International Journal of Foundations of Computer Science
© World Scientific Publishing Company

General Algorithms for Testing the Ambiguity of Finite Automata and the Double-Tape Ambiguity of Finite-State Transducers

CYRIL ALLAUZEN*

Google Research,
76 Ninth Avenue, New York, NY 10011, US.
allauzen@google.com

MEHRYAR MOHRI

Courant Institute of Mathematical Sciences,
251 Mercer Street, New York, NY 10012, US,
and
Google Research,
76 Ninth Avenue, New York, NY 10011, US.
mohri@cs.nyu.edu

ASHISH RASTOGI*

Goldman, Sachs & Co.,
200 West Street, New York, NY 10282, US.
ashish.rastogi@gs.com

We present efficient algorithms for testing the finite, polynomial, and exponential ambiguity of finite automata with ϵ-transitions. We give 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. Additionally, we give an algorithm to determine in time $O(|A|_E^3)$ the degree of polynomial ambiguity of a polynomially ambiguous automaton A and present an application of our algorithms to an approximate computation of the entropy of a probabilistic automaton.

We also study the double-tape ambiguity of finite-state transducers. We show that the general problem is undecidable and that it is NP-hard for acyclic transducers. We present a specific analysis of the double-tape ambiguity of transducers with bounded delay. In particular, we give a characterization of double-tape ambiguity for synchronized transducers with zero delay that can be tested in quadratic time and give an algorithm for testing the double-tape ambiguity of transducers with bounded delay.

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

1. Introduction

A finite automaton is ambiguous if it admits distinct accepting paths with the same label. 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 growth rate 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 growth rate.

The problem of testing ambiguity has been extensively analyzed in the past [10, 8, 17, 3, 7, 19, 16, 18, 20]. 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 [7] and Weber and Seidel [19]. The most efficient algorithms for testing polynomial and exponential ambiguity, thereby testing finite ambiguity, were given by Weber and Seidel [18, 20]. The algorithms they presented in [20] assume the input automaton to be ϵ-free, but they are extended by Weber to the case where the automaton has ϵ-transitions in [18]. In the presence of ϵ-transitions, the complexity of the algorithms given by Weber [18] 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 [14, 13]. 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 and present an application of our algorithms to an approximate computation of the entropy of a probabilistic automaton.

The notion of ambiguity is defined in a similar way for finite-state transducers if one is only interested in the ambiguity with respect to the input labels, or only the output labels, of a transducer. With that definition, all our results for automata apply directly to the transducer case as well. There is, however, another notion of interest for transducers that relates to both input and output labels and that we refer to as the *double-tape ambiguity* of a transducer. A transducer is double-tape ambiguous if it admits two distinct accepting paths with the same input label and the same output label. Double-tape ambiguity can lead to inefficiencies in a variety

of applications where transducers are now commonly used, e.g., machine translation, speech recognition, other language processing areas, and image processing.

This motivates our study of the double-tape ambiguity of finite-state transducers. We show that the general problem of double-tape ambiguity is undecidable and that it is NP-hard even for acyclic transducers. We also present a specific analysis of the double-tape ambiguity of transducers with bounded delay. In particular, we give a characterization of double-tape ambiguity for synchronized transducers with zero delay that can be tested in quadratic time and give an algorithm for testing the double-tape ambiguity of transducers with bounded delay.

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 deals with questions related to the double-tape ambiguity of finite-state transducers. Section 6 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 state $q \in Q$ is *accessible* if there exists a path from an initial state to q and *co-accessible* if there exists a path from q to a final state.

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, that is if every state is both accessible and co-accessible. 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 lying along an accepting path, there exists $x \in \Sigma^*$ such that $\mathrm{da}(A, x) = \infty$. Using a depth-first search of A restricted to ϵ-transitions, it can be decided in linear time if A contains such ϵ-cycles. Thus, in the following, we will assume, without loss of generality, that A is ϵ-cycle free.

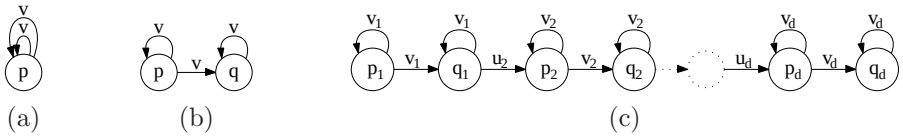

Fig. 1. Illustration of the properties: (a) (EDA); (b) (IDA); and (c) (IDA$_d$).

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 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 [7, 19, 18, 20].

Definition 2. *The properties (EDA), (IDA), and (IDA$_d$) 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))* [7].
- *(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))* [19, 18, 20].
- *(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))* [19, 18, 20].

Observe that (EDA) implies (IDA) as shown below. Indeed, 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 [19, 20], that A satisfies (EDA) iff A is exponentially ambiguous [7], and that A satisfies (IDA$_d$) iff $\mathrm{dpa}(A) \geq d$ [18, 20].

In the following, we show that these results can be extended to the case of automata with ϵ-transitions. To simplify the proofs, we first consider the case of multiset automata.

A *multiset automaton* or *m-automaton* is a 5-tuple (Σ, Q, E, I, F) as defined in Definition 1 except that E and F are multisets. We will denote by $\uplus$ the union of two

multisets ($\{1,2\} \uplus \{1,3\} = \{1,1,2,3\}$), by $\otimes$ the scalar multiplication of a multiset by a natural number ($2 \otimes \{1,1,2\} = \{1,1,1,1,2,2\}$), by $|X|_a$ the multiplicity of element a in the multiset X ($|\{1,1,2\}|_1 = 2$) and by $|X|$ the cardinality ($|\{1,1,2\}| = 3$) of X.

Lemma 3. *Let A be a trim ϵ-free m-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. Given a trim m-automaton $A = (\Sigma, Q, E, I, F)$, we construct a finite automaton $A' = (\Sigma \cup \{\#\}, Q', E', I, F')$ by inserting a transition labeled with # after each transition and from each final state as follows:

$$\begin{aligned} Q' &= Q \cup Q_E \cup Q_F \text{ with } Q_E = \{q_e \mid e \in E\} \text{ and } Q_F = \{q_f \mid f \in F\}, \\ E' &= \bigcup_{e \in E} \{(p[e], i[e], q_e), (q_e, \#, n[e])\} \cup \bigcup_{f \in F} \{(f, \#, q_f)\}, \text{ and} \\ F' &= Q_F. \end{aligned}$$

Observe that the cardinality of the set Q_E (resp. Q_F) is equal to the cardinality of the multiset E (resp. F). Each state q_E has only one incoming and one outgoing transition. The mapping $\alpha_E : e \mapsto (p[e], i[e], q_e)(q_e, \#, n[e])$ is an injection from E into E'^2 and the mapping $\alpha_F : f \mapsto (f, \#, q_f)$ an injection from F into E'.

Several key properties follow from the existence of these injections. (1) A' is trim since A is trim (follows from the existence of α_E and α_F). (2) There exists an injection $\beta : e_1 \dots e_n \mapsto \alpha_E(e_1) \dots \alpha_E(e_n)$ from the set of paths in A to the set of paths in A' such that the following conditions are equivalent: (a) (IDA) (resp. (EDA), (IDA$_d$)) holds for A, (b) (IDA) (resp. (EDA), (IDA$_d$)) holds for all paths in the image of β and (c) (IDA) (resp. (EDA), (IDA$_d$)) holds for A'. (3) The mapping $\gamma : x_1 x_2 \dots x_n \mapsto x_1 \# x_2 \# \dots x_n \# \#$ is a bijection from the language accepted by A to the language accepted by A' and (4) $\mathrm{da}(A, x) = \mathrm{da}(A', \gamma(x))$ for all $x \in \Sigma^*$ since the mapping $\delta : \pi \mapsto \beta(\pi)\alpha_F(n[\pi])$ is a bijection between the sets of accepting paths of A and A' such that $i[\delta(\pi)] = \gamma(i[\pi])$.

The proposition holds for A' since A' is a standard trim automaton as shown in [19, 20] for (i), [7] for (ii) and [20] for (iii). Hence, it follows from (2) and (4) that the proposition also hold for A. □

We will now show that Lemma 3 can be generalized to the case of m-automata with ϵ-transitions.

Lemma 4. *Let A be a trim ϵ-cycle free m-automaton.*

(i) A is infinitely ambiguous iff A satisfies (IDA).
(ii) A is exponentially ambiguous iff A satisfies (EDA).

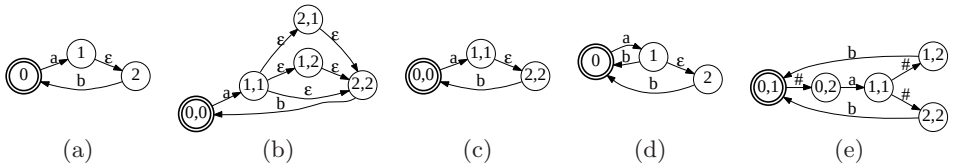

Fig. 2. *ϵ-filter and ambiguity*: (a) Finite automaton A; (b) $A \cap A$ without using ϵ-filter, which incorrectly makes A appear as exponentially ambiguous; (c) $A \cap A$ using an ϵ-filter. *Weber's processing of ϵ-transitions*: (d) Finite automaton B; (e) ϵ-free automaton B' such that $\mathrm{dpa}(B)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

(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 and follows from Lemma 3.

Assume now that A has $n+1$ ϵ-transitions, $n \geq 0$, and that the statement of the proposition holds for all m-automata with n ϵ-transitions. Select an ϵ-transition e_0 in A such that there are no outgoing ϵ-transitions in $n[e_0]$. Such a transition must exist since A is ϵ-cycle free. Let A' be the m-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 $e \in E[n[e_0]]$, i.e. the multiset E' of transitions of A' is defined as:

$$E' = (E \setminus \{e_0\}) \uplus \{(p[e_0], l[e], n[e]) \mid e \in E \text{ such that } p[e] = n[e_0]\}.$$

Finally, $p[e_0]$ is added to the multiset of final states as many times as the multiplicity of $n[e_0]$ in F, i.e. the multiset F' of final states of A' is defined as:

$$F' = F \uplus (|F|_{n[e_0]} \otimes \{p[e_0]\}).$$

It is clear that A and A' are equivalent and that there is a label and acceptance-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, Lemma 4 holds for A' and thus, it follows from (a) and (b) that Lemma 4 also holds for A. □

The case of finite automata with ϵ-transitions then follows as a corollary of Lemma 4.

Proposition 5. *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$_d$).

These characterizations have been used in [18, 20] to design algorithms for testing infinite, polynomial, and exponential ambiguity, and for computing the degree of

polynomial ambiguity in the case of ϵ-free finite automata.

Theorem 6 ([18, 20]) *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 6 has also been generalized by [18] to the case of automata with ϵ-transitions but with a significantly worse complexity.

Theorem 7 ([18]) *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.*

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, [18] proposed a reduction to the ϵ-free case. First, [18] 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, [18] 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).[a] 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 [18, 20] 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.

[a]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 to an even larger automaton. Instead, [18] 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.

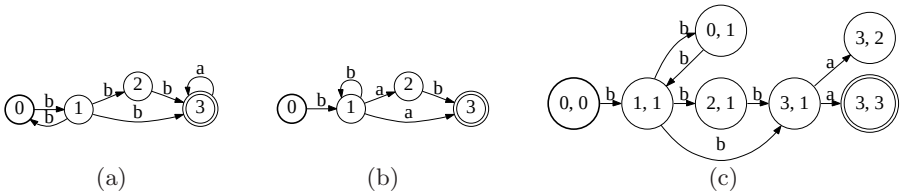

Fig. 3. Example of finite automaton intersection. (a) Finite automata A_1 and (b) A_2. (c) Result of the intersection of A_1 and A_2.

4.1. *Intersection of finite automata*

The intersection of finite automata is a special case of the general composition algorithm for weighted transducers [14, 13]. 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 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: $(q_1, a, q_1') \text{ and } (q_2, a, q_2') \Longrightarrow ((q_1, q_2), a, (q_1', q_2'))$. 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 [14, 13] 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 as ϵ_1, and let us augment A_1 with a self-loop labeled with ϵ_1 at all states and similarly, augment

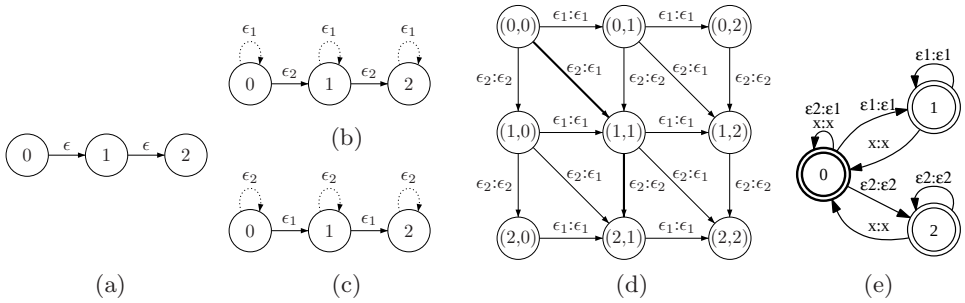

Fig. 4. Marking of automata, redundant paths and filter. (a) Automaton $A_1 = A_2$. (b) $\tilde{A}_1$: self-loop labeled with ϵ_1 added at all states of A_1, regular ϵs renamed to ϵ_2. (c) $\tilde{A}_2$: self-loop labeled with ϵ_2 added at all states of A_2, regular ϵs renamed to ϵ_1. (d) Redundant ϵ-paths: 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 ($A_1 \circ A_2$). (e) Filter transducer M allowing a unique ϵ-path. Each transition labeled $x : x$ represents transitions with input and output x of all x in Σ.

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 $(2,1)$. 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.

Proposition 8. *Let M be the transducer of Figure 4(d). M allows a unique ϵ-path between any two states (p,q) and (r,s), with $(p,q) \preceq (r,s)$.*

Proof. The proof of this proposition was previously given in [2]. Let a denote $(\epsilon_1 : \epsilon_1)$, b denote $(\epsilon_2 : \epsilon_2)$, c denote $(\epsilon_2 : \epsilon_1)$, and let x stand for any $(x : x)$, with $x \in \Sigma$. The following sequences must be disallowed by a shortest-path filter with matching transitions first: ab, ba, ac, bc. This is because, from any state, instead of the moves ab or ba, the matching or diagonal transition c can be taken. Similarly, instead of ac or bc, ca and cb can be taken for an earlier match. Conversely, it is clear from the grid or an immediate recursion that a filter disallowing these sequences accepts a unique path between two connected states of the grid.

Let L be the set of sequences over $\sigma = \{a, b, c, x\}$ that contain one of the

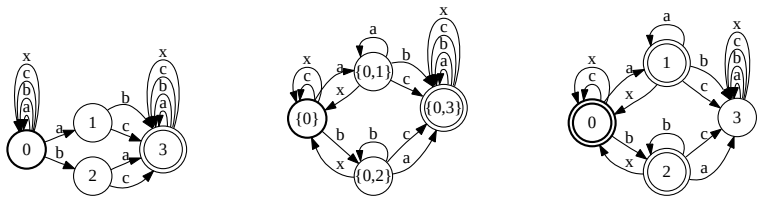

Fig. 5. (a) Finite automaton A representing the set of disallowed sequences. (b) Automaton B, result of the determinization of A. Subsets are indicated at each state. (c) Automaton C obtained from B by complementation, state 3 is not coaccessible.

disallowed sequence just mentioned as a substring that is $L = \sigma^*(ab+ba+ac+bc)\sigma^*$. Then $\overline{L}$ represents exactly the set of paths allowed by that filter and is thus a regular language. Let A be an automaton representing L (Figure 5(a)). An automaton representing $\overline{L}$ can be constructed from A by determinization and complementation (Figures 5(a)-(c)). The resulting automaton C is equivalent to the transducer M after removal of the state 3, which does not admit a path to a final state. □

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 (see section 5 for a formal definition of the composition of finite-state transducers). 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 in the size of the result of intersection.

Theorem 9. *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 8. □

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 $A^2 = A \cap A$. A state in A^2 is a triple (p, f, q), denoted by $(p, q)_f$ in the following, where p and q are states in A and f is a filter state.

Lemma 10. *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 $(p,p)_0$ and $(q,q')_f$, 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 $\pi''_2 = e_2\pi_2$. 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 $(p,p)_0$ to $(q,q')_f$. Similarly, since $i[\pi''_1] = i[\pi''_2]$, π''_1 and π''_2 are matched by intersection resulting in a path from $(q,q')_f$ to $(p,p)_0$. Thus, $(p,p)_0$ and $(q,q')_f$ 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 $(p,p)_0$ and $(q,q')_f$ are in the same strongly connected component of A^2. Let c be a cycle in $(p,p)_0$ going through $(q,q')_f$, c 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 $(r,r)_0$ making c' distinct from c, and since A is trim this would contradict Theorem 9. 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 10 to hold (see Figure 2). The lemma leads to a straightforward algorithm for testing exponential ambiguity.

Theorem 11. *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 10 that A is exponentially ambiguous iff there is a strongly connected component that contains two states of the form $(p,p)_0$ and $(q,q')_f$ 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. A state in A^3 is a 5-tuple (p,f,q,g,r), denoted by $(p,q,r)_{f,g}$ in the following, where p, q and r are states in A and f and g are filter states.

Lemma 12. *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 $(p, p, q)_{f.f'}$ to state $(p, q, q)_{g,g'}$.*

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 $(p[\pi_1], p[\pi_2], p[\pi_3])_{f,f'} = (p, p, q)_{f,f'}$ to $(n[\pi_1], n[\pi_2], n[\pi_3])_{g,g'} = (p, q, q)_{g,g'}$.

Conversely, if there is a non-ϵ path π from $(p, p, q)_{f,f'}$ to $(p, q, q)_{g,g'}$ 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 [9]. Finally, Theorem 11 and Lemma 12 can be combined to yield the following result.

Theorem 13. *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 11 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 12. 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 $(p, q, q)_{g,g'}$ to state $(p, p, q)_{f,f'}$ for every pair (p, q) of states in A such that $p \neq q$. It follows that A^3 verifies the condition in Lemma 12 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 14. *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 13 to test whether A is polynomially ambiguous and to compute all the pairs (p, q) that verify the condition of Lemma 12. 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, 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 $O(|A|_E^3)$. □

Finally, let us point out that A^2 can also be used to devise a simple test for the ambiguity of A based on the following observation.

Lemma 15. *Let A be a trim ϵ-cycle free finite automaton. A is unambiguous iff every coaccessible state in $A^2 = A \cap A$ is of the form $(p, p)_0$.*

Proof. Assume A is unambiguous and let $(p, q)_f$ be a coaccessible state in A^2. Since A_2 has been trimmed,[b] $(p, q)_f$ is both accessible and coaccessible. Hence, there exist a path π from the initial state to a final state of A^2 that goes through $(p, q)_f$. This path was obtained by matching two accepting paths π_1 and π_2 with the same label with π_1 going through p and π_2 going through q. If $p \neq q$ or $f \neq 0$, then π_1 and π_2 are distinct (by Theorem 9) and this contradicts A unambiguous. Hence, $p = q$ and $f = 0$.

Conversely, let us assume that every coaccessible state in A^2 is of the form $(p, p)_0$. Let us consider two accepting paths π_1 and π_2 sharing the same label. These two paths will be matched by composition to form an accepting path π in A^2. Since there cannot be multiple transitions with the same label between a given pair of states, the fact that all states along π are of the form $(p, p)_0$ implies that $\pi_1 = \pi_2$. Hence, A is unambiguous. □

Observe that here again the use of the ϵ-filter in composition is crucial for Lemma 15 to hold (see Figure 2).

Theorem 16. *Let A be a trim ϵ-cycle free finite automaton. It is decidable in time $O(|A|_E^2)$ whether A is ambiguous.*

Proof. The algorithms proceeds as follows. We first compute A^2 and perform a depth-first search to trim it. We can now check in $O(|A^2|_Q)$ time that each state is of the form $(p, p)_0$. Thus, the complexity of the algorithm is in $O(|A|_E^2)$. □

5. Double-Tape Ambiguity

The previous sections presented a comprehensive study of the ambiguity of finite automata. The notion of ambiguity is typically defined in the same way for finite-state transducers: a transducer is said to be ambiguous if it admits two accepting

[b] As mentioned in section 4.2, we always trim the result of intersection.

paths with the same input label. Thus, the results of the previous sections apply to the transducer case identically with that notion of ambiguity.

There is however another notion of ambiguity related to both tapes of a transducer that is of interest in applications, which we refer to as *double-tape ambiguity*. This section deals with that notion of double-tape ambiguity. It gives general decidability and hardness results for double-tape ambiguity, and presents a specific analysis for the case of transducers with bounded delay, including characterizations and algorithms for testing the double-tape ambiguity of such transducers. We start with the standard definition of a finite-state transducer.

Definition 17 (Finite-state transducers) *A* finite-state transducer T *is a 6-tuple* $(\Sigma, \Delta, Q, E, I, F)$ *where* Σ *is a finite input alphabet of the transducer;* 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; and* $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times Q$ *a finite set of transitions.*

We say that the transducer T accepts a pair $(x, y) \in \Sigma^* \times \Delta^*$ if T admits an accepting path with input label x and output label y and denote this by $(x, y) \in R(T)$. $R(T)$ is the rational relation defined by T.

Given a transducer T, we define the *inverse of* T, denoted by T^{-1}, the transducer obtained by swapping the input and output labels of each transition in T, thus $(x, y) \in R(T^{-1})$ iff $(y, x) \in R(T)$.

Let T_1 and T_2 be two finite-state transducers such that the input alphabet of T_2 coincides with the output alphabet of T_1. The result of the *composition of* T_1 *and* T_2 is a finite-state transducer denoted by $T_1 \circ T_2$ and specified for all x, y by: $(x, y) \in R(T_1 \circ T_2)$ iff there exists z such that $(x, z) \in R(T_1)$ and $(z, y) \in R(T_2)$.

The algorithm to compute the composition of two finite-state transducers is a slight modification of the intersection algorithm described in section 4. The following rule specifies how to compute a transition of $T_1 \circ T_2$ from appropriate transitions of T_1 and T_2 in the absence of output-ϵ transitions in T_1 and input-ϵ transitions in T_2: (q_1, a, b, q_1') and $(q_2, b, c, q_2') \implies ((q_1, q_2), a, c, (q_1', q_2'))$. The same epsilon-filtering technique described in section 4.2 is then used to deal with output-ϵ transitions in T_1 and input-ϵ transitons in T_2 [14, 13]. The notion of double-tape unambiguous transducers is defined as follows.

Definition 18 (Double-Tape Unambiguous Transducer) *A transducer* T *is said to be* double-tape unambiguous *if for all* $(x, y) \in \Sigma^* \times \Delta^*$*, it admits at most one accepting path in* T *with input label* x *and output label* y*.*

This notion clearly differs from the single-tape notion discussed in the previous sections for automata and often used for transducers. A transducer admitting multiple paths with the same input label x can still be double-tape unambiguous so long as the output labels of those paths are all distinct. The general problem of determining double-tape ambiguity turns out to be considerably harder than that of determining single-tape ambiguity however.

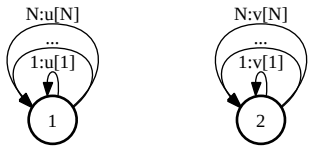

Fig. 6. The transducer constructed corresponding to a PCP problem with lists of strings $u_i, v_i \in \Sigma^*$ for $1 \leq i \leq N$. Both states 1 and 2 are initial and final.

5.1. *Undecidability Result*

We show that the general problem of determining if a transducer T is double-tape ambiguous is undecidable. When we restrict the transducer to be acyclic, then the problem becomes NP-hard. Our reduction is from the Post Correspondence Problem (PCP) [15].

Definition 19 (The Post Correspondence Problem [15]) *Given two list of strings $u_1, u_2, \ldots, u_N$ and $v_1, v_2, \ldots, v_N$, with $u_i, v_i \in \Sigma^*$ for $1 \leq i \leq N$, determine whether there exists a sequence of indices $(i_1, i_2, \ldots, i_K)$ with $K \geq 1$ and $1 \leq i_k \leq N$ such that:*

$$u_{i_1} u_{i_2} \ldots u_{i_K} = v_{i_1} v_{i_2} \ldots v_{i_K}.$$

Theorem 20 ([15, 11]) *PCP is undecidable in general. Furthermore, the problem remains undecidable even when restricted to a fixed number of strings in $(u_i)_{i=1}^N, (v_i)_{i=1}^N$, for $N \geq 7$.*

Theorem 21. *The problem of determining the double-tape ambiguity of an arbitrary finite-state transducer T is undecidable.*

Proof. Given a PCP problem instance over the alphabet Σ with strings $(u_i)_{i=1}^N$ and $(v_i)_{i=1}^N$, we construct a transducer T such that T is double-tape ambiguous if and only if the PCP problem has a solution. The transducer T is defined as follows (see Figure 6):

- The set of states $Q = \{1, 2\}$ with $I = F = Q$.
- The set of transitions E as:

  $$E = \{(1, i, u_i, 1) : 1 \leq i \leq N\} \cup \{(2, i, v_i, 2) : 1 \leq i \leq N\},$$

  where (q_i, a, b, q_j) denotes a transition from state q_i to q_j with input label a and output label b.[c]

If the PCP instance has a solution $(i_1, \ldots, i_K)$, then T is double-tape ambiguous since the pair $i_1 \ldots i_K : u_{i_1} \ldots u_{i_K}$ is accepted on two paths: one through the transitions $(1, i_k, u_{i_k}, 1)$ for $1 \leq k \leq K$, the other through $(2, i_k, v_{i_k}, 2)$ for $1 \leq k \leq K$.

[c]In order to simplify the proof we consider here a transducer with transition outputs in Δ^*. There straightforwardly exists an equivalent transducer with transition outputs in $\Delta \cup \{\epsilon\}$.

Conversely, if T is double-tape ambiguous then there exists two paths π_1 and π_2 with the same input and output labels. A path in T either remains at state 1 or at state 2. It is clear that if two distinct paths π_1 and π_2 have the same input labels, then they must be at different states. Let π_1 be the path that remains at state 1 and π_2 the path that remains at state 2. Let the input label on π_1 (and π_2) be the sequence $i_1 \dots i_K$. Since the output labels are the same on π_1 and π_2, it follows that $u_1 u_2 \dots u_{i_K} = v_1 v_2 \dots v_{i_K}$. Thus the PCP admits a solution and the proof is complete. □

It is natural to ask how hard the problem remains if we restrict our attention to more specific classes of transducers. We show that if we restrict ourselves to acyclic transducers, the problem is NP-hard.

Theorem 22. *The problem of determining the double-tape ambiguity of an arbitrary acyclic transducer T is NP-hard.*

Proof. The reduction is from bounded PCP: a variant of PCP in which we seek a sequence of indices $i_1 \dots i_K$ with $K \leq B$ for some fixed $B > 0$. The bounded PCP is NP-complete [6]. Instead of having self-loops at states 1 and 2 in the construction of Theorem 21, we simply unfold the loops B times. This shows that the problem for acyclic transducers is (at least) NP-hard. □

Note that this result does not imply that the problem is in NP, which in fact, most likely, is not the case.

5.2. *Bounded-delay transducers*

One natural class of transducers for which more positive results hold is that of transducers with *bounded delay*. This imposes a bound on the maximum difference of length between the input and output label of a path. The following gives a formal definition of the notion of *delay*.

Definition 23 (Delay) *The* delay *of a path π is defined as the difference of length between its input and output labels:*

$$\text{delay}(\pi) = \big||o[\pi]| - |i[\pi]|\big|. \tag{5}$$

A trim transducer T is said to have *bounded delay* if the delay of all paths of T is bounded. We then denote by $\text{delay}(T)$ the maximum delay of all paths in T.

A transducer T is *synchronized* if along any accepting path of T the delay is zero or increases strictly monotonically: for any accepting path $\pi = \pi_1 e \pi_2$, $\text{delay}(\pi_1) < \text{delay}(\pi_1 e)$ or $\text{delay}(\pi_1) = \text{delay}(\pi_1 e) = 0$. A transducer with bounded delay is *synchronizable*, that is it admits an equivalent synchronized transducer [12]. Given a transducer T, let T_s denote the synchronized transducer obtained from T using the synchronization algorithm of [12]. The complexity of the synchronization

algorithm is in $O(|T_s|)$. However, the size of T_s is exponential in the worst-case : $O(|T|(|\Sigma|^{\text{delay}(T)}+|\Delta|^{\text{delay}(T)}))$ where Σ is the input alphabet of T and Δ its output alphabet.

When T is a synchronized transducer with a delay of 0, we can give a characterization of double-tape ambiguity based on the form of the *identity paths* in $T^{-1} \circ T$. An identity path π is an accepting path with equal input and output labels: $i[\pi] = o[\pi]$.

Recall that a state in $T \circ T'$, the composition of two transducers T and T', is of the form $(p, q)_f$, where p is a state of T, q is a state of T', and f a state of the epsilon-filter.

Lemma 24 (Characterization) *Let T be a synchronized transducer with* $\text{delay}(T) = 0$. *T is double-tape ambiguous if and only if there exists a successful identity path in $T^{-1} \circ T$ going through a state of the form $(p, q)_f$ with $p \neq q$ or $f \neq 0$.*

Proof. Observe that since T is synchronized and has delay zero, every transition must have either both its input and output labels equal to ϵ, or both non-ϵ.

Assume that T is double-tape ambiguous. Then, T admits two accepting paths π_1 and π_2 with the same input and output labels, say x and y respectively. Since these two paths share the same input, they are matched by composition, which results in a path π in $T^{-1} \circ T$. Moreover, π is an identity path since π_1 and π have the same output label: $o[\pi_1] = o[\pi_2]$. Let e be the first transition along π that was obtained by matching two distinct transitions e_1 and e_2 in T. We shall show that $n[e]$ is a state of the form $(p, q)_f$ with $p \neq q$ or $f \neq 0$. If e_1 is a virtual transition corresponding to remaining at the same state without consuming any symbol while e_2 is an actual ϵ-transition in T, then the filter state of $n[e]$ is not 0, $f \neq 0$. Assume now that both e_1 and e_2 are actual transitions in T. Since e_1 and e_2 are distinct and $i[e_1] = i[e_2]$, we must have $n[e_1] \neq n[e_2]$ or $o[e_1] \neq o[e_2]$. Since T has a delay of 0, we must have $o[e_1] = o[e_2]$. Thus $n[e_1] \neq n[e_2]$ and $n[e]$ is of the form $(p, q)_f$ with $p \neq q$.

Conversely, assume that there exists an identity path π in $T^{-1} \circ T$ going through a state of the form $(p, q)_f$ with $f \neq 0$ or $p \neq q$. This path was obtained by matching in composition two paths π_1 and π_2 such that $i[\pi_1] = i[\pi_2]$ (since they are matched in composition) and $o[\pi_1] = o[\pi_2]$ (since π is an identity path). If π_1 and π_2 were equal, all the states along π would be of the form $(p, p)_0$. Thus, $\pi_1 \neq \pi_2$ and T is double-tape ambiguous. □

This characterization directly leads to an algorithm for testing the double-tape ambiguity of synchronized transducers.

Theorem 25. *The double-tape ambiguity of a synchronized transducer T can be decided in time $O(|T|^2)$, where $|T| = |Q| + |E|$ is the total number of states and transitions of T.*

Proof. A key property of a synchronized transducer T is that along any successful path, a transition with non-ϵ input and ϵ output can only be followed by transitions with non-ϵ input and ϵ output. Similarily, a transition of with ϵ input and non-ϵ output can only be followed by transitions with ϵ input and non-ϵ output. By replacing such ϵs with a special symbol not already in Σ or Δ, say $\#$, we obtain a synchronized transducer T' with a delay of 0 such that T is double-tape ambiguous iff T' is double-tape ambiguous.

The algorithm then consists of computing $T'^{-1} \circ T'$, deleting any transitions e such that $i[e] \neq o[e]$ and performing a depth-first search to verify that the states that are both accessible and co-accessible are all of the form $(p, 0, p)$. □

Finally, we can use the previous result to devise an effective algorithm for testing the double-tape ambiguity of bounded-delay transducers.

Corollary 26. *Let T be a bounded-delay transducer with input alphabet Σ and output alphabet Δ. It is decidable in time $O(|T|^2(|\Sigma|^{\text{delay}(T)} + |\Delta|^{\text{delay}(T)})^2)$ whether T is double-tape ambiguous.*

Proof. Since T has bounded delay, we can use the synchronization algorithm from [12] to compute an equivalent synchronized transducer T_s. The synchronization algorithms preserves double-tape ambiguity thus T_s is double-tape ambiguous iff T is double-tape ambiguous and by Theorem 25 we can decide the double-tape ambiguity of T in time $O(|T_s|^2)$. □

6. 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 27. *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

$$[\![A]\!](x) = \bigoplus_{\pi \in P(I,x,F)} \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]]. \tag{6}$$

The entropy $H(A)$ of a probabilistic automaton A is defined as:

$$H(A) = -\sum_{x\in\Sigma^*} [\![A]\!](x)\log([\![A]\!](x)). \tag{7}$$

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 $\mathbb{K}$,

$$(x_1, y_1)\oplus(x_2, y_2) = (x_1 + x_2, y_1 + y_2) \tag{8}$$

$$(x_1, y_1)\otimes(x_2, y_2) = (x_1x_2, x_1y_2 + x_2y_1). \tag{9}$$

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 $b_1, \ldots, b_k$:

$$\sum_{i=1}^{k} a_i \log\frac{a_i}{b_i} \geq \left(\sum_{i=1}^{k} a_i\right)\log\frac{\sum_{i=1}^{k} a_i}{\sum_{i=1}^{k} b_i}. \tag{10}$$

Lemma 28. *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,*

$$\sum_{i=1}^{k} w[\pi_i]\log w[\pi_i] \geq [\![A]\!](x)(\log[\![A]\!](x) - \log k). \tag{11}$$

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

$$\sum_{i=1}^{k} w[\pi_i]\log w[\pi_i] \geq \left(\sum_{i=1}^{k} w[\pi_i]\right)\log\frac{\sum_{i=1}^{k} w[\pi_i]}{k} = [\![A]\!](x)(\log[\![A]\!](x) - \log k). \tag{12}$$

□

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 29. *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 $\pi_1, \ldots, \pi_k$,

$$\sum_{i=1}^{k} w[\pi_i] \log(w(\pi_i)) \leq \Big(\sum_{i=1}^{k} w[\pi_i]\Big) \log\Big(\sum_{i=1}^{k} w[\pi_i]\Big). \tag{13}$$

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 28, we have

$$\sum_{i=1}^{l_x} w[\pi_i] \log w[\pi_i] \geq [\![A]\!](x)(\log [\![A]\!](x) - \log l_x) \geq [\![A]\!](x)(\log [\![A]\!](x) - \log k). \tag{14}$$

Thus,

$$S(A) = -\sum_{x \in \Sigma^*} \sum_{i=1}^{l_x} w[\pi_i] \log w[\pi_i] \leq H(A) + \sum_{x \in \Sigma^*} (\log k)[\![A]\!](x) = H(A) + \log k. \tag{15}$$

This proves the first statement of the theorem.

Next, assume that A is polynomially ambiguous with degree of polynomial ambiguity k. By Lemma 28, we have

$$\sum_{i=1}^{l_x} w[\pi_i] \log w[\pi_i] \geq [\![A]\!](x)(\log [\![A]\!](x) - \log l_x) \geq [\![A]\!](x)(\log [\![A]\!](x) - \log(|x|^k)). \tag{16}$$

Thus,

$$\begin{aligned} S(A) &\leq H(A) + \sum_{x \in \Sigma^*} k[\![A]\!](x) \log |x| = H(A) + k\mathbb{E}_A[\log |x|] && (17)\\ &\leq H(A) + k \log \mathbb{E}_A[|x|] = H(A) + k \log L, && \text{(by Jensen's inequality)} \end{aligned}$$

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 $w[e] \mapsto (w[e], w[e])$.

7. 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.

We also initiated the study of the double-tape ambiguity of finite-state transducers and gave a number of decidability and characterizations results as well as algorithms in the bounded delay case. These algorithms can be of interest in a number of modern applications where finite-state transducers are used.

Our algorithms also demonstrate the prominent role played by the intersection of finite automata or composition of finite-state transducers with ϵ-transitions [14, 13] 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] Cyril Allauzen and Mehryar Mohri. Efficient Algorithms for Testing the Twins Property. *Journal of Automata, Languages and Combinatorics*, 8(2):117–144, 2003.

[2] Cyril Allauzen and Mehryar Mohri. 3-way composition of weighted finite-state transducers. In *CIAA 2008*, volume 5148 of *LNCS*, pages 262–273. Springer, 2008.

[3] Tat-hung Chan and Oscar H. Ibarra. On the finite-valuedness problem for sequential machines. *Theoretical Computer Science*, 23:95–101, 1983.

[4] Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. On the computation of the relative entropy of probabilistic automata. *International Journal of Foundations of Computer Science*, 19(1):219–241, 2008.

[5] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, 1991.

[6] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[7] Oscar H. Ibarra and Bala Ravikumar. On sparseness, ambiguity and other decision problems for acceptors and transducers. In *STACS 1986*, volume 210 of *LNCS*, pages 171–179. Springer, 1986.

[8] Gérard Jacob. Un algorithme calculant le cardinal, fini ou infini, des demi-groupes de matrices. *Theoretical Computer Science*, 5(2):183–202, 1977.

[9] Werner Kuich. Finite automata and ambiguity. Technical Report 253, Institute für Informationsverarbeitung - Technische Universität Graz und ÖCG, 1988.

[10] Arnaldo Mandel and Imre Simon. On finite semigroups of matrices. *Theoretical Computer Science*, 5(2):101–111, 1977.

[11] Yuri Matiyasevich and Géraud Sénizergues. Decision problems for semi-Thue systems with a few rules. In *IEEE Symposium on Logic in Computer Science*, pages 523–531, 1996.

[12] Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(6):957–982,

2003.

[13] Mehryar Mohri, Fernando C. N. Pereira, and Michael 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.

[14] Fernando Pereira and Michael Riley. *Finite State Language Processing*, chapter Speech Recognition by Composition of Weighted Finite Automata. The MIT Press, 1997.

[15] Emil L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–268, 1946.

[16] Bala Ravikumar and Oscar 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.

[17] Christophe Reutenauer. Propriétés arithmétiques et topologiques des séries rationnelles en variable non commutative. Thèse de troisième cycle, Université Paris VI, 1977.

[18] Andreas Weber. Über die Mehrdeutigkeit und Wertigkeit von endlichen, Automaten und Transducern. Dissertation, Goethe-Universität Frankfurt am Main, 1987.

[19] Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. In *MFCS 1986*, volume 233 of *LNCS*, pages 620–629. Springer, 1986.

[20] Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991.