Languages and grammar

We use languages to communicate with each other but also with machines (e.g. prog lang to exec algo)

(syntax, semantics)

A alphabet Σ is a finite set e.g. $\{0,1\}$, $\{a,b,c\}$...

The set Σ^* is the set of all strings over Σ

A language L over the alphabet Σ is any subset $L \subseteq \Sigma^*$.

The languages we are interested in are usually infinite, but we want to describe them finitely. We do so using grammars.

Ex The following is a grammar for binary strings starting with 0.

Start symbol

terminals

$$\begin{aligned} S &\to 0A \\ A &\to 0A \\ A &\to 1A \\ A &\to \lambda \end{aligned}$$

← each rule is a production rule and tells you how to "rewrite" a string.

empty string

non-terminals

$$S \to 0A$$
$$A \to \lambda \mid 0A \mid 1A.$$

For a grammar G, $L(G)$ is the strings you can get by "rewriting" starting with S.

Ex

$$S \to \lambda \mid 0S1 \mid 1S1.$$

What language?

All pallindromes of even length.

To get all pallindromes?

$$S \to \lambda \mid 0S1 \mid 1S1 \mid 0 \mid 1.$$

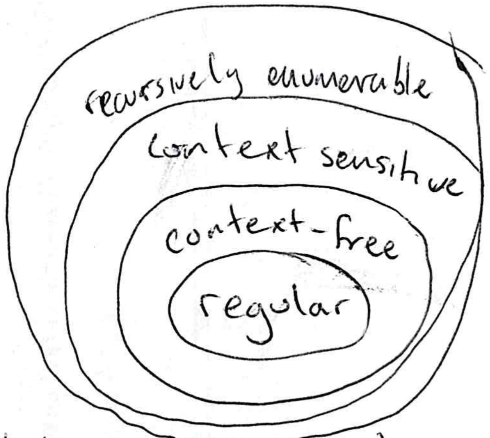
The Chomsky Heirarchy

Avram Noam Chomsky (born Dec 7, 1928)
(american linguist, philosopher, political commentator)
wanted to find a formalism for types of languages used.

It wasn't enough for natural languages but it is what was needed for interacting with computers.

- recursively enumerable
  - no restrictions on production rules
  - any algorithm can be written using these.
- Context-sensitive: $\alpha A \gamma \to \alpha w \gamma$ $(w \neq \lambda)$
- context-free: $A \to w$
  - this is all we will look at.
  - this is used for syntax of programming languages
- regular: $\underbrace{A \to a,\ A \to aB,\ A \to \lambda}_{\text{left regular}}$.

  regular if left or right regular.



Relating this back to combinatorics!

The generating function for a language L is $\sum_{w \in L} x^{|w|}$.

If L is regular then it is rational: $\frac{p(x)}{q(x)}$

so satisfies a linear recurrence with constant coefficients.

Thm (Chomsky-Schützenberger)

If L is unambiguous context-free it is algebraic.

unique derivation.

- the generating function f satisfies $P(f, x) = 0$. (with rational coefficients)

You may hope that the next will correspond to D-finite / holonomic

$[\,p_k(x) f^{(k)} + p_{k-1} f^{(k-1)} \dots p_1(x) f' + \dots\,]$

but it doesn't quite.

[See Mishna and Zabrocki, 2008 where they give a type of grammar for D-finite functions]

We will only work with context-free and regular.

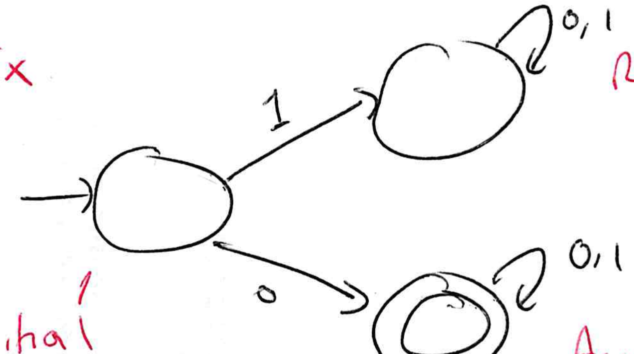
Finite Automata

What would a device for recognizing strings starting with 0 look like?

- Read input string, one sybol at a time and decide what to do based on what is read
- maintain some info on what has been read.
- having read it decide if it is language or not

(8)

Ex



Reject as start with 1.

initial state (depicted with incoming arrow)

Accept as start with 0. Draw two circles

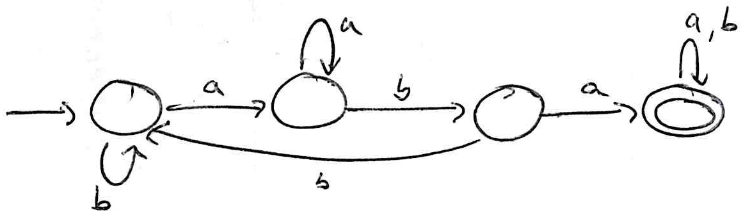
DFA: $M = (S, \Sigma, f, s_0, F)$

- S finite set of states
- ~~Σ~~ Σ is finite alphabet
- $f: S \times \Sigma \to S$ is transition function
- $s_0 \in S$ is start state
- $F \subseteq S$ set of final/accepting states

$L(M)$ = strings that end in accept state.

Ex

$L \subseteq \{a,b\}^*$ ~~#~~ of strings containing aba



Thm

~~Let~~ Let M be a DFA then $L(M)$ is regular.

Proof (sketch) algo to reg grammar

Thm

Let G be a regular grammar, then there is a DFA, M_G, s.t $L(G) = M_G$.

Proof (sketch)

construct (and define) an NFA. which is equiv to a DFA.

Insertion encoding

(Albert, Linton and Ruskuc, 2005)

(Vatter, 2012)

Every length n permutation can be created by inserting a new maximum to a length $n-1$ permutation.

This is true for permutation classes too.

Ex 325146

◇ ← this represents a future point

a slot

◇1◇

◇2◇1◇

32◇1◇

32◇14◇

32514◇

325146

these are called configuration.

together the evolution.

There are four ways to insert a new maximum

◇ ⟼ ◇n◇ represent by m (middle)

◇ ⟼ n◇ l (left)

◇ ⟼ ◇n r (right)

◇ ⟼ n f (fill or final)

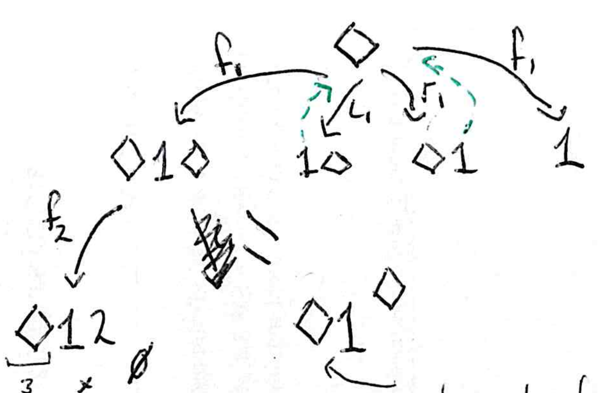
If we subscript by the slot we insert into, this encoding is unique

Ex 325146 is encoded by the word $m_1 m_1 f_1 l_2 f_1 f_1$.

This is the insertion encoding

For a permutation class C, $L(C)$ is the language formed by the insertion encodings of the permutations in C.

Example Av(312)



we have to fill this first

Discuss infinite automaton.

We can only use f_1, l_1, r_1, m_1.

$$S \to f \mid lS \mid rS \mid mSS$$

This is unambiguous so the generating function satisfies

$$F = x + xF + xF + xF^2$$

solving gives

$$F = \frac{1 - 2x - \sqrt{1-4x}}{x}$$

which is the generating function for catalan numbers.

Question: When does a permutation class have a regular insertion encoding?

If it is regular it is accepted by some DFA.

Let say it has k states.

For any prefix p of any word there exists a word of length at most $k + |p|$. Since can choose a word that doesn't revisit states.

This says, there is a maximum number of slots allowed on a configuration if it is regular.

Let $SB(k)$ be the set of permutations whose insertion encodings never has more than k slots.

This is the permutations avoiding all
$babab\ldots ab$
where $b \in \{k+1, \ldots, 2k+1\}$
$a \in \{1, \ldots, k\}$.

Theorem (Albert, Linton, and Ruškuc) 2005.

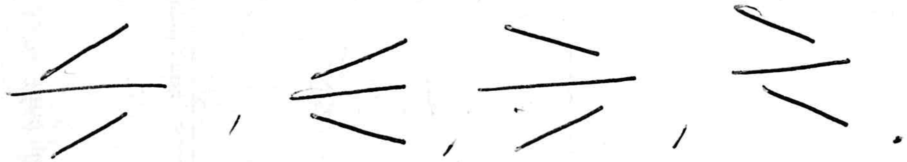
Every finitely based subclass of $SB(k)$ has a regular insertion encoding.

Proof

We defer this to later where we show how to build the DFA accepting it.

Proposition

A class $Av(B)$ has a regular insertion encoding if and only if there is a perm from each of the following in the basis



Proof Use the Erdős-Szekeres theorem.

($\Leftarrow$): these can be found in basis of $SB(k)$

($\Rightarrow$): Take pattern of form $abab\ldots ab$ (perhaps by extending).

for ≤

Let $k = n^4$ and θ be an element of basis of $SB(k)$

Amongst k smallest there is monotone sequence of size n^2,

Write $\underbrace{ab\,ab\ldots ab}_{\text{monoton}}$ of size $2n^2$

then b's contain monotone sequence of size n, so get element in one of classes above. □

Corollary

There is a linear time check if $Av(B)$ has a regular insertion encoding.

Side note

This result is similar to one of my favorites in PP. about polynomial permutation classes.

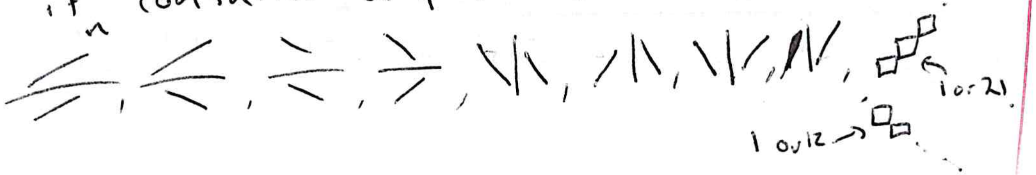
Thm Kaiser and Klazar

Fibonacci Dichotomy.

For a permutation class C, there is a polynomial $|C_n| = p(n)$ ← polynomial for all sufficiently large n if and only if $|C_n| < F_n$ for some n.

Thm Huczynska and Vatter (2013)

A class is polynomial ~~iff~~ if and only if ~~contains~~ its basis contains a perm from each of



So we know precisely when a class has a regular insertion encoding. Can we find the DFA accepting it, and thus get the rational generating function? Yes!

The idea will be to minimise the infinite automaton with states being acceptable ← contains some perm in C. configurations (initial is $\diamond$) with the transitions from letters.

Accepting states are slotless.

Clearly, this accepts the language. Do example on page 9 here.

Two configs with prefix p_1, p_2 are distinguishable if there is a word w s.t. $p_1 w \in L(C)$ and $p_2 w \notin L(C)$.

Thm Vatter (2012)

For a configuration $c = c_1 \dots c_n$, if $c \neq \diamond$ nor adjacent two $\diamond$s then it is decidable if c and $c - c_i$ are distinguishable.

Proof ~~[illegible]~~

Claim: If b is length of longest element in B, then distinguishable if and only if word of length at most $b-1$ that distinguishes.

No words then true.
Let p be prefix of c and p' for $c - c_i$
If $pw \in L(C) \Rightarrow p'w \in L(C)$, so
if w distinguishes then must have $p'w \in L(C)$ and $pw \notin L(C)$
pw will give a perm π as same slots, so this contains a pattern in B. Choose occurrence, and ignore entries in π that are neither in perm of c or occurrence.
We then have word of length at most $b-1$. ∎

Thm (Vatter)

$L(C)$ is regular $\iff$ for sufficiently long c some c_i s.t. c and $c - c_i$ are indistinguishable

Proof

$\Leftarrow$: we have a DFA now so regular

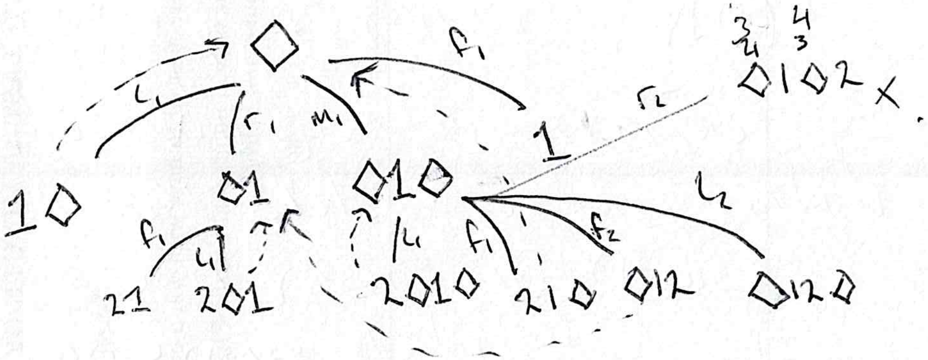
$\Rightarrow$: Slot bounded, say k.

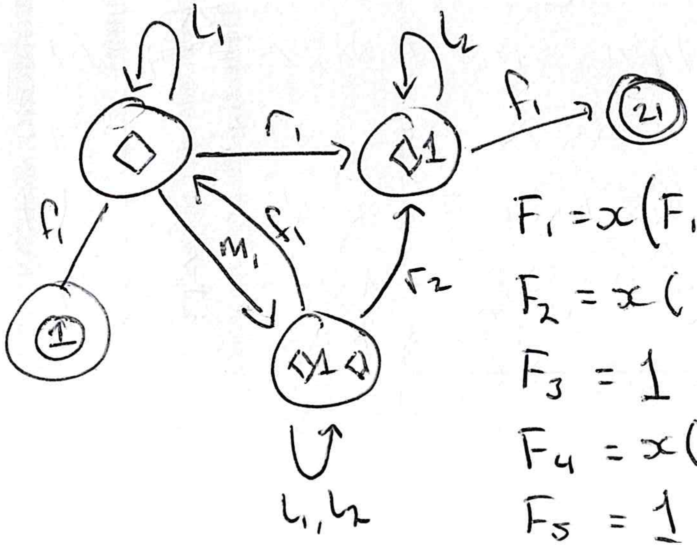
If c and $c-c_i$ are distinguishable, word w of length at most $b+k-1$.
Call w a witness for c_i. It can witness at most b entries of c.
For long enough some c_j with no witnesses so c and $c-c_j$ are indistinguishable. ∎

The implications is an algorithm that terminates.

I set as a exercise to implement your own version using Python and the comb spec searcher.

Ex Av(321, 3142)





$$F_1 = x(F_1 + F_2 + F_3 + F_4)$$
$$F_2 = x(F_2 + F_3)$$
$$F_3 = 1$$
$$F_4 = x(F_2 + F_1 + 2F_4)$$
$$F_5 = 1$$

Solving gives

$$F_1 = \frac{x - x^2}{1 - 3x + x^2} \leftarrow \frac{\text{fibonacci}}{F_{2n-1}}$$

The underlying idea to this algorithm is to find occurrences of patterns to see when configurations differ.

Why not keep track of them

This is what the telescope algorithm does from my thesis

Two states are equal if they have the same occurren

We are going pictoral now!