MA4199 Honours Project in Mathematics

Automatic Complexity

Choo XianJun, Davin[1]
Supervisor: Dr. Frank Stephan[1]

[1]Department of Computer Science
National University of Singapore

Overview of talk

Content

- Preliminaries, prior work and definitions
- General bounds for DFA, NFA, and CFG
- Morse-Thue sequence
- Palindromes
- Power length sequence
- Martin-Löf Random strings
- Program computation of explicit DFA bounds

Style

- Will use a combination of slides and board to draw diagrams

Enumeration schemes

$\{0,1\}^*$ means any permutation of 0's and 1's, including empty string ϵ.

- Natural numbers $\mathbb{N}$: $\{0,1,2,3,4,5,6,7,8,...\}$
- Binary numbers $\mathbb{B}$: $\{0,1,10,11,100,101,110,111,1000,...\}$
- Length-lexicographical $\mathbb{L}$: $\{\epsilon,0,1,00,01,10,11,000,001,...\}$

There exists bijections $b : \mathbb{N} \to \mathbb{B}$ and $ll : \mathbb{N} \to \mathbb{L}$.
Note: the $ran(b) \subset \{0,1\}^*$ while $ran(ll) = \{0,1\}^*$.

- $b(n) \to ll(n)$: Binary-add 1, remove msb of 1
- $ll(n) \to b(n)$: Add msb of 1, binary-minus 1

$$\begin{aligned} x &= x[0]x[1]x[2]x[3]... && \text{(Index values in } \mathbb{N}) \\ &= x(ll(0))x(ll(1))x(ll(2))x(ll(3))... \\ &= x(\epsilon)x(0)x(1)x(00)... && \text{(Index values in } \mathbb{L}) \end{aligned}$$

String operations

- Length $|x|$
- Prefix x_n
- Concatenation xy
- Repeat x^p
- Reverse x^R
- Bit flip $\overline{x}$
- Repeated bit flip $flip(x, n)$
- Automata reading $A(i)$
- Character/Bit indexing $x[j]$
- Substring indexing
  $x[a : b] = x[a]x[a + 1]...x[b]$

$$x = 111010$$

$$y = 110$$

Automata

Can be defined with a tuple: $(Q, \Sigma, \delta, q_0, F)$

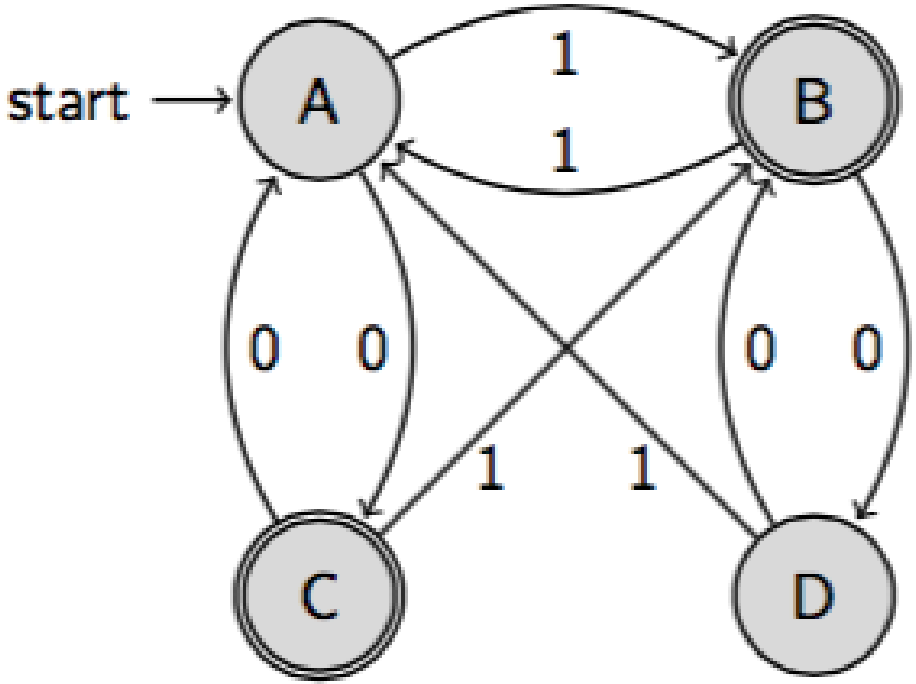
where	Q	$\subseteq$	$\mathbb{N}$	is the set of states
	Σ	$=$	$\{0, 1\}$	is the set of symbols
	δ	$:$	$Q \times \Sigma \to Q$	is the transition function
	q_0	$\in$	Q	is the initial/starting state
	F	$\subseteq$	Q	is the set of accepting/final states

DFA vs. NFA: Accepting conditions, Space of transition possibilities

Example of an automaton

This automata accepts $\{0, 1, 01, 000, 001, 100, 110, 111, ...\}$

$Q = \{A, B, C, D\}$
$\Sigma = \{0, 1\}$
$q_0 = A$
$F = \{B, C\}$
$\delta(A, 0) = C, \delta(A, 1) = B$
$\delta(B, 0) = D, \delta(B, 1) = A$
$\delta(C, 0) = A, \delta(C, 1) = B$
$\delta(D, 0) = B, \delta(D, 1) = A$



Prior work

$$A(x) = \min\{|M| : \forall y \in \{0,1\}^*, (|y| = |x|) \Rightarrow (M(y) = 1 \iff y = x)\}$$

- $|x| = n$
- M is an automaton
- Of all length n strings, M will only accept x and reject the rest
- Notes:
  - Domain of input to an automaton is $\mathbb{L} = \{0,1\}^*$
  - $M(y) = 1$ if M accepts y
  - $M(y) = 0$ if M rejects y

Prior work

$$A(x) = \min\{|M| : \forall y \in \{0,1\}^*, (|y| = |x|) \Rightarrow (M(y) = 1 \iff y = x)\}$$

- **(Shallit and Wang) M is a DFA**

  For almost all strings x, $A_D(x) \leq \frac{3}{4}|x| + \log(|x|)\sqrt{\frac{|x|}{8}}$

  $\exists$ constant C, such that for almost all strings x, $A_D(x) > C\frac{|x|}{\log(|x|)}$

- **(Hyde and Kjos-Hanssen) M is a NFA**

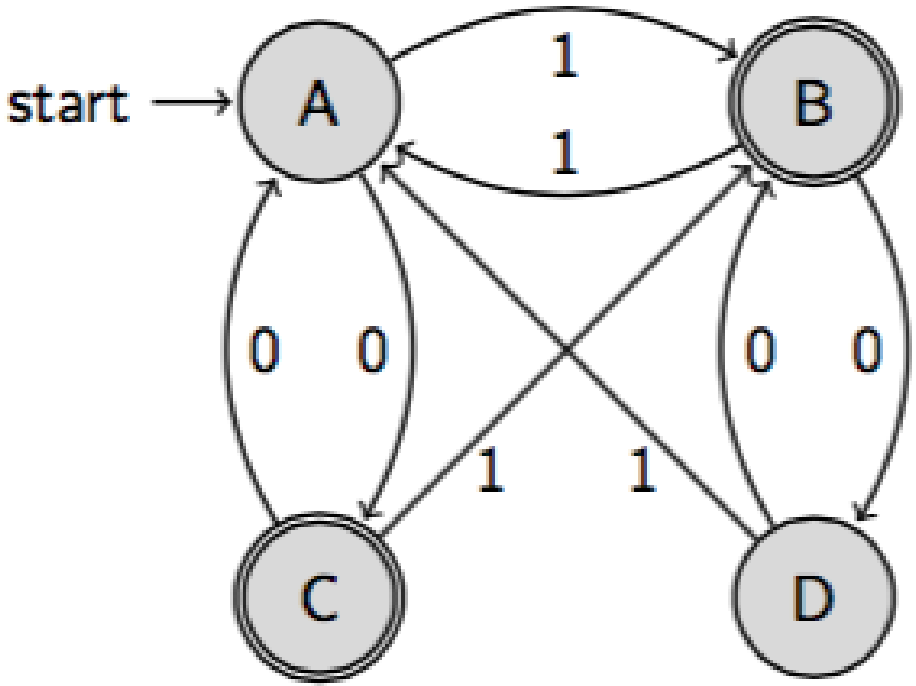
  For all strings x, $A_N(x) \leq \frac{|x|}{2} + 1$

  For almost all strings x, $\forall b > 0, b \neq 1$, $A_N(x) > \sqrt{\frac{|x|}{\log_b(|x|)}}$

Example illustrating prior work definition

This automata accepts $\{0, 1, 01, 000, 001, 100, 110, 111, ...\}$
Of all length 2 strings, only accepts 01. So, $A_D(01) \leq |Q| = 4$

$Q = \{A, B, C, D\}$
$\Sigma = \{0, 1\}$
$q_0 = A$
$F = \{B, C\}$
$\delta(A, 0) = C, \delta(A, 1) = B$
$\delta(B, 0) = D, \delta(B, 1) = A$
$\delta(C, 0) = A, \delta(C, 1) = B$
$\delta(D, 0) = B, \delta(D, 1) = A$



Definitions

There is 2 common methods to look at a binary string:

(i) Look at the bits themselves **(Prior work)**

(ii) Look at the indices referencing the bits **(Our approach)**

When we consider a characteristic function for a set $S \subseteq \mathbb{N}$:

(i) Element-wise: χ_S is a function s.t. $\chi_S(x) = 1 \iff x \in S$

(ii) Index-wise: $x_s \in \{0,1\}^*$ s.t. $x_s[i] = 1 \iff i \in S$

E.g. $S = \{\text{Even numbers}\} \subseteq \mathbb{N}$, then $x_S = 101010...$

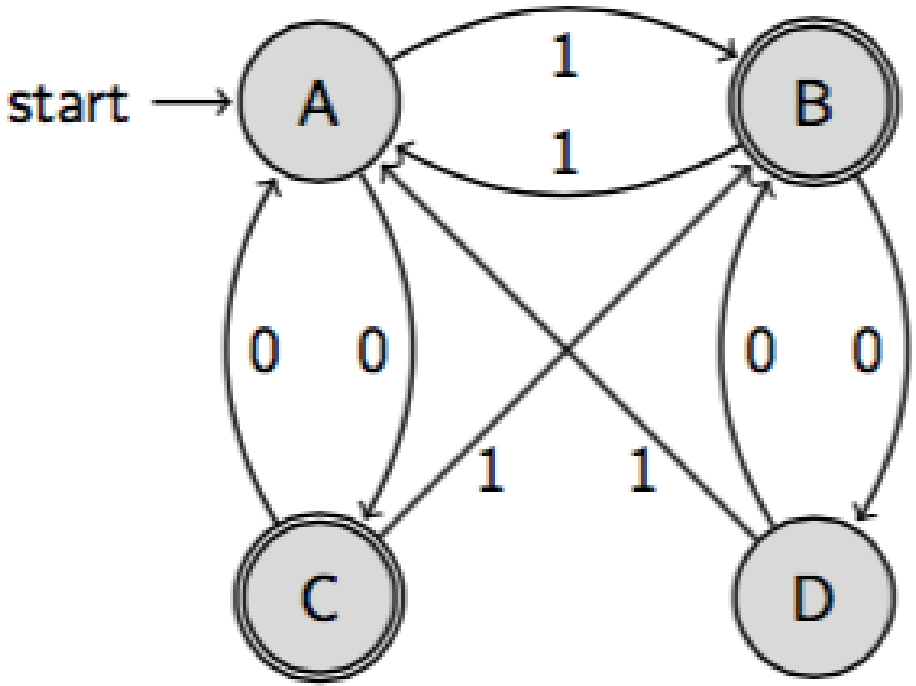
Definitions

- An automaton M is said to **produce** a binary string x if $M(ll(i)) = x[i], \forall i \in \mathbb{N}$
- For a NFA, $M(ll(i)) = 1 \iff$ $\exists$ an accepting path from q_0 in M_N upon reading $ll(i)$
- $D(x) = \min\{|M| : M \text{ is a DFA that produces } x\}$
- $N(x) = \min\{|M| : M \text{ is a NFA that produces } x\}$
- Trivial bound: $N(x) \leq D(x) \leq |x|$ (Proof: Trivial automaton T_x)

Example illustrating our definition

This automata accepts $\{0, 1, 01, 000, 001, 100, 110, 111, ...\}$
Hence, it produces $x = 0110100$. So, $D(0110100) \leq 4$.

$Q = \{A, B, C, D\}$
$\Sigma = \{0, 1\}$
$q_0 = A$
$F = \{B, C\}$
$\delta(A, 0) = C, \delta(A, 1) = B$
$\delta(B, 0) = D, \delta(B, 1) = A$
$\delta(C, 0) = A, \delta(C, 1) = B$
$\delta(D, 0) = B, \delta(D, 1) = A$



Trivial tree automaton T_x

For any given string x, we can build a **trivial tree automata** T_x that represents x such that $T_x(\mathit{II}(i)) = x[i], \forall i \in \mathbb{N}$

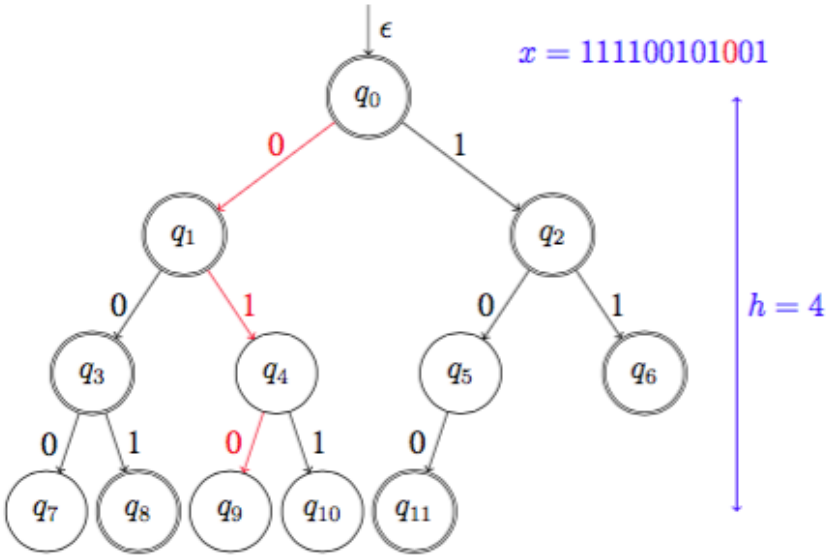


Figure 1.1: Trivial tree automaton of x

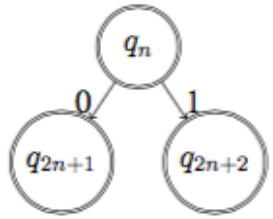


Figure 1.2: State transition in trivial tree automata

Definitions

We can define a context free grammar (CFG) by a tuple (NT, T, R, S),

$$
\begin{array}{lrcll}
\text{where} & NT & = & \{a_i : i \in \mathbb{N}\} & \text{is the set of non-terminals} \\
& T & = & \{\epsilon, 0, 1\} & \text{is the set of terminals} \\
& R & : & NT \to (NT \cup T)^* & \text{is the set of production rules} \\
& S & \in & NT & \text{is the starting symbol}
\end{array}
$$

Definitions

- **k-bounded CFG**
  For $k \geq 2$, a context free grammar is k-bounded if for every rule in R, the rules produce at most k symbols.
  i.e. If we have a rule $A \to w$, then $2 \leq |w| \leq k$.
- **Produce**
  A k-bounded CFG $G = (NT, T, R, S)$ *produces* a binary string x if

$$S \Rightarrow^* ll(i) \iff x[i] = 1, \forall i \in \mathbb{N}$$

- **k-bounded CFG complexity of binary string x**

$$C_k(x) = \min\{|NT| : G \text{ is a } k\text{-bounded CFG that generates } x\}$$

Notes

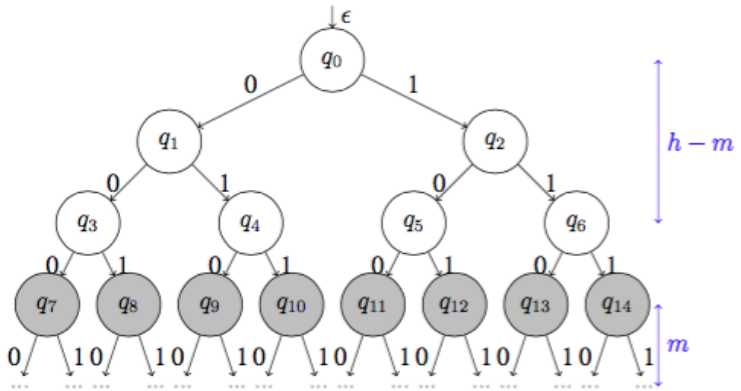
- We will now go through our results
- Due to time constraints, we only illustrate some proof sketches
- We will focus mainly on automata
- For CFG proof sketches, refer to report or ask me after the talk! :)

General bounds for DFA, NFA, and CFG

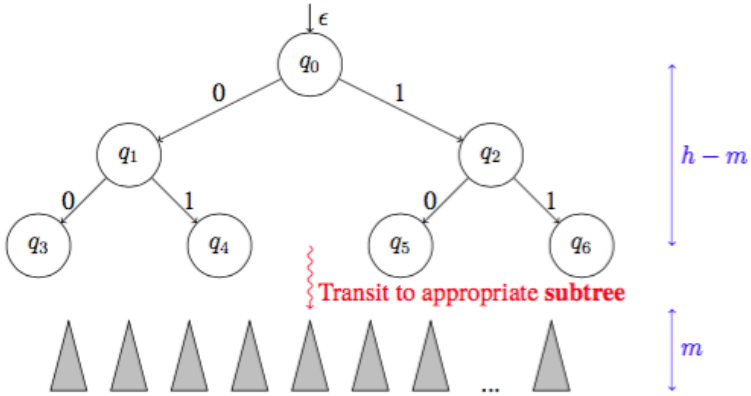
	Lower bound (existential)	Upper bound (for all strings)
DFA	$D(x) \geq \frac{\lvert x\rvert}{3 \cdot \log_2(\lvert x\rvert)}$	$D(x) \leq \frac{32 \cdot \lvert x\rvert}{\log_2(\lvert x\rvert)}$
NFA	$N(x) \geq \sqrt{\frac{\lvert x\rvert}{3}}$	$N(x) \leq 8 \cdot \sqrt{\lvert x\rvert}$
CFG	$C_k(x) \geq \sqrt[k+1]{\frac{\lvert x\rvert}{2}}$	$C_k(x) \leq 8 \cdot \sqrt[k+1]{\lvert x\rvert}$

Upper bound for DFA

Theorem: For any given string x such that $|x| \geq 3$, $D(x) \leq \frac{32|x|}{\log_2(|x|)}$



(a) The original T_x. We will *brute force* the gray states.

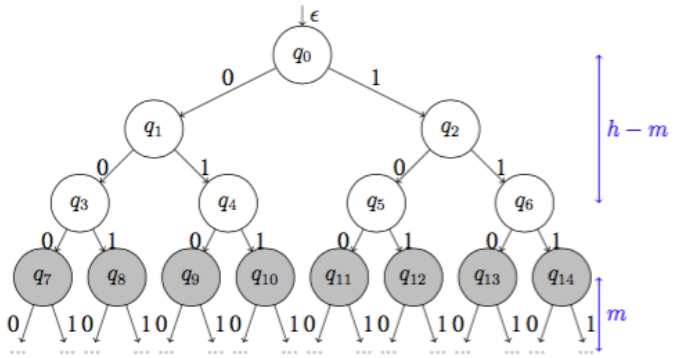


(b) Transformed automaton. We have 2^{2^m-1} possible subtrees (gray triangles).

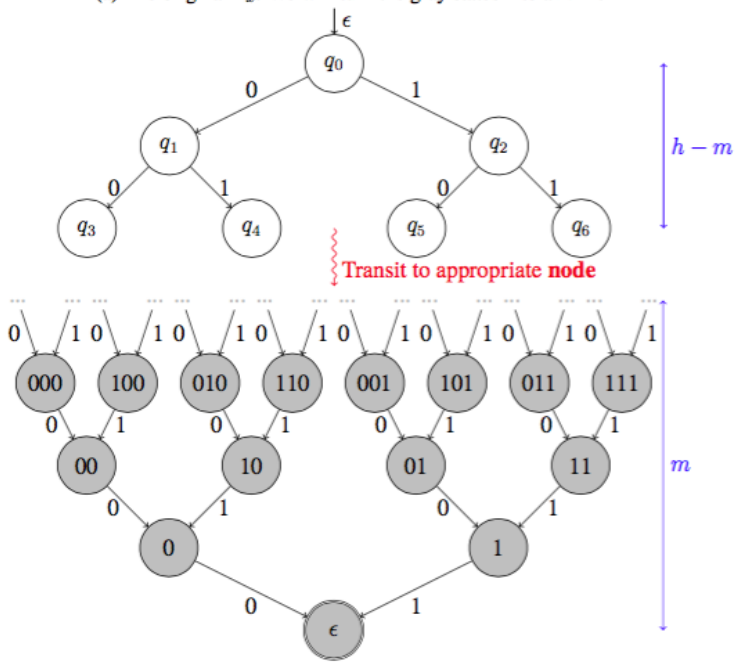
Figure 2.1: Illustration for proof of DFA Upper bound

Upper bound for NFA

Theorem: For any given string x such that $|x| \geq 3$, $N(x) \leq 8\sqrt{|x|}$



(a) The original T_x. We will turn the gray states into a NFA.



(b) Transformed automaton. We have $2^{m+1} - 1$ nodes in the inverted gray subtree.

Lower bound for DFA

Theorem: $\forall$ fixed length n, $\exists$ a string x of length n s.t. $D(x) \geq \frac{|x|}{3\log_2(|x|)}$
Proof sketch:

- Counting argument
- Total possible k-state DFA: $2^{2k\log_2(k)+k}$
- Fix k as smallest natural number such that $2^{2k\log_2(k)+k} \geq 2^{|x|}$
- Consider the string x such that $D(x) \geq k$

Definition

Method 1 (Bit flipping)

For $n \in \mathbb{N}$,

$$\begin{aligned} MT[0] &= 0 \\ MT[2^n : 2^{n+1} - 1] &= \overline{MT[0 : 2^n - 1]} \end{aligned}$$

Method 2 (Grammar generation)

For $n \in \mathbb{N}$,

$$\begin{aligned} MT[0] &= 0 \\ MT[2n] &= MT[n] \\ MT[2n+1] &= 1 - MT[n] = \overline{MT[n]} \end{aligned}$$

$$MT = 0|1|10|1001|10010110... = 0110100110010110...$$

4 states suffices!

Theorem: $D(MT) \leq 4$
Consider this automaton:

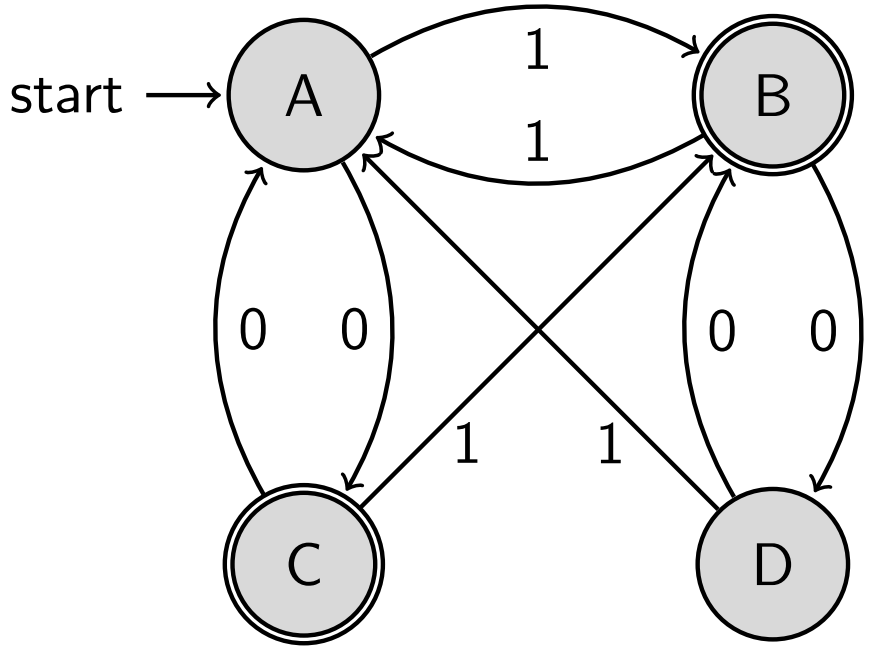


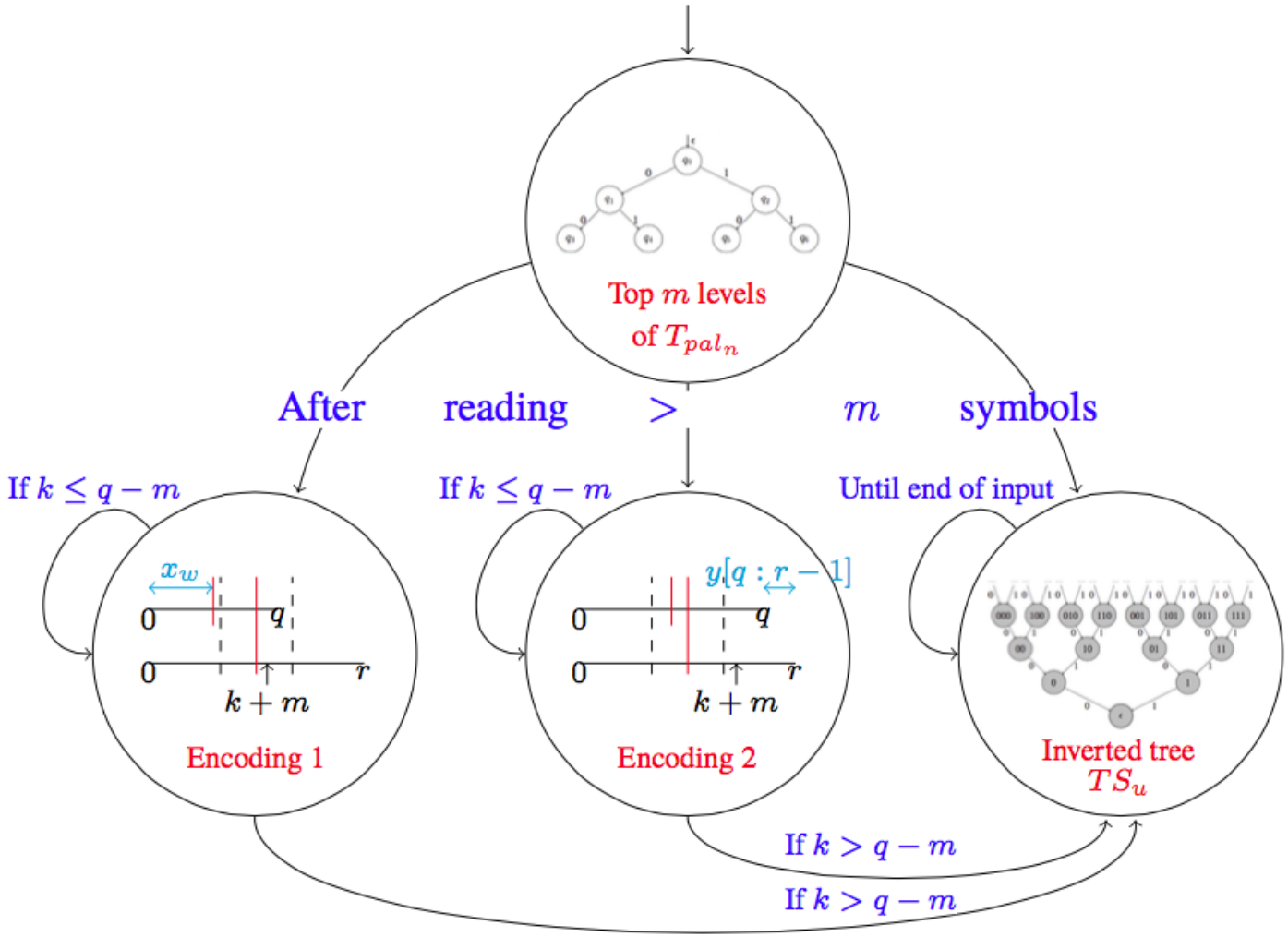
Figure : Automaton A_{MT} that produces the Morse-Thue sequence MT

Definition and basic facts

- x is a palindrome $\iff$ $x = x^R$
- Language of palindromes is a context free grammar
- Need at least a non-deterministic push down automata to recognise
- Consider the string pal_n such that
  $pal_n[i] = 1 \iff ll(i)$ is a palindrome

Upper Bound

Proposition: For a sufficiently large, fixed length n, $D(pal_n) \leq 5\sqrt{n}$



Lower Bound

Proposition: For a fixed length $n \geq 1$, $N(pal_n) \geq \frac{\sqrt{n}}{2}$

Proof sketch:

- After seeing the symbols $a_1 a_2 ... a_m$, for $1 \leq m \leq \lfloor \frac{\log_2(n)}{2} \rfloor$, the NFA must be in some state $S_{a_1 ... a_m}$ such that:
  - (a) The automaton can reach an accepting state by reading $a_m a_{m-1} ... a_1$
  - (b) The automaton cannot reach an accepting state by reading any other length m sequence of symbols
- By (a) and (b), $S_{a_1 ... a_m} = S_{b_1 ... b_m} \iff a_1 ... a_m = b_1 ... b_m$.
- For pal_n, there are $\sum_{m=0}^{\lfloor \frac{\log_2(n)}{2} \rfloor} 2^m \geq 2^{\frac{\log_2(n)}{2} - 1} = \frac{\sqrt{n}}{2}$ such states.

Bounds

- From before, we have:
  - $D(pal_n) \leq 5\sqrt{n}$
  - $\frac{\sqrt{n}}{2} \leq N(pal_n)$
- Since DFA is a NFA, $N(pal_n) \leq D(pal_n)$.
- Theorem:
  For a fixed length $n \geq 1$, $\frac{\sqrt{n}}{2} \leq N(pal_n) \leq D(pal_n) \leq 5\sqrt{n}$

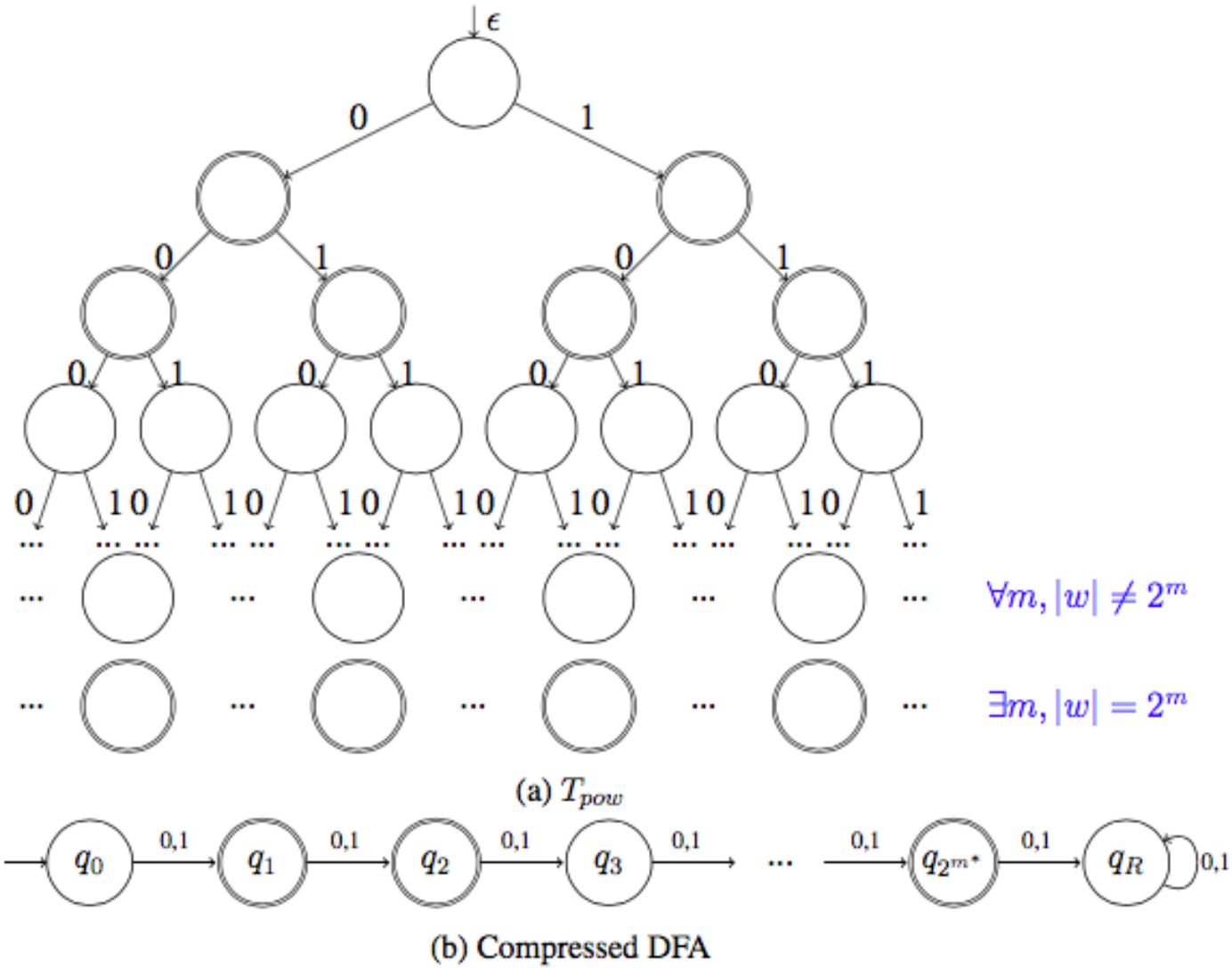
Definition

- $A_{2^m} = \{w \in \{0,1\}^* : |w| = 2^m, m \in \mathbb{N}\}$
- Represented by string pow: $pow[i] = 1 \iff |ll(i)|$ is a power of 2
- For a fixed prefix n, we define:
  $m^* = \max\{m : 2^{2^m} \leq n\}$

  $2^{m^*} = \lfloor \log_2(n) \rfloor =$ length of longest accepted $ll(i)$ indices in pow_n.

Upper bounds

Proposition: For a fixed length $n \geq 1$, $D(pow_n) \leq 2^{m^*} + 2$



(a) T_{pow}

(b) Compressed DFA

Upper bounds

Proposition: For a fixed length $n \geq 1$, $C_2(pow_n) \leq m^* + 1$
Proof sketch: Divide-and-conquer
Consider the following CFG defined by (NT, T, R, S):

$$\begin{array}{rrcl} \text{where} & NT & = & \{S\} \cup \{A_m : m \in \{0, 1, 2, ..., m^* - 1\}\} \\ & T & = & \{0, 1\} \\ & R & = & \{S \to A_0\} \cup \{S \to A_m A_m : m \in \{0, 1, 2, ..., m^* - 1\}\} \cup \\ & & & \{A_m \to A_{m-1} A_{m-1} : m \in \{1, 2, ..., m^* - 1\}\} \cup \\ & & & \{A_0 \to 0, A_0 \to 1\} \end{array}$$

Lower bounds

Proposition: For a fixed length $n \geq 1$, $N(pow_n) \geq \max\{1, 2^{m^*-1}\}$

Proof sketch:

- When $m^* = 0$, $N(pow_n) \geq 1$.
- When $m^* \geq 1$, consider an arbitrary automata that recognises pow_n.
  - $A = \{\text{Nodes } a : a \text{ accepts indices of length } 2^{m^*-1}\}$
  - $B = \{\text{Nodes } b : b \text{ accepts indices of length } 2^{m^*}\}$
  - At least 2^{m^*-1} distinct non-accepting states between $a \in A$ and $b \in B$.

Lower bounds

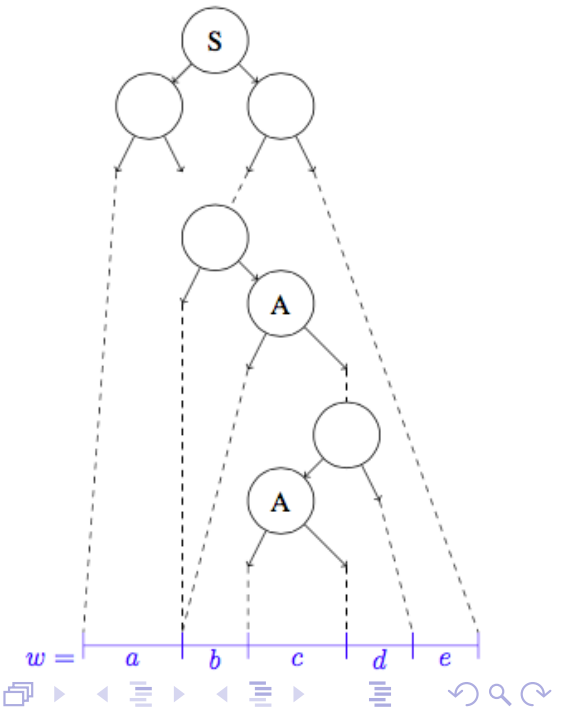
Proposition: For a fixed length $n \geq 1$, $C_2(pow_n) \geq m^*$

Proof sketch:

- Consider an arbitrary index $w \in pow_n$ such that $|w| = 2^{m^*}$.
- In the derivation of $S \Rightarrow^* w$, there are no repeated non-terminals.
- In the derivation of $S \Rightarrow^* w$, there are repeated non-terminals.
  - Pick smallest $k \in \mathbb{N}$ s.t. $0 \geq \max\{|b|, |d|\} \leq 2^k$
  - $S \Rightarrow^* abcde \in pow_n$
  - $S \Rightarrow^* abbcdde \notin pow_n$, so $|abbcdde| > 2^{m^*}$

$$\begin{aligned} 2^{m^*+1} &\leq |abbcdde| \\ &= |abcde| + |b| + |d| \\ &\leq 2^{m^*} + 2^k + 2^k \end{aligned}$$

  - So, $m^* - 1 \leq k$



Bounds

Theorem: For a fixed length $n \geq 1$,

- $\mathcal{O}(\log_2(n)) \leq D(pow_n) \leq \mathcal{O}(\log_2(n))$
- $\mathcal{O}(\log_2(n)) \leq N(pow_n) \leq \mathcal{O}(\log_2(n))$
- $\mathcal{O}(\log_2(\log_2(n))) \leq C_2(pow_n) \leq \mathcal{O}(\log_2(\log_2(n)))$

Proof sketch:

- From before, for a fixed length $n \geq 1$, we have:
  - $D(pow_n) \leq 2^{m^*} + 2$
  - $\max\{1, 2^{m^*-1}\} \leq N(pow_n)$
  - $m^* \leq C_2(pow_n) \leq m^*$
- $N(pow_n) \leq D(pow_n)$
- $m^* = \max\{m : 2^{2^m} \leq n\}$.

Definition

- Prefix-free machine V:
  - V is a partial mapping: $\{0,1\}^* \to \{0,1\}^*$
  - $Dom(V)$ is prefix-free
  - V is partial-recursive
- x is Martin-Löf Random (MLR) if and only if
  $\neg\exists V\ \exists^\infty n\ \exists p \in \{0,1\}^*\ [V \text{ is prefix-free} \wedge |p| < n \wedge V(p)\downarrow = x_n]$
- Roughly speaking:
  If x is *MLR*, you cannot *successfully* compress infinitely prefixes of x.

Encoding

Proposition:
For any string x_n, $n \geq 2^3 = 8$, we can encode it in a **prefix-free** way in $3\log_2(n) + \log_2(f(k))$ binary bits, where
k =complexity(x_n) in the chosen representation (DFA/NFA/CFG), and
$f(k)$ is the number of possible machines with parameter k.

x_n	$\|n\| - 2^3$	n	k	Index of $f(k)$ machines
#bits	$\log_2(n) - 3$	$\log_2(n) + 1$	$\log_2(n) + 1$	$\log_2(f(k)) + 1$

Bounds

Theorem:
Martin-Löf Random strings x meet the lower bound complexities of DFA/NFA/CFG, up to a constant factor, for **almost all** prefixes x_n of x.
Proof sketch:

- DFA: With k states, $f(k) = 2^{2k \log_2(k)+k}$
- NFA: With k states, $f(k) = 2^{k+2k^2}$
- CFG: With c non-terminals, $f(c) = 2^{2c^{k+1}}$
- Apply MLR definition:
  If x is MLR, then $\forall^\infty n[3 \log_2(n) + \log_2(f(k)) \geq n]$

The Program

Algorithm 1 MATCH(i, j, lst)

1: **if** $j \geq |lst|$ **then**
2: **return** True
3: **else**
4: $current\_match \leftarrow (lst[i] == lst[j])$
5: $left\_match \leftarrow \text{MATCH}(2 * i + 1, 2 * j + 1, lst)$
6: $right\_match \leftarrow \text{MATCH}(2 * i + 2, 2 * j + 2, lst)$
7: **return** ($current\_match$ and $left\_match$ and $right\_match$)
8: **end if**

The Program

Algorithm 2 FindPattern(seq)

```
1:  eq ← ∅
2:  seen ← ∅
3:  for i ∈ {0, 1, 2, ..., |seq| − 1} do
4:      if i ∉ seen then
5:          matches ← ∅
6:          for j ∈ {i + 1, i + 2, ..., |seq| − 1} do
7:              if Match(i, j, seq) then
8:                  matches ← matches ∪ {j}
9:                  seen ← seen ∪ {j}
10:             end if
11:         end for
12:         eq ← output ∪ {{i, matches}}
13:     end if
14: end for
15: return eq
```

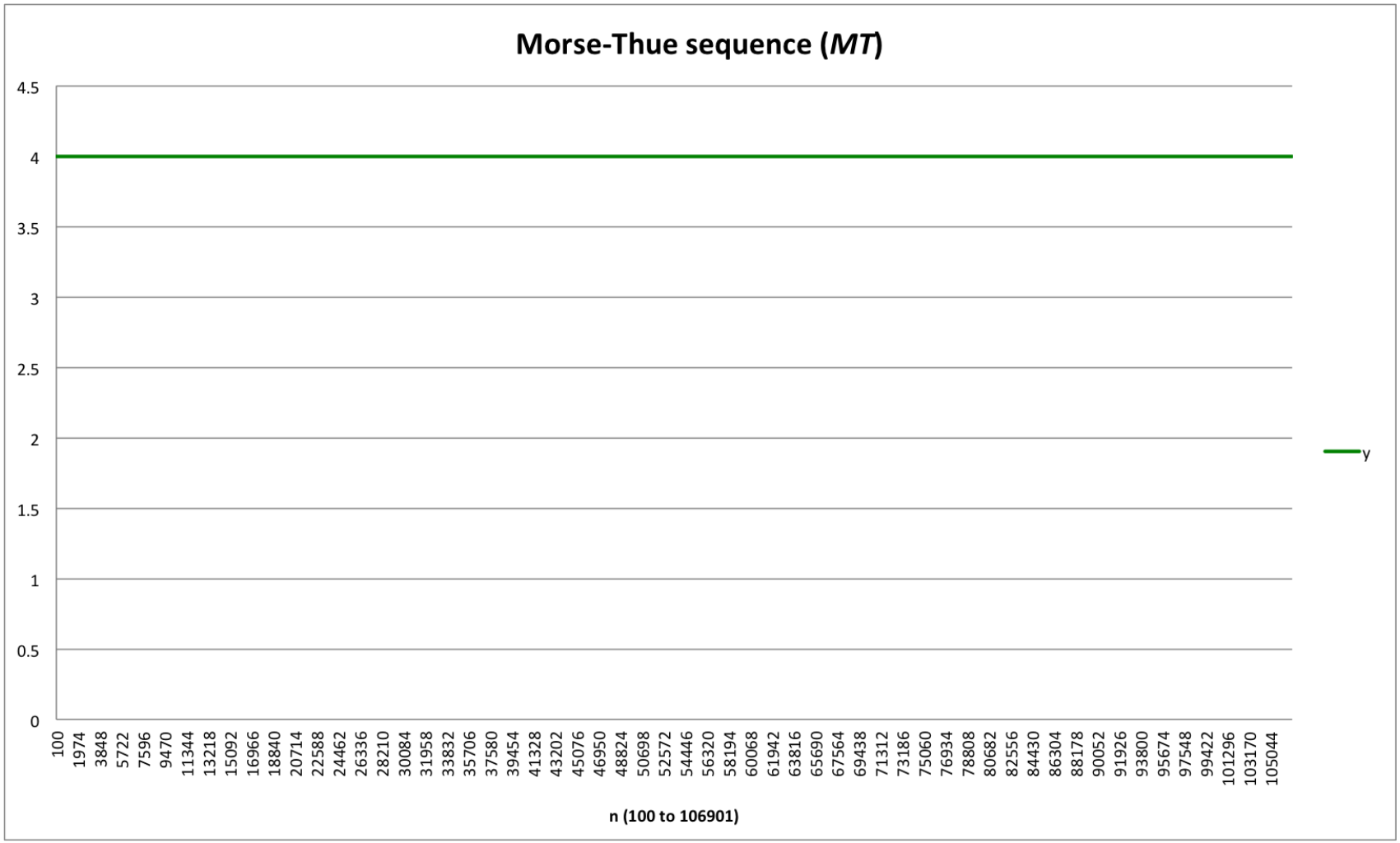
Parameters

The table below shows the parameters that are of interest to us.

Parameter	Description	Name in graph
$\lvert x \rvert$	Prefix length of interest	n
$\lvert eq \rvert$	Number of unique nodes needed	y
$\frac{y}{\log_2(\lvert x \rvert)}$	Ratio between y and the $\log_2(\lvert x \rvert)$ bound	*logRatio*
$\frac{y}{\frac{\lvert x \rvert}{\log_2(\lvert x \rvert)}}$	Ratio between y and the $\frac{\lvert x \rvert}{\log_2(\lvert x \rvert)}$ bound	*DFARatio*
$\frac{y}{\sqrt{\lvert x \rvert}}$	Ratio between y and the $\sqrt{\lvert x \rvert}$ bound	*NFARatio*

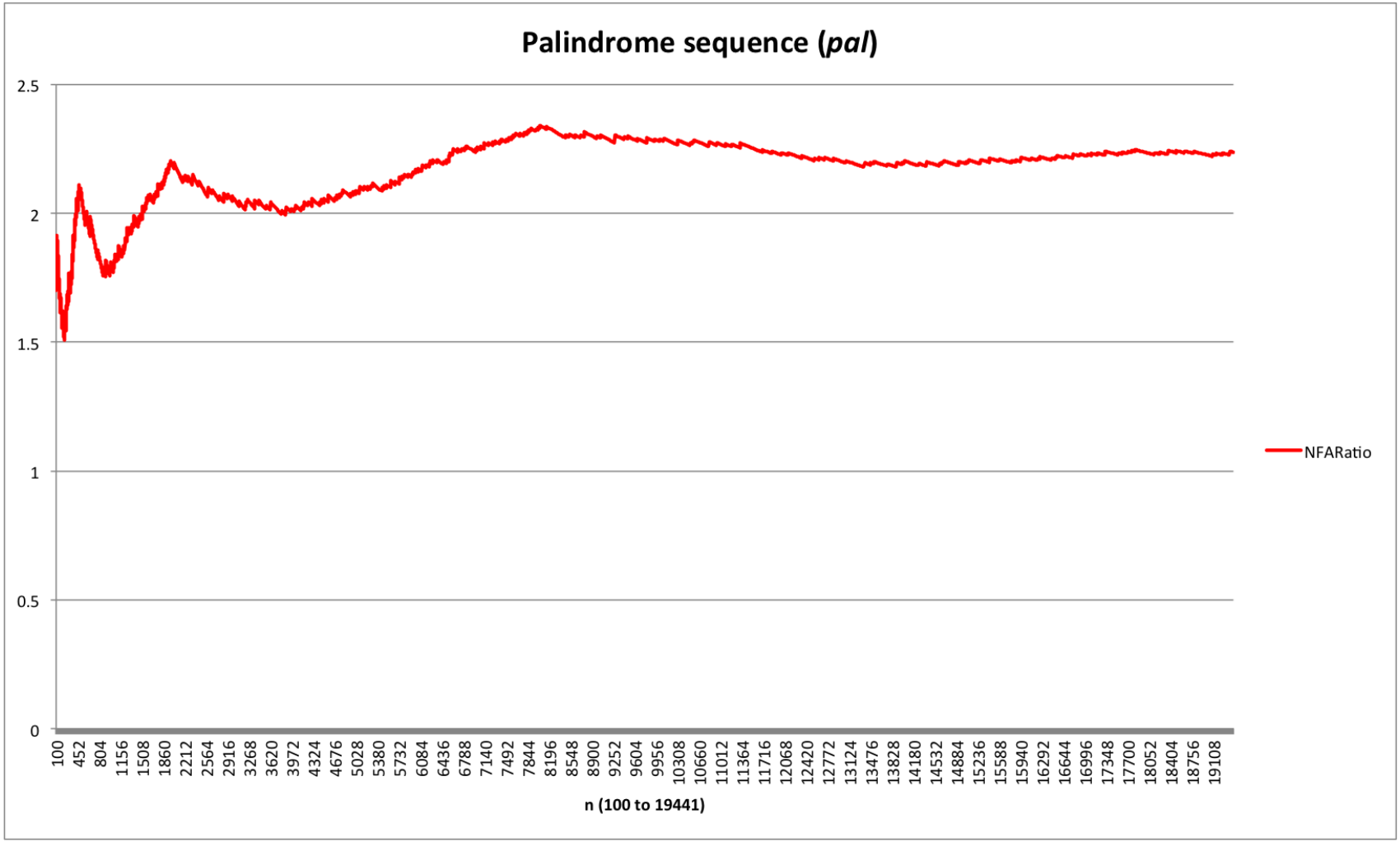
Empirical results

Morse-Thue sequence (MT): $MT = 0|1|10|1001|10010110...$



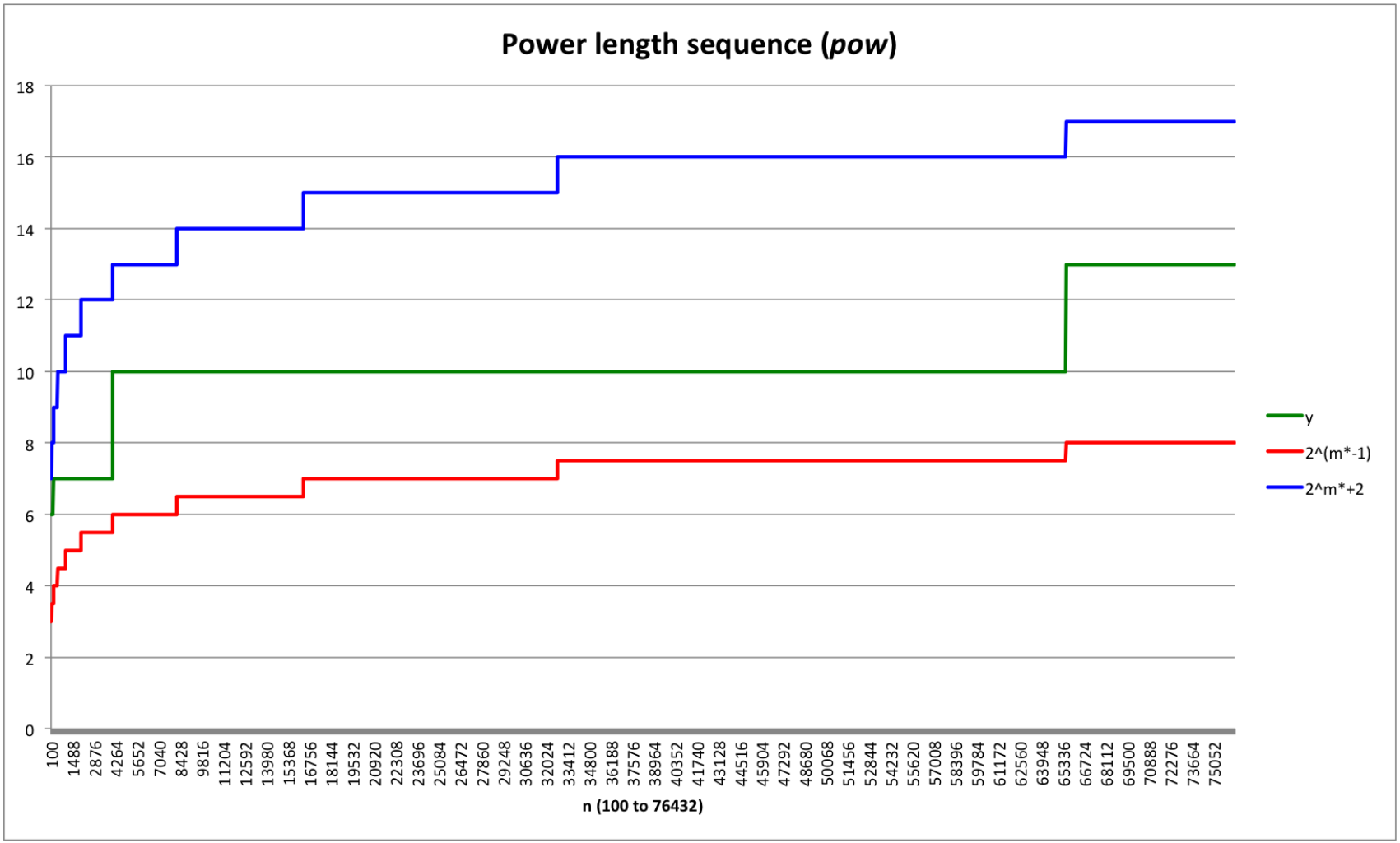
Empirical results

Palindrome sequence (pal): $pal_n[i] = 1 \iff ll(i)$



Empirical results

Power length sequence (pow): $pow[i] = 1 \iff |ll(i)|$ is a power of 2

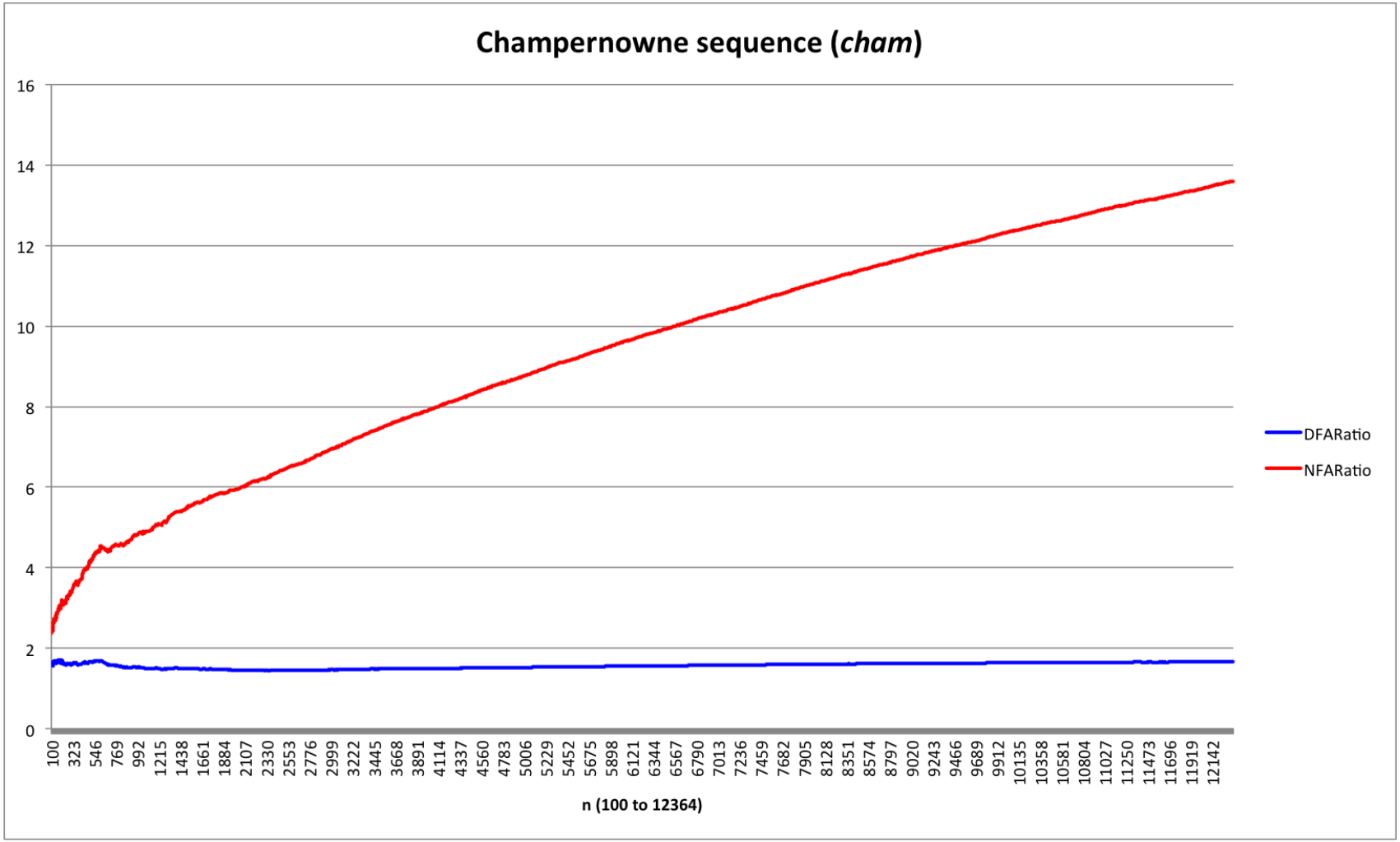


Champernowne sequence

- What is it?
  - Base-10 representation: 0.12345678910111213...
  - $\mathcal{L}$ representation: $cham = 0100011011000001010011...$
  - Has properties such as being normal, etc.
- Know: *pal* attains lower bound for NFA
- Hope: *cham* attains lower bound of DFA

Empirical results

Champernowne sequence (*cham*): $cham = 0100011011000001010011...$



Numerical bounds

- For $100 \leq n \leq 106901$,
  $4 \leq D(MT_n) \leq 4$
- For $100 \leq n \leq 19441$,
  $1.50 \cdot \sqrt{n} \leq D(pal_n) \leq 2.35 \cdot \sqrt{n}$
  For $4000 \leq n \leq 19441$,
  $2.00 \cdot \sqrt{n} \leq D(pal_n) \leq 2.35 \cdot \sqrt{n}$
- For $100 \leq n \leq 76432$,
  $0.5 \cdot \lfloor \log_2(n) \rfloor \leq D(pow_n) \leq \lfloor \log_2(n) \rfloor + 2$
- For $100 \leq n \leq 12364$,
  $1.436 \cdot \frac{|x|}{\log_2(|x|)} \leq D(cham_n) \leq 1.703 \cdot \frac{|x|}{\log_2(|x|)}$

Summary of thesis contribution

- Defined Kolmogorov complexity styled $\underline{\text{def}}^n$ for automata and CFG
- Studied the upper and lower bound complexities for general strings
- Looked at 4 specific examples
  - Morse-Thue sequence ($D(MT) \leq 4$.)
  - Palindromes
    (Matching, tight DFA and NFA bounds. Attain lower bound for NFA.)
  - Power length sequence
    (Matching, tight DFA and NFA bounds. CFG needs exponentially less.)
  - Martin-Löf Random strings (Attain lower bound complexities.)
- Made program to compute explicit DFA bounds for finite prefixes
  - Champernowne sequence (Empirically attain lower bound for DFA.)

Thank you for your time

- For references and more details, refer to report
- Questions?