows one to determinize transducers outputting pairs of strings and weights. The determinization algorithm for such transducers is illustrated in figures 13 and 14. Subsets in this algorithm are made of triples $(q, w, x) \in Q \times \Sigma^* \cup \{\infty\} \times \mathcal{R} \cup \{\infty\}$ where q is a state of the initial transducer, w a residual string and x a residual output weight.

3.7 Minimization

We here define a minimization algorithm for subsequential power series defined on the tropical semiring which extends the algorithm defined by Mohri (1994b) in the case of

9 Using the proof of the theorem of the previous section, it is easy to convince oneself that this assertion can be generalized to any rational subset Y of Σ^* such that the restriction of S, the function T realizes, to Y has bounded variation.

10 A *left semiring* is a semiring that may lack right distributivity.

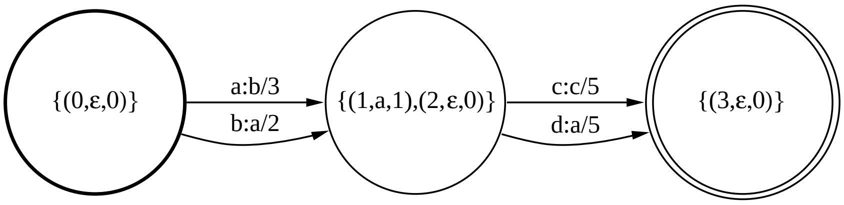

Figure 14
Sequential transducer τ_2 with outputs in $\Sigma^* \times \mathcal{R}$ obtained from β_1 by determinization.

string-to-string transducers. For any subset L of Σ^* and any string u we define $u^{-1}L$ by:

$$u^{-1}L = \{w : uw \in L\} \tag{22}$$

Recall that L is a regular language iff there exists a finite number of distinct $u^{-1}L$ (Nerode, 1958). In a similar way, given a power series S we define a new power series $u^{-1}S$ by[11]:

$$u^{-1}S \quad = \quad \sum_{w \in \Sigma^*} (S, uw)w \tag{23}$$

For any subsequential power series S we can now define the following relation on Σ^*:

$$\begin{aligned}\forall (u, v) \in \Sigma^*,\ u\ R_S\ v \Longleftrightarrow \exists k \in \mathcal{R}, \qquad & (u^{-1}supp(S) = v^{-1}supp(S))\ and \\ & ([u^{-1}S - v^{-1}S]_{/u^{-1}supp(S)} = k)\end{aligned} \tag{24}$$

It is easy to show that R_S is an equivalence relation. $(u^{-1}supp(S) = v^{-1}supp(S))$ defines the equivalence relation for regular languages. R_S is a finer relation. The additional condition in the definition of R_S is that the restriction of the power series $[u^{-1}S - v^{-1}S]$ to $u^{-1}supp(S) = v^{-1}supp(S)$ is constant. The following lemma shows that if there exists a subsequential transducer T computing S with a number of states equal to the number of equivalence classes of R_S, then T is a minimal transducer computing f.

Lemma 3
If S is a subsequential power series defined on the tropical semiring, R_S has a finite number of equivalence classes. This number is bounded by the number of states of any subsequential transducer realizing S.

Proof
Let $T = (Q, i, F, \Sigma, \delta, \sigma, \lambda, \rho)$ be a subsequential transducer realizing S. Clearly,

$$\begin{aligned}\forall (u, v) \in (\Sigma^*)^2,\ \delta(i, u) = \delta(i, v) \quad &\Rightarrow \quad \forall w \in u^{-1}supp(S), \delta(i, uw) \in F \Leftrightarrow \delta(i, vw) \in F \\ &\Leftrightarrow \quad u^{-1}supp(S) = v^{-1}supp(S)\end{aligned}$$

Also, if $u^{-1}supp(S) = v^{-1}supp(S), \forall (u, v) \in (\Sigma^*)^2$,

$$\begin{aligned}\delta(i, u) = \delta(i, v) \quad &\Rightarrow \quad \forall w \in u^{-1}supp(S),\ (S, uw) - (S, vw) = \sigma(i, u) - \sigma(i, v) \\ &\Leftrightarrow \quad [u^{-1}S - v^{-1}S]_{/u^{-1}supp(S)} = \sigma(i, u) - \sigma(i, v)\end{aligned}$$

11 One can prove that S, a power series defined on a field, is rational iff it admits a finite number of independent $u^{-1}S$ (Carlyle and Paz, 1971). This is the equivalent for power series of the Nerode's theorem for regular languages.

So $\forall (u,v) \in (\Sigma^*)^2,\ \delta(i,u) = \delta(i,v) \Rightarrow (uR_s v)$. This proves the lemma. ■

The following theorem proves the existence of a minimal subsequential transducer representing S.

Theorem 14
For any subsequential function S, there exists a minimal subsequential transducer computing it. Its number of states is equal to the index of R_S.

Proof
Given a subsequential power series S, we define a power series f by:

$$\begin{aligned} \forall u \in \Sigma^* : u^{-1} supp(S) = \emptyset,\ (f,u) &= 0 \\ \forall u \in \Sigma^* : u^{-1} supp(S) \neq \emptyset,\ (f,u) &= \min_{w \in u^{-1} supp(S)} (S, uw) \end{aligned}$$

We then define a subsequential transducer $T = (Q, i, F, \Sigma, \delta, \sigma, \lambda, \rho)$ by[12]:

- $Q = \{\bar{u} : u \in \Sigma^*\}$;
- $i = \bar{\epsilon}$;
- $F = \{\bar{u} : u \in \Sigma^* \cap supp(S)\}$;
- $\forall u \in \Sigma^*, \forall a \in \Sigma, \delta(\bar{u}, a) = \overline{ua}$;
- $\forall u \in \Sigma^*, \forall a \in \Sigma, \sigma(\bar{u}, a) = (f, ua) - (f, u)$;
- $\lambda = (f, \epsilon)$;
- $\forall q \in Q,\ \rho(q) = 0$.

Since the index of R_S is finite, Q and F are well-defined. The definition of δ does not depend on the choice of the element u in $\bar{u}$ since for any $a \in \Sigma$, $u\ R_S\ v$ implies $(ua)\ R_S\ (va)$. The definition of σ is also independent of this choice since by definition of R_S, if $uR_S v$, then $(ua)\ R_S\ (va)$ and there exists $k \in \mathcal{R}$ such that $\forall w \in \Sigma^*, (S, uaw) - (S, vaw) = (S, uw) - (S, vw) = k$. Notice that the definition of σ implies that:

$$\forall w \in \Sigma^*, \sigma(i, w) = (f, w) - (f, \epsilon) \tag{25}$$

So:

$$\forall w \in supp(S), \lambda + \sigma(i, w) + \rho(q) = (f, w) = \min_{w' \in w^{-1} supp(S)} (S, ww')$$

S is subsequential, hence: $\forall w' \in w^{-1} supp(S), (S, ww') \leq (S, w)$. Since $\forall w \in supp(S), \epsilon \in w^{-1} supp(S)$, we have:

$$\min_{w' \in w^{-1} supp(S)} (S, ww') = (S, w)$$

T realizes S. This ends the proof of the theorem. ■
Given a subsequential transducer $T = (Q, i, F, \Sigma, \delta, \sigma, \lambda, \rho)$, we can define for each state $q \in Q$, $d(q)$ by:

$$d(q) = \min_{\delta(q,w) \in F} (\sigma(q, w) + \rho(\delta(q, w))) \tag{26}$$

12 We denote by $\bar{u}$ the equivalence class of $u \in \Sigma^*$.

Definition
We define a new operation of *pushing* which applies to any transducer T. In particular, if T is subsequential the result of the application of pushing to T is a new subsequential transducer $T' = (Q, i, F, \Sigma, \delta, \sigma', \lambda', \rho')$ which only differs from T by its output weights in the following way:

- $\lambda' = \lambda + d(i)$;
- $\forall (q, a) \in Q \times \Sigma,\ \sigma'(q, a) = \sigma(q, a) + d(\delta(q, a)) - d(q)$;
- $\forall q \in Q, \rho'(q) = 0$.

According to the definition of d, we have:

$$\forall w \in \Sigma^* : \delta(q, aw) \in F, d(q) \leq \sigma(q, a) + \sigma(\delta(q, a), w) + \rho(\delta(\delta(q, a), w))$$

This implies that:

$$d(q) \leq \sigma(q, a) + d(\delta(q, a))$$

So, σ' is well-defined:

$$\forall (q, a) \in Q \times \Sigma, \sigma'(q, a) \geq 0$$

Lemma 4
Let T' be the transducer obtained from T by pushing. T' is a subsequential transducer which realizes the same function as T.

Proof
That T' is subsequential follows immediately its definition. Also,

$$\forall w \in \Sigma^*, q \in Q, \sigma'(q, w) = \sigma(q, w) + d(\delta(q, w)) - d(q)$$

Since $\delta(i, w) \in F \Rightarrow d(\delta(i, w)) = \rho(\delta(i, w))$, we have:

$$\lambda' + \sigma'(i, w) + \rho'(\delta(i, w)) = \lambda + d(i) + \sigma(q, w) + \rho(\delta(i, w)) - d(i) + 0$$

This proves the lemma. ■
The following theorem defines the minimization algorithm.

Theorem 15
Let T be a subsequential transducer realizing a power series on the tropical semiring. Then applying the following two operations:

1. pushing
2. automata minimization

leads to a minimal transducer. This minimal transducer is exactly the one defined in the proof of the theorem 14.

The automata minimization step in the theorem consists of considering pairs of input labels and associated weights as a single label and of applying classical minimization algorithms for automata (Aho, Hopcroft, and Ullman, 1974). We do not give the proof of the theorem. It can be proved in a way similar to what is indicated by Mohri (1994b).

In general, there are several distinct minimal subsequential transducers realizing the same function. Pushing introduces an equivalence relation on minimal transducers: $T \ R_p \ T'$ iff $p(T) = p(T')$, where $p(T)$ (resp. $p(T')$) denotes the transducer obtained from T (resp. T') by pushing. Indeed, if T and T' are minimal transducers realizing the same function, then $p(T)$ and $p(T')$ are both equal to the unique minimal transducer equivalent to T and T' as defined in theorem 14. So two equivalent minimal transducers only differ by their output labels, they have the same topology. They only differ by the way the output weights are spread along the paths.

Notice that if we introduce a new super final state Φ to which each final state q is connected by a transition of weight $\rho(q)$, then $d(q)$ in the definition of T' is exactly the length of a shortest path from Φ to q. Thus, T' can be obtained from T using the classical single-source shortest paths algorithms[13] such as that of Dijkstra (Cormen, Leiserson, and Rivest, 1992). In case the transducer is acyclic, a classical linear time algorithm based on a topological sort of the graph allows one to obtain d.

Once the function d is defined, the transformation of T into T' can be done in linear time, namely $O(|Q|+|E|)$, if we denote by E the set of transitions of T. The complexity of pushing is therefore linear ($O(|Q|+|E|)$) if the transducer is acyclic. In the general case, the complexity of pushing is $O(|E| \log |Q|)$ if we use classical heaps, $O(|E| + |Q| \log |Q|)$ if we use Fibonacci heaps, and $O(|E| \log \log |Q|)$ if we use the efficient implementation of priority queues by Thorup (1996). In case the maximum output weight W is small, one can use the algorithm of Ahuja et al. (1988). The complexity of pushing is then $O(|E| + |Q|\sqrt{|W|})$.

In case the transducer is acyclic, one can use a specific automata minimization algorithm (Revuz, 1992) with linear time complexity, $O(|Q| + |E|)$. In the general case an efficient implementation of Hopcroft's algorithm (Aho, Hopcroft, and Ullman, 1974) leads to $O(|E| \log |Q|)$.

Thus, the overall complexity of the minimization of subsequential transducers is always as good as that of classical automata minimization: $O(|Q| + |E|)$ in the acyclic case, and $O(|E| \log |Q|)$ in the general case.

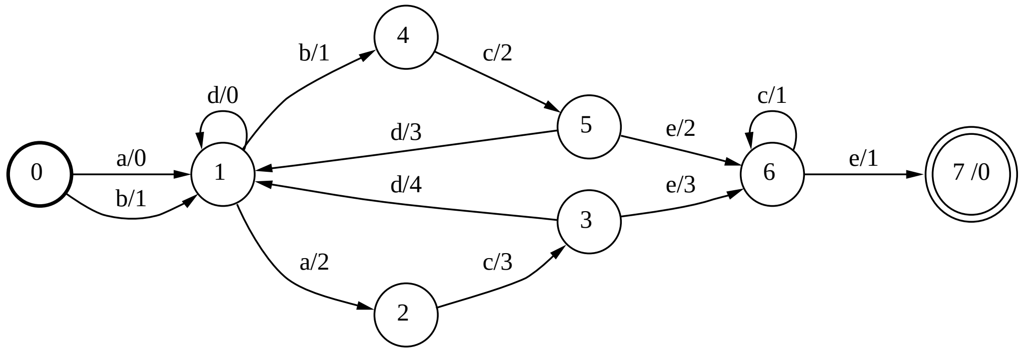

Figure 15
Transducer β_1.

Figures 15 to 17 illustrate the minimization algorithm. β_1 (figure 15) represents a subsequential string-to-weight transducer. Notice that the size of β_1 cannot be reduced

13 This algorithm can be extended to the case where weights are negative. If there is no negative cycle the Bellman-Ford algorithm can be used.

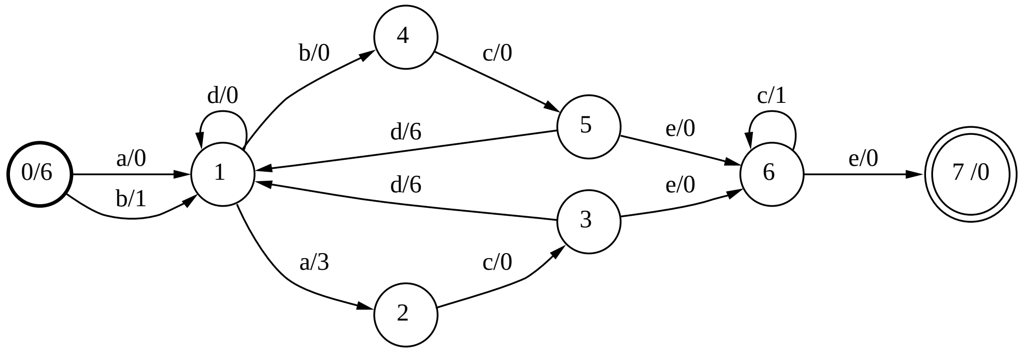

Figure 16
Transducer γ_1 obtained from β_1 by pushing.

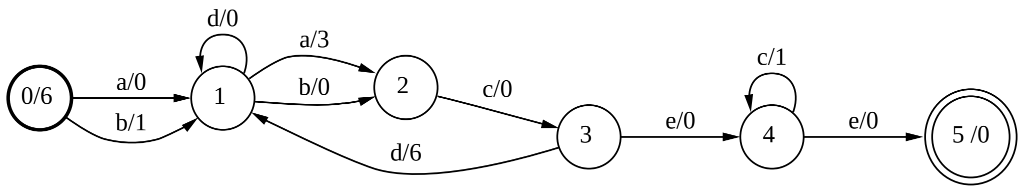

Figure 17
Minimal transducer δ_1 obtained from γ_1 by automata minimization.

using the automata minimization. γ_1 represents the transducer obtained by pushing, and δ_1 a minimal transducer realizing the same function as β_1 in the tropical semiring.

The transducer obtained by this algorithm is the one defined in the proof of the theorem 14 and has the minimal number of states. One can ask whether there exists a subsequential transducer with the minimal number of transitions and computing the same function as a given subsequential transducer T. The following corollary brings a specific answer to this question.

Corollary 1
A minimal subsequential transducer has also the minimal number of transitions among all subsequential transducers realizing the same function.

Proof
This generalizes the analogous theorem which holds in the case of automata. The proof is similar. Let T be a subsequential transducer with a minimal number of transitions. Clearly, pushing does not change the number of transitions of T and the automata minimization which consists of merging equivalent states reduces or does not change this number. Thus, the number of transitions of the minimal transducer equivalent to T as previously defined is less or equal to that of T. This proves the corollary since as previously pointed out equivalent minimal transducers all have the same topology: in particular, they have the same number of states and transitions. ■

Given two subsequential transducers, one might wish to test their equivalence. The importance of this problem was pointed out by Hopcroft and Ullman (1979) (page 284). The following corollary addresses this question.

Corollary 2
There exists an algorithm to determine if two subsequential transducers are equivalent.

Proof
The algorithm of theorem 15 associates a unique minimal transducer to each subsequential transducer T. More precisely this minimal transducer is unique up to a renumbering of the states. The identity of two subsequential transducers with different numbering of states can be tested in the same way as that of two deterministic automata. This can be done for instance by testing the equivalence of the automata and the equality of their number of states. There exists a very efficient algorithm to test the equivalence[14] of two deterministic automata (Aho, Hopcroft, and Ullman, 1974). Since the minimization of subsequential transducers was also shown to be efficient, this proves the corollary and also the efficiency of the test of equivalence. ■

Schützenberger (1961) gave an algorithm for minimizing the representation of power series. However, this algorithm can only be used when the semiring considered is a field. In particular, it cannot be used with the tropical semiring or the string semiring used in language and speech processing, since none of these semirings is a field. More precisely, a recent result of Krob (1994) states that such a minimization cannot be defined for transducers defined on the tropical semiring. Furthermore, we implemented the algorithm of Schützenberger (1961) and used it in the case of the semiring $(\mathcal{R}, +, \cdot, 0, 1)$. It has two important disadvantages in practice: it creates many transitions, and it can generate transitions with negative weights, even if the initial machine has none. The negative weights cannot be interpreted in terms of probability.

In the next section, we describe some of the applications of the algorithms we presented to speech recognition.

4. Application to speech recognition

In the previous sections we gave a theoretical description of the determinization and minimization algorithms for string-to-weight transducers. Here we indicate their use in practice. These algorithms have interesting applications in speech recognition. We briefly point out some of them.

4.1 Speech recognition with finite-state transducers

String to weight transducers are found at several stages of speech recognition. Phone lattices, language models, and word lattices are typically represented by such transducers. Weights in these graphs correspond to negative logarithms of probabilities. They are added along a path. For a given string there might be many different paths in a transducer. The minimum of the total weights of these paths is only considered as a relevant information. Thus, the main operations involved in the interpretation of these transducers are addition and $\min$, namely those of the tropical semiring. So the algorithms we defined in the previous sections apply to speech recognition.

14 The automata minimization step can in fact be omitted if one uses this equivalence algorithm, since it does not affect the equivalence of the two subsequential transducers, considered as automata.

The domain of the speech recognition systems above signal processing can be represented by a composition of finite-state transducers outputting weights, or both strings and weights (Pereira and Riley, 1996; Mohri, Pereira, and Riley, 1996):

$$G \circ L \circ C \circ A \circ O$$

where O represents the acoustic observations, A the acoustic model mapping sequences of acoustic observations to context-dependent phones, C the context-dependency model mapping sequences of context-dependent phones to (context independent) phones, L a pronunciation dictionary mapping sequences of phones to words, and G a language model or grammar mapping sequences of words to sentences.

In general, this cascade of compositions cannot be explicitely expanded, due to their size. One needs an approximation method to search it. Very often a *beam pruning* is used: only paths with weights within the beam (the difference of the weights from the minimum weight so far is less than a certain predefined threshold) are kept during the expansion of the cascade of composition. Furthermore, one is only interested in the best path or a set of paths of the cascade of transducers with the lowest weights.

A set of paths with the lowest weights can be represented by an acyclic string-to-weight transducer. Each path of that transducer corresponds to a sentence. The weight of the path can be intepreted as a negative $\log$ of the probability of that sentence given the sequence of acoustic observations (utterance). Such acyclic string-to-weight transducers are called *word lattices*.

4.2 Word lattices

For a given utterance, the word lattice obtained in such a way contains many paths labeled with the possible sentences and their associated weights. A word lattice often contains a lot of redundancy: many paths correspond to the same sentence but with different weights.

Word lattices can be directly searched to find the most probable sentences which correspond to the best paths, the paths with the smallest weights.

Figure 18 corresponds to a word lattice obtained in speech recognition for the 2,000-word ARPA ATIS Task. It corresponds to the following utterance: *Which flights leave Detroit and arrive at Saint-Petersburg around nine am ?* Clearly the lattice is complex. It contains about 83 million paths.

Usually, it is not enough to consider the best path of a word lattice. One needs to correct the best path approximation by considering the n best paths, where the value of n depends on the task considered. Notice that in case n is very large one would need here to consider all these 83 million paths. The transducer contains 106 states and 359 transitions.

Determinization applies to this lattice. The resulting transducer W_2 (figure 19) is sparser. Recall that it is equivalent to W_1. It realizes exactly the same function mapping strings to weights. For a given sentence s recognized by W_1 there are many different paths with different total weights. W_2 contains a path labeled with s and with a total weight equal to the minimum of the weights of the paths of W_1. Let us insist on the fact that no pruning, heuristics or approximation has been used here. The lattice W_2 only contains 18 paths. It is not hard to realize that the search stage in speech recognition is greatly simplified when applied to W_2 rather than W_1. W_2 admits 38 states and 51 transitions.

The transducer W_2 can still be minimized. The minimization algorithm described in the previous section leads to the transducer W_3. It contains 25 states and 33 transitions and of course the same number of paths as W_2, 18. The effect of minimization appears

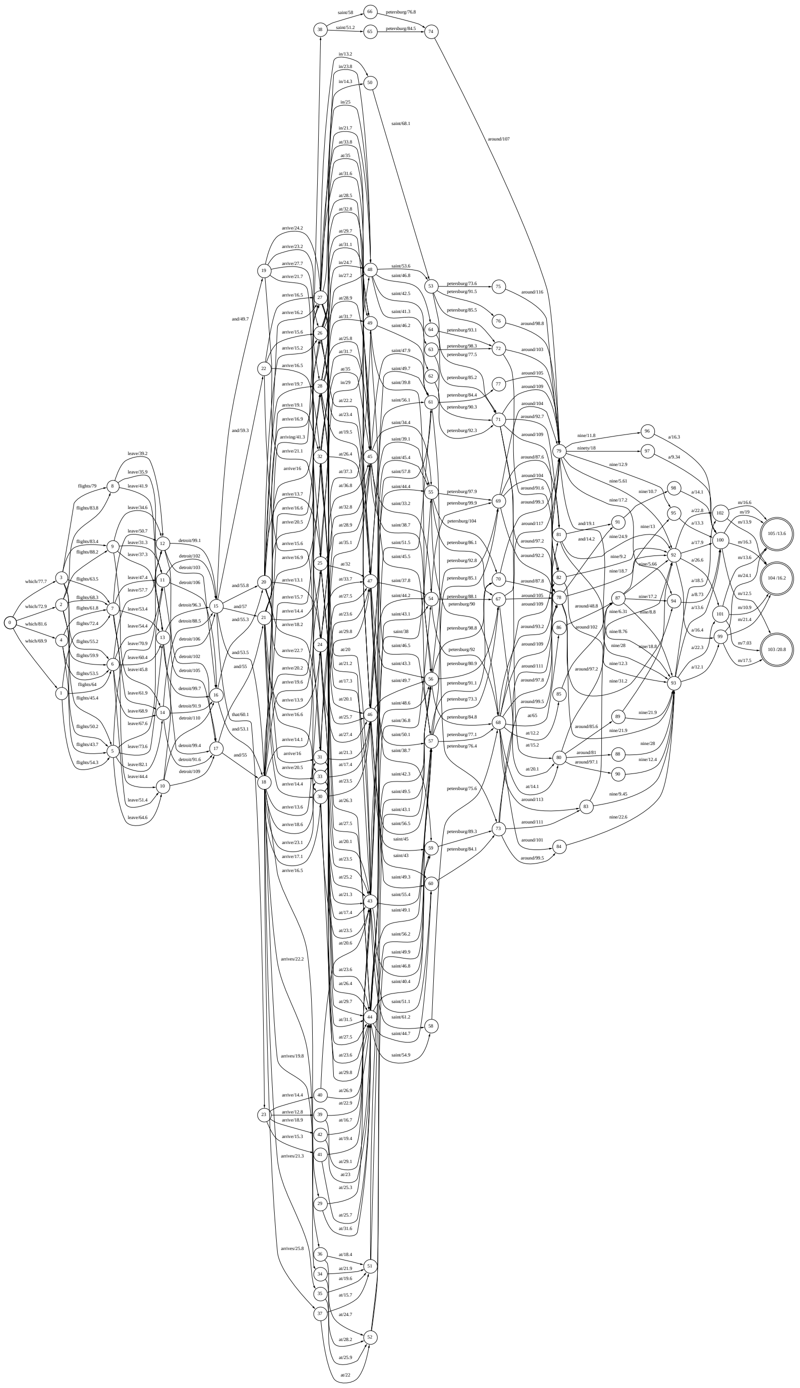

Figure 18
Word lattice W_1, ATIS task, for the utterance *Which flights leave Detroit and arrive at Saint-Petersburg around nine am ?*

less important. This is because determinization includes here a large part of the minimization by reducing the size of the first lattice. This can be explained by the degree of

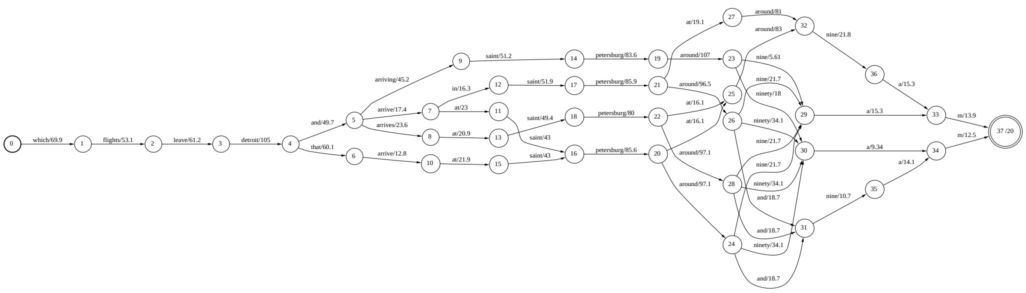

Figure 19
Equivalent word lattice W_2 obtained by determinization of W_1.

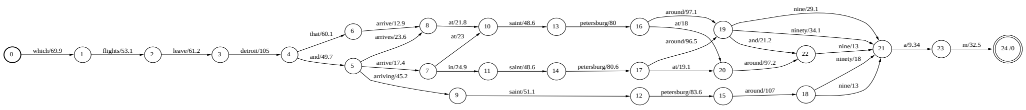

Figure 20
Equivalent word lattice W_3 obtained by minimization from W_2.

non-determinism [15] of word lattices such as W_1. Many states can be reached by the same set of strings. These states are grouped into a single subset during determinization.

Also, the complexity of determinization is exponential in general, but in the case of the lattices considered in speech recognition, this is not the case[16]. Since they contain a lot of redundancy the resulting lattice is smaller than the initial. In fact, the time complexity of determinization can be expressed in terms of the initial and resulting lattices W_1 and W_2 by $O(|\Sigma| \log |\Sigma| (|W_1||W_2|)^2)$, where $|W_1|$ and $|W_2|$ denote the sizes of W_1 and W_2. Clearly if we restrict determinization to the cases where $|W_2| \leq |W_1|$ its complexity is polynomial in terms of the size of the initial transducer $|W_1|$. The same remark applies to the space complexity of the algorithm. In practice the algorithm appears to be very efficient. As an example, it took about 0.02s on a Silicon Graphics[17] (Indy 100 MHZ Processor, 64 Mb RAM) to determinize the transducer of figure 18.

Determinization makes the use of lattices much faster. Since at any state there exists at most one transition labeled with the word considered, finding the weight associated with a sentence does not depend on the size of the lattice. The time and space complexity of such an operation is simply linear in the size of the sentence.

When dealing with large tasks, most speech recognition systems use a rescoring method (figure 21). This consists of first using a simple acoustic and grammar model to produce a word lattice, and then to reevaluate this word lattice with a more sophisticated model.

The size of the word lattice is then a critical parameter in the time and space efficiency of the system. The determinization and minimization algorithms we presented allow one to considerably reduce the size of such word lattices as seen in the examples.

We experimented with both determinization and minimization algorithms in the

15 The notion of ambiguity of a finite automaton can be formalized conveniently using the tropical semiring. Many important studies of the degree of ambiguity of automata have been done in connection with the study of the properties of this semiring (Simon, 1987).
16 A more specific determinization can be used in the cases very often encountered in natural language processing where the graph admits a loop at the initial state over the elements of the alphabet (Mohri, 1995).
17 Part of this time corresponds to I/O's and is therefore independent of the algorithm.

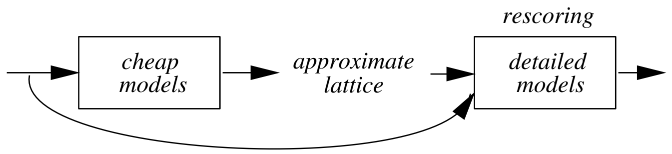

Figure 21
Rescoring.

Table 1
Word lattices in the ATIS task.

	Determinization	Determinization + Minimization
Objects	reduction factor	reduction factor
states	≈ 3	≈ 5
transitions	≈ 9	≈ 17
paths	$> \mathbf{2^{32}}$	$> \mathbf{2^{32}}$

ATIS task. Table 1 illustrates these results. It shows these algorithms to be very effective in reducing the redundancy of speech networks in this task. The reduction is also illustrated by an example in the ATIS task.

Example 1
Example of a word lattice in the ATIS task.

States:	187	$\rightarrow$	37
Transitions:	993	$\rightarrow$	59
Paths:	$> \mathbf{2^{32}}$	$\rightarrow$	6993

The number of paths of the word lattice before determinization was larger than that of the largest integer representable with 32 bit machines. We also experimented with the minimization algorithm by applying it to several word lattices obtained in the 60,000-word ARPA North American Business News task (NAB).

Table 2
Subsequential word lattices in the NAB task.

Minimization results	
Objects	reduction factor
states	≈ 4
transitions	≈ 3

These lattices were already determinized. Table 2 shows the average reduction factors we obtained when using the minimization algorithms with several subsequential lattices obtained for utterances of the NAB task. The reduction factors help to measure the gain of minimization alone since the lattices are already subsequential. The figures of example 2 correspond to a typical case. Here is an example of reduction we obtained.

Example 2
Example of a word lattice in NAB task.

Transitions:	108211	$\rightarrow$	37563
States:	10407	$\rightarrow$	2553

4.3 On-the-fly implementation of determinization

An important characteristic of the determinization algorithm is that it can be used *on-the-fly*. Indeed, the determinization algorithm is such that given a subset representing a state of the resulting transducer, the definition of the transitions leaving that state only depends on that state or equivalently on the states of that subset, and on the transducer to determinize. In particular, the definition and construction of these transitions do not depend directly on the previous subsets constructed.

We have produced an implementation of the determinization which allows one both to completely expand the result or to expand it on demand. Arcs leaving a state of the determinized transducer are expanded only if needed. This characteristic of the implementation is important. It can then be used for instance at any step in an on-the-fly cascade of composition of transducers in speech recognition to expand only the necessary part of a lattice or transducer (Pereira and Riley, 1996; Mohri, Pereira, and Riley, 1996). One of the essential implications of the implementation is that it contributes to saving space during the search stage. It is also very useful in speeding up the n-best decoder in speech recognition[18].

The determinization and minimization algorithms for string-to-weight transducers seem to have other applications in speech processing. Many new experiments can be done using these algorithms at different stages of the recognition. They might lead to reshape some of the methods used in this field in particular by providing a new success of the theory of automata and transducers.

5. Conclusion

We briefly presented the theoretical bases, algorithmic tools and the practical use of a set of devices which seem to fit the complexity of language and provide efficiency in space and time. From the theoretical point of view, the understanding of these objects is crucial. It helps to describe the possibilities they offer and to guide algorithmic choices. Many new theoretical issues arise when more precision is sought.

The notion of determinization can be generalized to that of *ϵ-determinization* for instance (chapter 3, exercise (Salomaa and Soittola, 1978)) requiring more general algorithms. It can also be extended to *local determinization*: determinization only at some states of a transducer that admit a predefined property, such as that of having a large number of outgoing transitions. An important advantage of local determinization is that it can be applied to any transducer without restriction. Furthermore, local determinization also admits an on-the-fly implementation. New characterizations of rational functions shed new light on some aspects of the theory of finite-state transducers (Reutenauer and Schützenberger, 1995). We started here a generalization of the operations we use based on that of the notions of semiring and power series. They help to simplify problems and algorithms used in various cases. In particular, the string semiring that we introduced makes it conceptually easier to describe many algorithms and properties.

Subsequential transducers admit very efficient algorithms. The determinization and minimization algorithms in the case of string-to-weight transducers presented here complete a large series of algorithms which have been shown to give remarkable results in natural language processing. Sequential machines lead to useful algorithms in many other areas of computational linguistics. In particular, subsequential power series allow one to achieve efficient results in indexation of natural language texts (Crochemore,

18 We describe this application of determinization elsewhere.

1986; Mohri, 1996b).

We briefly illustrated the application of these algorithms to speech recognition. More precision in acoustic modeling, finer language models, large lexicon grammars, and a larger vocabulary will lead in the near future to networks of much larger sizes in speech recognition. The determinization and minimization algorithms might help to limit the size of these networks while keeping their time efficiency.

These algorithms can also be used in text-to-speech synthesis. In fact, the same operations of composition of transducers (Sproat, 1995) and perhaps more important size issues can be found in this field.

6. Appendix

The determinization algorithm for power series can also be used to minimize transducers in many cases. Let us first consider the case of automata. Brzozowski (1962) showed that determinization can be used to minimize automata. This nice result has also been proved more recently in elegant papers by Bauer (1988) and Urbanek (1989). These authors refine the method to obtain better complexities[19].

Theorem 16 (Brzozowski, 1962)
Let A be a non-deterministic automaton. Then the automaton $A' = (Q', i', F', \Sigma, \delta')$ obtained by reversing A, applying determinization, rereversing the obtained automaton and determinizing it is the minimal deterministic automaton equivalent to A.

We generalize this theorem to the case of string-to-weight transducers. We say that a rational power series S is *bisubsequential* when S is subsequential and the power series $S^R = \sum_{w \in \Sigma^*} (S, w^R) w$ is also subsequential[20]. Not all subsequential transducers are bisubsequential. Figure 22 shows a transducer representing a power series S that is not bisubsequential. S is such that:

$$\begin{aligned} \forall n \in \mathcal{N},\ (S, ba^n) &= n+1 \\ \forall n \in \mathcal{N},\ (S, ca^n) &= 0 \end{aligned} \tag{27}$$

The transducer of figure 22 is subsequential so S is subsequential. But the reverse S^R is not because it does not have bounded variation. Indeed, since:

$$\begin{aligned} \forall n \in \mathcal{N},\ (S^R, a^n b) &= n+1 \\ \forall n \in \mathcal{N},\ (S^R, a^n c) &= 0 \end{aligned} \tag{28}$$

We have:

$$\forall n \in \mathcal{N},\ |(S^R, a^n b) - (S^R, a^n c)| = n+1$$

One can give a characterization of bisubsequential power series defined on the tropical semiring in a way similar to that of string-to-string transducers (Choffrut, 1978). In particular, the theorem of the previous sections shows that S is bisubsequential iff S and S^R have bounded variation.

We define similarly *bideterminizable* transducers as the transducers T defined on the tropical semiring admitting two applications of determinization in the following way:

1. The reverse of T, T^R can be determinized. We denote by $det(T^R)$ the resulting transducer.

19 See (Watson, 1993) for a taxonomy of minimization algorithms for automata.
20 For any string $w \in \Sigma^*$, we denote by w^R its reverse.

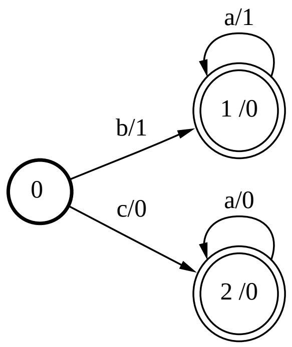

Figure 22
Subsequential power series S non bisubsequential.

2.The reverse of $det(T^R)$, $[det(T^R)]^R$ can also be determinized. We denote by $det([det(T^R)]^R)$ the resulting transducer.

In this definition, we assume that the reverse operation is performed simply by reversing the direction of the transitions and exchanging initial and final states. Given this definition, we can present the extension of the theorem of Brzozowski (1962) to bideterminizable transducers[21].

Theorem 17
Let T be a bideterminizable transducer defined on the tropical semiring. Then the transducer $det([det(T^R)]^R)$ obtained by reversing T, applying determinization, rereversing the obtained transducer and determinizing it is a minimal subsequential transducer equivalent to T.

Proof
We denote by:

- $T_1 = (Q_1, i_1, F_1, \Sigma, \delta_1, \sigma_1, \lambda_1, \rho_1)$ $det(T^R)$,
- $T' = (Q', i', F', \Sigma, \delta', \sigma', \lambda', \rho')$ $det([det(T^R)]^R)$
- $T'' = (Q'', i'', F'', \Sigma, \delta'', \sigma'', \lambda'', \rho'')$ the transducer obtained from T by pushing.

The double reverse and determinization algorithms clearly do not change the function T realizes. So T' is a subsequential transducer equivalent to T. We only need to prove that T' is minimal. This is equivalent to showing that T'' is minimal since T' and T'' have the same number of states. T_1 is the result of a determinization, hence it is a trim subsequential transducer. We show that $T' = det(T_1^R)$ is minimal if T_1 is a trim subsequential transducer. Notice that the theorem does not require that T be subsequential.

Let S_1 and S_2 be two states of T'' equivalent in the sense of automata. We prove that $S_1 = S_2$, namely that no two distinct states of T'' can be merged. This will prove that T'' is minimal. Since pushing only affects the output labels, T' and T'' have the same

21 The theorem also holds in the case of string-to-string bideterminizable transducers. We give the proof in the more complex case of string-to-weight transducers.

set of states: $Q' = Q''$. Hence S_1 and S_2 are also states of T'. The states of T' can be viewed as weighted subsets whose state elements belong to T_1, because T' is obtained by determinization of T_1^R.

Let $(q, c) \in Q_1 \times \mathcal{R}$ be a pair of the subset S_1. Since T_1 is trim there exists $w \in \Sigma^*$ such that $\delta_1(i_1, w) = q$, so $\delta'(S_1, w) \in F'$. Since S_1 and S_2 are equivalent, we also have: $\delta'(S_2, w) \in F'$. Since T_1 is subsequential, there exists only one state of T_1^R admitting a path labeled with w to i_1, that state is q. Thus, $q \in S_2$. Therefore any state q member of a pair of S_1 is also member of a pair of S_2. By symmetry the reverse is also true. Thus exactly the same states are members of the pairs of S_1 and S_2. There exists $k \geq 0$ such that:

$$\begin{aligned} S_1 &= \{(q_0, c_{10}), (q_1, c_{11}), ..., (q_k, c_{1k})\} \\ S_2 &= \{(q_0, c_{20}), (q_1, c_{21}), ..., (q_k, c_{2k})\} \end{aligned} \tag{29}$$

We prove that weights are also the same in S_1 and S_2. Let Π_j, ($0 \geq j \geq k$), be the set of strings labeling the paths from i_1 to q_i in T_1. $\sigma_1(i_1, w)$ is the weight output corresponding to a string $w \in \Pi_j$. Consider the accumulated weights c_{ij}, $1 \leq i \leq 2, 0 \leq j \leq k$, in determinization of T_1^R. Each c_{1j} for instance corresponds to the weight not yet output in the paths reaching S_1. It needs to be added to the weights of any path from $q_j \in S_1$ to a final state in *rev*(T_1). In other terms, the determinization algorithm will assign the weight $c_{1j} + \sigma_1(i_1, w) + \lambda_1$ to a path labeled with w^R reaching a final state of T' from S_1. T" is obtained by pushing from T'. Therefore the weight of such a path in T" is:

$$c_{1j} + \sigma_1(i_1, w) + \lambda_1 - \min_{0 \leq j \leq k, w \in \Pi_j} \{c_{1j} + \sigma_1(i_1, w) + \lambda_1\}$$

Similarly, the weight of a path labeled with the same string considered from S_2 is:

$$c_{2j} + \sigma_1(i_1, w) + \lambda_1 - \min_{0 \leq j \leq k, w \in \Pi_j} \{c_{2j} + \sigma_1(i_1, w) + \lambda_1\}$$

Since S_1 and S_2 are equivalent in T" the weights of the paths from S_1 or S_2 to F' are equal. Thus, $\forall w \in \Pi_j, \forall j \in [0, k]$,

$$c_{1j} + \sigma_1(i_1, w) - \min_{l \in [0,k], w \in \Pi_l} \{c_{1l} + \sigma_1(i_1, w)\} = c_{2j} + \sigma_1(i_1, w) - \min_{l \in [0,k], w \in \Pi_l} \{c_{2j} + \sigma_1(i_1, w)\}$$

$$\text{Hence: } \forall j \in [0, k],\ c_{1j} - \min_{l \in [0,k]} \{c_{1l}\} = c_{2j} - \min_{l \in [0,k]} \{c_{2l}\}$$

We noticed in the proof of the determinization theorem that the minimum weight of the pairs of any subset is 0.

$$\text{Therefore: } \forall j \in [0, k],\ c_{1j} = c_{2j}$$

and $S_2 = S_1$. This ends the proof of the theorem. ■

The following figures illustrate the minimization of string-to-weight transducers using the determinization algorithm. The transducer β_2 of figure 23 is obtained from that of figure 15, β_1, by reversing it. The application of determinization to β_2 results in β_3 (figure 24). Notice that since β_1 is subsequential, according to the theorem the transducer β_3 is minimal too. β_3 is then reversed and determinized (figure 25). The resulting transducer β_4 is minimal and equivalent to β_1. One can compare the transducer β_4 to the transducer of figure 17, δ_1. Both are minimal and realize the same function. δ_1 provides output weights as soon as possible. It can be obtained from β_4 by pushing.

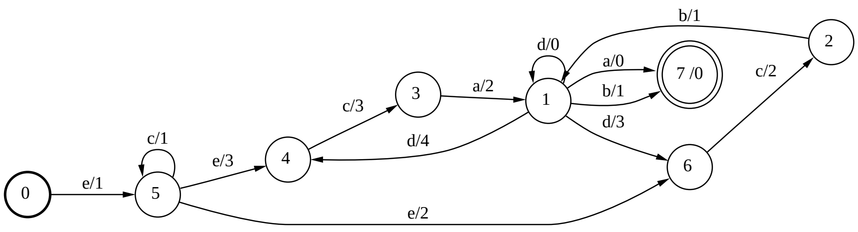

Figure 23
Transducer β_2 obtained by reversing β_1.

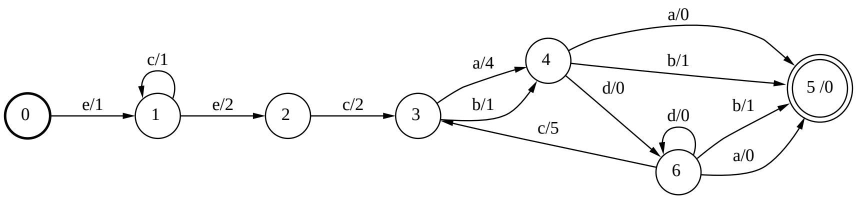

Figure 24
Transducer β_3 obtained by determinization of β_2.

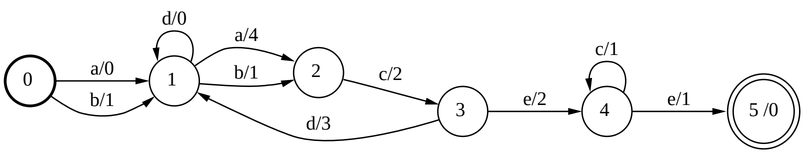

Figure 25
Minimal transducer β_4 obtained by reversing β_3 and applying determinization.

Acknowledgments
I thank Michael Riley, CL reviewers and Bjoern Borchardt, for their comments on earlier versions of this paper, Fernando Pereira and Michael Riley for discussions, Andrej Ljolje for providing the word lattices cited herein, Phil Terscaphen for useful advice, and Dominique Perrin for his help in bibliography relating to the minimization of automata by determinization.

References

Aho, Alfred V., John E. Hopcroft, and Jeffrey D. Ullman. 1974. *The design and analysis of computer algorithms*. Addison Wesley: Reading, MA.

Aho, Alfred V., Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers, Principles, Techniques and Tools*. Addison Wesley: Reading, MA.

Ahuja, Ravindra K., Kurt Mehlhorn, James B. Orlin, and Robert Tarjan. 1988. Faster algorithms for the shortest path problem. Technical Report 193, MIT Operations Research Center.

Bauer, W. 1988. On minimizing finite automata. *EATCS Bulletin*, 35.

Berstel, Jean. 1979. *Transductions and Context-Free Languages*. Teubner Studienbucher: Stuttgart.

Berstel, Jean and Christophe Reutenauer. 1988. *Rational Series and Their Languages*. Springer-Verlag: Berlin-New York.

Brzozowski, J. A. 1962. Canonical regular expressions and minimal state graphs for definite events. *Mathematical theory of Automata*, 12:529–561.

Carlyle, J. W. and A. Paz. 1971. Realizations by stochastic finite automaton. *Journal of Computer and System Sciences*, 5:26–40.

Choffrut, Christian. 1978. *Contributions à l'étude de quelques familles remarquables de fonctions rationnelles*. Ph.D. thesis, (thèse de doctorat d'Etat), Université Paris 7, LITP: Paris, France.

Cormen, T., C. Leiserson, and R. Rivest. 1992. *Introduction to Algorithms*. The MIT Press: Cambridge, MA.

Courcelle, Bruno, Damian Niwinski, and Andreas Podelski. 1991. A geometrical view of the determinization and minimization of finite-state automata. *Math. Systems Theory*, 24:117–146.

Crochemore, Maxime. 1986. Transducers and repetitions. *Theoretical Computer Science*, 45.

Eilenberg, Samuel. 1974-1976. *Automata, Languages and Machines*, volume A-B. Academic Press.

Elgot, C. C. and J. E. Mezei. 1965. On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9.

Ginsburg, S. and G. F. Rose. 1966. A characterization of machine mappings. *Canadian Journal of Mathematics*, 18.

Gross, Maurice. 1989. The use of finite automata in the lexical representation of natural language. *Lecture Notes in Computer Science*, 377.

Hopcroft, John E. and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley: Reading, MA.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3).

Karlsson, Fred, Atro Voutilainen, Juha Heikkila, and Atro Anttila. 1995. *Constraint Grammar, A language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

Karttunen, Lauri, Ronald M. Kaplan, and Annie Zaenen. 1992. Two-level morphology with composition. In *Proceedings of the fifteenth International Conference on Computational Linguistics (COLING'92), Nantes, France*. COLING.

Krob, Daniel. 1994. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Journal of Algebra and Computation*, 4.

Kuich, Werner and Arto Salomaa. 1986. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany.

Lothaire, M. 1990. *Mots*. Hermes: Paris, France.

Mohri, Mehryar. 1994a. Compact representations by finite-state transducers. In 32^{nd} *Meeting of the Association for Computational Linguistics (ACL 94), Proceedings of the Conference, Las Cruces, New Mexico*. ACL.

Mohri, Mehryar. 1994b. Minimization of sequential transducers. *Lecture Notes in Computer Science*, 807.

Mohri, Mehryar. 1994c. On some applications of finite-state automata theory to natural language processing: Representation of morphological dictionaries, compaction, and indexation.

Technical Report IGM 94-22, Institut Gaspard Monge, Noisy-le-Grand.

Mohri, Mehryar. 1994d. Syntactic analysis by local grammars automata: an efficient algorithm. In *Proceedings of the International Conference on Computational Lexicography (COMPLEX 94)*. Linguistic Institute, Hungarian Academy of Science: Budapest, Hungary.

Mohri, Mehryar. 1995. Matching patterns of an automaton. *Lecture Notes in Computer Science*, 937.

Mohri, Mehryar, 1996a. *Finite State Devices in Natural Language Processing*, chapter On The Use of Sequential Transducers in Natural Language Processing. The MIT Press, to appear.

Mohri, Mehryar. 1996b. On some applications of finite-state automata theory to natural language processing. *Journal of Natural Language Engineering*, 2:1–20.

Mohri, Mehryar, Fernando C. N. Pereira, and Michael Riley. 1996. Weighted automata in text and speech processing. In *ECAI-96 Workshop, Budapest, Hungary*. ECAI.

Mohri, Mehryar and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In 34*th Meeting of the Association for Computational Linguistics (ACL 96), Proceedings of the Conference, Santa Cruz, California*. ACL.

Nerode, Anil. 1958. Linear automaton transformations. In *Proceedings of AMS*, volume 9.

Pereira, Fernando C. N. and Michael Riley, 1996. *Finite State Devices in Natural Language Processing*, chapter Weighted Rational Transductions and their Application to Human Language Processing. The MIT Press, to appear.

Perrin, Dominique. 1990. Finite automata. In J. Van Leuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. Elsevier, Amsterdam, pages 1–57.

Reutenauer, Christophe and Marcel Paul Schützenberger. 1995. Variétés et fonctions rationnelles. *Theoretical Computer Science*, 145.

Revuz, Dominique. 1992. Minimisation of acyclic deterministic automata in linear time. *Theoretical Computer Science*, 92:181–189.

Roche, Emmanuel. 1993. *Analyse Syntaxique Transformationnelle du Français par Transducteurs et Lexique-Grammaire*. Ph.D. thesis, Université Paris 7.

Salomaa, Arto and Matti Soittola. 1978. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag: New York.

Schützenberger, Marcel Paul. 1961. On the definition of a family of automata. *Information and Control*, 4.

Schützenberger, Marcel Paul. 1977. Sur une variante des fonctions séquentielles. *Theoretical Computer Science*.

Schützenberger, Marcel Paul. 1987. Polynomial decomposition of rational functions. In *Lecture Notes in Computer Science*, volume 386. Lecture Notes in Computer Science, Springer-Verlag: Berlin Heidelberg New York.

Silberztein, Max. 1993. *Dictionnaires électroniques et analyse automatique de textes: le système INTEX*. Masson: Paris, France.

Simon, Imre. 1987. The nondeterministic complexity of finite automata. Technical Report RT-MAP-8073, Instituto de Matemática e Estatística da Universidade de São Paulo.

Sproat, Richard. 1995. A finite-state architecture for tokenization and grapheme-to-phoneme conversion in multilingual text analysis. In *Proceedings of the ACL SIGDAT Workshop, Dublin, Ireland*. ACL.

Thorup, Mikkel. 1996. On ram priority queues. In *Proceedings of SODA'96, Atlanta, Georgia*. ACM, SIAM, New York.

Urbanek, F. 1989. On minimizing finite automata. *EATCS Bulletin*, 39.

Watson, Bruce W. 1993. A taxonomy of finite automata minimization algorithms. Technical Report 93/44, Eindhoven University of Technology, The Netherlands.

Weber, Andreas and Reinhard Klemm. 1995. Economy of description for single-valued transducers. *Information and Computation*, 119.

Woods, W.A. 1970. Transition network grammars for natural language analysis. *Communications of the Association for the Computational Machinery*, 13(10).