

# OpenFst: A General and Efficient Weighted Finite-State Transducer Library

## (Extended Abstract of an Invited Talk)

Cyril Allauzen<sup>1</sup>, Michael Riley<sup>2,\*</sup>, Johan Schalkwyk<sup>2</sup>, Wojciech Skut<sup>2</sup>,  
and Mehryar Mohri<sup>1</sup>

<sup>1</sup> Courant Institute of Mathematical Sciences  
251 Mercer ST, New York, NY 10012, USA  
{allauzen,mohri}@cs.nyu.edu

<sup>2</sup> Google, Inc.  
111 Eighth AV, New York, NY 10011, USA  
{riley,johans,wojciech}@google.com

**Abstract.** We describe *OpenFst*, an open-source library for *weighted finite-state transducers* (WFSTs). OpenFst consists of a C++ template library with efficient WFST representations and over twenty-five operations for constructing, combining, optimizing, and searching them. At the shell-command level, there are corresponding transducer file representations and programs that operate on them. OpenFst is designed to be both very efficient in time and space and to scale to very large problems.

This library has key applications speech, image, and natural language processing, pattern and string matching, and machine learning.

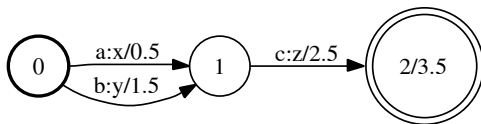
We give an overview of the library, examples of its use, details of its design that allow customizing the labels, states, and weights and the lazy evaluation of many of its operations.

Further information and a download of the OpenFst library can be obtained from <http://www.openfst.org>.

**Keywords:** weighted automata, finite-state transducers, rational power series.

## 1 Introduction

A *weighted finite-state transducer* (WFST) is a finite automaton for which each transition has an input label, an output label, and a weight. Figure 1 depicts a weighted finite state transducer:



**Fig. 1.** An example weighted finite-state transducer

---

\* Corresponding author.

The initial state is labeled 0. The final state is 2 with final weight of 3.5. Any state with non-infinite final weight is a final state. There is a transition from state 0 to 1 with input label  $a$ , output label  $x$ , and weight 0.5. This machine transduces, for instance, the string  $ac$  to  $xz$  with weight 6.5 (the sum of the arc and final weights).

Weighted finite-state transducers have been used in speech recognition and synthesis, machine translation, optical character recognition, pattern matching, string processing, machine learning, information extraction and retrieval among others. Having a comprehensive software library of weighted transducer representations and core algorithms is key for using weighted transducers in these applications and for the development of new algorithms and applications.

To our knowledge, the first such software library was the AT&T FSM Library developed by Mohri, Pereira, and Riley for their work on transducer algorithms and applications [1]. It is available from AT&T for non-commercial use as executable binary programs. Since then, there have been various other weighted transducer library efforts [2,3,4,5]. Our motivation for OpenFst was to create a library as comprehensive and efficient as the AT&T FSM Library, but that was an open-source project. We also sought to make this library as flexible and customizable as possible given the wide range of applications WFSTs have enjoyed in recent years. It is a C++ template library, allowing it to be both very customizable and efficient.

This paper is an overview of this new library. Section 2 introduces some definitions and notation. Section 3 describes the representation and construction of transducers in this library. Section 4 briefly outlines the available algorithms. Section 5 provides examples of the library's use and discusses some new, simplified implementations of several algorithms. Section 6 gives more detail about the transducer representation and discusses the lazy evaluation of algorithms.

The OpenFst library is available for download from <http://www.openfst.org> and is released under the Apache license. Detailed documentation is also available at this site.

## 2 Definitions and Notation

The OpenFst Library closely parallels its mathematical foundations in the theory of rational power series [6,7,8]. The library user can define the alphabets and weights that label transitions. The weights may represent any set so long as they form a *semiring*.

A semiring  $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$  is specified by a set of values  $\mathbb{K}$ , two binary operations  $\oplus$  and  $\otimes$ , and two designated values  $\bar{0}$  and  $\bar{1}$ . The operation  $\oplus$  is associative, commutative, and has  $\bar{0}$  as identity. The operation  $\otimes$  is associative, has identity  $\bar{1}$ , distributes with respect to  $\oplus$ , and has  $\bar{0}$  as annihilator: for all  $a \in \mathbb{K}$ ,  $a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$ . If  $\otimes$  is also commutative, we say that the semiring is *commutative*.

Table 1 lists some common semirings. All but the last are defined over subsets of the real numbers (extended with positive and negative infinity). In addition