The Separating Words Problem

Jeffrey Shallit
School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
shallit@cs.uwaterloo.ca
https://www.cs.uwaterloo.ca/~shallit

The Simplest Computational Problem?

Imagine a stupid computing device with very limited powers...

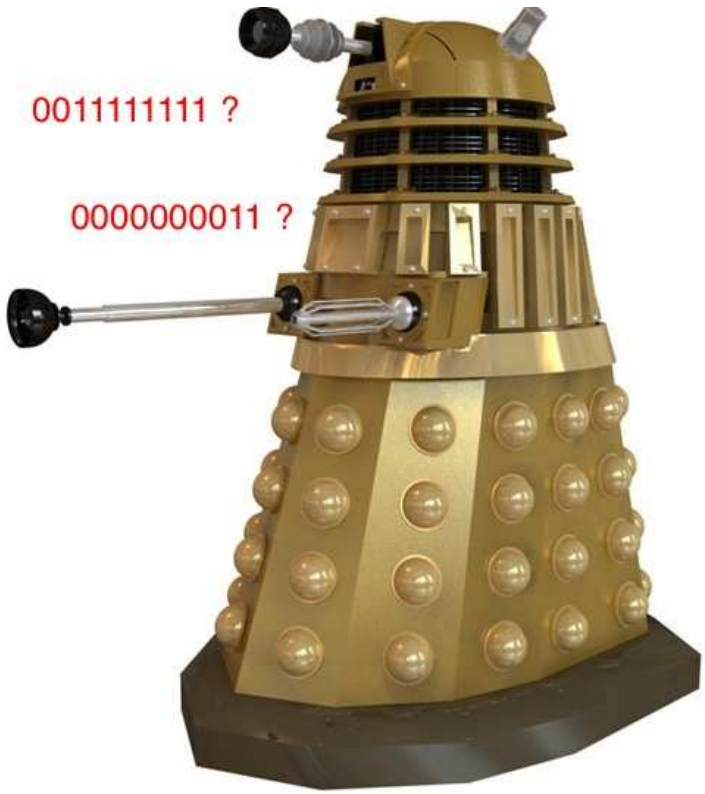
What is the simplest computational problem you could ask it to solve?

The Simplest Computational Problem?

- not the addition of two numbers

- not sorting

- it's telling two inputs apart - distinguishing them



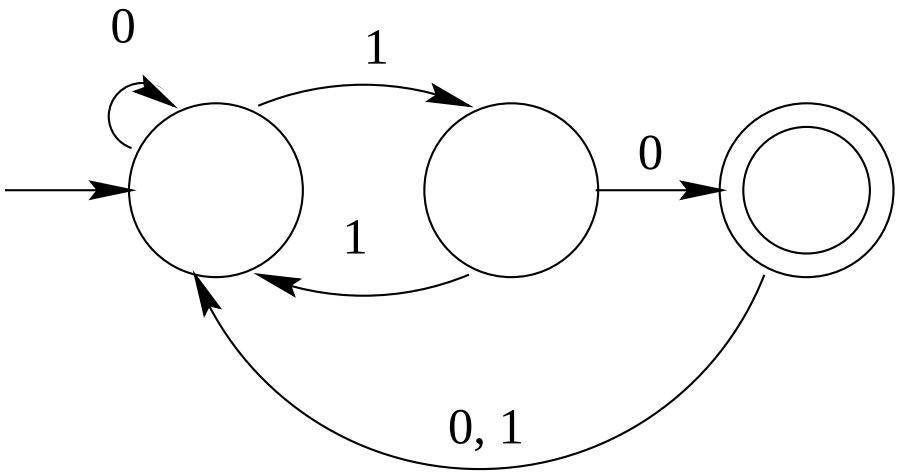
Our Computational Model: the DFA

Our computational model is the **deterministic finite automaton**, or DFA.

It consists of

- Q, a finite nonempty set of states
- q_0, an initial state
- F, a set of final states
- δ, a transition function that tells you how inputs move the machine from one state to another

An example of a DFA



– initial state has sourceless incoming arrow

– final states are denoted by double circles

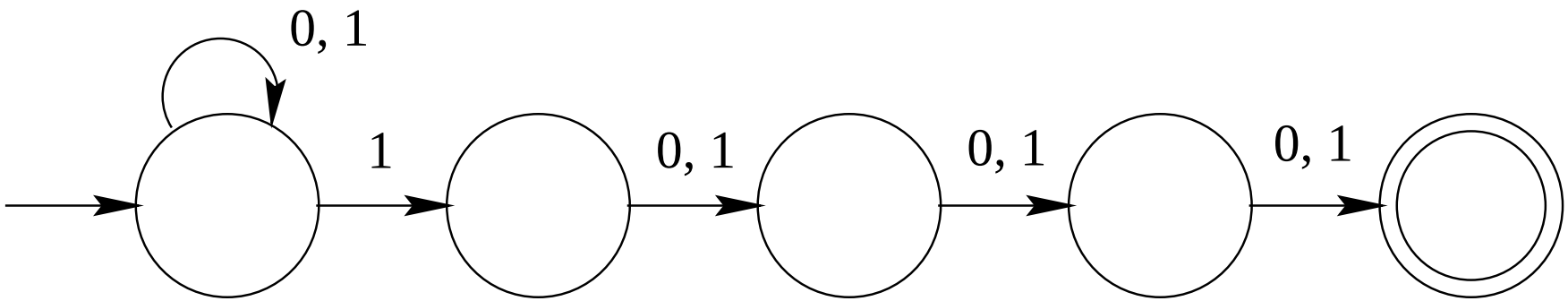
– a word is **accepted** if it labels a path from the initial state to a final state; otherwise it is **rejected**

DFA versus NFA

An automaton is **deterministic** if, for each state and input symbol, there is only one possible state that can be entered next. We call this a DFA.

Otherwise it is **nondeterministic**. We call this an NFA.

Example of an NFA



This NFA accepts all words having a 1 in a position that is 4 spots from the right end.

Motivation

We want to know how many states suffice to tell one length-n input from another.

On average, it's easy — but how about in the worst case?

Motivation: a classical problem from the early days of automata theory:

Given two automata, how long a word do we need to distinguish them?

Motivation

More precisely, given two DFA's M_1 and M_2, with m and n states, respectively, with $L(M_1) \neq L(M_2)$, what is a good bound on the length of the shortest word accepted by one but not the other?

- The cross-product construction gives an upper bound of $mn - 1$ (make a DFA for $L(M_1) \cap \overline{L(M_2)}$)
- But an upper bound of $m + n - 2$ follows from the usual algorithm for minimizing automata
- Furthermore, this bound is best possible.
- For NFA's the bound is exponential in m and n

Separating Words with Automata

Our problem is the inverse problem: given two distinct *words*, how big an automaton do we need to separate them?

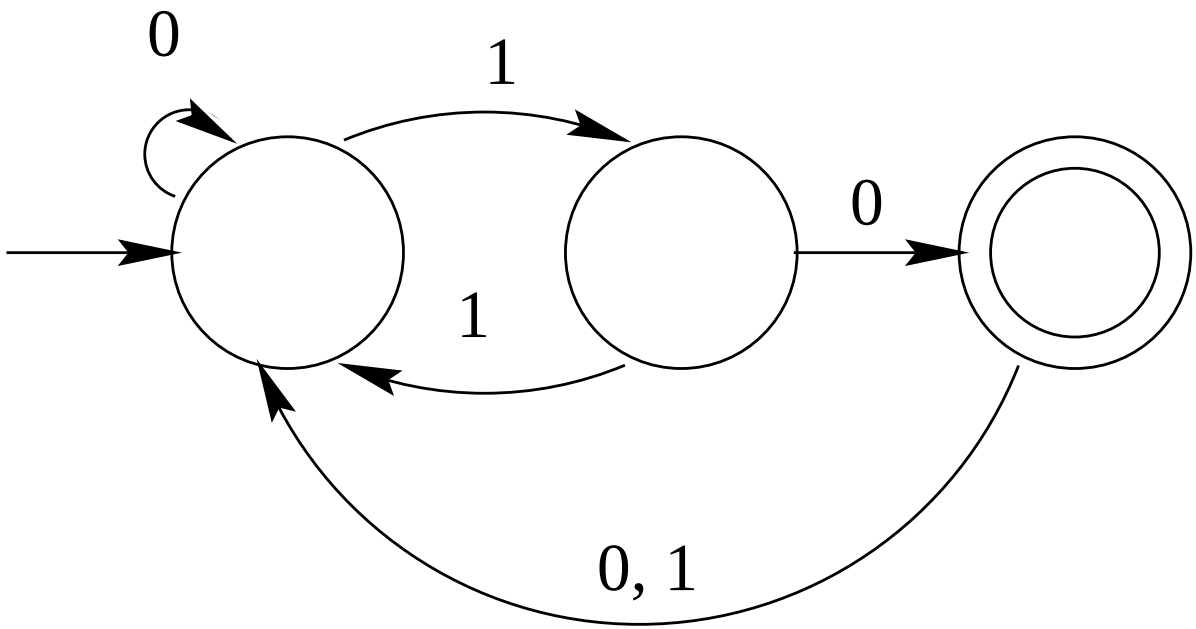
That is, given two words w and x of length $\leq n$, what is the smallest number of states in any DFA that accepts one word, but not the other?

Call this number $\mathrm{sep}(w, x)$.

Separation

A machine M **separates** the word w from the word x if M accepts w and rejects x, or vice versa.

For example, the machine below separates 0010 from 1000.



However, no 2-state DFA can separate these two words. So $\mathrm{sep}(1000, 0010) = 3$.

Separating Words of Different Length

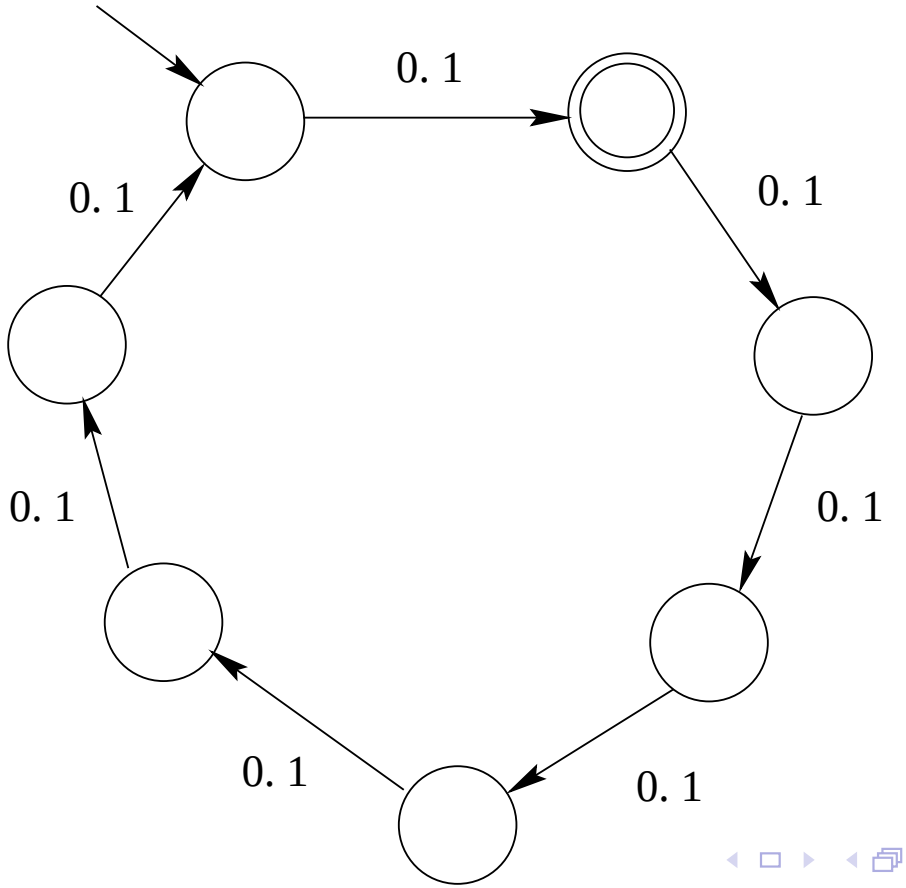
Easy case: if the two words are of different lengths, both $\leq n$, we can separate them with a DFA of size $O(\log n)$.

For by the prime number theorem, if $k \neq m$, and $k, m \leq n$ then there is a prime $p = O(\log n)$ such that $k \not\equiv m \pmod{p}$.

So we can accept one word and reject the other by using a cycle mod p, and the appropriate residue class.

Separating Words of Different Length

Example: suppose $|w| = 22$ and $|x| = 52$. Then $|w| \equiv 1 \pmod 7$ and $|x| \equiv 3 \pmod 7$. So we can accept w and reject x with a DFA that uses a cycle of size 7, as follows:



Separating Words with Different Prefix

For the remainder of the talk, then, we only consider the case where $|w| = |x|$.

We can separate w from x using $d + O(1)$ states if they differ in some position d from the start, since we can build a DFA to accept words with a particular prefix of length d.

Separating Words with Different Prefix

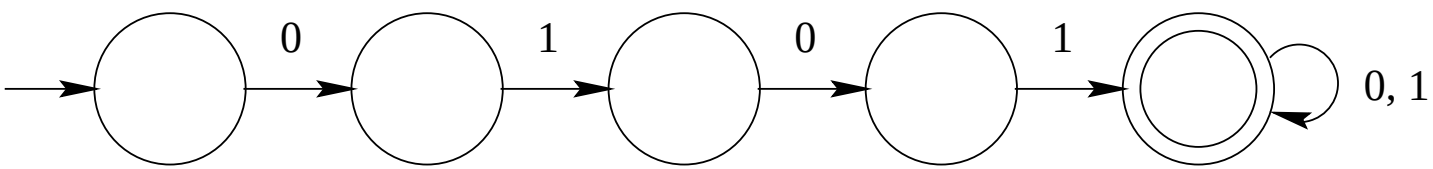
For example, to separate

$$01010011101100110000$$

from

$$01001111101011100101$$

we can build a DFA to recognize words that begin with 0101:



(Transitions to a dead state omitted.)

Separating Words With Different Suffix

Similarly, we can separate w from x using $d + O(1)$ states if they differ in some position d from the end.

The idea is to build a pattern-recognizer for the suffix of w of length d, ending in an accepting state if the suffix is recognized.

Separating Words With Different Suffix

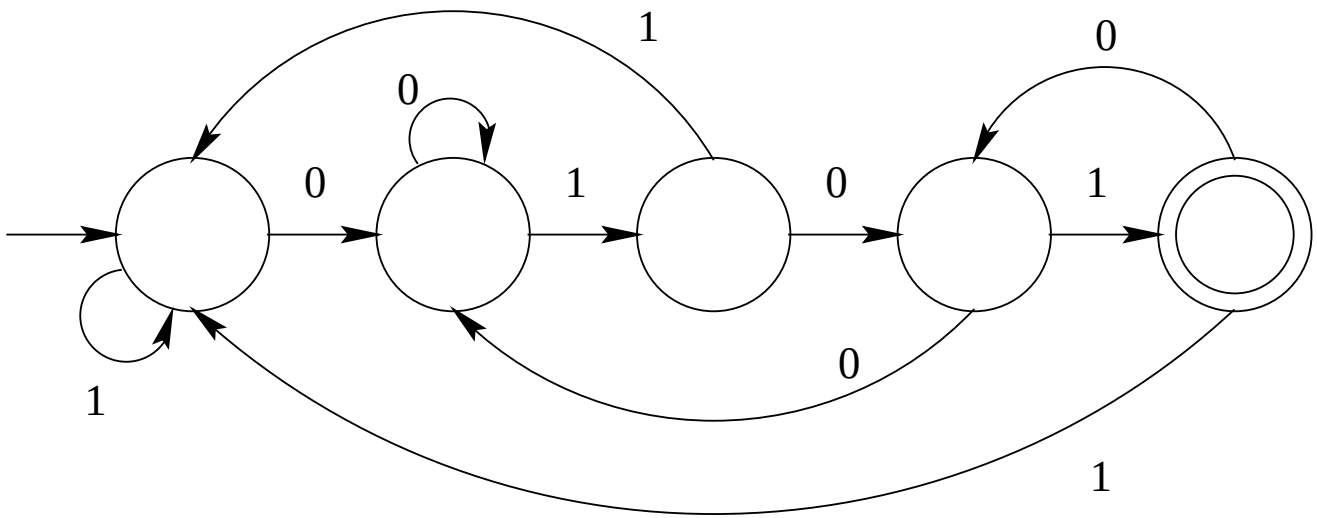
For example, to separate

$$11111010011001010101$$

from

$$11111011010010101101$$

we can build a DFA to recognize those words that end in 0101:



Separating Words With Differing Number of 1's

Can we separate two words having differing numbers of 1's?

Yes. By the prime number theorem, if $|w|, |x| = n$, and w and x have k and m 1's, respectively, then there is a prime $p = O(\log n)$ such that $k \not\equiv m \pmod{n}$.

So we can separate w from x just by counting the number of 1's, modulo p.

Separating Words with Differing Number of Patterns

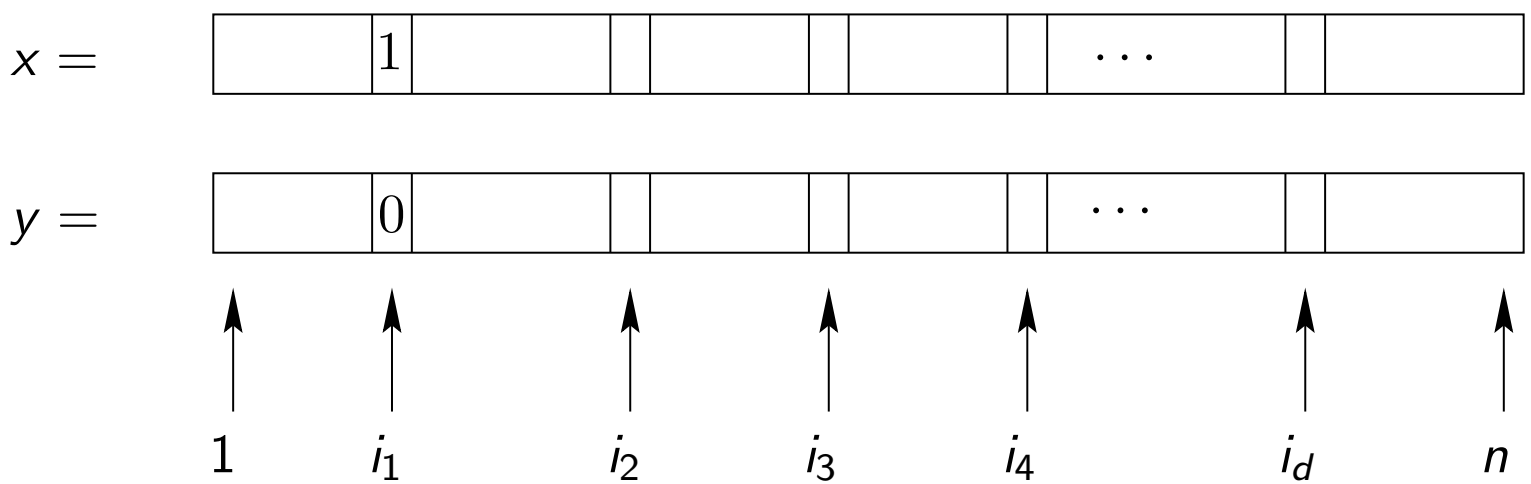
Similarly, we can separate two length-n words w, x using $O(d \log n)$ states if there is a pattern of length d occurring a differing number of times in w and x.

Separation of Very Similar Words

The *Hamming distance* between w and x is the number of positions where they differ.

If the Hamming distance between w and x is small, say $< d$, we can separate two length-n words using $O(d \log n)$ states.

The idea is as follows:



Let $i_1, i_2, \ldots, i_d$ be the positions where x and y differ.

Separation of Very Similar Words

Now consider $N = (i_2 - i_1)(i_3 - i_1) \cdots (i_d - i_1)$. Then $0 < N < n^{d-1}$.

So N is not divisible by some prime $p = O(\log N) = O(d \log n)$.

So $i_j \not\equiv i_1 \pmod{p}$ for $2 \leq j \leq d$.

Now count the number, modulo 2, of 1's occurring in positions congruent to $i_1 \pmod{p}$.

These positions do not include any of $i_2, i_2, \ldots, i_d$, by the way we chose p, and the two words agree in all other positions.

So x contains exactly one more 1 in these positions than w does, and hence we can separate the two words using $O(d \log n)$ states.

The Separation Number

- Let

$$S(n) := \max_{\substack{|w|=|x|=n \\ w \neq x}} \operatorname{sep}(w, x),$$

  the smallest number of states required to separate any two words of length n.

- The separation problem was first studied by Goralcik and Koubek 1986, who proved $S(n) = o(n)$.

- In 1989 Robson obtained the best known upper bound: $S(n) = O(n^{2/5}(\log n)^{3/5})$.

Dependence on Alphabet Size

For equal-length words, $S(n)$ doesn't depend on alphabet size (provided it is at least 2).

To see this, let $S_k(n)$ be the maximum number of states needed to separate two length-n words over an alphabet of size k.

Suppose x, y are distinct words of length n over an alphabet Σ of size $k > 2$.

Then x and y must differ in some position, say for $a \neq b$,

$$\begin{aligned} x &= x' \; a \; x'' \\ y &= y' \; b \; y''. \end{aligned}$$

Dependence on Alphabet Size

$$\begin{aligned} x &= x' \; a \; x'' \\ y &= y' \; b \; y''. \end{aligned}$$

Map a to 0, b to 1 and assign all other letters arbitrarily to either 0 or 1.

This gives two new distinct binary words X and Y of the same length.

If X and Y can be separated by an m-state DFA, then so can x and y, by renaming transitions to be over Σ instead of 0 and 1.

Thus $S_k(n) \leq S_2(n)$. But clearly $S_2(n) \leq S_k(n)$, since every binary word can be considered as a word over a larger alphabet. So $S_k(n) = S_2(n)$.

Robson's Upper Bound

Robson's upper bound of $O(n^{2/5}(\log n)^{3/5})$ is hard to explain. But he also proved:

Theorem (Robson, 1996). We can separate words by computing the parity of the number of 1's occurring in positions congruent to $i \pmod{j}$, for $i, j = O(\sqrt{n})$.

This gives the bound $S(n) = O(n^{1/2})$.

Open Problem 1: Improve Robson's bound of $O(n^{2/5}(\log n)^{3/5})$ on $S(n)$.

Robson's proof

Idea of the proof:

Fix an integer N.

Consider all the squarefree positive integers $\leq N$.

For each such integer n, consider all the numbers a that are relatively prime to n.

For each such pair (a, n) construct a machine $M_{a,n}$ of $O(n)$ states that accepts if the number of 1's occurring in positions congruent to $a \pmod{n}$ is odd and rejects otherwise.

Robson's proof (2)

There are

$$t(N) := \sum_{n \leq N} \mu(n)^2 \varphi(n)$$

such machines, and one can prove that this number is asymptotically CN^2, where $C \doteq 0.214$. Here μ is the Möbius function and φ is the Euler totient function.

We argue that for each pair of strings x, y of length $\leq t(N) - 1$ there is a machine $M_{a,n}$ in the set above that distinguishes them.

Step 1: First we define a map from binary strings x (of any length) to binary strings of length $t(N)$. This map res(x) encodes the result (accept $= 1$, reject $= 0$) of each machine on the input x. We define an equivalence relation on strings by writing $x \equiv y$ if res(x) $=$ res(y).

Robson's proof (3)

Step 2: We argue that the equivalence relation $\equiv$ has $2^{t(N)}$ equivalence classes. This is done in two parts:

(a) first, we show how to construct, for all pairs (a, n) a word x such that res(x) has exactly one 1 in the position corresponding to the machine $M_{a,n}$.

Let $P = \prod_{p \leq N} p$ be the product of all primes $\leq N$. For each squarefree $n \leq N$, construct a word of length P with all 0's, except that we put a 1 in any position that is equal to
– either 0 or a modulo each of the primes dividing n, and
– equal to 0 modulo P/n.

We claim this string has the desired properties.

Robson's proof (4)

(b) second, we argue that for two strings x, y of the same length, we have $\text{res}(x \oplus y) = \text{res}(x) \oplus \text{res}(y)$, where $\oplus$ is XOR.

Example: let $N = 6$. Then $P = 2 \cdot 3 \cdot 5 = 30$.

Here are the strings corresponding to $\text{res}(x)$ having exactly one 1 in the position corresponding to each machine:

(a, n)	x	$\text{res}(x)$
(0,1)	100000000000000000000000000000	1000000000
(1,2)	100000000000001000000000000000	0100000000
(1,3)	100000000100000000000000000000	0010000000
(2,3)	100000000000000000001000000000	0001000000
(1,5)	100000100000000000000000000000	0000100000
(2,5)	100000000000100000000000000000	0000010000
(3,5)	100000000000000000100000000000	0000001000
(4,5)	100000000000000000000000100000	0000000100
(1,6)	100000000100001000000000010000	0000000010
(5,6)	100001000000000100001000000000	0000000001

Robson's proof (5)

Thus, for example, to get the vector of all 1's as the result res(x) we just XOR all the strings above together:

$$x = 0000011000001001001000000110000$$
$$\text{res}(x) = 1111111111$$

Step 3: We let w_0 be the shortest string such that $w_0 \equiv \epsilon$ and w_0 has at least one 1. Note that w_0 ends with 1.

Example: for $N = 6$ this shortest word is $w_0 = 11111011111$.

Robson's proof (6)

It is easy to see that every string of the form $w_0 0^i$ is also equivalent to the empty word. Slightly harder is to show that every string of the form $0^i w_0$ is also equivalent to the empty word.

Step 4: We argue that every equivalence class of $\equiv$ has a word of length $|w_0| - 1$.

Take any shortest element x of an equivalence class. If $|x| \leq |w_0| - 1$, append 0's on the right to get a word of the right length. If $|x| \geq |w_0|$, and ends with 0, deleting the 0 gives an equivalent shorter word, a contradiction. So x ends with 1. Now consider $x \oplus 0^{|x|-|w_0|} w_0$. This is also equivalent to the empty word, but ends with 0, so we can delete the 0 on the right end of $x \oplus 0^{|x|-|w_0|} w_0$ to get an equivalent shorter word, a contradiction.

Robson's proof (7)

Step 5: Show $|w_0| - 1 = T(N)$: Every equivalence class of $\equiv$ has exactly one word of length $|w_0| - 1$. For if there were two such words, their XOR would be equivalent to the empty word, contradicting the definition of w_0. So each of the $2^{|w_0|-1}$ words of length $|w_0| - 1$ are inequivalent under $\equiv$. But there are $2^{T(N)}$ equivalence classes. So $T(N) = |w_0| - 1$.

Step 6: We now know that any two words of the same length $< |w_0|$ are separated by at least one machine in our list, for otherwise by forming the XOR we would have have a word equivalent to the empty word of length $< |w_0|$.

Step 7: Two words of length $\leq T(N)$ can be distinguished by a machine on our list. Each machine has $O(N)$ states and $T(N) \sim CN^2$. So this gives $O(\sqrt{N})$ separation.

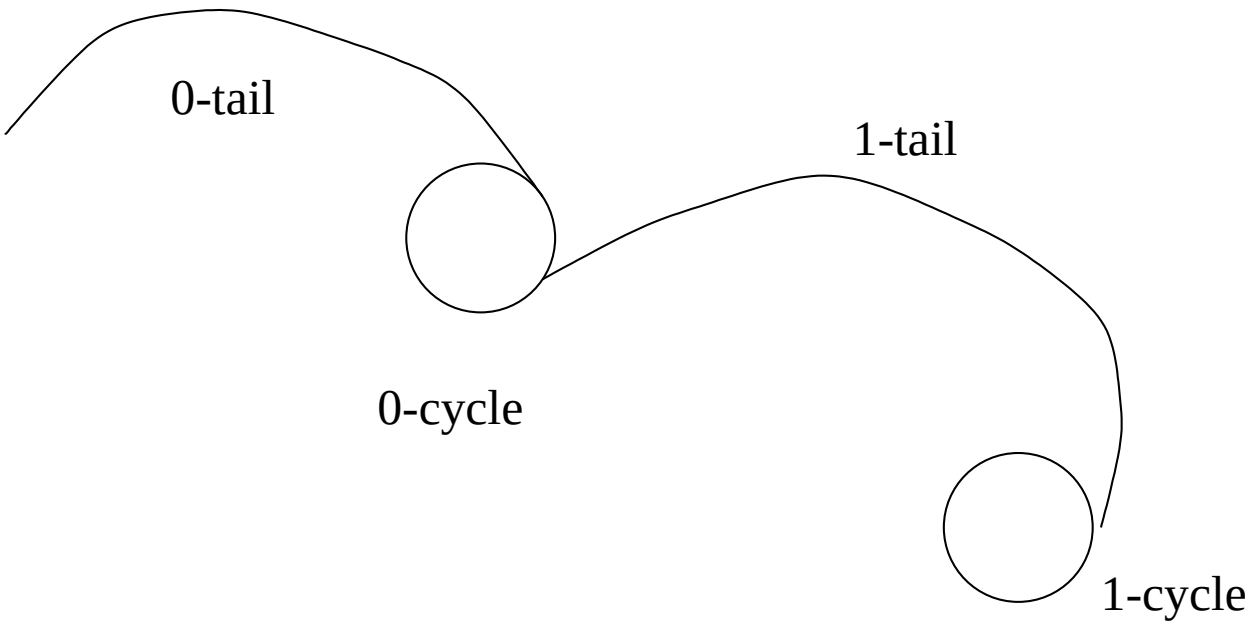
Lower Bounds

Let's now turn to lower bounds.

- Claim: $S(n) = \Omega(\log n)$.
- To see this, consider the two words

$$0^{t-1+\mathrm{lcm}(1,2,\ldots,t)}1^{t-1} \quad \text{and} \quad 0^{t-1}1^{t-1+\mathrm{lcm}(1,2,\ldots,t)}.$$

Proof in pictures:



Lower Bounds

So no t-state machine can distinguish these words.

Now $\operatorname{lcm}(1, 2, \ldots, t) = e^{t+o(t)}$ by the prime number theorem, and the lower bound $S(n) = \Omega(\log n)$ follows.

Separating Words With Automata

Some data:

n	$S(n)$	n	$S(n)$
1	2	10	4
2	2	11	4
3	2	12	4
4	3	13	4
5	3	14	4
6	3	15	4
7	3	16	4
8	3	17	4
9	3	18	5

Separating a Word from Its Reverse

Maybe it's easier to separate a word w from its reverse w^R, than the general case of two words.

However, no better upper bound is known.

We still have a lower bound of $\Omega(\log n)$ for this problem:

Consider separating

$$w = 0^{t-1}10^{t-1+\mathrm{lcm}(1,2,\ldots t)}$$

from

$$w^R = 0^{t-1+\mathrm{lcm}(1,2,\ldots t)}10^{t-1}.$$

Then no DFA with $\leq t$ states can separate w from w^R.

Reverses of Two Words

- Must $\mathrm{sep}(w^R, x^R) = \mathrm{sep}(w, x)$?
- No, for $w = 1000$, $x = 0010$, we have

$$\mathrm{sep}(w, x) = 3$$

- but

$$\mathrm{sep}(w^R, x^R) = 2.$$

Open Problem 2: Is

$$\left|\mathrm{sep}(x, w) - \mathrm{sep}(x^R, w^R)\right|$$

unbounded?

Separating a Word from Its Conjugates

- Two words w, w' are *conjugates* if one is a cyclic shift of the other.
- For example, the English words `enlist` and `listen` are conjugates
- Is the separating words problem any easier if restricted to pairs of conjugates?
- There is still a lower bound of $\Omega(\log n)$ for this problem, as given by

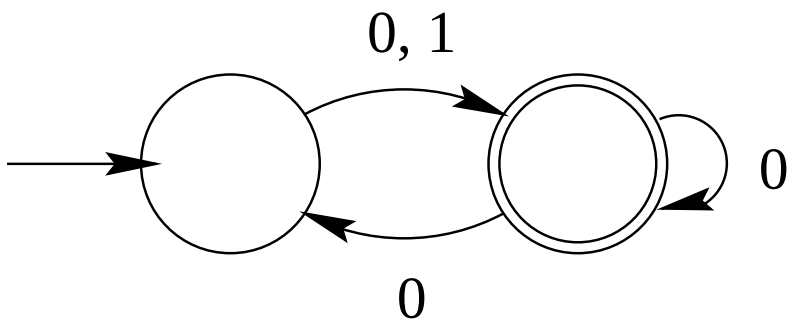
$$w = 0^{t-1}10^{t-1+\mathrm{lcm}(1,2,\ldots t)}1$$

and

$$w' = 0^{t-1+\mathrm{lcm}(1,2,\ldots t)}10^{t-1}1.$$

Separation by NFA's

- We can define $\mathrm{nsep}(w, x)$ in analogy with sep: the number of states in the smallest NFA accepting w but rejecting x.
- Now there is an asymmetry in the inputs: $\mathrm{nsep}(w, x)$ need not equal $\mathrm{nsep}(x, w)$.
- For example, the following 2-state NFA accepts $w = 000100$ and rejects $x = 010000$, so $\mathrm{nsep}(w, x) \leq 2$.



- But there is no 2-state NFA accepting x and rejecting w, so $\mathrm{nsep}(x, w) \geq 3$.

Separation by NFA's

- Do NFA's give more power?
- Yes,

$$\mathrm{sep}(0001, 0111) = 3$$

but

$$\mathrm{nsep}(0001, 0111) = 2.$$

More Variations on Separating Words

Is

$$\mathrm{sep}(x, w)/\mathrm{nsep}(x, w)$$

unbounded?

Yes.

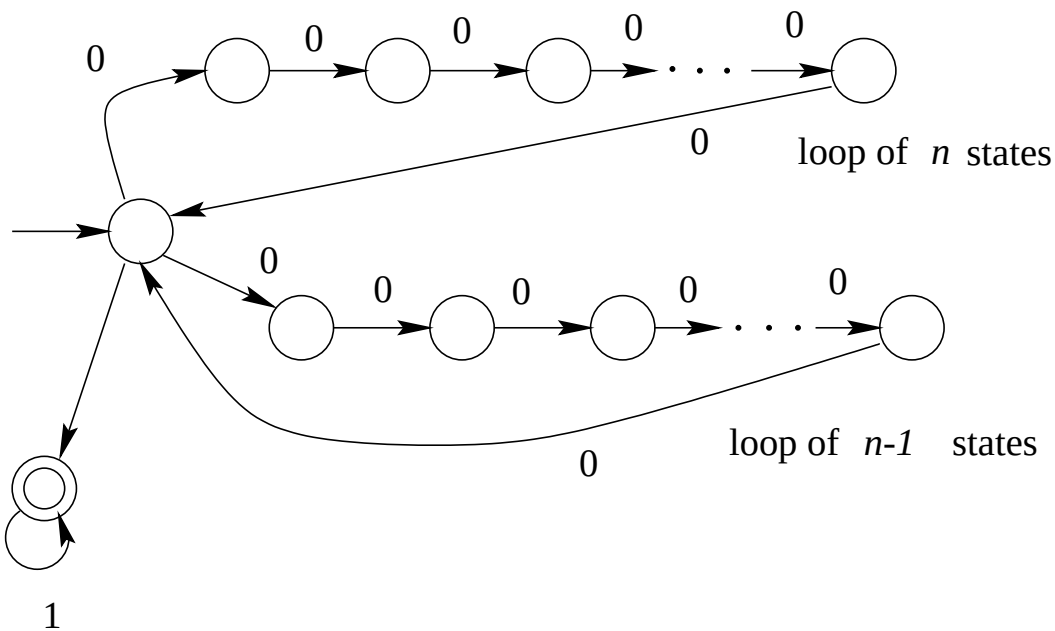
Consider once again the words

$$w = 0^{t-1+\mathrm{lcm}(1,2,\ldots,t)}1^{t-1} \quad \text{and} \quad x = 0^{t-1}1^{t-1+\mathrm{lcm}(1,2,\ldots,t)}$$

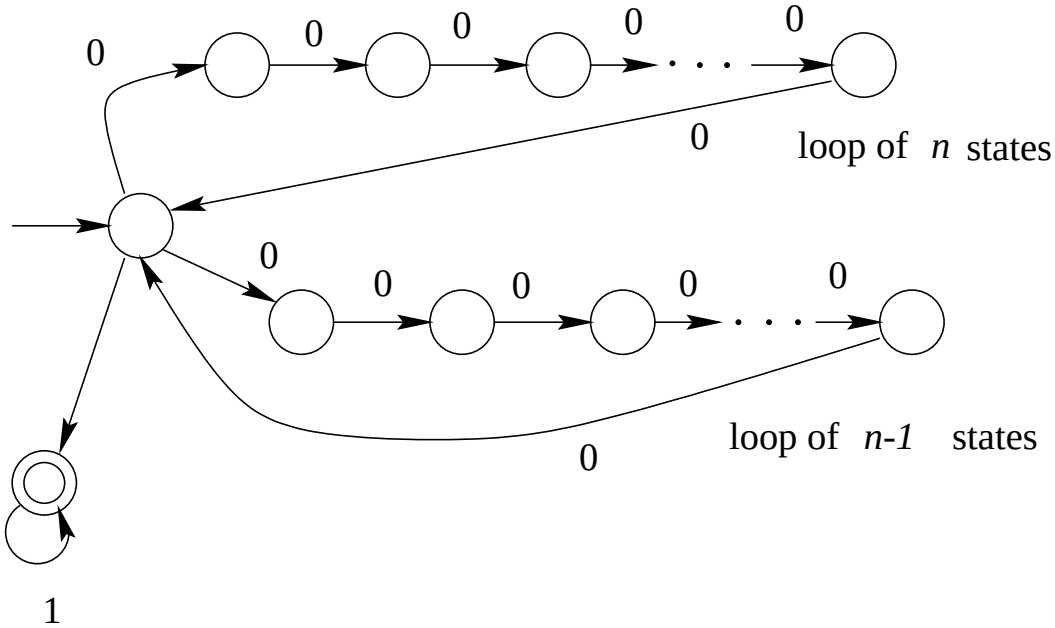
where $t = n^2 - 3n + 2$, $n \geq 4$.

We know from before that any DFA separating these words must have at least $t + 1 = n^2 - 3n + 3$ states.

Now consider the following NFA M:



The language accepted by this NFA is $\{0^a : a \in A\}1^*$, where A is the set of all integers representable by a non-negative integer linear combination of n and $n - 1$.



But $t - 1 = n^2 - 3n + 1 \notin A$ (compute mod $n - 1$ and mod n).

On the other hand, every integer $\geq t$ is in A. Hence
$w = 0^{t-1+\mathrm{lcm}(1,2,\ldots,t)}1^{t-1}$ is accepted by M but
$x = 0^{t-1}1^{t-1+\mathrm{lcm}(1,2,\ldots,t)}$ is not.

M has $2n = \Theta(\sqrt{t})$ states, so
$\mathrm{sep}(x, w)/\mathrm{nsep}(x, w) \geq \sqrt{t} = \Omega(\sqrt{\log |x|})$, which is unbounded.

Lower Bound for Nondeterministic Separation

Theorem. No NFA of n states can separate

$$0^{n^2}1^{n^2+\mathrm{lcm}(1,2,\ldots,n)}$$

from

$$0^{n^2+\mathrm{lcm}(1,2,\ldots,n)}1^{n^2}.$$

Proof. We use the same argument as for DFA's, and the fact (Chrobak) that any unary n-state NFA can be simulated by a DFA with with a "tail" of at most n^2 states and a cycle of size dividing $\mathrm{lcm}(1, 2, \ldots, n)$.

This gives a lower bound of $\Omega(\log n)$ on nondeterministic separation.

Nondeterministic Separation of Reversed Words

A result of Sarah Eisenstat (MIT), May 2010:

Theorem. We have $\mathrm{nsep}(w, x) = \mathrm{nsep}(w^R, x^R)$.

Proof. Let M be an NFA with the smallest number of states accepting w and rejecting x. Now make a new NFA M' with initial state equal to any one element of $\delta(q_0, w)$ and final state q_0, and all other transitions of M reversed. Then M' accepts w^R. But M' rejects x^R. For if it accepted x^R then M would also accept x, which it doesn't.

Open Questions about Nondeterministic Separation

Open Problem 3: Find better bounds on $\mathrm{nsep}(w, x)$ for $|w| = |x| = n$, as a function of n.

Open Problem 4: Find better bounds on $\mathrm{sep}(w, x)/\mathrm{nsep}(w, x)$.

Permutation Automata

Instead of arbitrary automata, we could restrict our attention to automata where each letter induces a permutation of the states ("permutation automata").

For an n-state automaton, the action of each letter can be viewed as an element of S_n, the symmetric group on n elements.

Turning the problem around, then, we could ask: what is the shortest pair of distinct equal-length binary words w, x, such that for all morphisms $\sigma : \{0,1\}^* \to S_n$ we have $\sigma(w) = \sigma(x)$?

You might suspect that the answer is $\mathrm{lcm}(1, 2, \ldots, n)$.

But for $n = 4$, here is a shorter pair (of length 11): 00000011011 and 11011000000.

A Problem in Groups

Now if $\sigma(w) = \sigma(x)$ for all σ, then (if we define $\sigma(x^{-1}) = \sigma(x)^{-1}$) $\sigma(wx^{-1}) =$ the identity permutation for all σ.

Call any nonempty word y over the letters $0, 1, 0^{-1}, 1^{-1}$ an *identical relation* if $\sigma(y) =$ the identity for all morphisms σ.

We say y is *nontrivial* if y contains no occurrences of 00^{-1} and 11^{-1}.

What is the length ℓ of the shortest nontrivial identical relation over S_n?

Recently Gimadeev and Vyalyi proved $\ell = 2^{O(\sqrt{n}\log n)}$.

Separation by Context-Free Grammars

- Given two words w, x, what's the smallest CFG generating w but not x?
- Size of grammar is measured by number of productions
- Problem: right-hand sides can be arbitrarily complicated
- Solution: Use CFG's in Chomsky normal form (CNF), where all productions are of the form $A \to BC$ or $A \to a$.

Separation by Context-Free Grammars

In 1999 Currie, Petersen, Robson and JOS proved:

- If $|w| \neq |x|$ then there is a CFG in CNF with $O(\log \log n)$ productions separating w from x. Furthermore, this bound is optimal.
- Idea: again, if w and x are of different lengths, both $\leq n$, there is a prime $p = O(\log n)$ such that $i = |w| \not\equiv |x| \pmod{p}$.
- We can generate Σ^p in $O(\log p)$ productions in a CFG.
- So we can generate $(\Sigma^p)^*\Sigma^i$ in $O(\log \log n)$ productions.
- There is a matching lower bound.

More on Context-Free Separation

- If $|w| = |x|$ there is a CFG in CNF with $O(\log n)$ productions separating w from x.
- There is a lower bound of $\Omega(\frac{\log n}{\log \log n})$.
- Upper bound is similar to before
- For the lower bound, we use a counting argument.

Open Problem 5: Find matching upper and lower bounds in the case $|w| = |x|$.

Dessert: Another Kind of Separation

Suppose you have regular languages R_1, R_2 with $R_1 \subseteq R_2$ and $R_2 - R_1$ infinite.

Then it is easy to see that there is a regular language R_3 such that $R_1 \subseteq R_3 \subseteq R_2$ such that $R_2 - R_3$ and $R_3 - R_1$ are both infinite.

This is a kind of topological separation property.

Another Kind of Separation

In 1980, Bucher asked:

Open Problem 6: Is the same true for context-free languages?

That is, given context-free languages L_1, L_2 with $L_1 \subseteq L_2$ and $L_2 - L_1$ infinite, need there be a context-free language L_3 such that $L_1 \subseteq L_3 \subseteq L_2$ such that $L_2 - L_3$ and $L_3 - L_1$ are both infinite?

For Further Reading

- J. M. Robson, Separating words with machines and groups, *RAIRO Info. Theor. Appl.* **30** (1996), 81–86.
- J. Currie, H. Petersen, J. M. Robson, and J. Shallit, Separating words with small grammars, *J. Autom. Lang. Combin.* **4** (1999), 101–110.