# Workfile on Separation Problem

Given two finite words $u, v \in \Sigma^*$, let $\mathsf{Sep}(u, v)$ be the minimum number of states in a dfa accepting exactly one of the words $u$, $v$. Furthermore, let $\mathsf{Sep}(n) = \max_{u,v\in\Sigma^{\le n}} \mathsf{Sep}(u, v)$.

**Problem.** Describe the asymptotics of $\mathsf{Sep}(n)$.

**Comment.** We consider dfa as a quadruple $\mathcal{A} = \{\Sigma, Q, \delta, s\}$, omitting the set of accepting states. We write $q.w$ for the state of $\mathcal{A}$ obtained by reading the word $w \in \Sigma^*$ starting in the state $q \in Q$. Thus, we say that $\mathcal{A}$ *separates* $u$ and $v$ if $s.u \ne s.v$. Since we are seeking for small separating dfa's, we consider only *reachable* ones (such that any $q \in Q$ is equal to $s.w$ for an appropriate word $w$).

An algebraic viewpoint: let $T_k$ denote the full transformation semigroup of an $k$-element set.

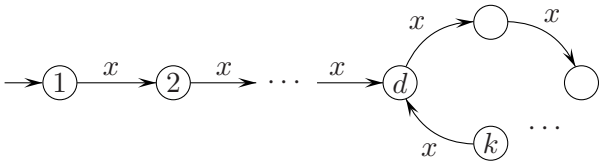
**Observation 1.** The semigroup $T_k$ satisfies an identity $u = v$ with $n = \max\{|u|, |v|\}$ if and only if $\mathsf{Sep}(n) > k$.

*Proof.* The transition monoid of any $k$-state dfa is a submonoid of $T_k$. Hence, if $T_k$ satisfies $u = v$, then in this dfa one has $s.u = s.v$. For the converse, note that some words $u$ and $v$ of length $\le n$ are not separated by any $k$-state automaton, inclusing those having the transition monoid $T_k$. □

**Observation 2.** The shortest unary identity for $T_k$ is

$$x^{k-1} = x^{k-1+\kappa(k)}, \text{ where } \kappa(k) = \mathsf{lcm}\{1, \dots, k\}. \tag{1}$$

*Proof.* We should prove that (1) is the shortest identity satisfied by any $k$-element cyclic semigroup. The Cayley graphs of cyclic semigroups are exactly reachable unary dfa's:




(2)

(here $1 \le d \le k$). After reading $x^{k-1}$ by such a dfa, we reach the cycle of length $k-d+1 \le k$. Since $k-d+1$ divides $\kappa(k)$, we return to the same state after reading $x^{\kappa(k)}$, obtaining (1). Further, any word $x^{k-i}$ with $i > 1$ is separated from any other unary word by the $k$-state dfa with a loop on the $k$th state. Finally, any two numbers $i$ and $j$ such that $k - 1 \le i < j < k - 1 + \kappa(k)$ are different modulo $d$ for some $d \le k$. Then, $x^i$ and $x^j$ are separated by a cycle of $d$ vertices. □

Since $\log \kappa(k) = k + o(k)$ by the Prime Number Theorem, we have

**Corollary 1.** $\mathsf{Sep}(n) = \Omega(\log n)$.

In general, almost nothing is known about short identities in $T_k$. See [R. Pöschel, M.V. Sapir, N.W. Sauer, M.G. Stone, M.V. Volkov. Identities in full transformation semigroups. Algebra Universalis 31 (1994), 580–588]. Volkov says that (i) there was no significant progress in the area since then, (ii) the general belief is that (1) is the shortest identity.

If $|u| = |v|$ is a non-unary identity for $T_k$, then $x^{|u|} = x^{|v|}$ is an identity as well. Due to Observation 2, when searching for short non-unary identities it is sufficient to analyze the case $|u| = |v|$ (such identities are called *uniform*). Furthermore, from a nontrivial $m$-ary identity one can easily derive a nontrivial binary identity of the same length by identifying all letters but one. Thus we have shown

**Observation 3.** The semigroup $T_k$ satisfies an identity $u = v$ with $\max\{|u|, |v|\} < k - 1 + \kappa(k)$ if and only if it satisfies a uniform binary identity with this property.

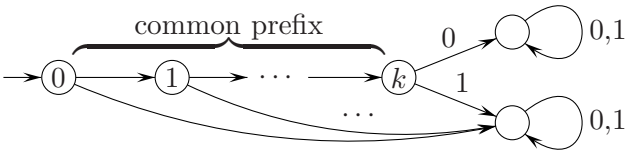
Thus, in what follows we consider separation of binary words $u, v \in \{0, 1\}^*$ of equal length $n$.

# 1 Separators

In this section we consider the known tricks which can be implemented in dfa's and used for separating words.

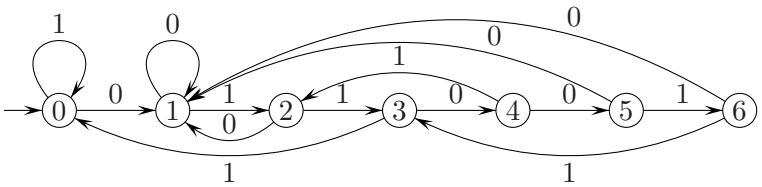
### Basic constructions

All dfa's below are widely used as parts of more complicated separators.
*Basic counter*. Two words of different length can be separated by some dfa (2) with $k = \log n$ and all edges labeled by both 0 and 1. Such a dfa counts letters in the processed word modulo the length of the cycle. Usually, the "tail" in basic counters is redundant, and just a cycle of the required length is used. In addition, (2) can be used for separation by counting only zeroes: in this case the edges are labeled with 0's, and a loop with the label 1 is attached to each vertex.
*Prefixer*. If the longest common prefix of $u$ and $v$ has length $k$, then $u$ and $v$ can be trivially separated using $k+3$ states:




(3)

*Suffixer*. If the longest common suffix of $u$ and $v$ has length $k$, then $u$ and $v$ can be separated using $k+2$ states. This is not so obvious and requires the use of the *Knuth-Morris-Pratt (KMP) automaton*, which is a special case of the *Aho-Corasick automaton*. These automata were designed for pattern matching; the KMP automaton for a word $z$ has $|z|+1$ states and enter the rightmost state if and only if the processed word ends with $z$. (The automaton for $z = 011001$ is drawn below as an example. Note that an edge from a state $j$ to a state $i$ with a label $a$ means that $z[1..i]$ is the longest prefix of $z$ among the suffixes of the word $z[1..j]a$.) Thus, the KMP automaton for $u[n-k..n]$ (or for $v[n-k..n]$) separates $u$ and $v$.




(4)

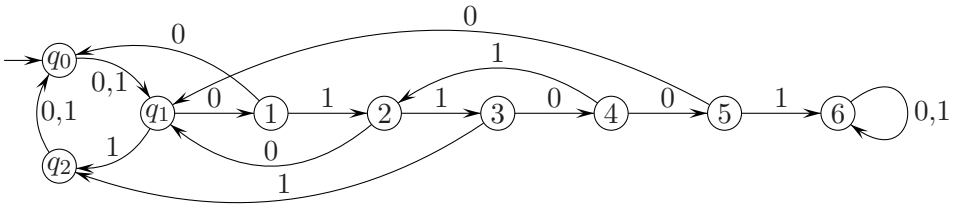
**Detectors**

By detectors I mean dfa's which find an element of structure (e.g., a factor) which appears in exactly one of the words $u$, $v$, or appears in one word earlier than in the other. The simplest situation is when one word (say, $u$) has a factor $z$ of length $k$ which does not occur in $v$. Then we can build the KMP automaton for $z$ (see (4)) and replace both edges outgoing from the rightmost vertex by loops. The obtained $(k{+}1)$-state detector will reach (and stay in) the rightmost state when processing $u$, but will never reach this state processing $v$.

The above analysis shows the main feature of detectors: the sought element of structure is found at the moment when the dfa enters some state $q$ for the first time. Hence, the transitions from $q$ do not affect the search, and we can change them arbitrarily. If the dfa never reaches $q$ on $v$, as in the example above, then it is enough to make $q$ a sink state. In the other case ($q$ is reached on $v$, but later than on $u$) we attach an appropriate basic counter to $q$ on the expense of additional $\log n$ states.

*Robson's detector* [Rob89]. Suppose that $u$ has a factor $z$ of length $k$ beginning in a position equal to $r$ modulo $d$, while all occurrences of $z$ in $v$ begin in positions unequal to $r$ modulo $d$. Then there is a detector with $k + d$ states, separating $u$ and $v$. Such a detector can be combined from a basic counter modulo $d$ and the KMP automaton as follows. The cycle of length $d$ with the vertices $s = q_0, \dots, q_{d-1}$ and edges labeled by $\{0, 1\}$ is taken; the edge from $q_{r-1}$ with the label $z[1]$ is deleted; instead, the KMP automaton for $z$ is attached to this loop, identifying its initial vertex with $q_{r-1}$; the "fail" edges of the KMP automaton are redefined by the following rule: if $(j{+}1{-}i) \bmod d \neq r$, then the edge $j \to i$ is replaced by $j \to q_{(j+r) \bmod d}$; finally, the edges from $k$, which is the rightmost vertex, are replaced with a double loop. It is easy to see that $k$ is reached only when $z$ is read from the position which equals $r$ modulo $d$.

An example from [Rob89] for $z = 011001$, $d = 3$, and $r = 2$ is reproduced below. Compare to (4).




(5)

Using this class of detectors, Robson showed directly that $\mathsf{Sep}(n) = O((n \log n)^{1/2})$ and, with a more elaborate proof, that $\mathsf{Sep}(n) = O(n^{2/5} \log n^{3/5})$. The latter is the best known general upper bound on $\mathsf{Sep}(n)$.

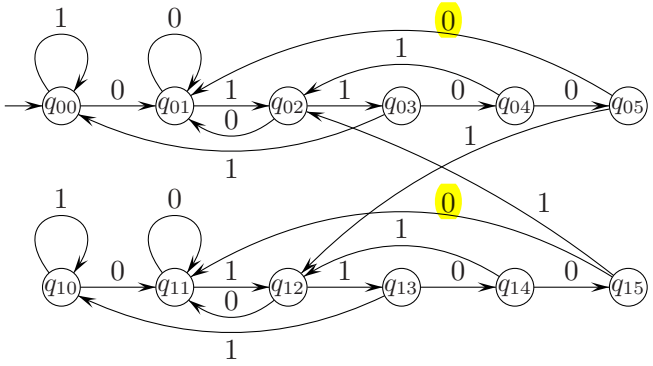
*A-factor detector.* By an arithmetic factor (*A-factor*) of a word $u$ we mean its subword of the form $z = u[i]u[i{+}d] \cdots u[i{+}(k{-}1)d]$ for some $i, d$, and $k$ (setting the difference $d$ to 1, one gets usual factors). Suppose that $z$ is an A-factor of $u$, $r = i \bmod d$, and $v$ has no A-factor $z$ with the same difference $d$ and beginning in a position which equals $r$ modulo $d$. Then there is a detector with $kd + 1$ states, separating $u$ and $v$. This detector is obtained from the KMP automaton for $z$ by replacing each vertex (except for the rightmost one) with a $d$-cycle to count the current position modulo $d$. Thus, $Q = \{q_{ij} \mid 0 \le i < d, 0 \le j < k\} \cup k$, $s = q_{00}$. The first index in the vertex is equal to the length of the processed part of the input modulo $d$. To guarantee this property, any edge of the form $q_{ij} \to q_{i'j'}$ satisfies $i' = i{+}1 \bmod d$. Furthermore, if $i \neq r{-}1$, then $j' = j$ (the current letter cannot belong to the sought A-factor). If $i = r{-}1$, then $j'$ is defined as in

the KMP automaton, compare (4), (5). Namely, $j' = j+1$ if the current letter is $z[j+1]$, and $j'$ is defined by the failure rule otherwise. The remaining edges are $q_{r-1,k-1} \to k$ and the double loop on $k$. Clearly, for this automaton $s.u = k \neq s.v$.

### Counters

Counters form another class of separators. They compare, modulo some small integer, the numbers of occurrences of some object in $u$ and $v$. Any factor or A-factor occurs in $u$ (or in $v$) at most $n$ times. Hence, if such an object occurs $n_u$ times in $u$, $n_v$ times in $v$, and $n_u > n_v$, then $n_u \neq n_v \pmod p$ for some prime (or prime power) $p \leq \log n$.

*Factor counter.* Assume that a factor $z$ occurs $n_u$ times in $u$, $n_v$ times in $v$, and $n_u \neq n_v \pmod p$. Then there is a counter with $kp$ states, separating $u$ and $v$. The set of states of this counter resembles the one of A-factor detector: we take $Q = \{q_{ij} \mid 0 \leq i < p, 0 \leq j < k\}$, $s = q_{00}$. The first index counts modulo $p$ the number of occurrences of $z$ in the processed part of the input (this property guarantees that $s.u \neq s.v$). Hence, for an edge $q_{ij} \to q_{i'j'}$ we should have $i' = i+1 \bmod p$ only when finishing the reading of $z$; otherwise, $i' = i$. The second indices simulate the work of the KMP automaton for $z$, with a slight difference: the vertex $k$ is absent. Since the edge from $(k-1)$ to $k$ in the KMP automaton finishes a new occurrence of $z$, we direct the corresponding edge to the next layer of our counter: $q_{i,k-1} \xrightarrow{z[k]} q_{i'j'}$, where $i' = i+1 \bmod p$ and $j'$ is such that $z[1..j']$ is the longest prefix of $z$ among the proper suffixes of $z$. For example, the factor counter for $z = 011001$ and $p = 2$ looks as follows:




(6)

*A-factor counter.* Assume that we count occurrences of an A-factor $z$ with difference $d$ in $u$ and $v$, and only the occurrences beginning in positions equal to $r$ modulo $d$ are counted. Further, assume that the obtained numbers for $u$ and $v$ differ modulo $p$. Then there is a counter with $kdp$ states, separating $u$ and $v$. This counter is a "hybrid" of a factor counter and an A-factor detector. We take $Q = \{q_{ijl} \mid 0 \leq i < d, 0 \leq j < p, 0 \leq l < k\}$, $s = q_{000}$. The first index counts the length of the processed word modulo $d$, the second one counts the number of occurrences of $z$ modulo $p$, and the third one indicates the length of a proper prefix of $z$ found so far. Thus, the states $s.u$ and $s.v$ will have different values of the second index. In the transitions, the first index is always increased by 1 (modulo $d$). If $i \neq r-1$, the current letter should be skipped in the search for $z$, so the last two indices of a vertex remain unchanged in a transition. Finally, if $i = r-1$, then the last two indices change as in the factor counter for $z$.

*Hamming counter.* If the Hamming distance between $u$ and $v$ is $d$, then they can be separated in $O(d \log n)$ states [E.D. Demaine, S. Eisenstat, J. Shallit, D.A. Wilson. Remarks on Separating Words. DCFS 2011, LNCS 6808.]. Let $i_1, \ldots, i_d$ be the positions in which $u$ and $v$ differ. Then $N = (i_2 - i_1) \cdots (i_d - i_1) < n^{d-1}$ is not divisible by some

$l \le \log N < d \log n$. Then some maximal A-factors of $u$ and $v$ with difference $l$ differ in exactly one position ($i_1$), so we can apply the A-factor counter for $z = 1$, $d = l$, $p = 2$ to get the required bound.

So, in this direction we have the following particular

**Question 1.** What upper and lower bounds can be obtained for separation by counters and detectors only?

Note that Robson's bound used only detectors.

# 2 Numerical Separation

Here we consider two approaches related to counters.

Words that are not separated by a factor counter with $k \log n$ states are exactly $k$-*Abelian equivalent* words. Two words are said to be $k$-Abelian equivalent if they have the same number of occurrences of any factor up to length $k$. The notion was introduced by Karhumäki in the 1980s and became quite popular in combinatorics of words in the very last years. The bad news are that two words of length $n$ can be $\Omega(n)$-Abelian equivalent without being equal. The extreme example is $(0^{n/2}10^{n/2-1}, 0^{n/2-1}10^{n/2})$, a pair of $(n/2)$-Abelian equivalent words.

### Equations

Words that are not separated by an A-factor counter with $kd \log n$ states satisfy a much stronger condition: they are $k$-Abelian equivalent together with all their maximal A-factors with differences $2, \dots, d$. This condition can be equivalently represented as a system of polynomial equations over $\mathbb{Z}$, in which the variables $x_1, \dots, x_n$ are letters of some word, and hence can take only boolean values. The equations look like:

$$x_1 + x_2 + \cdots + x_n = b : \text{ the word contains } b \text{ 1's}$$
$$x_1(1 - x_2)x_3 + x_3(1 - x_4)x_5 + \cdots = c : \text{ the word contains } c \text{ factors 101 in odd positions}$$

In total, we have $\Theta(d^2 2^k)$ equations; if the system has a unique solution, then the corresponding word can be separated from everything by an A-factor counter (with parameters $d, k$, and $p \le \log n$); if the number of solutions is greater than 1, then such a separation is impossible.

Of course, some equations in the system are dependent. Can any bounds on the number of equations guaranteeing a unique solution be derived from universal algebra, CSP theory, etc?

All I know is some "refined" set of factors ensuring $k$-Abelian equivalence [Cassaigne, Karhumäki, Saarela, 2014]. It consists of all factors that begin and end with 1; in addition, the prefix and the suffix of length $k-1$ must be fixed. They proved that for a constant $k$ this set of factors is a "basis". So, the number of "independent" equations is quite big anyway.

### Anti-Hamming

This is what we tried in 2011: building a set $P$ of positions such that a Hamming counter will not find a small number for the role of $p$ if $u$ and $v$ diverge exactly in the positions

from $P$. I found a script from that time, and reproduce the main idea (and the failure) below.

Let $d \in \mathbb{N}$, $T = \kappa(d) \approx e^d$. Assume that $p_1 < p_2 < \ldots < p_h$ are all primes between $d{+}1$ and $d^2$, and $h$ is odd. By the Prime Number Theorem, $h \approx \frac{d^2-2d}{2\log d}$.

Consider all sums of distinct numbers $p_i$, there are $2^h$ of them. Many of them coincide, because $\bar{p} = \sum_{i=1}^{d} p_i \ll 2^h$. The obtained set of numbers is our set $P$. If two sums coincide, they have the same parity of the number of summands. We assign '0' [resp., '1'] to all numbers that are sums of even [resp., odd] number of $p_i$'s. There is a natural bijection between the numbers with zeroes and ones: $n \to \bar{p} - n$. Hence, the number of 0's equals the number of 1's. Consider some word $w$, $|w| = T$, and build $u$ and $v$ from the word $w^{\bar{p}+2}$ as follows:

- for any number $j \in P$, $u[(j{+}1)T]$ is set to the bit assigned to $j$, and $v[jT]$ is set to the opposite bit;
- all other letters in $u$ and $v$ coincide with the corresponding letters in $w^{\bar{p}+2}$.

The initial idea was to prove that $u$ and $v$ cannot be separated with $o(d^2)$ states (at least for some $w$). I do not believe much in this, but may be we can prove this for separation by counters only? It is clear that

- $u$ and $v$ are $T$-Abelian equivalent: each factor of length $\le T$ contains at most one position divisible by $T$, and $u$ has the same number of 0's and 1's in these positions as $v$ (note that $u$ and $v$ are not $(T{+}1)$-Abelian equivalent, since $u[1..T] \ne v[1..T]$);
- for any $d' \le d$, all pairs of corresponding maximal A-factors with difference $d'$ in $u$ and $v$ are $T/d'$-Abelian equivalent: since $d'$ divides $T$, such pairs consist of equal words, except the rightmost pair; for this latter one the argument is the same as for $u$ and $v$;
- so, no A-factor counters with the difference $\le d$ will work.

On the other hand, when we consider A-factors with difference $p_1$ it may happen that it will be enough to count 1's modulo 2 in some of these factors, getting $2p_1$ states in an A-factor counter. The reason for such a fail is the following. Consider an A-factor with difference $p_1$ which contains some positions divisible by $T$, and let $(r{+}1)T$ be the leftmost of these positions. Then the other positions are of the form $(r{+}1{+}lp_1)T$ for some $l > 0$. The problem is that we cannot guarantee that the number of elements of $P$ among the numbers $r{+}lp_1$ is even. For example, $r{+}ip_1 \in P$ and all its representations as the sum of $p_i$'s do not include $p_1$. Then $r{+}(i{+}1)p_1 \in P$ as well; suppose that this number also has a representation without $p_1$, while $r{+}(i{+}2)p_1 \in P$ can be represented only with $p_1$. Then we have just three positions in which the considered A-factors of $u$ and $v$ differ[1].

# 3 Permutation Automata

Note that counters (except for the basic counter) are permutation automata. So, a separate problem is how efficiently words can be separated with permutation automata (pdfa).

[1] Test on the set 7,11,13,17,19 showed that no "miracle" happens: the class 1 mod 7 contains 3 elements of $P$.

(Or, in algebraic terms, what is the length $n(k)$ of the shortest uniform semigroup identity in the $k$-element symmetric group. It is known that $n(3) = 4$, $n(4) = 11$. I observed that minimal identities have the form $u = \overleftarrow{u}$ and ask a student to write a program checking such equalities. He found shortest such $u$ for $k = 5$ ($u = xy(xyyx)^3(yxxy)^2yx(yxxy)^2$) and $k = 6$ ($u = (xy)^6(yx)^{10}(xy)^4$), so we have $n(5) \leq 32$, $n(6) \leq 40$. Anyway, there is no idea how these identities look in general.)

Robson [Rob96] proved that any two words can be separated by a pdfa with $O(\sqrt{n})$ states. He used the same A-factor counters for $z = 1$ and $p = 2$ (and $d$ depending on $u$ and $v$) as Demaine et al. for the Hamming counter.

**Superautomaton**

The product of dfa's $\mathcal{A}_1$, with states $1, \ldots, k$, and $\mathcal{A}_2$, with states $1, \ldots, m$, is the dfa $\mathcal{A}_1 \times \mathcal{A}_2$ with $km$ states of the form $(i, j)$, where $1 \leq i \leq k$, $1 \leq j \leq m$, and the transition $(i, j) \xrightarrow{a} (i', j')$ exists if and only if $i \xrightarrow{a} i'$ in $\mathcal{A}_1$ and $j \xrightarrow{a} j'$ in $\mathcal{A}_2$. It is clear that

- the product of two pdfa's is a pdfa;
- if $\mathcal{A}_1 \times \mathcal{A}_2$ separates $u$ and $v$, then at least one of $\mathcal{A}_1$, $\mathcal{A}_2$ separates them.

So, the idea is to build a product of many pdfa's of equal size $k$ such that this product "separates everything". If this "superautomaton" is built, then "everything" can be separated in $k$ states. The summary of this "work in progress" is below. Some scanned proofs are in RandomAutomata_summary.pdf.

- There are $(k!)^2$ ways to define the transition function for a $k$-state binary pdfa. At most $k!$ ways give the same (up to isomorphism) dfa, so the number of non-isomorphic pdfa's is at least $k!$.
- For pdfa's, strong and weak connectivity are equivalent. A random pdfa is connected (and hence is reachable) with probability almost $1 - 1/k$. A connected pdfa has at most $k$ automorphisms (fixing the image of a vertex, you fix everything). This gives us at most $(k-1)k!$ non-isomorphic connected pdfa's. Almost all[2] of them are *primitive*, that is, the gcd of lengths of all cycles is 1, and for any big enough $n$ any vertex can be reached by a word of length $n$.
- It seems probable that I will be able to prove that the product of two non-isomorphic connected primitive pdfa's is connected (and most, but not all, of them are primitive). This allows one to build products iteratively: $((\mathcal{A}_1 \times \mathcal{A}_2) \times (\mathcal{A}_3 \times \mathcal{A}_4)) \times \cdots$ (note that non-isomorphic terms give non-isomorphic products). So, in principle, one can build a product of, say, $2^k$ connected primitive pdfa's, which is a connected primitive pdfa $\mathcal{S}$ with $k^{2^k}$ states.
- The graph of $\mathcal{S}$ (and of any pdfa, in general) has a matrix which is a double-stochastic matrix multiplied by 2. Connectedness means that 2 is a simple root, so $(1, \ldots, 1)$ is the corresponding row eigenvector. This means that the number of walks of length $n$ from the initial vertex to $i$th vertex is asymptotically $2^n/(k^{2^k})$ for any $i$. Thus, $\mathcal{S}$ divides all words into the asymptotically equal classes. This means that $\mathcal{S}$ is quite good in terms of separation. The problem is that this does not guarantee separation of all words of some exponential in $k$ length. (What is the expected diameter of $\mathcal{S}$?)

[2]The probability is $1 - O(\sqrt{n}/2^n)$.

# 4 Random Automata

An obvious Moser-Tardos-like idea: take a random dfa; if it does not separate $u$ and $v$, redefine some transition(s) randomly and check again, until successful separation.

*Pro:* a dfa and the result of reading a word of fixed length can be easily written as a boolean formula or CSP formula, which are natural objects for local resampling algorithms. Here is a possible representation as a CSP (which can be converted into SAT using the bits of CSP variables as boolean variables).
*Domain*: $0,\dots,k{-}1$ (states of the dfa).
*Variables*: $x_{0i}$, $x_{1i}$ ($i = 0,\dots,k{-}1$) for transitions from the $i$th vertex; $s_0,\dots,s_n$ $[t_0,\dots,t_n]$ for the states after reading the $j$th letter of $u$ [resp., $v$].
*Constants (binary)*: $u_1,\dots,u_n,v_1,\dots,v_n$ (letters of $u$ and $v$).
*Constraints*: $s_0 = t_0 = 0$; $s_n \neq t_n$; $s_{j+1} = x_{1s_j}u_{j+1} + x_{0s_j}(1{-}u_{j+1})$, $t_{j+1} = x_{1t_j}v_{j+1} + x_{0t_j}(1{-}v_{j+1})$, $j = 1,\dots,n$ (some auxiliary variables are needed to write these constraints in a stricter way, without variables in indices).

*Contras:* from recent studies on synchronizing automata it is known that a random binary automaton (i) is synchronizing with high probability, (ii) is synchronized by a random word of polynomial length w.h.p. So, it is doubtful that such an automaton will separate any long words; we may be forced to switch from arbitrary automata to some subclass (permutation automata?). Another problem is that the dependencies in the obtained formula are not local (what to resample?).

—

One more possible approach using randomization.

**Observation 4.** Assume that $n$ is the minimum length of a word $u$ with the following property: in any $k$-state dfa, both outgoing edges of the vertex $s.u$ are traversed while reading $u$. Then $\mathsf{Sep}(n) \le k + \log n$.

*Proof.* Consider the longest common prefix $w$ of two arbitrary unequal words $u$ and $v$ of length $n$. By the choice of $n$, there is a $k$-state dfa $\mathcal{A}$ in which the path from the initial state with the label $w$ does not use at least one edge from the vertex $s.w$. Then we upgrade $\mathcal{A}$, adding the basic counter of an appropriate length $d \le \log n$ and redirecting the unused edge to a vertex of this counter. The paths for $u$ and $v$ reach this counter in different time, so we get a sort of detector. □

The possible use of this observation is as follows: take a word of length $n$ and try to build a $k$ state dfa with the required property by a resampling algorithm; find the bound on $k$ which makes such an algorithm converging.