Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# A disambiguation algorithm for weighted automata

Mehryar Mohri [a,b], Michael D. Riley [b,*]

[a] Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012, United States
[b] Google Research, 76 Ninth Avenue, New York, NY 10011, United States

ARTICLE INFO

*Article history:*
Received 29 November 2015
Received in revised form 18 August 2016
Accepted 22 August 2016
Available online xxxx

*Keywords:*
Weighted automata
Weighted automata algorithms
Automata theory
Rational power series

ABSTRACT

We present a disambiguation algorithm for weighted automata. The algorithm admits two main stages: a pre-disambiguation stage followed by a transition removal stage. We give a detailed description of the algorithm and the proof of its correctness. The algorithm is not applicable to all weighted automata but we prove sufficient conditions for its applicability in the case of the tropical semiring by introducing the *weak twins property*. In particular, the algorithm can be used with any weighted automaton over the tropical semiring for which the weighted determinization algorithm terminates and with any acyclic weighted automaton over an arbitrary weakly left divisible cancellative and commutative semiring. While disambiguation can sometimes be achieved using weighted determinization, our disambiguation algorithm in some cases can return a result that is exponentially smaller than any equivalent deterministic automaton. We also present some empirical evidence of the space benefits of disambiguation over determinization in speech recognition and machine translation applications.



## 1. Introduction

Weighted finite automata and transducers are widely used in applications. Most modern speech recognition systems used for hand-held devices or spoken-dialog applications use weighted automata and their corresponding algorithms for the representation of their models and their efficient combination and search [19,2]. Similarly, weighted automata are commonly used for a variety of tasks in machine translation [10] and other natural language processing applications [11], computational biology [7], image processing [1], optical character recognition [5], and many other areas.

A problem that arises in several applications is that of *disambiguation of weighted automata*: given an input weighted automaton, the problem consists of computing an equivalent weighted automaton that is *unambiguous*, that is one with no two accepting paths labeled with the same string. Fig. 1 shows two equivalent weighted automata, one ambiguous and one not.

The need for disambiguation is often motivated by the common problem of determining the most probable string, or more generally the $n$ most likely strings of a *lattice*, that is an acyclic weighted automaton generated by a complex model, such as those used in machine translation, speech recognition, information extraction, and many other natural language processing and computational biology systems. A lattice compactly represents the model's most likely hypotheses. It defines a probability distribution over the strings and is used as follows: the weight of an accepting path is obtained by multiplying the weights of its component transitions and the weight of a string obtained by summing up the weights of accepting paths

* Corresponding author.
*E-mail addresses:* mohri@cims.nyu.edu (M. Mohri), riley@google.com (M.D. Riley).

http://dx.doi.org/10.1016/j.tcs.2016.08.019
0304-3975/

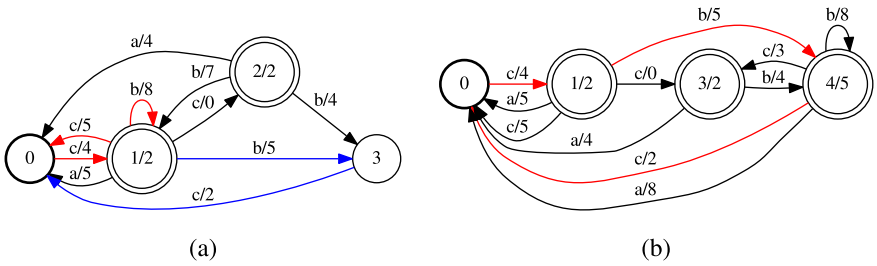


**Fig. 1.** Example of (a) an ambiguous weighted automaton and (b) an equivalent, unambiguous automaton. In figures here, initial states are depicted by a bold circle (always with initial weight $\bar{1}$) and final states by double circles containing their final weight. Transitions are labeled with their symbol and weight. In this example, weights are over the tropical semiring. The string `cbcc` labels two accepting paths, colored as `cbcc` and `cbcc`, in (a) but only one path, `cbcc`, in (b). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

labeled with that string. In general, there may be many accepting paths labeled with a given string. Clearly, if the lattice were unambiguous, a standard shortest-paths or $n$-shortest-paths algorithm [9] could be used to efficiently determine the $n$ most likely strings. When the lattice is not unambiguous, the problem is more complex and can be solved using weighted determinization [20]. An alternative solution, which we will show has benefits, consists of first finding an unambiguous weighted automaton equivalent to the lattice and then running an $n$-shortest-paths algorithm on the resulting weighted automaton. A similar need for disambiguation appears when computing the marginals of a given weighted transducer.

Another common problem where disambiguation is needed is that of sampling strings from a weighted automaton according to the probability distribution it induces. This weighted automaton may be defined over a semiring different from the probability semiring but with the same weight set and the same multiplicative operation. This problem arises, for example, in the context of on-line learning with path experts [6]. Sampling from that weighted automaton directly is a difficult problem. But, if instead an equivalent unambiguous weighted automaton can be computed, then the additive operation of the semiring would be inconsequential for that weighted automaton. One can then equivalently work in the probability semiring and use a straightforward sampling method.

In general, one way to determine an equivalent unambiguous weighted automaton is to use the weighted determinization algorithm [17]. This, however, admits several drawbacks. First, weighted determinization cannot be applied to all weighted automata. This is both because not all weighted automata admit an equivalent deterministic weighted automaton but also because even for some that do, the weighted determinization algorithm may not halt. Sufficient conditions for the application of the algorithm have been given [17,3]. In particular the algorithm can be applied to all acyclic weighted automata over an arbitrary semiring and to all weighted automata over the tropical semiring admitting the twins property. Nevertheless, a second issue is that in some cases where weighted determinization can be used, the size of the resulting deterministic automaton is prohibitively large.

This paper presents a new disambiguation algorithm for weighted automata extending to the weighted case the algorithm of [18] – the weighted case is significantly more complex and this extension non-trivial. As we shall see, our disambiguation algorithm applies to a broader family of weighted automata than determinization in the tropical semiring: we show that if a weighted automaton can be determinized using the algorithm of [17], then it can also be disambiguated using the algorithm presented in this paper (see Section 6). Furthermore, for some weighted automata, the size of the unambiguous weighted automaton returned by our algorithm is exponentially smaller than that of any equivalent deterministic weighted automata. In particular, our algorithm leaves the input unchanged if it is unambiguous, while the size of the automaton returned by determinization for some unambiguous weighted automata is exponentially larger. An example is given in Section 7. We also present empirical evidence showing the benefits of weighted disambiguation over determinization in applications. Our algorithm applies in particular to unweighted finite automata. Note that it is known that for some non-deterministic finite automata of size $n$ the size of an equivalent unambiguous automaton is at least $\Omega(2^{\sqrt{n}})$ [22], which gives a lower bound on the time and space complexity of any disambiguation algorithm for finite automata.

Our disambiguation algorithm for weighted automata is presented in a general way and for a broad class of semirings. Nevertheless, the algorithm is limited in several ways. First, not all weighted automata admit an equivalent unambiguous weighted automaton. But, even for some that do, our algorithm may not succeed. The situation is thus similar to that of weighted determinization. However, we present sufficient conditions based on a new notion of *weak twins property* under which our algorithm can be used. In particular, our algorithm applies to all acyclic weighted automata and more generally to all weighted automata for which the weighted determinization algorithm of [17] terminates. Our algorithm admits two stages. The first stage called *pre-disambiguation* constructs a weighted automaton with several key properties, including the property that paths leaving the initial state and labeled with the same string have the same weight. The second stage consists of removing some transitions to make the result unambiguous. Our disambiguation algorithm can be applied whenever pre-disambiguation terminates.

The paper is organized as follows. In Section 2, we review previous work on this and related problems. In Section 3, we introduce some preliminary definitions and notation relevant to the description of our algorithm. Section 4 describes our pre-disambiguation algorithm and proves some key properties of its result. We describe in fact a family of pre-

Please cite this article in press as: M. Mohri, M.D. Riley, A disambiguation algorithm for weighted automata, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2016.08.019