Towards a Cardinality Theorem for Finite Automata

Till Tantau

Technische Universität Berlin
Fakultät für Elektrotechnik und Informatik
10623 Berlin, Germany
tantau@cs.tu-berlin.de

Abstract. Kummer's cardinality theorem states that a language is recursive if a Turing machine can exclude for any n words one of the $n+1$ possibilities for the number of words in the language. This paper gathers evidence that the cardinality theorem might also hold for finite automata. Three reasons are given. First, Beigel's nonspeedup theorem also holds for finite automata. Second, the cardinality theorem for finite automata holds for $n = 2$. Third, the restricted cardinality theorem for finite automata holds for all n.

1 Introduction

How difficult is it to compute the function $\#_A^n$ for a given language A? This function takes n words as input and yields the number of words in A as output. This counting problem, raised in this general form by Gasarch [6], plays an important role in a variety of proofs both in complexity theory [18, 11, 21, 8, 14] and recursion theory [16, 17, 4].

One way of quantifying the complexity of $\#_A^n$ is to consider its enumeration complexity. For the enumeration complexity of a function f we ask for the smallest m such that f is m-enumerable. This notion, which was first defined by Cai and Hemaspaandra [5] in the context of polynomial-time computations and which was only later transfered to recursive computations, is defined as follows. A function f, taking n-tuples of words as input, is *m-enumerable* if there exists a Turing machine that on input $x_1, \ldots, x_n$ starts a possibly infinite computation during which it prints words onto an output tape. At most m different words may be printed and one of them must be $f(x_1, \ldots, x_n)$.

Intuitively, the larger m, the easier it should be to m-enumerate $\#_A^n$. This intuition is wrong. Kummer's cardinality theorem, see below, states that even n-enumerating $\#_A^n$ is just as hard as deciding A. Intriguingly, the intuition *is* correct for polynomial-time computations and the work of Gasarch, Hoene, and Nickelsen [6, 10, 19] shows that a polynomial-time version of the cardinality theorem does not hold.

Cardinality Theorem ([16]). *If $\#_A^n$ is n-enumerable, then A is recursive.*

1.1 History of the Cardinality Theorem

The proof of the cardinality theorem combines ideas from different areas. Several less general results were already proved when Kummer wrote his paper 'A Proof of Beigel's Cardinality Conjecture' [16]. The title of Kummer's paper refers to the fact that Richard Beigel [3] was the first to conjecture the cardinality theorem as a generalisation of his so-called nonspeedup theorem. In the following formulation of the nonspeedup theorem, χ_A^n denotes the n-fold characteristic function of A. Note that the nonspeedup theorem is a consequence of the cardinality theorem.

Nonspeedup Theorem ([3]). *If χ_A^n is n-enumerable, then A is recursive.*

Owings [20] succeeded in proving the cardinality theorem for $n = 2$. For larger n he could only show that if $\#_A^n$ is n-enumerable, then A is recursive in the halting problem. Harizanov et al. [7] have formulated a restricted cardinality theorem, see below, whose proof is somewhat simpler than the proof of the full cardinality theorem.

Restricted Cardinality Theorem ([7]). *If $\#_A^n$ is n-enumerable via a Turing machine that never enumerates both 0 and n simultaneously, then A is recursive.*

1.2 Finite Automata and the Cardinality Theorem

In this paper I gather evidence that the cardinality theorem might also hold for finite automata, see Conjecture 1.1 below.

The conjecture refers to the notion of n-enumerability by finite automata, which was introduced in [22] and which is defined as follows. A function f, taking n-tuples of words as input, is *m-fa-enumerable* if there exists a finite automaton for which for every input tuple $(x_1, \ldots, x_n)$ the output attached to the last state reached is a set of at most m values that contains $f(x_1, \ldots, x_n)$. The different components of the tuple are put onto n different tapes, shorter words padded with blanks at the end, and the automaton scans the tapes synchronously. A more detailed definition of fa-enumerability is given at the beginning of the next section.

Conjecture 1.1. If $\#_A^n$ is n-fa-enumerable, then A is regular.

The following three results support the conjecture:

1. The nonspeedup theorem also holds for finite automata, see Fact 2.1.
2. The conjecture holds for $n = 2$, see Theorem 2.2.
3. The restricted form of the conjecture holds for all n, see Theorem 2.3.

Together these results bring us as near to a proof of Conjecture 1.1 as did the results in recursion theory before Kummer's breakthrough proof.

1.3 Reformulations of the Cardinality Theorem

Results on cardinality computations are 'separation results in disguise'. The above results can be reformulated purely in terms of separation and disjointness of certain relations, which will be called regular. A *regular relation* is a relation on words that can be accepted by a multi-tape automata with synchronously read tapes. Two relations A and B are *fa-separable* if there exists a regular relation C such that $A \subseteq C \subseteq \bar{B}$. Relations $A_1, \ldots, A_k$ are *disjoint* if $A_1 \cap \cdots \cap A_k$ is empty. We will only refer to this notion of disjointness in the following, not to the more restrictive and more common notion of pairwise disjointness.

We will see that the condition '$\#_A^2$ is 2-fa-enumerable' is equivalent to 'there exist disjoint regular supersets of $A \times A$, $A \times \bar{A}$, and $\bar{A} \times \bar{A}$'. The condition '$\#_A^n$ is n-fa-enumerable via a finite automaton that never enumerates both 0 and n simultaneously' is equivalent to '$A^{(n)}$ and $\bar{A}^{(n)}$ are fa-separable'. Here $A^{(n)}$ denotes the set of n-tuples of pairwise distinct elements of A. These equivalences allow us to reformulate the last two supporting results for Conjecture 1.1, Theorems 2.2 and 2.3, as follows.

$2'$. If there exist disjoint regular supersets of $A \times A$, $A \times \bar{A}$, and $\bar{A} \times \bar{A}$, then A is regular, see Theorem 3.2.
$3'$. If $A^{(n)}$ and $\bar{A}^{(n)}$ are fa-separable, then A is regular, see Theorem 3.3.

Both statements are deceptively innocent-looking and the reader is invited to try to prove them without referring to Theorems 2.2 and 2.3.

The statements are 'optimal' in the sense that if we slightly relax the imposed conditions, then the set A need no longer be regular. Indeed, A can even become *nonrecursive* as the following two results show.

- There exists a nonrecursive set A for which there exist disjoint regular supersets of $A \times A$, $A \times \bar{A}$, $\bar{A} \times A$, and $\bar{A} \times \bar{A}$, see Theorem 3.4.
- For each $n \geq 2$ there exist disjoint, recursively inseparable sets A and B such that $A^{(n)}$ and $B^{(n)}$ are fa-separable, see Theorem 3.5.

The idea of the construction for the last result can be extended to show that there exist disjoint, recursively inseparable sets that are $(3, 5)$-fa-separable, see Theorem 4.1 and Section 4 for definitions. The existence of such sets is somewhat surprising, since an old result of Kinber [15, Theorem 2] states that such sets do not exist.

This paper is organised as follows. In Section 2 we prove the cardinality theorem for finite automata for $n = 2$ and the restricted cardinality theorem for finite automata for all n. In Section 3 we reformulate the finite automata versions of the cardinality theorem in terms of regular relations. In Section 4 we construct a counter-example to the above-mentioned result of Kinber [15, Theorem 2].

2 Finite Automata Versions of the Cardinality Theorem

This section presents the three results mentioned in the introduction that support Conjecture 1.1.

We first fix notations. The i-th bit of a bitstring $b \in \{0,1\}^n$ will be denoted $b[i]$. Let $b[i_1,\ldots,i_k] := b[i_1]\ldots b[i_k]$. For a set P of bitstrings let $P[i_1,\ldots,i_k] := \{b[i_1,\ldots,i_k] \mid b \in P\}$. The cardinality of a set P will be denoted $|P|$. The *n-fold characteristic function* χ_A^n of a language $A \subseteq \Sigma^*$ maps each word tuple $(x_1,\ldots,x_n)$ to the bitstring $b \in \{0,1\}^n$ for which $b[i] = 1$ iff $x_i \in A$. The function $\#_A^n$ maps each word tuple $(x_1,\ldots,x_n)$ to $\big|\{x_1,\ldots,x_n\} \cap A\big|$. Note that if words $x_1,\ldots,x_n$ are not pairwise different, then $\#_A^n(x_1,\ldots,x_n) < n$.

Finite automata compute functions as follows. As done in [1, 2, 22] we use deterministic automata that instead of just accepting or rejecting a word may produce many different outputs, depending on the type of the last state reached. Formally, instead of a set of accepting states the automata are equipped with output functions $\gamma\colon Q \to \Gamma$ that assign an output $\gamma(q) \in \Gamma$ to each state $q \in Q$. The *output* $M(x)$ of an automaton M on input x is the value $\gamma(q)$ attached to the last state q reached by M upon input x. An automaton M *computes a function* f if $f(x) = M(x)$ for all x.

To fa-compute a function that takes n-tuples of words as input we use n input tapes. Each word is put onto a different tape, shorter words padded with blanks at the end such that all words have the same length. Then the automaton scans the words *synchronously*, meaning that in each step all heads advance exactly one symbol. Equivalently, one may also think of the input words being fed to a single-tape automaton in an interleaved fashion: first the first symbols of all input words, then the second symbols, then the third symbols, and so forth.

A function f, taking n-tuples of words as input, is *m-fa-enumerable* if there exists a deterministic n-tape automaton M such that for all words $x_1,\ldots,x_n$ the set $M(x_1,\ldots,x_n)$ has size at most m and contains $f(x_1,\ldots,x_n)$.

Fact 2.1 ([22]). *If χ_A^n is n-fa-enumerable, then A is regular.*

Theorem 2.2. *If $\#_A^2$ is 2-fa-enumerable, then A is regular.*

Proof. If $\#_A^2$ is 2-fa-enumerable, then χ_A^2 is 3-fa-enumerable via an automaton M that always outputs one of the three sets $\{00,01,10\}$, $\{00,11\}$, and $\{01,10,11\}$.

Consider an automaton N that gets *three* input words x, y, and z and computes (in parallel) what M would output on the three inputs (x,y), (x,z), and (y,z). The automaton N outputs a set P of possibilities for $\chi_A^3(x,y,z)$ defined as follows: $P := \big\{b \in \{0,1\}^3 \mid b[1,2] \in M(x,y), b[1,3] \in M(x,z), b[2,3] \in M(y,z)\big\}$.

We now employ an argument that is similar to Kadin's [13] easy-hard arguments. Easy-hard arguments have been used in complexity theory in different proofs, see for example [9]. In such an argument one shows that either all words in Σ^* are *easy* (in a sense to be defined), in which case the language A is, well, easy; or there exist some hard words, which allow us to decide all *other* words, provided we know the characteristic values of the hard words.

Let us call a pair (x,y) of words *easy* if there exists a word z such that $Q := N(x,y,z)[1,2]$ has size at most 2. The word z will be called an *advisor* for (x,y). If a pair is not easy, that is, if there does not exist an advisor for it, we call the pair *hard*. We now distinguish two cases, depending on the existence of certain hard pairs.

Case 1. Suppose there exists a hard pair (x, y) with $\chi_A(x) \neq \chi_A(y)$, that is, $\chi_A^2(x, y) = 01$ or $\chi_A^2(x, y) = 10$. We only consider the case $\chi_A^2(x, y) = 01$, the other case is symmetric. We claim that A can be decided by an automaton that works as follows: on input of a single word z it computes $R := N(x, y, z)$. As we will argue in the following paragraph, R will contain at most one bitstring that begins with 01. But then we can output the last bit of this bitstring, which must be the correct value of $\chi_A(z)$. Thus A is regular.

To see that R contains only one bitstring starting with 01, suppose we had both $010 \in R$ and $011 \in R$. Then $000 \notin R$, since otherwise $R[2, 3] \supseteq \{10, 11, 00\}$, contradicting the assumption that one possibility has been excluded for $\#_A^2(y, z)$. Likewise, $101 \notin R$ and also $111 \notin R$, since otherwise $R[1, 3] \supseteq \{00, 01, 11\}$. Since (x, y) was a hard pair, either $R[1, 2] = \{00, 01, 10\}$ or $R[1, 2] = \{01, 10, 11\}$. In the first case, since $000 \notin R$ and $00 \in R[1, 2]$, we must have $001 \in R$. Likewise, since $101 \notin R$ and $10 \in R[1, 2]$, we must have $100 \in R$. But then $R \supseteq \{010, 011, 001, 100\}$ and thus $R[2, 3] \supseteq \{10, 11, 01, 00\}$, a contradiction. Similarly, in the second case we must have $100 \in R$ and $110 \in R$ and thus $R \supseteq \{010, 011, 100, 110\}$, which yields $R[2, 3] \supseteq \{10, 11, 00\}$, also a contradiction. This shows that R contains only one bitstring starting with 01.

Case 2. We must now show that A is regular if $\chi_A(x) = \chi_A(y)$ for every hard pair (x, y). The rough idea is as follows. Our aim is to show that χ_A^2 is 2-fa-enumerable, because then A is regular by Fact 2.1. So we wish to output a set of at most two possibilities for the characteristic string of any two input words. On input of two words x and y we first check whether the pair (x, y) is hard. How such a check can be performed is explained later. If the pair is hard, by assumption we know that $\chi_A(x) = \chi_A(y)$ and we can output the set $\{00, 11\}$. Otherwise, the pair is easy. In this case we know that there exists an advisor z such that $Q := N(x, y, z)[1, 2]$ has size at most 2. Thus if we can find the advisor, we can output Q. How an advisor can be found is explained next.

To find the advisor z, we employ a method that is also used in [22] for the proof of Fact 2.1. The idea is to first construct a *nondeterministic automaton I with ε-moves*. This automaton has two input tapes, on which it finds two words x and y. As its first step it nondeterministically branches to all states the automaton N reaches when it reads the first symbols of x and y on its first and second tape and some arbitrary symbol on the third tape (including the blank symbol, which corresponds to guessing the end of the word on the third tape). In the second step I branches to the states that N reaches upon then reading the second symbols of x and y plus some arbitrary symbol on the third tape and so on. When the ends of both x and y have been reached, I may go on simulating N using ε-moves: it nondeterministically branches to the states N would reach reading blanks on the first two tapes and some arbitrary symbol on the third tape. Note that on some nondeterministic path the automaton I will reach the state reached by N upon input (x, y, z).

We turn I into a deterministic automaton J using the standard power set construction on its states. Upon input of a pair (x, y) the automaton J will end its computation in a 'power state' p that is exactly the set of all states

reached by I upon input (x, y). We define the output attached to p in J as the intersection of all sets $\gamma(q)[1, 2]$ with $q \in p$, where γ is the output function of N. This ensures that $\chi_A^2(x, y)$ is always an element of $J(x, y)$. Due to the existence of the advisor z, there must exist a state $q \in p$ such that $\gamma(q)[1, 2]$ has size at most 2. Thus the intersection that is output by J has size at most 2.

It remains to show how we can check whether (x, y) is a hard pair. Consider the automaton I once more and once more turn it into a deterministic automaton J'. Only this time we have a closer look at the outputs $\gamma(q)$ attached to the states $q \in p$ of the power state p. We check whether for one of these outputs $\gamma(q)[1, 2]$ has size at most 2. This is the case iff (x, y) is an easy pair. □

The next proof adapts several ideas that were previously used in a new proof by Austinat et al. [2, Proposition 1] of an old result of Kinber. Kinber's original proof turns out to be wrong, see Section 4.

Theorem 2.3. *If $\#_A^n$ is n-fa-enumerable via a finite automaton that never enumerates both 0 and n simultaneously, then A is regular.*

Proof. We prove the claim by induction on n. For $n = 1$ the claim is trivial. So suppose the claim has already been shown for $n - 1$.

Let $\#_A^n$ be n-fa-enumerable via a finite automaton M that never enumerates both 0 and n simultaneously. Then on input of pairwise different words $x_1, \ldots, x_n$ the output of M either misses 0, which can be interpreted as '$\{x_1, \ldots, x_n\}$ intersects A', or it misses n, which can be interpreted as '$\{x_1, \ldots, x_n\}$ intersects $\bar{A}$'.

Similarly to the proof of Theorem 2.2 we now distinguish two cases, depending on the existence of certain hard words. Let us call a tuple $(y_1, \ldots, y_n)$ an *advisor* for a tuple $(x_1, \ldots, x_{n-1})$, if all of these words are pairwise different and if M makes the following $n + 1$ claims: it claims '$\{y_1, \ldots, y_n\}$ intersects $\bar{A}$' and for $i \in \{1, \ldots, n\}$ it claims '$\{x_1, \ldots, x_{n-1}, y_i\}$ intersects A'. Note that an advisor can only, but need not, exist if at least one x_i is in A. Let us call a tuple $(x_1, \ldots, x_{n-1})$ of pairwise different words *easy* if (a) at least one x_i is not in A or (b) there exists an advisor for it, and let us call the tuple *hard* if neither (a) nor (b) holds.

Case 1. Suppose that there exists a hard tuple $(x_1, \ldots, x_{n-1})$. Since (a) does not hold for it, all x_i are in A. Consider an automaton N that works as follows. On input y it simulates M on the input $(x_1, \ldots, x_{n-1}, y)$ and if the output is '$\{x_1, \ldots, x_{n-1}, y\}$ intersects A', it accepts, otherwise rejects. We show that $\mathrm{L}(N)$, the set of all words accepted by N, is a finite variation of A, which is hence regular. For $y \in A$ the automaton M *must* output '$\{x_1, \ldots, x_{n-1}, y\}$ intersects A' and thus $y \in \mathrm{L}(N)$, except possibly for $y \in \{x_1, \ldots, x_{n-1}\}$. For $y \notin A$, since (b) does not hold, the automata M can output '$\{x_1, \ldots, x_{n-1}, y\}$ intersects A' at most $n - 1$ times. Thus $y \notin \mathrm{L}(N)$ whenever $y \notin A$, except for these finitely many exceptions.

Case 2. Suppose all tuples of pairwise different words are easy. We show that $\#_A^{n-1}$ is $(n - 1)$-fa-enumerable via an automaton M' that never outputs 0 and

$n-1$ simultaneously. Then A is regular by the induction hypothesis. So let n input words $x_1, \ldots, x_{n-1}$ be given. If they are not pairwise different, we can immediately output the set $\{0, \ldots, n-2\}$. Otherwise, via the detour of a nondeterministic automaton, we search for an advisor as in the proof of Theorem 2.2. If we find one, we output '$\{x_1, \ldots, x_{n-1}\}$ intersects A'. The existence of the advisor ensures that this output is correct. If we fail to find an advisor, which can only happen because (a) holds, we output '$\{x_1, \ldots, x_{n-1}\}$ intersect $\bar{A}$'. Clearly this output is also correct. □

3 Regular Relations and Cardinality Computations

The introduction claims that 'results on cardinality computations are separation results in disguise'. This section explains what is meant by this claim. It starts with a lemma that shows how cardinality computations can be reformulated in terms of separation and disjointness of regular relations. Then we apply this lemma and show how the previously obtained results can be reformulated without referring to cardinality computations. At the end of the section, we show that the obtained results are optimal in certain senses.

Recall that we called a relation R *regular* if its characteristic function χ_R, defined by $\chi_R(x_1, \ldots, x_n) = 1$ iff $(x_1, \ldots, x_n) \in R$, is fa-computable. We called two relations A and B *fa-separable* if there exists a regular relation C such that $A \subseteq C \subseteq \bar{B}$.

Lemma 3.1. *For every language A the following equivalences hold:*

1. *The function $\#_A^n$ is n-fa-enumerable via an automaton that never enumerates both 0 and n simultaneously, iff $A^{(n)}$ and $\bar{A}^{(n)}$ are fa-separable.*
2. *The function $\#_A^2$ is 2-fa-enumerable, iff there exist disjoint regular supersets of $A \times A$, $A \times \bar{A}$, and $\bar{A} \times \bar{A}$.*
3. *The function χ_A^2 is 3-fa-enumerable, iff there exist disjoint regular supersets of $A \times A$, $A \times \bar{A}$, $\bar{A} \times A$, and $\bar{A} \times \bar{A}$.*

Proof. The first equivalence follows easily from the definitions.

For the second equivalence, first assume that there exist disjoint regular supersets $R_2 \supseteq A \times A$, $R_1 \supseteq A \times \bar{A}$, and $R_0 \supseteq \bar{A} \times \bar{A}$. An automaton that witnesses the 2-fa-enumerability of $\#_A^2$ works as follows: on input (x, y) with $x \neq y$ it outputs the set Q that contains 0 if both $(x, y) \in R_0$ and $(y, x) \in R_0$, that contains 1 if either $(x, y) \in R_1$ or $(y, x) \in R_1$, and that contains 2 if both $(x, y) \in R_2$ and $(y, x) \in R_2$. Note that $|Q| \leq 2$. Suppose $x \notin A$ and $y \notin A$. Then $(x, y) \in R_0$ and $(y, x) \in R_0$. Thus Q will contain 0 as required. Likewise, if $x \in A$ and $y \in A$ then $2 \in Q$. Finally, if $\chi_A(x) \neq \chi_A(y)$, then either $(x, y) \in R_1$ or $(y, x) \in R_1$ and thus $1 \in Q$. In all cases we have $\#_A^2(x, y) \in Q$. For the second direction, let $\#_A^2$ be 2-fa-enumerated by M. For $i \in \{0, 1, 2\}$ define sets $S_i := \{(x, y) \mid i \in M(x, y)\}$ and let $\Delta := \{(x, x) \mid x \in \Sigma^*\}$. The desired disjoint supersets are given by $S_2 \cup \Delta \supseteq A \times A$, $S_1 \setminus \Delta \supseteq A \times \bar{A}$, and $S_0 \cup \Delta \supseteq \bar{A} \times \bar{A}$.

For the third equivalence, let R_{11}, R_{10}, R_{01}, and R_{00} denote disjoint supersets of $A \times A$, $A \times \bar{A}$, $\bar{A} \times A$, and $\bar{A} \times \bar{A}$, respectively. Then χ_A^2 can be 3-fa-enumerated

by an automaton M that works as follows: on input (x, y) it checks (in parallel) for which b we have $(x, y) \in R_b$. It then outputs the set of all such b. Clearly, $\chi_A^2(x, y)$ will be an element of this set and this set will have size at most 3. For the second direction, if χ_A^2 is 3-fa-enumerable via M, we can define R_b with $b \in \{00, 01, 10, 11\}$ as the set of all pairs (x, y) such that $b \in M(x, y)$. These sets have the required properties. □

Lemma 3.1 allows us to reformulate Theorems 2.2 and 2.3 as follows.

Theorem 3.2. *If there exist disjoint regular supersets of $A \times A$, $A \times \bar{A}$, and $\bar{A} \times \bar{A}$, then A is regular.*

Theorem 3.3. *If $A^{(n)}$ and $\bar{A}^{(n)}$ are fa-separable, then A is regular.*

In the following we prove that both results are 'optimal'. For the first theorem this means that if we add a superset of $\bar{A} \times A$ to the list of disjoint sets, then the theorem fails—even quite dramatically as Theorem 3.4 shows. For the second theorem this means that we cannot replace $\bar{A}$ by an arbitrary set B.

Theorem 3.4. *There exists a nonrecursive set A such that there exist disjoint regular supersets of $A \times A$, $A \times \bar{A}$, $\bar{A} \times A$, and $\bar{A} \times \bar{A}$.*

Proof. As shown in [1, 22], for every infinite bitstring b the set A of all bitstrings that are lexicographically smaller than b has the property that χ_A^2 is 3-fa-enumerable. Such a set A is also called a *standard left cut* in the literature. By Lemma 3.1, since χ_A^2 is 3-fa-enumerable, there exist of disjoint regular supersets of $A \times A$, $A \times \bar{A}$, $\bar{A} \times A$, and $\bar{A} \times \bar{A}$. Since there exist uncountably many standard left cuts [12], there must exist a nonrecursive one. □

Theorem 3.5. *For each $n \geq 2$ there exist disjoint, recursively inseparable sets A and B such that $A^{(n)}$ and $B^{(n)}$ are fa-separable.*

Proof. For an infinite bitstring b let A_b denote the set of all nonempty prefixes of b and let B_b denote the set of all nonempty prefixes of b with the last bit toggled. Then any two words in A_b are comparable with respect to the prefix ordering $\sqsubseteq$, whereas no two different words in B_b are comparable with respect to $\sqsubseteq$. Thus for every bitstring b the relation $A_b^{(2)}$ is a subset of $\sqsubseteq$, whereas $B_b^{(2)}$ is a subset of the complement of $\sqsubseteq$. In particular, $A_b^{(n)}$ and $B_b^{(n)}$ are fa-separable for every b and all $n \geq 2$.

We construct an infinite bitstring b such that $A := A_b$ and $B := B_b$ are not recursively separable. This bitstring is constructed in stages. Let $(M_i)_{i \in \mathbb{N}}$ be an enumeration of all Turing machines (the enumeration need not be effective). In stage i we guarantee that $\mathrm{L}(M_i)$, the set of all words accepted by M_i, does not separate A_b and B_b, that is, either $A_b \not\subseteq \mathrm{L}(M_i)$ or $B_b \not\subseteq \overline{\mathrm{L}(M_i)}$.

Suppose we have already constructed $b^i := b[1, \ldots, i]$ and must now decide how to define $b[i + 1]$. We check whether both $b^i 0 \in \mathrm{L}(M_i)$ and $b^i 1 \notin \mathrm{L}(M_i)$ hold. If this is the case, let $b[i + 1] := 1$, which will ensure both $A_b \not\subseteq \mathrm{L}(M_i)$ and $B_b \not\subseteq \overline{\mathrm{L}(M_i)}$. If this is not the case, let $b[i + 1] := 0$, which will ensure either $A_b \not\subseteq \mathrm{L}(M_i)$ or $B_b \not\subseteq \overline{\mathrm{L}(M_i)}$. In either case we guarantee that $\mathrm{L}(M_i)$ does not separate A_b and B_b. □

4 Counter-Example to a Theorem of Kinber

In this section we extend the ideas used in the proof of Theorem 3.5 to construct a counter-example to an old result of Kinber [15, Theorem 2]. Kinber's theorem states that if two sets are (m,n)-fa-separable for $m > n/2$, then they are fa-separable. This claim is wrong, since we will construct sets that are $(3,5)$-fa-separable, but not even recursively separable. Before we prove this, we first review Kinber's notion of (m,n)-fa-separability, which is a generalisation of fa-separability.

For two sets A and B let us call a pair (x,b), consisting of a word $x \in \Sigma^*$ and a bit b, *bad* if $b = 1$ and $x \in B$ and if $b = 0$ and $x \in A$. Two disjoint sets A and B are called (m,n)-*separable*, respectively (m,n)-*fa-separable*, if there exists an n-ary function f computable by a Turing machine, respectively a finite automaton, that on input of any n pairwise different words $x_1, \ldots, x_n$ outputs a bitstring $b \in \{0,1\}^n$ such that at most $n-m$ pairs $\big(x_i, b[i]\big)$ are bad. The intuition behind this definition is that an (m,n)-separating function must output 1 for words in A and 0 for words in B and it may make up to $n-m$ mistakes. Words that are neither in A nor in B play no role. Note that sets are $(1,1)$-fa-separable iff they are fa-separable.

Kinber shows that for all $m < n$ there are (m,n)-separable sets that are not recursively separable. He claims that the situation for finite automata is different, since he claims that for $m > n/2$ all (m,n)-fa-separable sets are fa-separable. Theorem 4.1 shows that this claim is wrong.

Theorem 4.1. *There exist* $(3,5)$*-fa-separable, but not recursively separable sets.*

Proof. We show that the recursively inseparable sets A and B constructed in the proof of Theorem 3.5 are $(3,5)$-fa-separable. Recall that these sets were of the form $A = A_b$ and $B = B_b$ for some infinite bitstring b. We must construct an automaton M that on input of five words $x_1, \ldots, x_5$ will claim '$x_i \in A$' or '$x_i \in B$' for each i, such that among the claims for words in $A \cup B$ at most 2 are wrong.

Let $X := \{x_1, \ldots, x_5\}$. Let y_i denote the word x_i without the last bit (if x_i is the empty string, then $x_i \notin A \cup B$ and we can ignore it). We will say that x_i is *associated* with y_i. Let us call two words x_i and x_j *siblings* if they are associated with the same vertex $y_i = y_j$. Let $Y := \{y_1, \ldots, y_5\}$.

The automaton scans the forest structure of $(Y, \sqsubseteq)$, that is, for each pair (i,j) it finds out whether $y_i \sqsubseteq y_j$ holds. Then it considers all branches in the tree $(Y, \sqsubseteq)$ for which at least three words are associated with the vertices on this branch. Given such a branch, let y denote the last vertex on this branch. Then all vertices on this branch are prefixes of y. The automaton assigns outputs to some of the input words according to the following rule: for each $i \in \{1, \ldots, 5\}$, if y_i is a proper prefix of y and $x_i \sqsubseteq y$ we claim '$x_i \in A$' and if y_i is a proper prefix of y and $x_i \not\sqsubseteq y$ we claim '$x_i \in B$'. Since a word may be associated with a vertex that lies on more than one branch, the just given rule may assign conflicting outputs to a word x_i. Also, we may not have assigned any output to x_i. In either

case the automaton outputs '$x_i \in A$'. Note that in both cases this ensures that if x_i has a sibling x_j, the automaton also outputs '$x_j \in A$'.

According to the construction, the output of the automaton for a word $x_i \in A \cup B$ can be *incorrect* only if y_i is not a proper prefix of the last vertex of one of the above-mentioned branches, or if two of these branches 'split' exactly at y_i. Note furthermore that if $x_i \in A \cup B$ has a sibling, at least one output will be correct for the sibling pair.

We now argue that the described procedure $(3, 5)$-fa-separates A and B. Let $X' := X \cap (A \cup B)$ be the words for which our algorithm must produce a correct output with an error margin of 2. Since for $|X'| \leq 2$ we can output anything, the interesting cases are $3 \leq |X'| \leq 5$.

For $|X'| = 5$, there can only be a mistake for one word associated with the top vertex. There cannot be any splits. Thus we can make at most one mistake.

For $|X'| = 4$, there can also be only one mistake for one word associated with the top vertex, but there can be another mistake caused by a split earlier on the branch to which the words in X' are associated. In total, we can make at most two mistakes.

For $|X'| = 3$, if a sibling pair is associated with any vertex on the branch, at least one output is correct and we are done. Otherwise, again one mistake is possible for the word associated with the top vertex. If there is no split at the root vertex, we make at most one additional mistake at the 'middle' vertex. So assume that there is a split at the root vertex. Then *two* additional input words must be associated with the branch leading away in the wrong direction from the root (since we considered only branches to which at least three words are associated). But then there cannot be another split at the middle vertex of our main branch and the output for this middle element must be correct. □

Although Theorem 2 of Kinber's paper fails, corollaries of this theorem can still be true. For example, Kinber's claim is true if instead of arbitrary disjoint sets A and B we consider A and $\bar{A}$: if a set and its complement are (m, n)-fa-separable for $m > n/2$, then it is regular. Austinat et al. [2] were the first to give a (correct) proof of this corollary. The result can also easily be derived from the more general Theorem 3.3: if A and $\bar{A}$ are (m, n)-fa-separable for $m > n/2$, then $A^{(n)}$ and $\bar{A}^{(n)}$ are fa-separable via majority voting and thus A is regular.

5 Conclusion

This paper raises two questions that I would like to recommend for further research. First, does the cardinality theorem hold for finite automata? Second, for which m and n do there exist (m, n)-fa-separable, but not fa-separable sets?

One promising approach to prove the cardinality theorem for finite automata seems to be the employment of proof techniques used in the recursive setting. This approach was successfully taken in [22] to prove the nonspeedup theorem for finite automata. Unfortunately, the transferal of the proofs appears to be highly nontrivial. For example, Kummer's proof of the cardinality theorem is based on

the following *r.e. tree lemma*: if a tree is recursively enumerable (r.e.) and some finite tree cannot be embedded into it, then all its branches are recursive. It is not clear what the correct transferal of this lemma to finite automata might be.

The idea of using regular relations to reformulate the cardinality theorem for finite automata can also be applied to the original cardinality theorem. Using a similar argument as in the proof of Lemma 3.1, one can show that a set A is recursive iff there exist disjoint r.e. supersets of $A \times A$, $A \times \bar{A}$, and $\bar{A} \times \bar{A}$. More generally, let $A^{(k,n)}$ denote the set of all tuples $(x_1, \ldots, x_n)$ of pairwise different words such that exactly k of them are in A. Then the cardinality theorem can be reformulated as follows: a set A is recursive iff there exists disjoint r.e. supersets of $A^{(0,n)}, \ldots, A^{(n,n)}$. In this relational formulation we can also ask whether the cardinality theorem holds in different contexts. For example, we can ask whether a set A must be context-free if there exist disjoint context-free supersets of $A \times A$, $A \times \bar{A}$, and $\bar{A} \times \bar{A}$.

Acknowledgements. I would like to thank Holger Austinat and Ulrich Hertrampf for helpful discussions and pointers to the literature.

References

1. H. Austinat, V. Diekert, and U. Hertrampf. A structural property of regular frequency computations. *Theoretical Comput. Sci.*, to appear 2002.
2. H. Austinat, V. Diekert, U. Hertrampf, and H. Petersen. Regular frequency computations. In *Proc. RIMS Symposium on Algebraic Syst., Formal Languages and Computation*, volume 1166 of *RIMS Kokyuroku*, pages 35–42, Research Inst. for Math. Sci., Kyoto University, Japan, 2000.
3. R. Beigel. *Query-Limited Reducibilities.* PhD thesis, Stanford University, USA, 1987.
4. R. Beigel, W. Gasarch, M. Kummer, G. Martin, T. McNicholl, and F. Stephan. The complexity of ODD_n^A. *J. Symbolic Logic*, 65(1):1–18, 2000.
5. J. Cai and L. Hemachandra. Enumerative counting is hard. *Inform. Computation*, 82(1):34–44, 1989.
6. W. Gasarch. Bounded queries in recursion theory: A survey. In *Proc. 6th Structure in Complexity Theory Conf.*, pages 62–78, 1991. IEEE Computer Society Press.
7. V. Harizanov, M. Kummer, and J. Owings. Frequency computations and the cardinality theorem. *J. Symbolic Logic*, 52(2):682–687, 1992.
8. L. Hemachandra. The strong exponential hierarchy collapses. *J. Comput. Syst. Sci.*, 39(3):299–322, 1989.
9. E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. A downward collapse in the polynomial hierarchy. *SIAM J. Comput.*, 28(2):383–393, 1998.
10. A. Hoene and A. Nickelsen. Counting, selecting, and sorting by query-bounded machines. In *Proc. 10th Symposium on Theoretical Aspects of Comput. Sci.*, volume 665 of *Lecture Notes in Comput. Sci.*, pages 196–205. Springer-Verlag, 1993.
11. N. Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, 1988.
12. C. Jockusch, Jr. *Reducibilities in Recursive Function Theory.* PhD thesis, Massachusetts Inst. of Technology, USA, 1966.

13. J. Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM J. Comput.*, 17(6):1263–1282, 1988.
14. J. Kadin. $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ and sparse Turing-complete sets for NP. *J. Comput. Syst. Sci.*, 39(3):282–298, 1989.
15. E. Kinber. Frequency computations in finite automata. *Cybernetics*, 2:179–187, 1976.
16. M. Kummer. A proof of Beigel's cardinality conjecture. *J. Symbolic Logic*, 57(2): 677–681, 1992.
17. M. Kummer and F. Stephan. Effective search problems. *Math. Logic Quarterly*, 40(2):224–236, 1994.
18. S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *J. Comput. Syst. Sci.*, 25(2):130–143, 1982.
19. A. Nickelsen. On polynomially $\mathcal{D}$-verbose sets. In *Proc. 14th Symposium on Theoretical Aspects of Comput. Sci.*, volume 1200 of *Lecture Notes in Comput. Sci.*, pages 307–318. Springer-Verlag, 1997.
20. J. Owings, Jr. A cardinality version of Beigel's nonspeedup theorem. *J. Symbolic Logic*, 54(3):761–767, 1989.
21. R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988.
22. T. Tantau. Comparing verboseness for finite automata and Turing machines. In *Proc. 19th Symposium on Theoretical Aspects of Comput. Sci.*, volume 2285 of *Lecture Notes in Comput. Sci.*, pages 465–476. Springer-Verlag, 2002.