General Algorithms for Testing the Ambiguity of Finite Automata

Cyril Allauzen[1,⋆], Mehryar Mohri[2,1], and Ashish Rastogi[1,⋆]

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

Abstract. This paper presents efficient algorithms for testing the finite, polynomial, and exponential ambiguity of finite automata with ϵ-transitions. It gives an algorithm for testing the exponential ambiguity of an automaton A in time $O(|A|_E^2)$, and finite or polynomial ambiguity in time $O(|A|_E^3)$, where $|A|_E$ denotes the number of transitions of A. These complexities significantly improve over the previous best complexities given for the same problem. Furthermore, the algorithms presented are simple and based on a general algorithm for the composition or intersection of automata. We also give an algorithm to determine in time $O(|A|_E^3)$ the degree of polynomial ambiguity of a polynomially ambiguous automaton A. Finally, we present an application of our algorithms to an approximate computation of the entropy of a probabilistic automaton.

1 Introduction

The question of the ambiguity of finite automata arises in a variety of contexts. In some cases, the application of an algorithm requires an input automaton to be finitely ambiguous, in others, the convergence of a bound or guarantee relies on finite ambiguity, or the asymptotic rate of increase of ambiguity as a function of the string length. Thus, in all these cases, an algorithm is needed to test the ambiguity, either to determine if it is finite, or to estimate its asymptotic rate of increase.

The problem of testing ambiguity has been extensively analyzed in the past [9, 7, 13, 3, 6, 15, 12, 14, 16]. The problem of determining the degree of ambiguity of an automaton with finite ambiguity was shown by Chan and Ibarra to be PSPACE-complete [3]. However, testing finite ambiguity can be achieved in polynomial time using a characterization of exponential and polynomial ambiguity given by Ibarra and Ravikumar [6] and Weber and Seidel [15]. The most efficient algorithms for testing polynomial and exponential ambiguity, thereby testing finite ambiguity, were given by Weber and Seidel [14, 16]. The algorithms they presented in [16] assume the input automaton to be ϵ-free, but they are

⋆ Research done at the Courant Institute, partially supported by the New York State Office of Science Technology and Academic Research (NYSTAR).

extended by Weber to the case where the automaton has ϵ-transitions in [14]. In the presence of ϵ-transitions, the complexity of the algorithms given by Weber [14] is $O((|A|_E + |A|_Q^2)^2)$ for testing the exponential ambiguity of an automaton A and $O((|A|_E + |A|_Q^2)^3)$ for testing polynomial ambiguity, where $|A|_E$ stands for the number of transitions and $|A|_Q$ the number of states of A.

This paper presents significantly more efficient algorithms for testing finite, polynomial, and exponential ambiguity for the general case of automata with ϵ-transitions. It gives an algorithm for testing the exponential ambiguity of an automaton A in time $O(|A|_E^2)$, and finite or polynomial ambiguity in time $O(|A|_E^3)$. The main idea behind our algorithms is to make use of the composition or intersection of finite automata with ϵ-transitions [11, 10]. The ϵ-filter used in these algorithms crucially helps in the analysis and test of the ambiguity. The algorithms presented in this paper would not be valid and would lead to incorrect results without the use of the ϵ-filter. We also give an algorithm to determine in time $O(|A|_E^3)$ the degree of polynomial ambiguity of a polynomially ambiguous automaton A. Finally, we present an application of our algorithms to an approximate computation of the entropy of a probabilistic automaton.

The remainder of the paper is organized as follows. Section 2 presents general automata and ambiguity definitions. In Section 3, we give a brief description of existing characterizations for the ambiguity of automata and extend them to the case of automata with ϵ-transitions. In Section 4, we present our algorithms for testing finite, polynomial, and exponential ambiguity, and the proof of their correctness. Section 5 shows the relevance of the computation of the polynomial ambiguity to the approximation of the entropy of probabilistic automata.

2 Preliminaries

Definition 1. *A* finite automaton A *is a 5-tuple* (Σ, Q, E, I, F) *where* Σ *is a finite alphabet;* Q *is a finite set of states;* $I \subseteq Q$ *the set of initial states;* $F \subseteq Q$ *the set of final states; and* $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ *a finite set of transitions, where* ϵ *denotes the empty string.*

We denote by $|A|_Q$ the number of states, by $|A|_E$ the number of transitions, and by $|A| = |A|_E + |A|_Q$ the size of an automaton A. Given a state $q \in Q$, $E[q]$ denotes the set of transitions leaving q. For two subsets $R \subseteq Q$ and $R' \subseteq Q$, we denote by $P(R, x, R')$ the set of all paths from a state $q \in R$ to a state $q' \in R'$ labeled with $x \in \Sigma^*$. We also denote by $p[\pi]$ the origin state, by $n[\pi]$ the destination state, and by $i[\pi] \in \Sigma^*$ the label of a path π.

A string $x \in \Sigma^*$ is accepted by A if it labels an accepting path, that is a path from an initial state to a final state. A finite automaton A is said to be *trim* if all its states lie on some accepting path. It is said to be *unambiguous* if no string $x \in \Sigma^*$ labels two distinct accepting paths; otherwise, it is said to be *ambiguous*. The *degree of ambiguity* of a string x in A is denoted by $\mathrm{da}(A, x)$ and defined as the number of accepting paths in A labeled by x. Note that if A contains an ϵ-cycle, there exists $x \in \Sigma^*$ such that $\mathrm{da}(A, x) = \infty$. Using a depth-first search

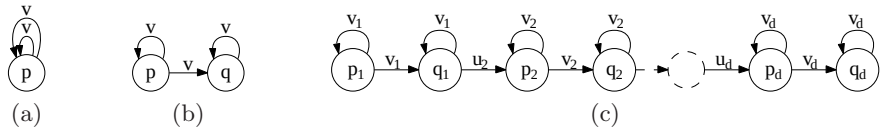

Fig. 1. Illustration of the properties: (a) (EDA); (b) (IDA); and (c) (IDA$_d$).

of A restricted to ϵ-transitions, it can be decided in linear time if A contains ϵ-cycles. Thus, in the following, we will assume, without loss of generality, that A is ϵ-cycle free.

The *degree of ambiguity* of A is defined as $\mathrm{da}(A) = \sup_{x \in \Sigma^*} \mathrm{da}(A, x)$. A is said to be *finitely ambiguous* if $\mathrm{da}(A) < \infty$ and *infinitely ambiguous* if $\mathrm{da}(A) = \infty$. It is said to be *polynomially ambiguous* if there exists a polynomial h in $\mathbb{N}[X]$ such that $\mathrm{da}(A, x) \leq h(|x|)$ for all $x \in \Sigma^*$. The minimal degree of such a polynomial is called the *degree of polynomial ambiguity* of A and is denoted by $\mathrm{dpa}(A)$. By definition, $\mathrm{dpa}(A) = 0$ iff A is finitely ambiguous. When A is infinitely ambiguous but not polynomially ambiguous, it is said to be *exponentially ambiguous* and $\mathrm{dpa}(A) = \infty$.

3 Characterization of infinite ambiguity

The characterization and test of finite, polynomial, and exponential ambiguity of finite automata without ϵ-transitions are based on the following three fundamental properties [6, 15, 14, 16].

Definition 2. *The properties (EDA), (IDA), and (EDA) for A are defined as follows.*

(a) (EDA): *there exists a state q with at least two distinct cycles labeled by some $v \in \Sigma^*$ (see Figure 1(a)) [6].*
(b) (IDA): *there exist two distinct states p and q with paths labeled with v from p to p, p to q, and q to q, for some $v \in \Sigma^*$ (see Figure 1(b)) [15, 14, 16].*
(c) (IDA$_d$): *there exist $2d$ states $p_1, \ldots p_d, q_1, \ldots, q_d$ in A and $2d-1$ strings $v_1, \ldots, v_d$ and $u_2, \ldots u_d$ in Σ^* such that for all $1 \leq i \leq d$, $p_i \neq q_i$ and $P(p_i, v_i, p_i)$, $P(p_i, v_i, q_i)$, and $P(q_i, v_i, q_i)$ are non-empty, and, for all $2 \leq i \leq d$, $P(q_{i-1}, u_i, p_i)$ is non-empty (see Figure 1(c)) [15, 14, 16].*

Observe that (EDA) implies (IDA). Assuming (EDA), let e and e' be the first transitions that differ in the two cycles at state p, then, since Definition 1 disallows multiple transitions between the same two states with the same label, we must have $n[e] \neq n[e']$. Thus, (IDA) holds for the pair $(n[e], n[e'])$.

In the ϵ-free case, it was shown that a trim automaton A satisfies (IDA) iff A is infinitely ambiguous [15, 16], that A satisfies (EDA) iff A is exponentially ambiguous [6], and that A satisfies (IDA$_d$) iff $\mathrm{dpa}(A) \geq d$ [14, 16]. In the following proposition, these characterizations are straightforwardly extended to the case of automata with ϵ-transitions.

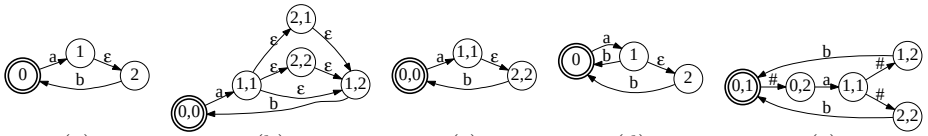

Fig. 2. *ϵ-filter and ambiguity*: (a) Finite automaton A; (b) $A \cap A$ without using ϵ-filter, which incorrectly makes A appear as exponentially ambiguous; (c) $A \cap A$ using an ϵ-filter. *Weber's processing of ϵ-transitions*: (d) Finite automaton B; (e) ϵ-free automaton B' such that $\mathrm{dpa}(B) = \mathrm{dpa}(B')$.

Proposition 1. *Let A be a trim ϵ-cycle free finite automaton.*

(i) A is infinitely ambiguous iff A satisfies (IDA).
(ii) A is exponentially ambiguous iff A satisfies (EDA).
(iii) $\mathrm{dpa}(A) \geq d$ *iff A satisfies* (IDA_d).

Proof. The proof is by induction on the number of ϵ-transitions in A. If A does not have any ϵ-transition, then the proposition holds as shown in [15, 16] for (i), [6] for (ii) and [16] for (iii).

Assume now that A has $n+1$ ϵ-transitions, $n \geq 0$, and that the statement of the proposition holds for all automata with n ϵ-transitions. Select an ϵ-transition e_0 in A, and let A' be the finite automaton obtained after application of ϵ-removal to A limited to transition e_0. A' is obtained by deleting e_0 from A and by adding a transition $(p[e_0], l[e], n[e])$ for every transition $e \in E[n[e_0]]$. It is clear that A and A' are equivalent and that there is a label-preserving bijection between the paths in A and A'. Thus, (a) A satisfies (IDA) (resp. (EDA), (IDA_d)) iff A' satisfies (IDA) (resp. (EDA), (IDA_d)) and (b) for all $x \in \Sigma^*$, $\mathrm{da}(A, x) = \mathrm{da}(A', x)$. By induction, Proposition 1 holds for A' and thus, it follows from (a) and (b) that Proposition 1 also holds for A. □

These characterizations have been used in [14, 16] to design algorithms for testing infinite, polynomial, and exponential ambiguity, and for computing the degree of polynomial ambiguity in the ϵ-free case.

Theorem 1 ([14, 16]). *Let A be a trim ϵ-free finite automaton.*

1. *It is decidable in time $O(|A|_E^3)$ whether A is infinitely ambiguous.*
2. *It is decidable in time $O(|A|_E^2)$ whether A is exponentially ambiguous.*
3. *The degree of polynomial ambiguity of A,* $\mathrm{dpa}(A)$*, can be computed in $O(|A|_E^3)$.*

The first result of Theorem 1 has also been generalized by [14] to the case of automata with ϵ-transitions but with a significantly worse complexity.

Theorem 2 ([14]). *Let A be a trim ϵ-cycle free finite automaton. It is decidable in time $O((|A|_E + |A|_Q^2)^3)$ whether A is infinitely ambiguous.*

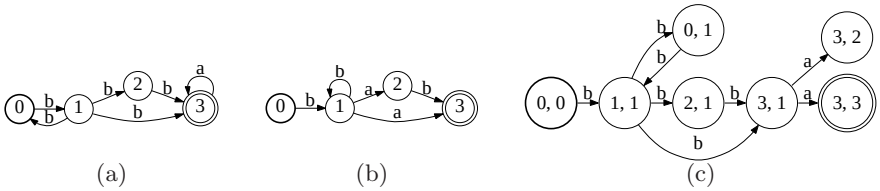

Fig. 3. Example of finite automaton intersection. (a) Finite automata A_1 and (b) A_2. (c) Result of the intersection of A_1 and A_2.

The algorithms designed for the ϵ-free case cannot be readily used for finite automata with ϵ-transitions since they would lead to incorrect results (see Figure 2(a)-(c)). Instead, [14] proposed a reduction to the ϵ-free case. First, [14] gave an algorithm to test if there exist two states p and q in A with two distinct ϵ-paths from p to q. If that is the case, then A is exponentially ambiguous (complexity $O(|A|_Q^4 + |A|_E)$). Otherwise, [14] defined from A an ϵ-free automaton A' over the alphabet $\Sigma \cup \{\#\}$ such that A is infinitely ambiguous iff A' is infinitely ambiguous, see Figure 2(d)-(e).[3] However, the number of transitions of A' is $|A|_E + |A|_Q^2$. This explains why the complexity in the ϵ-transition case is significantly worse than in the ϵ-free case. The same approach can be used to test the exponential ambiguity of A in time $O((|A|_E + |A|_Q^2)^2)$ and to compute $\mathrm{dpa}(A)$ when A is polynomially ambiguous in $O((|A|_E + |A|_Q^2)^3)$. Note that we give tighter estimates of the complexity of the algorithms of [14, 16] where the authors gave complexities using the loose inequality: $|A|_E \leq |\Sigma|\,|A|_Q^2$.

4 Algorithms

Our algorithms for testing ambiguity are based on a general algorithm for the composition or intersection of automata, which we briefly describe in the following section.

4.1 Intersection of finite automata

The intersection of finite automata is a special case of the general composition algorithm for weighted transducers [11, 10]. States in the intersection $A_1 \cap A_2$ of two finite automata A_1 and A_2 are identified with pairs of a state of A_1

[3] Observe that A' is not the result of applying the classical ϵ-removal algorithm to A, since ϵ-removal does not preserve infinite ambiguity and would lead be an even larger automaton. Instead [14] used a more complex algorithm where ϵ-transitions are replaced by regular transitions labeled with a special symbol while preserving infinite ambiguity, $\mathrm{dpa}(A) = \mathrm{dpa}(A')$, even though A' is not equivalent to A. States in A' are pairs (q, i) with q a state in A and $i \in \{1, 2\}$. There is a transition from $(p, 1)$ to $(q, 2)$ labeled by $\#$ if q belongs to the ϵ-closure of p and from $(p, 2)$ to $(q, 1)$ labeled by $\sigma \in \Sigma$ if there was such a transition from p to q in A.

and a state of A_2. The following rule specifies how to compute a transition of $A_1 \cap A_2$ in the absence of ϵ-transition from appropriate transitions of A_1 and A_2: (q_1, a, q_1') and $(q_2, a, q_2') \Longrightarrow ((q_1, q_2), a, (q_1', q_2'))$. Figure 3 illustrates the algorithm. A state (q_1, q_2) is initial (resp. final) when q_1 and q_2 are initial (resp. final). In the worst case, all transitions of A_1 leaving a state q_1 match all those of A_2 leaving state q_2, thus the space and time complexity of composition is quadratic: $O(|A_1||A_2|)$, or $O(|A_1|_E|A_2|_E)$ when A_1 and A_2 are trim.

4.2 Epsilon-filtering

A straightforward generalization of the ϵ-free case would generate redundant ϵ-paths. This is a crucial issue in the more general case of the intersection of weighted automata over a non-idempotent semiring, since it would lead to an incorrect result. The weight of two matching ϵ-paths of the original automata would then be counted as many times as the number of redundant ϵ-paths generated in the result, instead of once. It is also a crucial problem in the unweighted case since redundant ϵ-paths can affect the test of infinite ambiguity, as we shall see in the next section. A critical component of the composition algorithm of [11, 10] consists however of precisely coping with this problem using an *epsilon-filtering* mechanism.

Figure 4(c) illustrates the problem just mentioned. To match ϵ-paths leaving q_1 and those leaving q_2, a generalization of the ϵ-free intersection can make the following moves: (1) first move forward on an ϵ-transition of q_1, or even a ϵ-path, and remain at the same state q_2 in A_2, with the hope of later finding a transition whose label is some label $a \neq \epsilon$ matching a transition of q_2 with the same label; (2) proceed similarly by following an ϵ-transition or ϵ-path leaving q_2 while remaining at the same state q_1 in A_1; or, (3) match an ϵ-transition of q_1 with an ϵ-transition of q_2.

Let us rename existing ϵ-labels of A_1 as ϵ_2, and existing ϵ-labels of A_2 ϵ_1, and let us augment A_1 with a self-loop labeled with ϵ_1 at all states and similarly, augment A_2 with a self-loop labeled with ϵ_2 at all states, as illustrated by Figures 4(a) and (b). These self-loops correspond to remaining at the same state in that machine while consuming an ϵ-label of the other transition. The three moves just described now correspond to the matches (1) $(\epsilon_2 : \epsilon_2)$, (2) $(\epsilon_1 : \epsilon_1)$, and (3) $(\epsilon_2 : \epsilon_1)$. The grid of Figure 4(c) shows all the possible ϵ-paths between intersection states. We will denote by $\tilde{A}_1$ and $\tilde{A}_2$ the automata obtained after application of these changes.

For the result of intersection not to be redundant, between any two of these states, all but one path must be disallowed. There are many possible ways of selecting that path. One natural way is to select the shortest path with the diagonal transitions (ϵ-matching transitions) taken first. Figure 4(c) illustrates in boldface the path just described from state $(0, 0)$ to state $(1, 2)$. Remarkably, this filtering mechanism itself can be encoded as a finite-state transducer such as the transducer M of Figure 4(d). We denote by $(p, q) \preceq (r, s)$ to indicate that (r, s) can be reached from (p, q) in the grid.

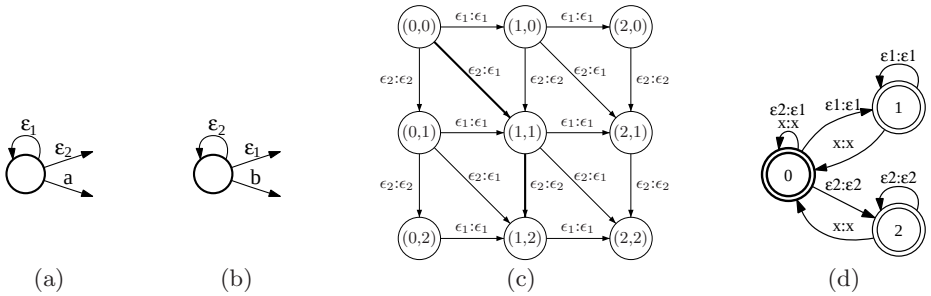

Fig. 4. Marking of automata, redundant paths and filter. (a) $\tilde{A}_1$: self-loop labeled with ϵ_1 added at all states of A_1, regular ϵs renamed to ϵ_2. (b) $\tilde{A}_2$: self-loop labeled with ϵ_2 added at all states of A_2, regular ϵs renamed to ϵ_1. (c) Redundant ϵ-paths: a straightforward generalization of the ϵ-free case could generate all the paths from $(0,0)$ to $(2,2)$ for example, even when composing just two simple transducers. (d) Filter transducer M allowing a unique ϵ-path.

Proposition 2. *Let M be the transducer of Figure 4(d). M allows a unique path between any two states (p,q) and (r,s), with $(p,q) \preceq (r,s)$.*

Proof. The full proof of this proposition is given in [2]. □

Thus, to intersect two finite automata A_1 and A_2 with ϵ-transitions, it suffices to compute $\tilde{A}_1 \circ M \circ \tilde{A}_2$, using the ϵ-free rules of composition. States in the intersection are now identified with triplets made of a state of A_1, a state of M, and a state of A_2. A transition (q_1, a_1, q_1') in $\tilde{A}_1$, a transition (f, a_1, a_2, f') in M, and a transition (q_2, a_2, q_2') in $\tilde{A}_2$ are combined to form the following transition in the intersection: $((q_1, f, q_2), a, (q_1', f', q_2'))$, with $a = \epsilon$ if $\{a_1, a_2\} \subseteq \{\epsilon_1, \epsilon_2\}$ and $a = a_1 = a_2$ otherwise. In the rest of the paper, we will assume that the result of intersection is trimmed after its computation, which can be done in linear time.

Theorem 3. *Let A_1 and A_2 be two finite automata with ϵ-transitions. To each pair (π_1, π_2) of accepting paths in A_1 and A_2 sharing the same input label $x \in \Sigma^*$ corresponds a unique accepting path π in $A_1 \cap A_2$ labeled with x.*

Proof. This follows straightforwardly from Proposition 2. □

4.3 Ambiguity Tests

We start with a test of the exponential ambiguity of A. The key is that the (EDA) property translates into a very simple property for $A^2 = A \cap A$.

Lemma 1. *Let A be a trim ϵ-cycle free finite automaton. A satisfies* (EDA) *iff there exists a strongly connected component of $A^2 = A \cap A$ that contains two states of the form (p,p) and (q,q'), where p, q and q' are states of A with $q \neq q'$.*

Proof. Assume that A satisfies (EDA). There exist a state p and a string v such that there are two distinct cycles c_1 and c_2 labeled by v at p. Let e_1 and e_2 be the first edges that differ in c_1 and c_2. We can then write $c_1 = \pi e_1 \pi_1$ and $c_2 = \pi e_2 \pi_2$. If e_1 and e_2 share the same label, let $\pi'_1 = \pi e_1$, $\pi'_2 = \pi e_2$, $\pi''_1 = \pi_1$ and $\pi''_2 = \pi_2$. If e_1 and e_2 do not share the same label, exactly one of them must be an ϵ-transition. By symmetry, we can assume without loss of generality that e_1 is the ϵ-transition. Let $\pi'_1 = \pi e_1$, $\pi'_2 = \pi$, $\pi''_1 = \pi_1$ and $\pi''_2 = \epsilon_2 \pi_2$. In both cases, let $q = n[\pi'_1] = p[\pi''_1]$ and $q' = n[\pi'_2] = p[\pi''_2]$. Observe that $q \neq q'$. Since $i[\pi'_1] = i[\pi'_2]$, π'_1 and π'_2 are matched by intersection resulting in a path in A^2 from (p, p) to (q, q'). Similarly, since $i[\pi''_1] = i[\pi''_2]$, π''_1 and π''_2 are matched by intersection resulting in a path from (q, q') to (p, p). Thus, (p, p) and (q, q') are in the same strongly connected component of A^2.

Conversely, assume that there exist states p, q and q' in A such that $q \neq q'$ and that (p, p) and (q, q') are in the same strongly connected component of A^2. Let c be a cycle in (p, p) going through (q, q'), it has been obtained by matching two cycles c_1 and c_2. If c_1 were equal to c_2, intersection would match these two paths creating a path c' along which all the states would be of the form (r, r), and since A is trim this would contradict Theorem 3. Thus, c_1 and c_2 are distinct and (EDA) holds. □

Observe that the use of the ϵ-filter in composition is crucial for Lemma 1 to hold (see Figure 2). The lemma leads to a straightforward algorithm for testing exponential ambiguity.

Theorem 4. *Let A be a trim ϵ-cycle free finite automaton. It is decidable in time $O(|A|_E^2)$ whether A is exponentially ambiguous.*

Proof. The algorithm proceeds as follows. We compute A^2 and, using a depth-first search of A^2, trim it and compute its strongly connected components. It follows from Lemma 1 that A is exponentially ambiguous iff there is a strongly connected component that contains two states of the form (p, p) and (q, q') with $q \neq q'$. Finding such a strongly connected component can be done in time linear in the size of A^2, *i.e.* in $O(|A|_E^2)$ since A and A^2 are trim. Thus, the complexity of the algorithm is in $O(|A_E|^2)$. □

Testing the (IDA) property requires finding three paths sharing the same label in A. As shown below, this can be done in a natural way using the automaton $A^3 = (A \cap A) \cap A$, obtained by applying twice the intersection algorithm.

Lemma 2. *Let A be a trim ϵ-cycle free finite automaton. A satisfies* (IDA) *iff there exist two distinct states p and q in A with a non-ϵ path in $A^3 = A \cap A \cap A$ from state (p, p, q) to state (p, q, q).*

Proof. Assume that A satisfies (IDA). Then, there exists a string $v \in \Sigma^*$ with three paths $\pi_1 \in P(p, v, p)$, $\pi_2 \in P(p, v, q)$ and $\pi_3 \in P(q, v, p)$. Since these three paths share the same label v, they are matched by intersection resulting in a path π in A^3 labeled with v from $(p[\pi_1], p[\pi_2], p[\pi_3]) = (p, p, q)$ to $(n[\pi_1], n[\pi_2], n[\pi_3]) = (p, q, q)$.

Conversely, if there is a non-ϵ path π form (p,p,q) to (p,q,q) in A^3, it has been obtained by matching three paths π_1, π_2 and π_3 in A with the same input $v = i[\pi] \neq \epsilon$. Thus, (IDA) holds. □

This lemma appears already as Lemma 5.10 in [8]. Finally, Theorem 4 and Lemma 2 can be combined to yield the following result.

Theorem 5. *Let A be a trim ϵ-cycle free finite automaton. It is decidable in time $O(|A|_E^3)$ whether A is finitely, polynomially, or exponentially ambiguous.*

Proof. First, Theorem 4 can be used to test whether A is exponentially ambiguous by computing A^2. The complexity of this step is $O(|A|_E^2)$.

If A is not exponentially ambiguous, we proceed by computing and trimming A^3 and then testing whether A^3 verifies the property described in Lemma 2. This is done by considering the automaton B on the alphabet $\Sigma' = \Sigma \cup \{\#\}$ obtained from A^3 by adding a transition labeled by # from state (p,q,q) to state (p,p,q) for every pair (p,q) of states in A such that $p \neq q$. It follows that A^3 verifies the condition in Lemma 2 iff there is a cycle in B containing both a transition labeled by # and a transition labeled by a symbol in Σ. This property can be checked straightforwardly using a depth-first search of B to compute its strongly connected components. If a strongly connected component of B is found that contains both a transition labeled with # and a transition labeled by a symbol in Σ, A verifies (IDA) but not (EDA) and thus A is polynomially ambiguous. Otherwise, A is finitely ambiguous. The complexity of this step is linear in the size of B: $O(|B|_E) = O(|A_E|^3 + |A_Q|^2) = O(|A_E|^3)$ since A and B are trim.

The total complexity of the algorithm is $O(|A|_E^2 + |A|_E^3) = O(|A|_E^3)$.

When A is polynomially ambiguous, we can derive from the algorithm just described one that computes $\mathrm{dpa}(A)$.

Theorem 6. *Let A be a trim ϵ-cycle free finite automaton. If A is polynomially ambiguous,* $\mathrm{dpa}(A)$ *can be computed in time $O(|A|_E^3)$.*

Proof. We first compute A^3 and use the algorithm of Theorem 5 to test whether A is polynomially ambiguous and to compute all the pairs (p,q) that verify the condition of Lemma 2. This step has complexity $O(|A|_E^3)$.

We then compute the component graph G of A, and for each pair (p,q) found in the previous step, we add a transition labeled with # from the strongly connected component of p to the one of q. If there is a path in that graph containing d edges labeled by #, then A verifies (IDA_d). Thus, $\mathrm{dpa}(A)$ is the maximum number of edges marked by # that can be found along a path in G. Since G is acyclic, this number can be computed in linear time in the size of G, *i.e.* in $O(|A|_Q^2)$. Thus, the overall complexity of the algorithm is $O(|A|_E^3)$. □

5 Application to Entropy Approximation

In this section, we describe an application in which determining the degree of ambiguity of a *probabilistic* automaton helps estimate the quality of an approximation of its entropy. Weighted automata are automata in which each transition

carries some weight in addition to the usual alphabet symbol. The weights are elements of a semiring, that is a ring that may lack negation. The following is a more formal definition.

Definition 3. *A* weighted automaton A *over a semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is a* 7*-tuple* $(\Sigma, Q, I, F, E, \lambda, \rho)$ *where* Σ *is a finite alphabet,* Q *a finite set of states,* $I \subseteq Q$ *the set of initial states,* $F \subseteq Q$ *the set of final states,* $E \subseteq Q \times \Sigma \cup \{\epsilon\} \times \mathbb{K} \times Q$ *a finite set of transitions,* $\lambda : I \to \mathbb{K}$ *the initial weight function mapping* I *to* $\mathbb{K}$*, and* $\rho : F \to \mathbb{K}$ *the final weight function mapping* F *to* $\mathbb{K}$*.*

Given a transition $e \in E$, we denote by $w[e]$ its weight. We extend the weight function w to paths by defining the weight of a path as the $\otimes$-product of the weights of its constituent transitions: $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. The weight associated by a weighted automaton A to an input string $x \in \Sigma^*$ is defined by $[\![A]\!](x) = \bigoplus_{\pi \in P(I,x,F)} \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]]$. The entropy $H(A)$ of a probabilistic automaton A is defined as:

$$H(A) = -\sum_{x \in \Sigma^*} [\![A]\!](x) \log([\![A]\!](x)). \tag{1}$$

The system $(\mathbb{K}, \oplus, \otimes, (0,0), (1,0))$ with $\mathbb{K} = (\mathbb{R} \cup \{+\infty, -\infty\}) \times (\mathbb{R} \cup \{+\infty, -\infty\})$ and $\oplus$ and $\otimes$ defined as follows defines a commutative semiring called the *entropy semiring* [4]: for any two pairs (x_1, y_1) and (x_2, y_2) in $\mathbb{K}$, $(x_1, y_1) \oplus (x_2, y_2) = (x_1 + x_2, y_1 + y_2)$ and $(x_1, y_1) \otimes (x_2, y_2) = (x_1 x_2, x_1 y_2 + x_2 y_1)$. In [4], the authors showed that a generalized shortest-distance algorithm over this semiring correctly computes the entropy of an unambiguous probabilistic automaton A. The algorithm starts by mapping the weight of each transition to a pair where the first element is the probability and the second the entropy: $w[e] \mapsto (w[e], -w[e] \log w[e])$. The algorithm then proceeds by computing the generalized shortest-distance defined over the entropy semiring, which computes the $\oplus$-sum of the weights of all accepting paths in A.

Here, we show that the same shortest-distance algorithm yields an approximation of the entropy of an ambiguous probabilistic automaton A, where the approximation quality is a function of the degree of polynomial ambiguity, $\mathrm{dpa}(A)$. Our proofs make use of the standard log-sum inequality [5], a special case of Jensen's inequality, which holds for any positive reals $a_1, \ldots, a_k$, and $b_1, \ldots, b_k$:

$$\sum_{i=1}^{k} a_i \log \frac{a_i}{b_i} \geq \left(\sum_{i=1}^{k} a_i \right) \log \frac{\sum_{i=1}^{k} a_i}{\sum_{i=1}^{k} b_i}. \tag{2}$$

Lemma 3. *Let* A *be a probabilistic automaton and let* $x \in \Sigma^+$ *be a string accepted by* A *on* k *paths* $\pi_1, \ldots, \pi_k$*. Let* $w[\pi_i]$ *be the probability of path* π_i*. Clearly,* $[\![A]\!](x) = \sum_{i=1}^{k} w[\pi_i]$*. Then,* $\sum_{i=1}^{k} w[\pi_i] \log w[\pi_i] \geq [\![A]\!](x)(\log [\![A]\!](x) - \log k)$.

Proof. The result follows straightforwardly from the log-sum inequality, with $a_i = w[\pi_i]$ and $b_i = 1$:

$$\sum_{i=1}^{k} w[\pi_i] \log w[\pi_i] \geq \left(\sum_{i=1}^{k} w[\pi_i] \right) \log \frac{\sum_{i=1}^{k} w[\pi_i]}{k} = [\![A]\!](x)(\log [\![A]\!](x) - \log k). \tag{3}$$

□

Let $S(A)$ be the quantity computed by the generalized shortest-distance algorithm over the entropy semiring or a probabilistic automaton A. When A is unambiguous, it is shown by [4] that $S(A) = H(A)$.

Theorem 7. *Let A be a probabilistic automaton and let L denote the expected length of the strings accepted by A (i.e. $L = \sum_{x\in\Sigma^*} |x|[\![A]\!](x)$). Then,*

1. *if A is* finitely ambiguous *with* $\mathrm{da}(A) = k$ *for some $k \in \mathbb{N}$, then $H(A) \leq S(A) \leq H(A) + \log k$;*
2. *if A is* polynomially ambiguous *with* $\mathrm{dpa}(A) = k$ *for some $k \in \mathbb{N}$, then $H(A) \leq S(A) \leq H(A) + k \log L$.*

Proof. The lower bound $S(A) \geq H(A)$ follows from the observation that for a string x that is accepted in A by k paths $\pi_1, \ldots, \pi_k$,

$$\sum_{i=1}^{k} w[\pi_i] \log(w(\pi_i)) \leq \left(\sum_{i=1}^{k} w[\pi_i]\right) \log\left(\sum_{i=1}^{k} w[\pi_i]\right). \tag{4}$$

Since the quantity $-\sum_{i=1}^{k} w[\pi_i] \log(w[\pi_i])$ is string x's contribution to $S(A)$ and the quantity $-(\sum_{i=1}^{k} w[\pi_i]) \log(\sum_{i=1}^{k} w[\pi_i])$ its contribution to $H(A)$, summing over all accepted strings x, we obtain $H(A) \leq S(A)$.

Assume that A is finitely ambiguous with degree of ambiguity k. Let $x \in \Sigma^*$ be a string that is accepted on $l_x \leq k$ paths $\pi_1, \ldots, \pi_{l_x}$. By Lemma 3, we have $\sum_{i=1}^{l_x} w[\pi_i] \log w[\pi_i] \geq [\![A]\!](x)(\log [\![A]\!](x) - \log l_x) \geq [\![A]\!](x)(\log [\![A]\!](x) - \log k)$. Thus, $S(A) = -\sum_{x\in\Sigma^*} \sum_{i=1}^{l_x} w[\pi_i] \log w[\pi_i] \leq H(A) + \sum_{x\in\Sigma^*} (\log k)[\![A]\!](x) = H(A) + \log k$. This proves the first statement of the theorem.

Next, assume that A is polynomially ambiguous with degree of polynomial ambiguity k. By Lemma 3, we have $\sum_{i=1}^{l_x} w[\pi_i] \log w[\pi_i] \geq [\![A]\!](x)(\log [\![A]\!](x) - \log l_x) \geq [\![A]\!](x)(\log [\![A]\!](x) - \log(|x|^k))$. Thus,

$$\begin{aligned} S(A) &\leq H(A) + \sum_{x\in\Sigma^*} k[\![A]\!](x) \log |x| = H(A) + k\mathbb{E}_A[\log |x|] \qquad (5)\\ &\leq H(A) + k \log \mathbb{E}_A[|x|] = H(A) + k \log L, \qquad \text{(by Jensen's inequality)} \end{aligned}$$

which proves the second statement of the theorem. □

The theorem shows in particular that the quality of the approximation of the entropy of a polynomially ambiguous probabilistic automaton can be estimated by computing its degree of polynomial ambiguity, which can be achieved efficiently as described in the previous section. This also requires the computation of the expected length L of an accepted string. L can be computed efficiently for an arbitrary probabilistic automaton using the entropy semiring and the generalized shortest-distance algorithms, using techniques similar to those described in [4]. The only difference is in the initial step, where the weight of each transition in A is mapped to a pair of elements by $w[e] \mapsto (w[e], w[e])$.

6 Conclusion

We presented simple and efficient algorithms for testing the finite, polynomial, or exponential ambiguity of finite automata with ϵ-transitions. We conjecture that the time complexity of our algorithms is optimal. These algorithms have a variety of applications, in particular to test a pre-condition for the applicability of other automata algorithms. Our application to the approximation of the entropy gives another illustration of their usefulness. Our algorithms also demonstrate the prominent role played by the intersection or composition of automata and transducers with ϵ-transitions [11, 10] in the design of *testing algorithms*. Composition can be used to devise simple and efficient testing algorithms. We have shown elsewhere how it can be used to test the functionality of a finite-state transducer, or the twins property for weighted automata and transducers [1].

References

1. C. Allauzen and M. Mohri. Efficient Algorithms for Testing the Twins Property. *Journal of Automata, Languages and Combinatorics*, 8(2):117–144, 2003.
2. C. Allauzen and M. Mohri. 3-way composition of weighted finite-state transducers. In *CIAA 2008*, volume 5148 of *LNCS*, pages 262–273. Springer, 2008.
3. T. Chan and O. H. Ibarra. On the finite-valuedness problem for sequential machines. *Theoretical Computer Science*, 23:95–101, 1983.
4. C. Cortes, M. Mohri, A. Rastogi, and M. Riley. Efficient computation of the relative entropy of probabilistic automata. In *LATIN 2006*, volume 3887 of *LNCS*, pages 323–336. Springer, 2006.
5. T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, 1991.
6. O. H. Ibarra and B. Ravikumar. On sparseness, ambiguity and other decision problems for acceptors and transducers. In *STACS 1986*, volume 210 of *LNCS*, pages 171–179. Springer, 1986.
7. G. Jacob. Un algorithme calculant le cardinal, fini ou infini, des demi-groupes de matrices. *Theoretical Computer Science*, 5(2):183–202, 1977.
8. W. Kuich. Finite automata and ambiguity. Technical Report 253, Institute für Informationsverarbeitung - Technische Universität Graz und ÖCG, 1988.
9. A. Mandel and I. Simon. On finite semigroups of matrices. *Theoretical Computer Science*, 5(2):101–111, 1977.
10. M. Mohri, F. C. N. Pereira, and M. Riley. Weighted Automata in Text and Speech Processing. In *Proceedings of ECAI-96, Workshop on Extended finite state models of language, Budapest, Hungary*. John Wiley and Sons, 1996.
11. F. Pereira and M. Riley. *Finite State Language Processing*, chapter Speech Recognition by Composition of Weighted Finite Automata. The MIT Press, 1997.
12. B. Ravikumar and O. H. Ibarra. Relating the type of ambiguity of finite automata to the succintness of their representation. *SIAM Journal on Computing*, 18(6):1263–1282, 1989.
13. C. Reutenauer. Propriétés arithmétiques et topologiques des séries rationnelles en variable non commutative. Thèse de troisième cycle, Université Paris VI, 1977.
14. A. Weber. Über die Mehrdeutigkeit und Wertigkeit von endlichen, Automaten und Transducern. Dissertation, Goethe-Universität Frankfurt am Main, 1987.
15. A. Weber and H. Seidl. On the degree of ambiguity of finite automata. In *MFCS 1986*, volume 233 of *LNCS*, pages 620–629. Springer, 1986.
16. A. Weber and H. Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991.