Cofinite Unary Languages

for nondeterministic finite automata

Zan Xu, Eric Shen

University of Maryland, College Park

Introduction

What are finite automata?

Imagine a computer with a very limited memory. It can only be set to one of several *states*.

The only things it can know are its current state and the next input.

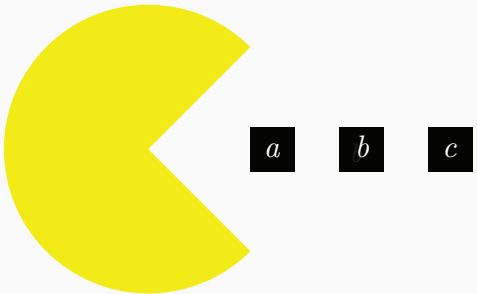
This is how most real-world systems work: think chemical reactions, game theory, regex.

Input and output

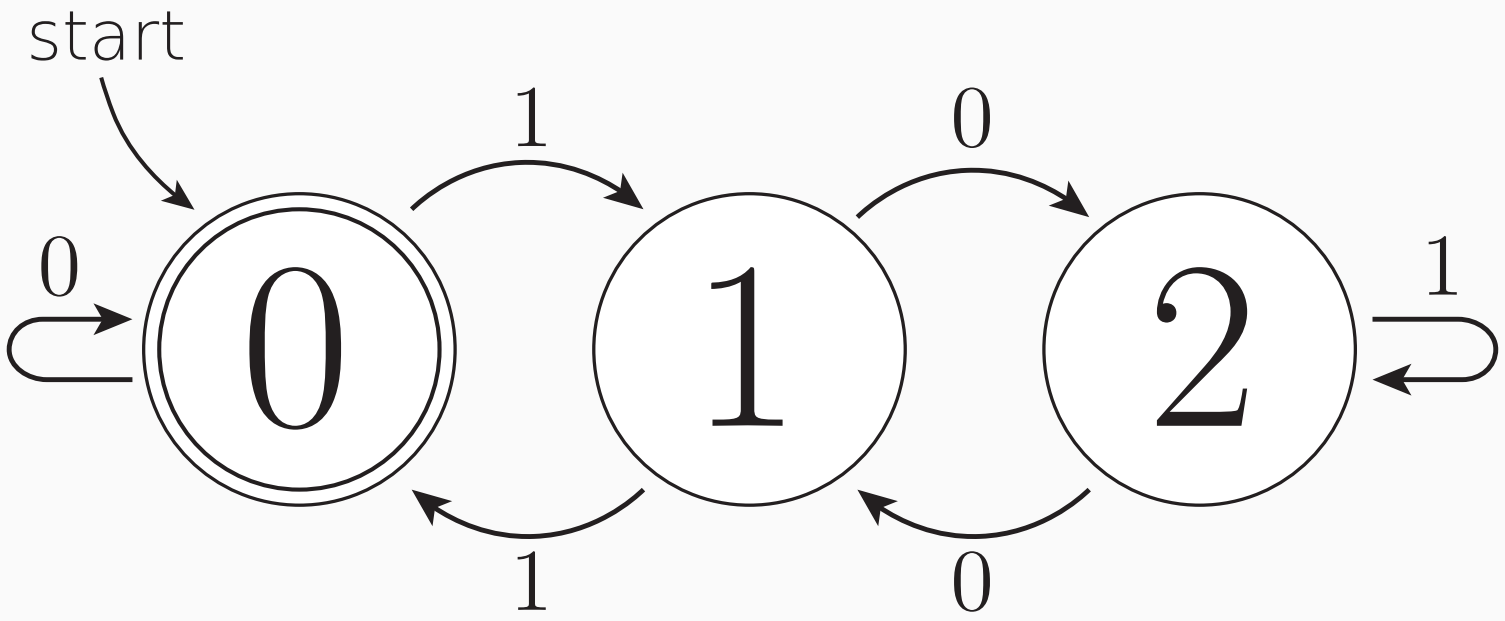
At each step, it changes states based on the current state and the next input *character*.

- Characters are limited to an *alphabet*.
- A sequence of characters is known as a *word*.

The automaton outputs once it has processed the entire word. Its output is boolean; it either *accepts* or *rejects* based on the final state.



Example



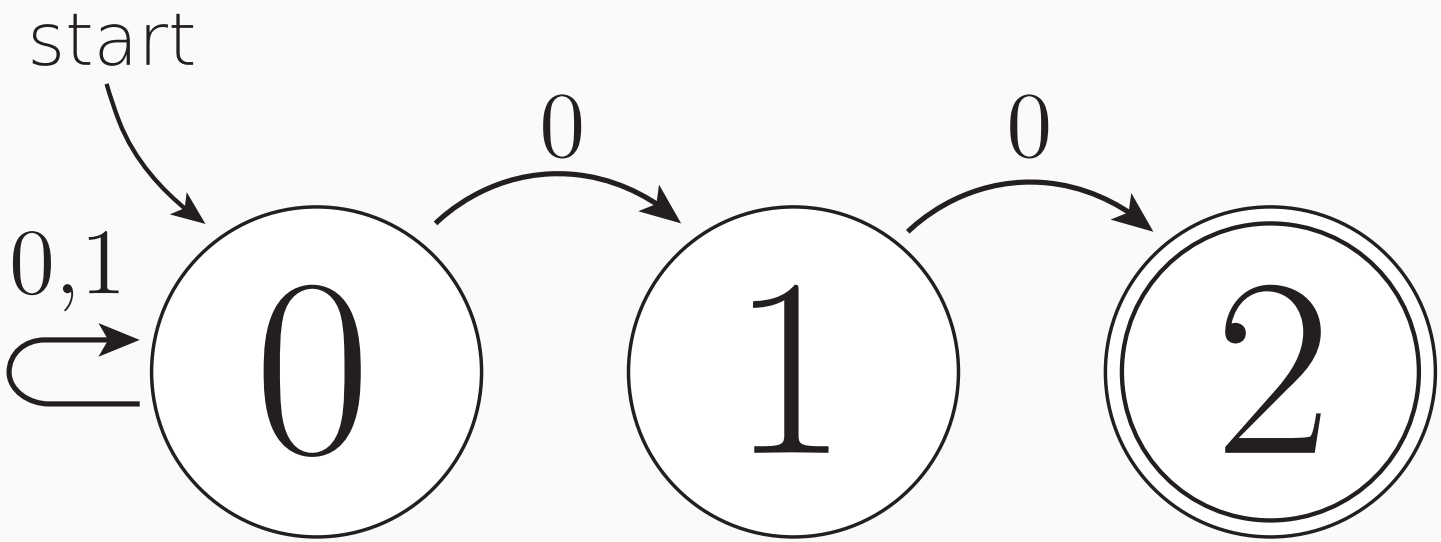
An automaton for divisibility by 3. The input word should be a binary string.

Adding nondeterminism

That was a deterministic finite automaton. For every (state, character) pair, the automaton will move to another specific state.

But what if we could move to multiple states at once?

NFA example



This NFA accepts only strings that end in ‘00’. If at least one of the states the machine ends on is an accept state, then the entire automaton accepts.

Research Problem

Cofinite Unary Languages

Definitions

Cofinite Rejects a finite set of words

Unary Our alphabet only has one character

Language A set of words that an automaton accepts

We'll call only character a. We'll use a^i to refer to a word of length a^i, so a^{30} would be the word "*aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa*".

Problem

Given a cofinite unary language, how small can we make our NFA?

Background

Frobenius coin problem

- If I have some coprime denominations of coins, what is the biggest amount of money I cannot make using these types of coins?

Frobenius coin problem

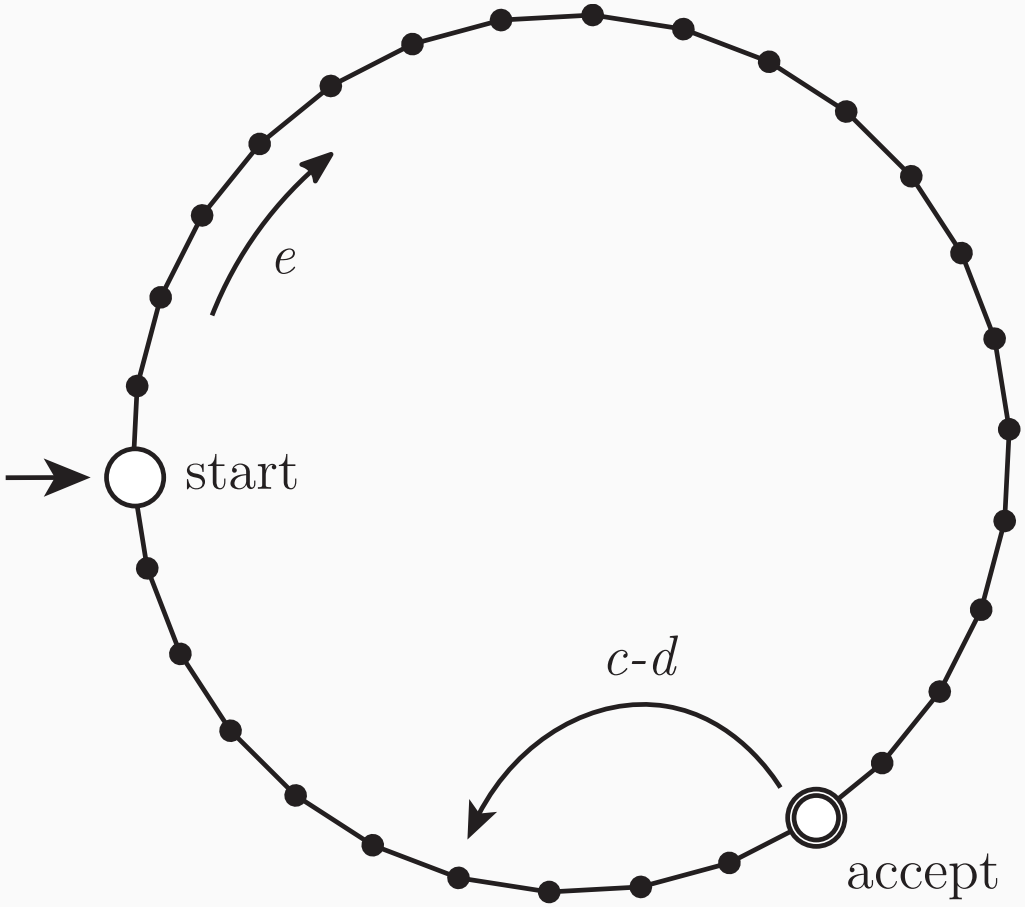
- If I have some coprime denominations of coins, what is the biggest amount of money I cannot make using these types of coins?
- For the 2 coin case, if the coins are m and n the biggest amount you cannot make is $mn - m - n$.

LOOP(c, d, e)

Consider 2 loops of c and d states respectively, starting from the same accept state. The lengths of words accepted are diophantine linear combinations of c, d, i.e., $Cc + Dd = n$ given discrete values.

We don't have to start at the shared state though. Imagine if we started e behind the accept state. Then we'd need our solutions to have e more states to work, so our loops accept anything that's $Cc + Dd + e$.

LOOP(c, d, e) Diagram



What's the longest unary string not accepted by this loop?

Landau's function

Landau's function, $g(n)$ is defined to be the maximum LCM of a partition of n. Basically, for a given n, we want to find some prime powers with the biggest product and sum $\leq n$.

Example

Let $n = 15$. Then the best partition is $\{3, 5, 7\}$ with product 105.

Prime power loops

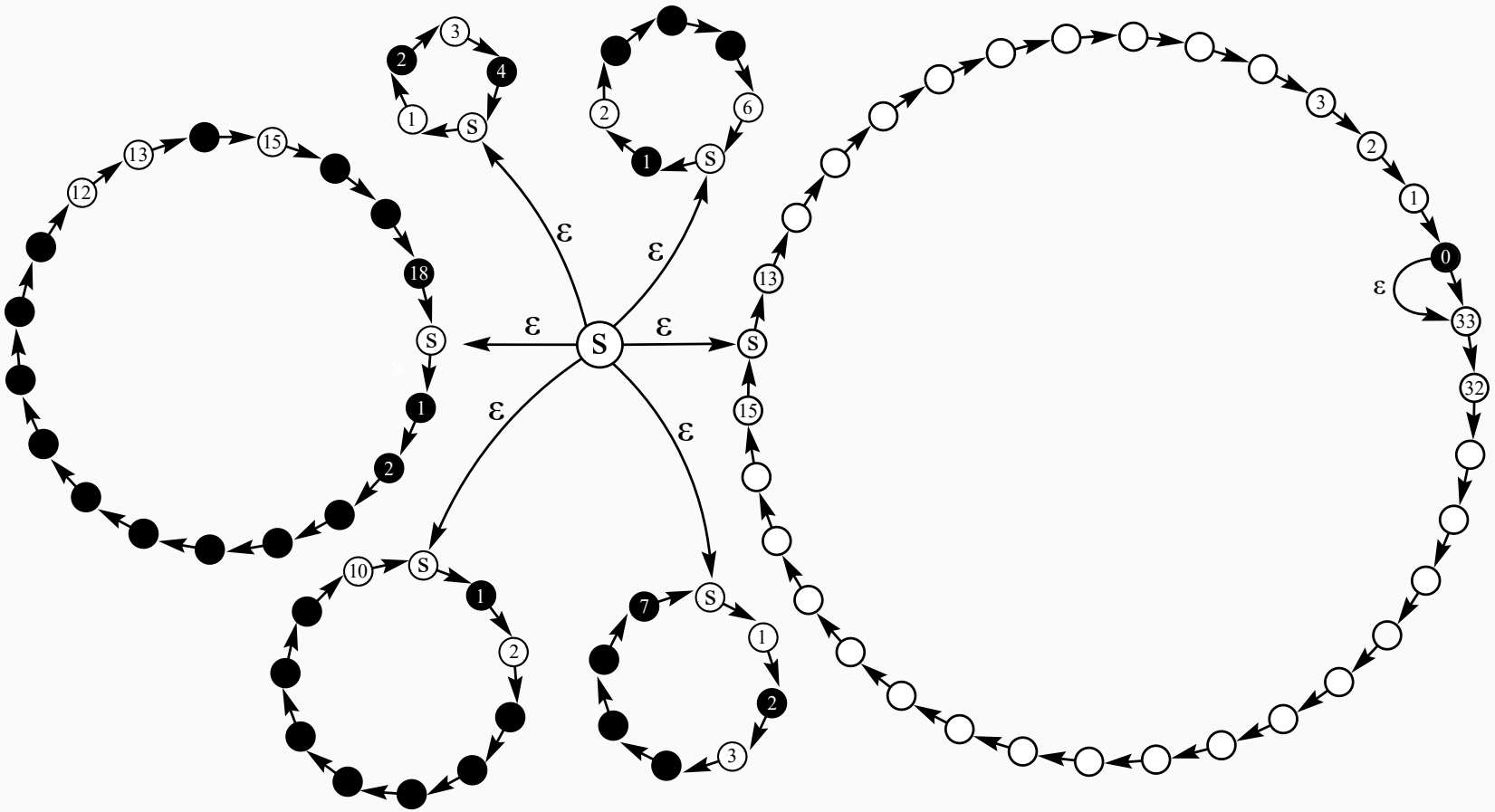
Let's run our word of length n through some (independent) loops of size $p_1, p_2, \ldots, p_m$. And, let's construct these loops such that they accept everything *except* for n mod p_i for each loop p_i.

Then the only string that's rejected by all of them is the one that's equal to n mod p_i for all p_i.

Chinese Remainder Theorem

If you know n mod p_i for some coprime $p_1, p_2, \ldots, p_m$, then you can uniquely determine n under $\prod_1^m p_i$.

$MN(1000, 1001, 1003)$



NFA for language $\{a^i \forall i \notin \{1000, 1001, 1003\}\}$

$MN(\{n\})$

In general, if we want to reject only some set of lengths S, we can't do better than $\sqrt{\max(S)}$ states. proof

If we're only rejecting a single word a^n:

- The big c, d, e loop takes no more than $\lceil \sqrt{n} + 2 \rceil$ states. proof
- For a given number of states Landau's function gives us the biggest n we can make prime powers for. This works out to be under $\ln^2 n$.

$MN(\{n, kn\})$ and beyond

If we're rejecting a pair of words a^n, a^{kn}:

- The big c, d, e loop takes no more than $\lceil \sqrt{n} + 3 \rceil$ states. proof
- Landau's no longer works. However, the greedy result still yields a result of $O(\log^2 n + 1)$ for the prime loops (not necessarily prime powers anymore)

In general, the modular loops portion of a set of size ℓ uses at most $\ell^2 \log^2 n + O(1)$ states. proof

Future Research

- Non-sparse missing numbers
- k<2 case

Ask questions
Ask for proofs

Frobenius Questions

Why are there chicken nuggets?

Before the introduction of the 4-piece chicken McNuggets, McNuggets were sold in boxes of 6, 9, and 20. The most chicken nuggets you cannot buy is 43. Finding the numbers of chicken nuggets you cannot buy from McDonalds is an example of the Frobenius problem.

back to slide

Landau's Questions

How can the sum be less than n?

Because we can't possibly decrease the LCM if we just include the remainder.

Why is the optimal partition coprime?

Consider if they were ab and ac. Then we could use ab and c for a smaller sum and the same product.

And prime powers?

If it's composite e.g. ab we could use a and b separately for a more efficient solution.

back to slide

Real-life applications

Why is this useful?

no

Proof of Chinese Remainder Thm

Proof

$n \mapsto \{n \bmod p_1, n \bmod p_2, \dots\}$ is injective over the domain $[1, \prod p]$ because you can compute a unique set of mods for any n. By fundamental counting principle are only $\prod p$ possible sets of mods. Since the size of the domain and codomain are equal there is a n for every set of mods.

back to slide

Bounds I: Minimum states required

Proof

Previous research [Chroback] showed that we can convert an NFA of size m for a unary cofinite language to a chain of size $\leq m^2$ terminating in some disjoint loops via ϵ transitions. If we could recognize $MN(S)$ in fewer than $\sqrt{n}$ states we could convert it into a chain of size $\leq n$ terminating in some cycles. Therefore if we plug a^n in it will terminate in the cycles. But since our loops are periodic that would mean there is an infinite number of other words also accepted, and so our language is no longer cofinite. By contradiction our NFA must take at least $\sqrt{n}$ states.

back to slide

Bounds II: c, d, e loop takes $\sqrt{n} + O(1)$

Proof

The loop $c = \lceil \sqrt{n} + 1 \rceil$, $d = c + 1$, $e \equiv n + 1 \pmod{c}$ must reject n. If it accepts n then by definition for some integer solutions:

$$cC + (c+1)D + e = n$$

$$n \equiv D + e \pmod{c}$$

$$n - e \equiv D \equiv n - (n+1) \equiv -1 \pmod{c}$$

Therefore, since D is positive, $D \geq c - 1$. But algebraically, $(c+1)D > n$, yielding a contradiction.

back to slide

Bounds III: c, d, e loop takes $\sqrt{n} + O(1)$ for n, kn

Proof Outline

1. We solve the 1-element case for kn (see Bounds II).
2. We show that given that c,d,e, the loop rejects n if it falls in a certain range $\pmod{c}$.
3. We re-express this range in base $c + 1$.
4. We show that for all but an edge case if it does not fall in the range from step 2, then incrementing c,d and recalculating e works if we reapply step 1.
5. We patch this edge case by changing base again and doing it all over.

back to slide

Bounds IV: Mod loops must take $\ell^2 \log^2 n$

Proof Outline

1. Over all coprime sets $B \subset \mathbb{N}$ with $\prod B > N$, $\min \sum B \leq \log^2 N$
2. Then there's a set of loops with $N = n^\ell$
3. Construct this and plug in elements of S
4. Show that every other word that also goes through this NFA must have size $> N^\ell$

back to slide

Bibliography

[1] M. Chrobak. *Finite Automata and Unary languages*. Theoretical Computer Science, 47:149– 158, 1986.

[2] J.-P. Massias, J.-L. Nicolas, and G. Robin. *Effective Bounds for the Maximal Order of an Element in the Symmetric Group* Math. Comp., 53(188):665–678, 1989.