An Optimal Pre-Determinization Algorithm for Weighted Transducers

Cyril Allauzen and Mehryar Mohri

AT&T Labs – Research,
180 Park Avenue, Florham Park, NJ 07932-0971, USA

Abstract

We present a general algorithm, *pre-determinization*, that makes an arbitrary weighted transducer over the tropical semiring or an arbitrary unambiguous weighted transducer over a cancellative commutative semiring determinizable by inserting in it transitions labeled with special symbols. After determinization, the special symbols can be removed or replaced with ϵ-transitions. The resulting transducer can be significantly more efficient to use. We report empirical results showing that our algorithm leads to a substantial speed-up in large-vocabulary speech recognition. Our pre-determinization algorithm makes use of an efficient algorithm for testing a *general twins property*, a sufficient condition for the determinizability of all weighted transducers over the tropical semiring and unambiguous weighted transducers over cancellative commutative semirings. Based on the transitions marked by this test of the twins property, our pre-determinization algorithm inserts new transitions just when needed to guarantee that the resulting transducer has the twins property and thus is determinizable. It also uses a single-source shortest-paths algorithm over the min-max semiring for carefully selecting the positions for insertion of new transitions to benefit from the subsequent application of determinization. These positions are proved to be *optimal* in a sense that we describe.

Key words: finite automata, finite-state transducers, weighted finite-state transducers, determinization, twins property

1 Introduction

Weighted transducers are used in many applications such as text, speech, or image processing for the representation of various information sources [10,13].

Email addresses: allauzen@research.att.com (Cyril Allauzen), mohri@research.att.com (Mehryar Mohri).
URLs: http://www.research.att.com/~allauzen (Cyril Allauzen), http://www.research.att.com/~mohri (Mehryar Mohri).

They are combined to create large and complex systems such as an information extraction or a speech recognition system using a general composition algorithm for weighted transducers [15,16].

The efficiency of such systems is dramatically increased when *subsequential* or *deterministic* transducers are used, i.e. weighted transducers with a unique initial state and with no two transitions sharing the same input label at any state. A generic determinization algorithm for weighted transducers was introduced by [13]. The algorithm can be viewed as a generalization of the classical subset construction used for unweighted finite automata, it outputs a deterministic transducer equivalent to the input weighted transducer. But, unlike unweighted automata, not all weighted transducers can be determinized using that algorithm – this is clear since some weighted transducers do not even admit an equivalent subsequential one, they are not *subsequentiable*, but some subsequentiable transducers cannot be determinized either using that algorithm.

We present a general algorithm, *pre-determinization*, that makes an arbitrary weighted transducer over the tropical semiring or an arbitrary unambiguous weighted transducer over a cancellative commutative semiring determinizable by inserting in it transitions labeled with special symbols. After determinization, the special symbols can be removed or replaced with ϵ-transitions. The resulting transducer can be significantly more efficient to use. We report empirical results showing that our algorithm leads to a substantial speed-up in large-vocabulary speech recognition.

Our pre-determinization algorithm makes use of an efficient algorithm for testing a *general twins property* [2], which is a characterization of the determinizability of functional finite-state transducers and that of unambiguous weighted automata over the tropical semiring or any cancellative commutative semiring, and also a sufficient condition for the determinizability of all weighted transducers over the tropical semiring.

The algorithm for testing the twins property determines and marks some transitions whose presence violates the twins property. Transitions with new symbols need not be inserted at those positions however. There is some degree of freedom in the choice of those positions and their choice is critical to ensure greater benefits from the application of determinization. Based on the transitions marked by this test of the twins property, our algorithm inserts new transitions just when needed to guarantee that the resulting transducer has the twins property and thus is determinizable. It uses a single-source shortest-paths algorithm over the min-max semiring for carefully selecting the positions for insertion of new transitions to benefit from the subsequent application of determinization. These positions are proved to be *optimal* in a sense that we describe.

2 Preliminaries

A *semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is a ring that may lack negation [12]. It has two associative operations $\oplus$ and $\otimes$ with identity elements $\overline{0}$ and $\overline{1}$. $\oplus$ is commutative, $\otimes$ distributes over $\oplus$ and $\overline{0}$ is an annihilator for $\otimes$. A semiring is said to be *commutative* when $\otimes$ is commutative. A commutative semiring is said to be *cancellative* when for all a, b, c in $\mathbb{K}$ with $c \neq \overline{0}$, $a \otimes c = b \otimes c$ implies $a = b$. The *tropical semiring* $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ or the *real semiring* $(\mathbb{R}, +, \times, 0, 1)$ are classical examples of cancellative commutative semirings.

A *weighted transducer* $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ over a semiring $\mathbb{K}$ is an 8-tuple where Σ is a finite input alphabet, Δ is a finite output alphabet, Q is a finite set of states, $I \subseteq Q$ the set of initial states, $F \subseteq Q$ the set of final states, $E \subseteq Q \times \Sigma \times \Delta \times \mathbb{K} \times Q$ a finite set of transitions, $\lambda : I \rightarrow \mathbb{K}$ the initial weight function mapping I to $\mathbb{K}$, and $\rho : F \rightarrow \mathbb{K}$ the final weight function mapping F to $\mathbb{K}$ [18,12]. *Weighted automata* can be defined in a similar way by simply omitting the output labels.

The results presented in this paper hold similarly for weighted transducers over the tropical semiring and unambiguous weighted transducers over a cancellative commutative semiring, cases where our algorithm for testing the twins property can be used [2]. However, to simplify and shorten the presentation, in the following, all definitions, proofs, and examples will be given for weighted transducers over the tropical semiring.

Given a transition $e \in E$, we denote by $i[e]$ its input label, $o[e]$ its output label, $w[e]$ its weight, $p[e]$ its origin or previous state and $n[e]$ its destination state or next state. Given a state $q \in Q$, we denote by $E[q]$ the set of transitions leaving q. A *path* $\pi = e_1 \cdots e_k$ in A is an element of E^* with consecutive transitions: $n[e_{i-1}] = p[e_i]$, $i = 2, \ldots, k$. We extend n and p to paths by setting: $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$. A *cycle* π is a path whose origin and destination states coincide: $n[\pi] = p[\pi]$. We denote by $P(q, q')$ the set of paths from q to q' and by $P(q, x, q')$ and $P(q, x, y, q')$ the set of paths from q to q' with input label $x \in \Sigma^*$ and output label y (transducer case). These definitions can be extended to subsets $R, R' \subseteq Q$, by: $P(R, x, R') = \cup_{q \in R,\, q' \in R'} P(q, x, q')$. The labeling functions i (and similarly o) and the weight function w can also be extended to paths by defining the label of a path as the concatenation of the labels of its constituent transitions, and the weight of a path as the sum of the weights of its constituent transitions: $i[\pi] = i[e_1] \cdots i[e_k]$, $w[\pi] = w[e_1] + \cdots + w[e_k]$. We also extend w to any finite set of paths Π by setting: $w[\Pi] = \min_{\pi \in \Pi} w[\pi]$. The weight associated by a transducer T to an input string $x \in \Sigma^*$ and output string $y \in \Delta^*$ is:

$$[\![T]\!](x, y) = \min_{\pi \in P(I,x,y,F)} (\lambda[p[\pi]] + w[\pi] + \rho[n[\pi]]) \tag{1}$$

A *successful path* in a weighted transducer T is a path from an initial state to a final state. A state q of T is *accessible* if it can be reached from I. It is *coaccessible* if a final state can be reached from q. A weighted transducer T is *trim* if it contains no transition with weight ∞ and if all its states are both accessible and coaccessible. T is *unambiguous* if, for any string $x \in \Sigma^*$, it admits at most one successful path with input label x. T is *cycle-unambiguous* if at any state q there is at most once cycle with a given label $x \in \Sigma^*$. A cycle c in T is an *ϵ-cycle* if both its input and output are labeled with ϵ, *i.e.* $i[c] = o[c] = \epsilon$. A state q in T is said to be *cycle-accessible* if a non ϵ-cycle can be reached from q. The *inverse* T^{-1} of a weighted transducer T is obtained by swapping the input and output labels of each transition in T and its *negation* $-T$ by negating the cost of every transition in T. [1]

Composition is a general operation for combining weighted finite-state transducers [6,11,18,12]. The result of the composition of two weighted transducers T_1 and T_2 over the tropical semiring is the weighted transducer defined as follows. States in the composition $T_1 \circ T_2$ of T_1 and T_2 are identified with pairs of a state of T_1 and a state of T_2. [2] Leaving aside transitions with ϵ inputs or outputs, the following rule specifies how to compute a transition of $T_1 \circ T_2$ from appropriate transitions of T_1 and T_2: [3]

$$(q_1, a, b, w_1, q_1') \text{ and } (q_2, b, c, w_2, q_2') \Longrightarrow ((q_1, q_2), a, c, w_1 + w_2, (q_1', q_2'))$$

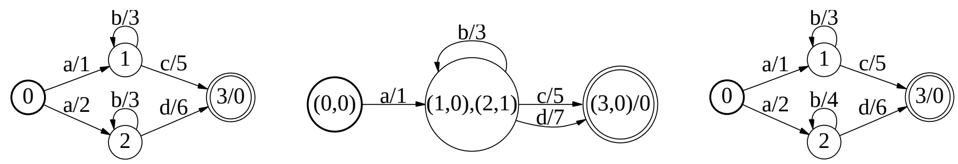
When $T_2 = -T_1^{-1}$, we say that a state (q_1, q_2) of the composed transducer is a *diagonal state* if $q_1 = q_2$. Similarly, a transition is said to be a *diagonal transition* when it is obtained by merging a transition (q_1, a, b, w_1, q_1') with its negative inverse $(q_1, b, a, -w_1, q_1')$ and more generally a path is said to be a *diagonal path* if all its constituent transitions are diagonal.

The following definitions [17] will also be needed in the next sections. An alphabet Σ can be extended by associating to each symbol $a \in \Sigma$ a new symbol denoted by a^{-1} and defining Σ^{-1} as: $\Sigma^{-1} = \{a^{-1} : a \in \Sigma\}$. $X = (\Sigma \cup \Sigma^{-1})^*$ is then the set of strings written over the alphabet $(\Sigma \cup \Sigma^{-1})$. If we set $aa^{-1} = a^{-1}a = \epsilon$, then X forms a group called the *free group generated by* Σ and is denoted by $\Sigma^{(*)}$. Note that the inverse of a string $x = a_1 \cdots a_n$ is then simply $x^{-1} = a_n^{-1} \cdots a_1^{-1}$. Two strings x and y in Σ^* *commute* if $xy = yx$, we then write $x \equiv y$.

[1] Any commutative cancellative semiring can be embedded in a commutative semiring whose multiplicative operation admits an inverse [2]. Since the multiplicative operation of the semiring $\mathbb{K}$ is cancellative, an inverse can be simulated externally by considering the semiring $\mathbb{K}' = (\mathbb{K} \times \mathbb{K})/\equiv$ where $\equiv$ denotes the equivalence relation defined by $(x, y) \equiv (z, t)$ iff $x \otimes t = y \otimes z$. $\mathbb{K}$ can then be embedded into $\mathbb{K}'$, indeed each $x \in \mathbb{K}$ can then be identified with $(x, \overline{1})$ and admits $(\overline{1}, x)$ as an inverse. In the particular case of the tropical semiring, this inverse can be identified with the natural negation of real numbers.

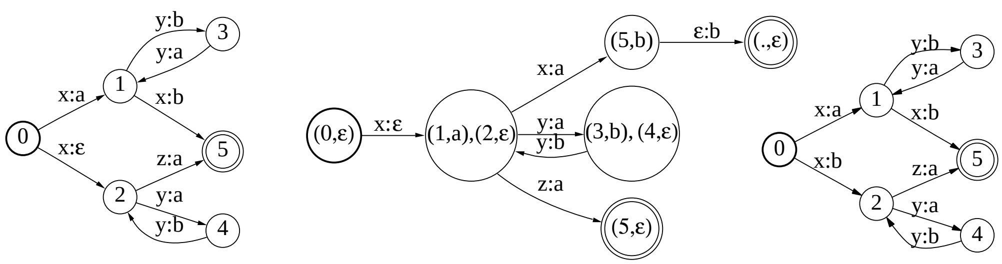
[2] We use a *matrix notation* for the definition of composition as opposed to a *functional notation*.

[3] See [15,16] for a detailed presentation of the algorithm including the use of a filter for dealing with ϵ-paths.



(a) (b) (c)

Fig. 1. Determinization of weighted automata. (a) Weighted automaton over the tropical semiring A. (b) Equivalent weighted automaton B obtained by determinization of A. (c) Non-determinizable weighted automaton over the tropical semiring, states 1 and 2 are non-twin siblings.



(a) (b) (c)

Fig. 2. Determinization of finite-state transducers. (a) Finite-State transducer T. (b) Equivalent transducer T' obtained by determinization of T. (c) Non-determinizable finite-state transducer, states 1 and 2 are non-twin siblings.

3 Determinization and the Twins Property

3.1 Determinization

A weighted automaton or transducer is said to be *deterministic* if it has a unique initial state and if no two transitions leaving the same state have the same input label. There exists a generic determinization algorithm for weighted automata and transducers [13]. The algorithm is a generalization of the classical subset construction [1].

Figure 1 illustrates the determinization of a weighted automaton. The states of the output weighted automaton correspond to *weighted subsets* of the type $\{(q_0, w_0), \ldots, (q_n, w_n)\}$ where each $q_k \in Q$ is a state of the input machine, and w_k a remainder weight. The algorithm starts with the subset reduced to $\{(p, 0)\}$ where p is an initial state and proceeds by creating a transition labeled with $a \in \Sigma$ and weight w leaving $\{(q_0, w_0), \ldots, (q_n, w_n)\}$ if there exists at least one state q_k admitting an outgoing transition labeled with a, w being defined by: $w = \min\{w_k + w[e] : e \in E[q_k],\ i[e] = a\}$.

Similarly, Figure 2 illustrates the determinization of a finite-state transducer. Here, the states of the resulting transducer are *string subsets* of the type

$\{(q_0, x_0), \ldots, (q_n, x_n)\}$, where each $q_k \in Q$ is a state of the input machine, and x_k a remainder string. We refer the reader to [13] for a more detailed presentation of these algorithms.

Unlike the unweighted automata case, not all weighted automata or finite-state transducers are *determinizable*, that is the determinization algorithm does not halt with some inputs. Figure 1(c) shows an example of a non-determinizable weighted automaton and Figure 2(c) a non-determinizable finite-state transducer. Note that the automaton of Figure 1(c) differs from that of Figure 1(a) only by the weight of the self-loop at state 2. The difference between that weight and that of the similar loop at state 1 is the cause of the non-determinizability.

3.2 The twins property

There exists a characterization of the determinizability of weighted transducers based on a *general twins property* and an efficient algorithm for testing that property under some general conditions [13,2].

The twins property was originally introduced by [7,8,6] to give a characterization of the determinizability of unweighted functional finite-state transducers.[4] The definition of the twins property and the characterization results were later extended by [13] to the case of weighted automata. The general twins property for weighted transducers presented here combines both sets of definitions and characterizations [2].

Two states q and q' are said to be *siblings* when they can be reached from the initial states I by paths sharing the same input label and when there exists a cycle at q and a cycle at q' labeled with the same input. Figure 3(a) illustrates this definition. Two sibling states q and q' of a weighted finite-state transducer are said to be *twins* if the following two conditions hold for any paths π from I to q and π' from I to q' and for any cycles c at q and c' at q' such that $i[\pi] = i[\pi']$ and $i[c] = i[c']$:

$$o[\pi]^{-1} o[\pi'] = o[\pi c]^{-1} o[\pi' c'] \tag{2}$$

$$w[P(q, i[c], q)] = w[P(q', i[c'], q')] \tag{3}$$

T is said to have the *twins property* if any two siblings in T are twins. Note that in this definition q may be equal to q' and that we may have $\pi = \pi'$ or $c = c'$, or that π or π' can be the empty path if q, or q', is the initial state.

For weighted automata, only condition 3 on the equality of the cycle weights is required. For unweighted transducers, only condition 2 on the output labels is needed. The twins property is a sufficient condition for the determinizability of weighted automata or weighted transducers over the tropical semiring [13]. It

[4] The twins property was recently shown to provide a characterization of the determinizability of all unweighted finite-state transducers [3].

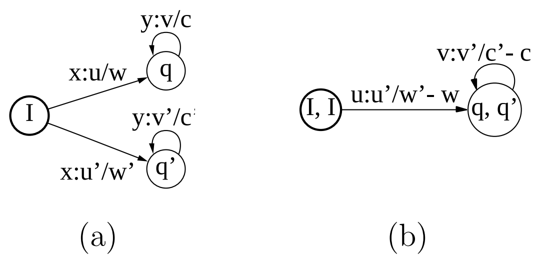


Fig. 3. (a) Two sibling states q and q' in T. (b) The corresponding configuration in $-T^{-1} \circ T$.

is a necessary and sufficient condition for the determinizability of unweighted transducers [3] and that of unambiguous weighted automata or weighted transducers over the tropical semiring [13,2].

Polynomial-time algorithms were given by [19,5] to test the twins property for unweighted transducers. A more efficient algorithm for testing the twins property for weighted and unweighted transducers was given by [2]. The algorithm is based on the composition of T with its negative inverse $-T^{-1}$. Assume that T is a trim cycle-unambiguous weighted transducer over the tropical semiring, then T has the twins property if and only if the following conditions hold for any state q, any path π from the initial state to q, and any cycle c at q in $-T^{-1} \circ T$ [2]: ¡

$$i[\pi]^{-1}o[\pi] = i[\pi c]^{-1}o[\pi c] \tag{4}$$
$$w[c] = 0 \tag{5}$$

Figures 3(a)-(b) illustrate these conditions. Note that condition 4 trivially holds for any path π if c is an ϵ-cycle. If T is a cycle-ambiguous weighted transducer over the tropical semiring, conditions 4 and 5 become sufficient conditions for T to have the twins property and hence for T to be determinizable.

4 Pre-Determinization Algorithm

This section describes a general algorithm, *pre-determinization*, to make an arbitrary weighted transducer T over the tropical semiring or an arbitrary unambiguous weighted transducer T over a cancellative commutative semiring determinizable. The key steps of our algorithm are the following. We first augment the algorithm for testing the twins property for weighted transducers to tag with distinct marks the transitions of the transducer $-T^{-1} \circ T$ that are found by the algorithm to violate the twins property. These marks are then used to *disconnect* some paths of $-T^{-1} \circ T$ by inserting transitions with special symbols in T. We use a single-source shortest-first algorithm over the min-max semiring to disconnect simple cycles at a favorable position and in the desired order of visit of the simple cycles.

4.1 Marking transitions of the composed transducer

The algorithm for testing the twins property computes the composed transducer $S = -T^{-1} \circ T$ and determines paths violating condition (4) or (5). We augment this algorithm to tag the transitions of S found to violate these conditions with distinct marks. More precisely, we use the following marks. If a transition e in S is marked by

i) M_l, then there exist a cycle c containing e and a path π such that the label condition (4) does not hold;
ii) M_w, then there exist a cycle c containing e and a path π such that the weight condition (5) does not hold;
iii) M_a, then there exists a path π_0 containing e such that the label condition (4) does not hold for all non ϵ-cycles c accessible by a path π admitting π_0 as a prefix.

Marks are not exclusive, a transition may be assigned several marks or none. We denote by $M[e]$ the set of marks assigned to a transition e by the augmented test of the twins property.

We now describe in detail how the algorithms given in [2] can be augmented to mark transitions as specified above starting with the algorithm for checking condition 5. For each strongly connected component U in S, the algorithm checks if the weight of each cycle in U is 0. Let q_U be an arbitrary state in U, this is equivalent to checking if for any state q in U, all paths from q_U to q have the same weight.

The pseudocode of the algorithm is given below. For each state q, we maintain two attributes: $W[q]$, which denotes the weight of the first path from q_U to $n[e]$ found in a depth-first search (DFS) of U, and a Boolean attribute $m[q]$ which is set to be TRUE if all paths from $n[e]$ to q_U, $e \in E[q]$, contain a transition marked with M_w.

We assume that, for all $q \in U - \{q_U\}$, $W[q]$ is initialized to some undefined value UNDEFINED, $W[q_U]$ is initialized to 0 and $m[q]$ to FALSE for all $q \in U$. The initial call is Cycle_Identity(q_U, U).

Cycle_Identity(q, U)

$m \leftarrow$ TRUE
for each $e \in E[q]$ such that $n[e] \in U$
    do if $(W[n[e]] =$ UNDEFINED$)$
        then $W[n[e]] \leftarrow W[p[e]] + w[e]$
            Cycle_Identity$(n[e], U)$
    else if $(W[n[e]] \neq W[p[e]] + w[e]$ **and** $m[n[e]] =$ FALSE$)$
        then $M[e] \leftarrow M[e] \cup \{M_w\}$
    if $(M_w \notin M[e]$ and $m[n[e]] =$ FALSE$)$
        then $m \leftarrow$ FALSE
$m[q] \leftarrow m$

In line 1, m is set to TRUE and keeps this value unless an unmarked path from q to q_U is found (lines 8-9). Lines 3-5 define $W[n[e]]$ as the weight of the first path from q_U to $n[e]$ found in the current DFS of U. Lines 6-7 check that the weight of any other path from q_U to q found in a DFS of U equals $W[n[e]]$ and otherwise mark e with M_w if all the paths from $n[e]$ to q_U are not already marked. If e is not marked with M_l and not all paths from $n[e]$ to q_U are marked, then m must be set to FALSE (lines 8-9). Finally, line 10 sets $m[q]$ to m.

The algorithm for checking condition 4 is based on the notion of *residue*. The residue of a path π is defined as the element of the free group $i[\pi]^{-1}o[\pi]$. In [2], it is shown that it is sufficient to compute for each cycle-accessible state q at most two distinct paths residues reaching q from the initial state of S. These residues must verify some combinatorial identities that are checked by the algorithm whose pseudocode is given below.

Residue(q, k)

$m \leftarrow \{M_l, M_a\}$
for each $e \in E[q]$ such that $cyacc[n[e]] =$ TRUE and $M_a \notin m[n[e]] \cup M[e]$ **do**
    $R \leftarrow i[e]^{-1} R_k[p[e]] o[e]$
    if $(R \notin (\Sigma \cup \Sigma^{-1})^*)$
        then $M[e] \leftarrow M[e] \cup \{M_a\}$
    else if $(R_k[n[e]] = \infty)$
            then $R_k[n[e]] \leftarrow R$
                Residue$(n[e], k)$
    else if $(scc[n[e]] = scc[p[e]]$ **and** $R_k[n[e]] \neq R$ **and** $M_l \notin m[n[e]])$
            then $M[e] \leftarrow M[e] \cup \{M_l\}$
        if $(scc[p[e]] \neq scc[n[e]]$ **or** $M_l \in M[e] \cup m[n[e]])$
            then if $(k = 1$ **and** $R_1[n[e]] \neq \infty$ **and** $R_2[n[e]] = \infty$
                    and $R_1[n[e]] \neq R)$
                then $R_2[n[e]] \leftarrow R$
                    Residue$(n[e], 2)$
                else if $(R_1[n[e]] \neq \infty$ **and** $R_2[n[e]] \neq \infty)$
                    then if $(R_1[n[e]]^{-1} R_2[n[e]] \not\equiv R_1[n[e]]^{-1} R)$
                        then $M[e] \leftarrow M[e] \cup \{M_a\}$
    if $(scc[p[e]] = scc[n[e]])$
        then $m \leftarrow m \cap (m[n[e]] \cup M[e])$
    else $m \leftarrow m \cap (m[n[e]] \cup M[e] \cup \{M_l\})$
$m[q] \leftarrow m$

The algorithm uses a DFS of S to compute two distinct residues R_1 and R_2, initialized to an undefined value ∞, for each cycle-accessible state in S. The initial call is Residue$(i, 1)$ where i is the initial state of S.

It also maintains an attribute $m[q]$ for each state. If $M_l \in m[q]$, then all cycles at q contain a transition marked with M_l. If $M_a \in m[q]$, then all paths going through q from the initial state to a cycle-accessible state contain a transition marked with M_a. For each state q, $m[q]$ is initialized to the empty set.

The original call is Residue$(i, 1)$ where i is the initial state. We also assume that the cycle-accessible states q of T have been marked with $cyacc[q] = \text{TRUE}$, this can be done in linear time with respect to the size of T. In line 1, m is initialized to $\{M_l, M_a\}$, m is a temporary variable meant to hold the value of $m[q]$ that is being computed. The search is only necessary for transitions e such that $n[e]$ is cycle-accessible and $M_a \notin m[n[e]] \cup M[e]$ (line 2). The new residue $R = i[e]^{-1}R_k[p[e]]o[e]$ is computed in line 3. If R is not in $\Sigma^* \cup (\Sigma^{-1})^*$, e is marked with M_a (lines 4-5). When $R_k[n[e]]$ is undefined and R is in $\Sigma^* \cup (\Sigma^{-1})^*$, it is set to R and the computation of R_k continues with the call Residue$(n[e], k)$ (lines 6-8). If q and $n[e]$ are in the same strongly connected component, $R_k[n[e]]$ has already been computed and is not equal to R, thus e is marked with M_l unless $M_l \in m[n[e]]$ (lines 9-10).

Lines 11-17 correspond to the case where q and $n[e]$ are in distinct strongly connected components or $M_l \in M[e] \cup m[n[e]]$. When $R_1[n[e]]$ has been already computed, $R_2[n[e]]$ is undefined and $R \neq R_1[n[e]]$, $R_2[n[e]]$ is set to R and the computation of the second residue R_2 continues with the call Residue$(n[e], 2)$ (lines 12-14). If both $R_1[n[e]]$ and $R_2[n[e]]$ are defined, $R_1[n[e]]^{-1}R_2[n[e]]$ and $R_1[n[e]]^{-1}R$ must commute otherwise e is marked with M_a (lines 15-17). Lines 18-20 update m such that $M_a \in m$ if for all transitions e considered so far, $M_a \in M[e] \cup m[n[e]]$ and that $M_l \in m$ if for all transitions e considered so far such that $scc[p[e]] = scc[n[e]]$, $M_l \in M[e] \cup m[n[e]]$. At line 21, after all transitions leaving q have been considered, m is assigned to $m[q]$.

4.2 Disconnecting Paths

By definition of composition, a path $\pi = e_1 \cdots e_n$ in the composed transducer S is the result of matching the input label of a path $\pi_1 = e_1^1 \cdots e_1^n$ of T with the input label of a path $\pi_2 = e_2^1 \cdots e_2^n$ of T. Assume that π is not a diagonal path, then π can be eliminated from the composed machine S by inserting a new transition with a special symbol in π_1 or π_2, at any position i, $1 \leq i \leq n$, such that e_i is not a diagonal transition ($e_1^i \neq e_2^i$), since this would prevent π_1 or π_2 to match. We then say that path π has been *disconnected* and will often use the transition e_1^i (or e_2^i) to refer to the position of insertion of that special transition in T. Each of these special transitions will have for input label a distinct special symbol that is not part of the original input alphabet Σ and that will not be used to label any other special transition. The choice of the position e_1^i (or e_2^i) is critical for the subsequent application of determinization and will be discussed in detail in Section 4.3.

Proposition 1 (Correctness) *Let T be a weighted transducer over the tropical semiring or an unambiguous weighted transducer over a cancellative commutative semiring, let S be the corresponding composed transducer, and let T' be the transducer obtained from T after application of the following operations:*

(1) if $M[e] \cap \{M_w\} \neq \emptyset$, disconnect all simple non-diagonal cycles containing e in S.

(2) if $M[e] \cap \{M_l\} \neq \emptyset$, disconnect all simple non-diagonal non ϵ-cycles containing e in S.
(3) if $M[e] \cap \{M_l\} \neq \emptyset$, disconnect all simple non-diagonal paths from an initial state leading to a diagonal cycle containing e in S.
(4) if $M[e] \cap \{M_a\} \neq \emptyset$, disconnect all simple non-diagonal cycles in S reachable from e, and all simple non-diagonal paths containing e in S from the initial state to a diagonal cycle.

Then T' has the twins property and if we replace the special symbols in T' by ϵ, then T' becomes equivalent to T.

Proof. The proof follows directly the definition of the twins property and the proof of the correctness of the algorithm to test for the twins property from [2]. □

In what follows, we will focus on the algorithm for disconnecting all the simple non-diagonal cycles containing a transition e in S with $M[e] \cap \{M_w\} \neq \emptyset$ (the first item of Proposition 1), or similarly all the simple non-diagonal non ϵ-cycles containing a transition e in S with $M[e] \cap \{M_l\} \neq \emptyset$ (the second item of Proposition 1). A similar algorithm can be used to disconnect the paths leading to a diagonal cycle containing a transition e with $M[e] \cap \{M_l\} \neq \emptyset$ (third item). Disconnecting the paths defined by the fourth item of Proposition 1 can be done using the same algorithms. It requires first determining all the strongly connected components reachable from a transition e with $M[e] \cap \{M_a\} \neq \emptyset$. This can be done in time linear in the size of S by computing a topological order of the component graph of S [9].

Comments. Our test of the twins property marks violating transitions and not paths. Ideally, one would mark just the violating paths instead. Our algorithm inserts auxiliary transitions just where needed given the transitions marked by the test of the twins property. However, in some cases, disconnecting some paths makes it unnecessary to insert symbols at other transitions. Ideally, one would disconnect just the paths that need to be disconnected, but this is difficult to determine and is likely to be computationally hard.

4.3 Positions for Insertion of Transitions

As mentioned earlier, different positions may be chosen to disconnect a non-diagonal simple cycle C of S. Our choice is motivated by the subsequent application of determinization. We wish the longest possible paths to be merged by determinization in order to improve the efficiency of use of the resulting transducer.

For any transition e in T, we define its *merging power*, $m[e]$, as the minimum length of the paths that can be merged with a path containing e if a special symbol is inserted at e. Thus, if the choice is between two transitions e_1 and e_2 for the insertion of a special symbol, with $m[e_1] < m[e_2]$, e_2 is preferable since it can allow longer paths to be merged. We then say that e_2 is a *more favorable position for determinization* than e_1.

Since composition merges pairs of paths with matching labels, the merging power of a transition e can be naturally defined in terms of the composed transducer S. Let E_S denote the set of transitions of S and denote by (e, e') a transition of E_S obtained by matching the negative inverse of the transition e and the transition e' in composition. The level of each transition $(e, e') \in E_S$ in a breadth-first search tree of S can be computed in linear time in the size of S [9]. Let $L[(e, e')]$ denote the level of (e, e'). For any transition e in T, let $\Phi[e]$ be the set of non-diagonal transitions of E_S obtained by matching e with some other transition e'. The merging power of a transition e of T can then be defined by:

$$m[e] = \begin{cases} \min\{L[(e', e'')] : (e', e'') \in \Phi[e]\} & \text{if } (\Phi[e] \neq \emptyset) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

And a simple cycle C in S should be disconnected at a transition (e, e') such that e (or e') is the most favorable position for determinization:

$$e = \text{argmax}\{m[e] : \Phi[e] \cap C \neq \emptyset\} \tag{7}$$

Since disconnecting one cycle may affect another, it is also important to determine in what order simple cycles are disconnected. To avoid disconnecting a cycle more than once, we must start with the simple cycle whose most favorable insertion position e has the minimum merging power. Note that the max operation is used to determine the most favorable position along a cycle and the min operation to determine the order in which these cycles are visited and disconnected.

For each strongly connected component, we must disconnect each simple non-diagonal cycle containing a transition marked with M_w in the order just defined. Enumerating all simple cycles explicitly to disconnect them can be very costly. Instead, since the operations used are min and max and since the min-max semiring is 0-closed, we can use a single-source shortest-distance algorithm over $(\mathbb{N} \cup \{\infty\}, \min, \max, \infty, 0)$ to visit and disconnect simple cycles in the desired order [14]. For the purpose of determining the order of visit of the cycles, we can assign to each transition $(e_1, e_2) \in E_S$ the weight $\max\{m[e_1], m[e_2]\}$. By definition, the *shortest-first* order of this algorithm then coincides exactly with the desired order and guarantees that all simple cycles are visited as described. A simple cycle is disconnected at the transition with the maximum merging power if it was marked to be disconnected and is not already disconnected as a result of the disconnection of another cycle.

For each state s in a strongly connected component Γ, we use a depth-first search from s to identify the set X_s of transitions that belong to a simple non-diagonal cycle at s containing a transition marked with M_w: these are the transitions along the paths containing a transition marked with M_w that are either not a back edge or a back edge with destination state s. We use a single-source shortest-distance algorithm over $(\mathbb{N} \cup \{\infty\}, \min, \max, \infty, 0)$ from s restricted to transitions in X_s to compute for each state q the shortest distance $d_s[q]$ from s to q [14]. We use the same algorithm on the reverse graph to compute the shortest distance $f_s[q]$ from each state q to s. The pseudocode of the algorithm is given below.

It is derived from the pseudocode of the generic single-source shortest distance algorithm presented in [14] restricted to a set of transitions X_s. The algorithm computes the shortest distance from s to q, i.e., the minimal weight of a path in X_s from s to q. For each state q we maintain the attribute $d_s[q]$ which is the current estimate of the shortest-distance from s to q, initialized in line 1-3. We use a queue S with a shortest-first discipline to maintain the set of states whose transitions need to be relaxed. The weight of a state q in the queue is $d_s[q]$. S is initialized to $\{s\}$ (Line 4). The state q with the minimal $d_s[q]$ is extracted from S (lines 5-7). At lines 9-12, each transition in $E[q] \cap X$ is relaxed. If $\max(d_s[q], w[e])$ is less than $d_s[n[e]]$, then $d_s[n[e]$ is updated and if $n[e]$ is not already in S, it is added to S so that its outgoing transitions can be later relaxed.

ShortestDistance(G, s, X_s)

1 **for** $e \in X_s$
2   **do** $d_s[p[e]] \leftarrow d[n[e]] \leftarrow \infty$
3 $d_s[s] \leftarrow 0$
4 $S \leftarrow \{s\}$
5 **while** $S \neq \emptyset$
6   **do** $q \leftarrow \mathrm{head}(S)$
7   Dequeue(S)
8   **for** each $e \in E[q] \cap X_s$
9   **do if** $d_s[n[e]] > \max(d_s[q], w[e])$
10   **then** $d_s[n[e]] \leftarrow \max(d_s[q], w[e])$
11   **if** $n[e] \notin S$
12   **then** Enqueue$(S, n[e])$

Once the distances d_s and f_s have been computed for each state s in Γ, we can use them to identify the potential positions for insertion. A transition e is said to be *maximal* if there exists a state s such that $w[e] \geq \max(d_s[q], f_s[q])$. We consider all the maximal transitions e in the order of increasing weights $w[e]$ and apply the following operations. When e is maximal for a state s, then we disconnect e since it is indeed the most favorable position for the cycle $\pi_1 e \pi_2$, unless $p[e]$ has been made non-accessible by some previous disconnection or the shortest path π_1 from s to $p[e]$ or π_2 from $n[e]$ to s has been disconnected.

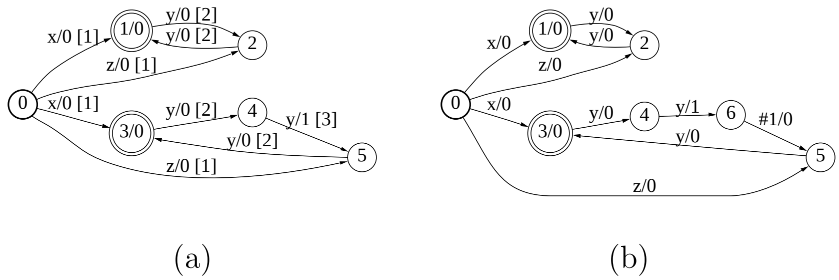


Fig. 4. (a) Non-determinizable weighted automaton A over the tropical semiring. The merging power $m[e]$ of each transition e is indicated in square brackets. (b) Weighted automaton B, output of the pre-determinization algorithm applied to A.

To keep track of that, after disconnecting a transition e, we mark all states q whose shortest path to (or from) a state s is thereby disconnected. We also keep track of which states become non-accessible when e is disconnected.

Proposition 2 (Optimality) *Let T be a weighted transducer over the tropical semiring or an unambiguous weighted transducer over a cancellative commutative semiring and let T' be the result of the application of the pre-determinization algorithm to T. Let e_s be a special transition in T' inserted at position e. Then, e_s cannot be moved from e to a position e' in T' more favorable for determinization without violating the hypothesis of Proposition 1.*

Proof. By definition of the pre-determinization algorithm, there exists a path π in S that contains a transition in $\Phi(e)$, that must be disconnected according to Proposition 1 and for which e was the most favorable position. If another position e' along π is selected for inserting e_s, then, by definition of the min-max single-source shortest paths algorithm, $m[e'] \leq m[e]$, thus e' is not more favorable than e. If the position for the insertion of e_s is not along π then π is not disconnected and the hypotheses of Proposition 1 are not verified. $\square$

Example. Let A be the weighted automaton over the tropical semiring shown in Figure 4(a). Figure 5 shows the composed automaton $-A^{-1} \circ A$. A does not have the twins property since $-A^{-1} \circ A$ admits non-zero cycles: the cycle at state $(1, 3)$ has weight 2 and the symmetric cycle at state $(3, 1)$ has weight -2. The algorithm for testing the twins property marks with M_w one of the transitions of each one of these cycles, e.g., the transitions from $(2, 5)$ and $(5, 2)$ labeled with y. A single-source shortest-distance algorithm over the min-max semiring from $(1, 3)$ identifies the transition leaving state $(4, 1)$ as a maximal transition since it has the largest value (3) and hence as the position for the insertion of a special symbol. This corresponds to inserting a new transition with the new special symbol $\#_1$ at the transition leaving state 4 in A. This insertion disconnects in fact both cycles with non-zero weight, thus no other disconnection is needed. Figure 4(b) shows B, the result of the

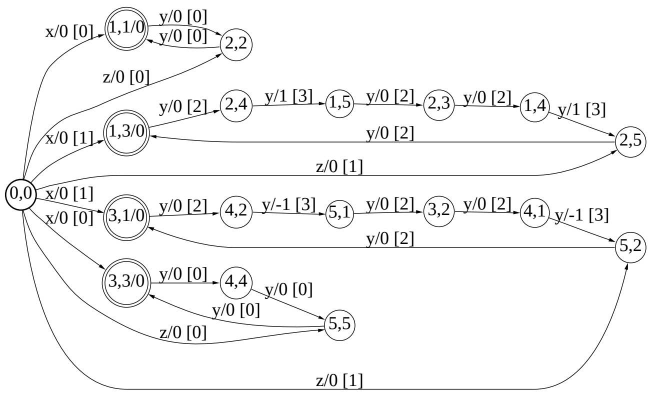


Fig. 5. The negative composition $-A^{-1} \circ A$ where A is the weighted automaton of Figure 4. For each transition (e_1, e_2), $\max\{m[e_1], m[e_2]\}$ is indicated in square brackets.

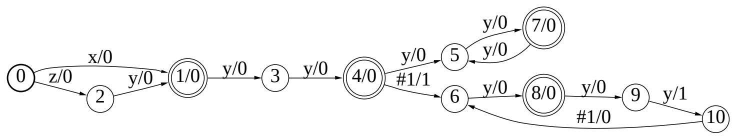


Fig. 6. The result of the determinization of the weighted automaton B of Figure 4(b).

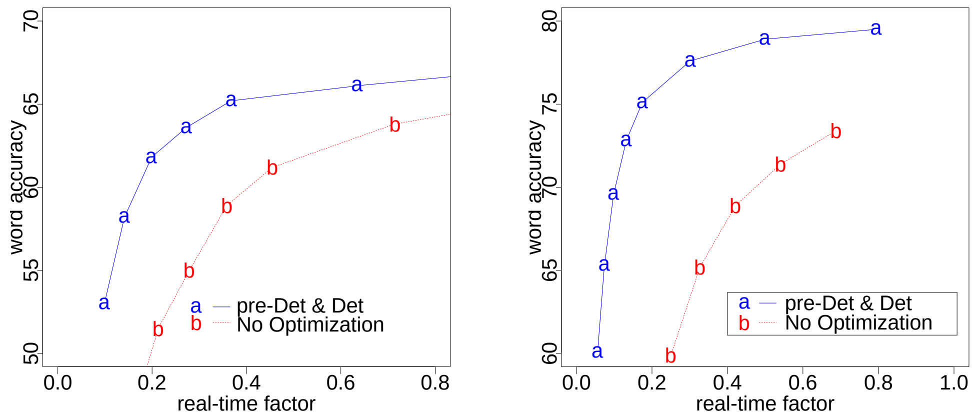
application of the pre-determinization algorithm to A. B has the twins property and is thus determinizable. Figure 6 shows the automaton obtained by determinizing B.

4.4 Complexity

Let Q be the set of states and E the set of transitions of the weighted transducer T. In the worst case, the composed transducer $S = -T^{-1} \circ T$ may have as many as $|Q|^2$ states and $|E|^2$ transitions. The worst-case complexity of the algorithm for testing the twins property and marking the transitions is quadratic in the size of S: $O(|Q|^2(|Q|^2 + |E|^2))$ [2]. The algorithm for disconnecting paths and cycles is based on multiple applications of a single-source shortest-distance algorithm over $(\mathbb{N} \cup \{\infty\}, \min, \max, \infty, 0)$ whose worst case complexity is in $O(|Q|^2 \log |Q| + |E|^2)$ when the graph it applies to has order $|Q|^2$ states and $|E|^2$ transitions [14]. The algorithm also requires computing the component graph of S and its topological order which can be done in linear time in the size of S. Thus, the overall complexity of our pre-determinization algorithm is in $O(|Q|^2(|Q|^2 \log |Q| + |E|^2))$.

5 Experimental Results

We have fully implemented the test of the twins property described and applied it to pre-determinization. We measured its benefits by testing it in the



(a) (b)

Fig. 7. Comparison of the optimization based on our pre-determinization algorithm and determinization versus no optimization in the 5,500-word vocabulary HMIHY 0300 task with (a) a class-based language model and (b) a phrase-based model.

5,500-word vocabulary HMIHY 0300 speech recognition task [4]. The class-based statistical language models used in that task are not determinizable and lead to other non-determinizable machines when combined with the weighted transducers representing information sources such as the pronunciation dictionary.

Our experiments showed that pre-determinization leads to a substantial recognition speed-up in this task. Figure 7 gives recognition accuracy as a function of recognition time, in multiples of real-time on a single processor of a 1GHz Intel Pentium III Linux cluster with 256 KB of cache and 2 GB of memory. Figure 7(a) shows the results corresponding to a class-based language model. Using our algorithm, the accuracy achieved by the old non-optimized integrated transducer at .4 times real-time is reached by the new system using our optimization at about .15 times real-time, that is more than 2.6 times faster. Figure 7(b) shows similar plots when a phrase-based language model is used. The accuracy achieved by the non-optimized transducer at .45 times real-time is achieved by the optimized transducer at .1 times real-time.

The optimization of the weighted transducer T obtained by composing the pronunciation dictionary and the phrase-based language model plays a crucial role in the improvement of the speech recognition speed. To measure the benefits of our algorithm, we computed the *input multiplicity* of the transitions of T and that of the transitions of T' obtained by applying to T our pre-determinization algorithm followed by determinization and removal of auxiliary transitions. The *input multiplicity* of a transition e is defined as the number of transitions sharing the same input label and the same origin state as e. Figure 8 shows the distribution of the input multiplicities in T and T' in quantiles. The transducer T we are starting from in this experiment is not very

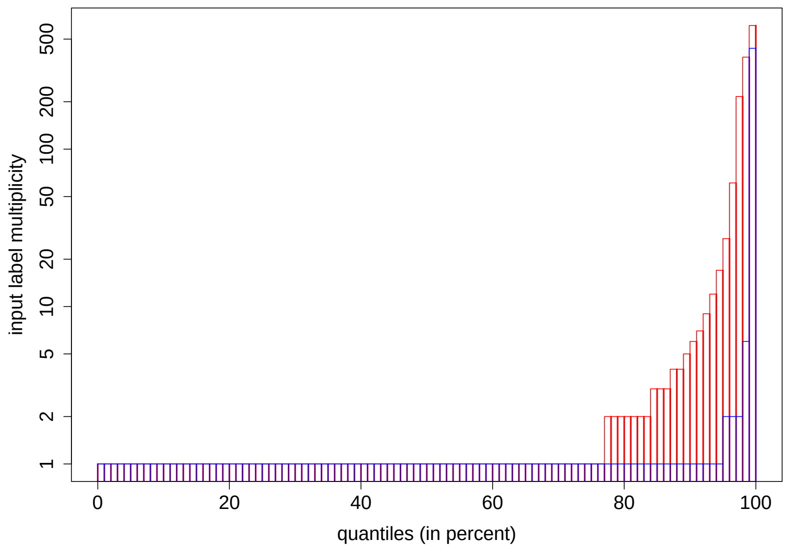


Fig. 8. Distribution of input multiplicities in T (in red) and T' (in blue).

non-deterministic since 76% of its transitions have input multiplicity 1. In T', this proportion increases to 94% and more generally the resulting distribution of multiplicities is much more favorable for T'.

6 Conclusion

A general algorithm was presented that makes an arbitrary weighted transducer over the tropical semiring or any unambiguous weighted transducer over a cancellative commutative semiring determinizable by inserting in it auxiliary symbols and transitions just when needed to ensure that it has the twins property. The auxiliary symbols are inserted at carefully selected positions to increase the benefits of the subsequent determinization. After determinization, the auxiliary symbols can be removed or simply replaced by the empty string.

Experiments in large-vocabulary speech recognition show that the resulting transducer can lead to a substantial recognition speed-up when the original weighted transducer is not determinizable.

References

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

[2] Cyril Allauzen and Mehryar Mohri. Efficient Algorithms for Testing the Twins Property. *Journal of Automata, Languages and Combinatorics*, 8(2):117–144, 2003.

[3] Cyril Allauzen and Mehryar Mohri. Finitely Subsequential Transducers. *International Journal of Foundations of Computer Science*, 14(6):983–994, 2003.

[4] Cyril Allauzen and Mehryar Mohri. Generalized Optimization Algorithm for Speech Recognition Transducers. In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003)*, volume 1, pages 352–355, 2003.

[5] Marie-Pierre Béal, Olivier Carton, Christophe Prieur, and Jacques Sakarovitch. Squaring transducers: An efficient procedure for deciding functionality and sequentiality. *Theoretical Computer Science*, 292:45–63, 2003.

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

[7] Christian Choffrut. Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles. *Theoretical Computer Science*, 5:325–338, 1977.

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

[9] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press: Cambridge, MA, 1992.

[10] Karel Culik II and Jarkko Kari. Digital Images and Formal Languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 599–616. Springer, 1997.

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

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

[13] Mehryar Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23(2), 1997.

[14] Mehryar Mohri. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[15] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted Automata in Text and Speech Processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96), Workshop on Extended finite state models of language, Budapest, Hungary*. ECAI, 1996.

[16] Fernando C. N. Pereira and Michael D. Riley. Speech recognition by composition of weighted finite automata. In *Finite-State Language Processing*, pages 431–453. MIT Press, Cambridge, Massachusetts, 1997.

[17] Dominique Perrin. Words. In M. Lothaire, editor, *Combinatorics on words*, Cambridge Mathematical Library. Cambridge University Press, 1997.

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

[19] Andreas Weber and Reinhard Klemm. Economy of Description for Single-Valued Transducers. *Information and Computation*, 118(2):327–340, 1995.