

Restricted Turing Machines and Language Recognition

Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano, Italy

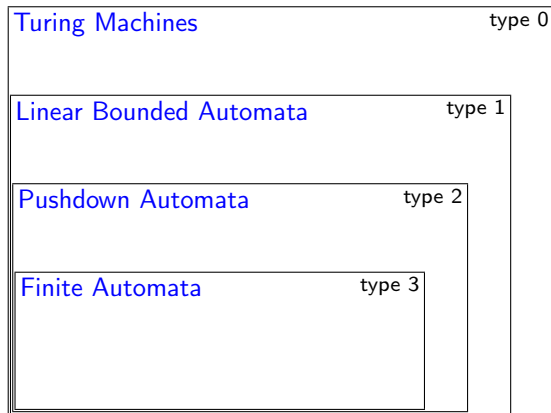
LATA 2016 – Prague
March 14-18, 2016



UNIVERSITÀ DEGLI STUDI
DI MILANO

Introduction

The Chomsky Hierarchy



One-tape Turing machines with restricted rewritings

Definition

Fixed an integer $d \geq 1$, a *d-limited automaton* is

- ▶ a one-tape Turing machine
- ▶ which is allowed to rewrite the content of each tape cell *only in the first d visits*

Limited Automata [Hibbard '67]

One-tape Turing machines with restricted rewritings

Definition

Fixed an integer $d \geq 1$, a *d-limited automaton* is

- ▶ a one-tape Turing machine
- ▶ which is allowed to rewrite the content of each tape cell *only in the first d visits*

Limited Automata [Hibbard '67]

One-tape Turing machines with restricted rewritings

Definition

Fixed an integer $d \geq 1$, a *d-limited automaton* is

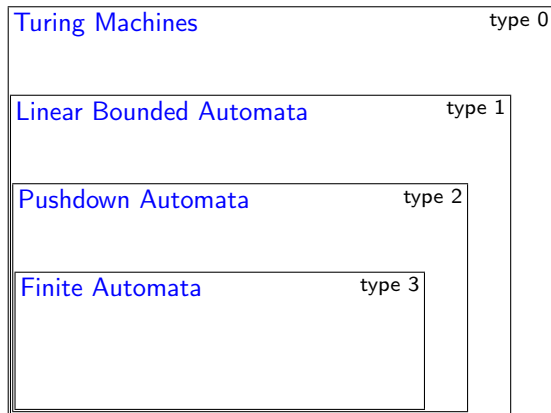
- ▶ a one-tape Turing machine
- ▶ which is allowed to rewrite the content of each tape cell *only in the first d visits*

Computational power

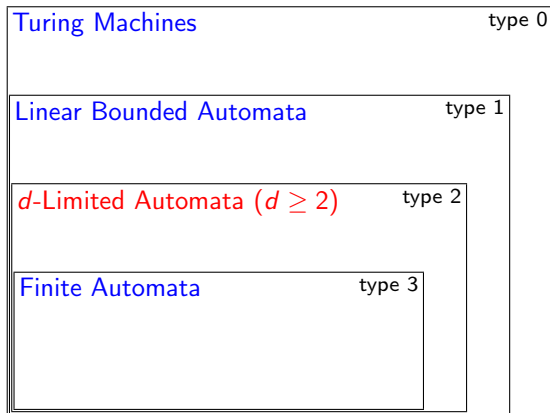
- ▶ For each $d \geq 2$, *d-limited automata* characterize context-free languages

[Hibbard '67]

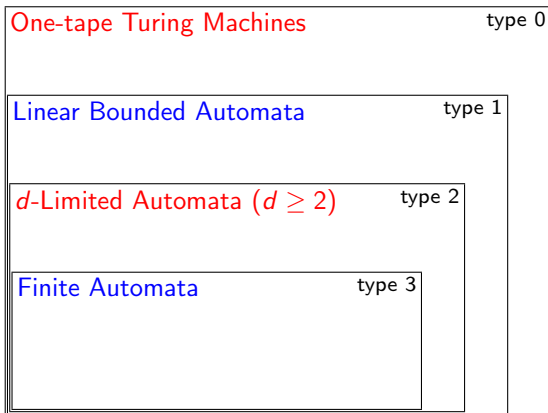
The Chomsky Hierarchy



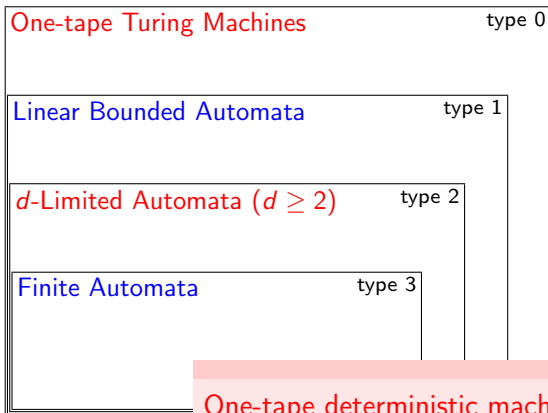
The Chomsky Hierarchy



The Chomsky Hierarchy

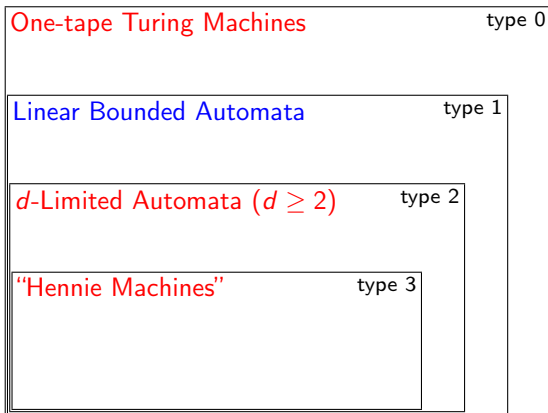


The Chomsky Hierarchy



One-tape deterministic machines working in *linear time* can recognize only regular languages [Hennie '65]

The Chomsky Hierarchy



Part I: Fast One-Tape Turing Machines

Hennie Machines & C

Part II: One-Tape Turing Machines
with Rewriting Restrictions

Limited Automata & C

Outline

- ▶ **One-Tape Turing machines**
- ▶ Time complexity: different measures
- ▶ Crossing sequences
- ▶ Lower bounds for nonregular languages recognition
- ▶ Optimality
- ▶ Fast recognition of unary nonregular languages
- ▶ Final remarks: other complexity measures

Part I: Fast One-Tape Turing Machines

Outline

- ▶ One-Tape Turing machines
- ▶ Time complexity: different measures
- ▶ Crossing sequences
- ▶ Lower bounds for nonregular languages recognition
- ▶ Optimality
- ▶ Fast recognition of unary nonregular languages
- ▶ Final remarks: other complexity measures

Part I: Fast One-Tape Turing Machines

Outline

- ▶ One-Tape Turing machines
- ▶ Time complexity: different measures
- ▶ **Crossing sequences**
- ▶ Lower bounds for nonregular languages recognition
- ▶ Optimality
- ▶ Fast recognition of unary nonregular languages
- ▶ Final remarks: other complexity measures

Part I: Fast One-Tape Turing Machines

Outline

- ▶ One-Tape Turing machines
- ▶ Time complexity: different measures
- ▶ Crossing sequences
- ▶ Lower bounds for nonregular languages recognition
- ▶ Optimality
- ▶ Fast recognition of unary nonregular languages
- ▶ Final remarks: other complexity measures

Part I: Fast One-Tape Turing Machines

Outline

- ▶ One-Tape Turing machines
- ▶ Time complexity: different measures
- ▶ Crossing sequences
- ▶ Lower bounds for nonregular languages recognition
- ▶ **Optimality**
- ▶ Fast recognition of unary nonregular languages
- ▶ Final remarks: other complexity measures

Part I: Fast One-Tape Turing Machines

Outline

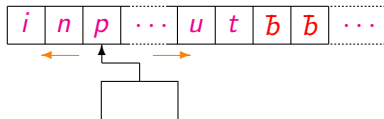
- ▶ One-Tape Turing machines
- ▶ Time complexity: different measures
- ▶ Crossing sequences
- ▶ Lower bounds for nonregular languages recognition
- ▶ Optimality
- ▶ Fast recognition of unary nonregular languages
- ▶ Final remarks: other complexity measures

Part I: Fast One-Tape Turing Machines

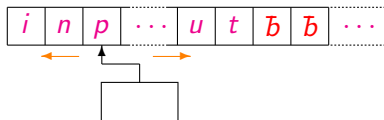
Outline

- ▶ One-Tape Turing machines
- ▶ Time complexity: different measures
- ▶ Crossing sequences
- ▶ Lower bounds for nonregular languages recognition
- ▶ Optimality
- ▶ Fast recognition of unary nonregular languages
- ▶ Final remarks: other complexity measures

One-Tape Turing Machines



One-Tape Turing Machines



- ▶ *Finite state control*

- ▶ *Semi-infinite tape*

at the beginning:

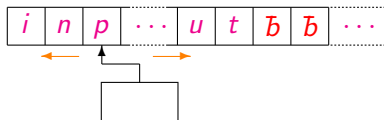
- input string (leftmost part)
- blank symbol (remaining squares)

- ▶ *Computation step*

- change of state
- nonblank symbol written in the scanned tape cell
- head moved either to the left, or to the right, or kept on the same cell

- ▶ *Accepting and rejecting strings and computation steps*

One-Tape Turing Machines



- ▶ *Finite state control*

- ▶ *Semi-infinite tape*

at the beginning:

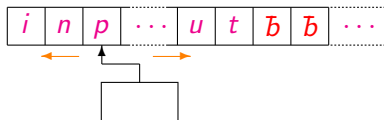
- input string (leftmost part)
- blank symbol (remaining squares)

- ▶ *Computation step*

- change of state
- nonblank symbol written in the scanned tape cell
- head moved either to the left, or to the right, or kept on the same cell

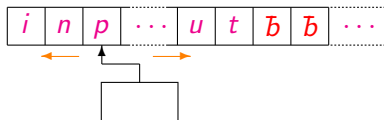
- ▶ *Accepting and rejecting states: the computation stops*

One-Tape Turing Machines



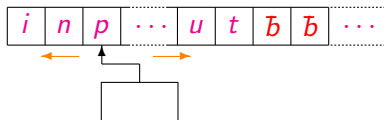
- ▶ *Finite state control*
- ▶ *Semi-infinite tape*
at the beginning:
 - input string (leftmost part)
 - blank symbol (remaining squares)
- ▶ *Computation step*
 - change of state
 - nonblank symbol written in the scanned tape cell
 - head moved either to the left, or to the right, or kept on the same cell
- ▶ *Accepting and rejecting states*: the computation stops

One-Tape Turing Machines



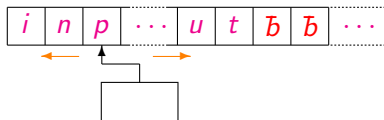
- ▶ *Finite state control*
- ▶ *Semi-infinite tape*
at the beginning:
 - input string (leftmost part)
 - blank symbol (remaining squares)
- ▶ *Computation step*
 - change of state
 - nonblank symbol written in the scanned tape cell
 - head moved either to the left, or to the right, or kept on the same cell
- ▶ *Accepting and rejecting states*: the computation stops

One-Tape Turing Machines



- ▶ Deterministic version (**dTM**)
- ▶ Nondeterministic version (**nTM**)

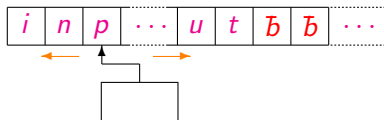
One-Tape Turing Machines



Time complexity:

- ▶ $t(\mathcal{C})$ number of moves in the computation \mathcal{C}

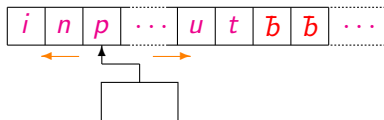
One-Tape Turing Machines



Time complexity:

- ▶ $t(C)$ number of moves in the computation C
- ▶ $t(x)$ for an input x
- ▶ $t(n)$ for inputs of length n

One-Tape Turing Machines



Time complexity:

- ▶ $t(\mathcal{C})$ number of moves in the computation \mathcal{C}
- ▶ $t(x)$ for an input x
- ▶ $t(n)$ for inputs of length n

Nondeterministic case: several computations on a same input

How to define $t(x)$ and $t(n)$?

Complexity Measures

strong measure: costs of *all computations* on x

$$t(x) = \max\{t(\mathcal{C}) \mid \mathcal{C} \text{ is a computation on } x\}$$

worst case!

weak measure: *minimum* cost of accepting x

$$t(x) = \begin{cases} \min\{t(\mathcal{C}) \mid \mathcal{C} \text{ is accepting on } x\} & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$$

best case for acceptance!

accept measure: costs of *all accepting* computations on x

$$t(x) = \begin{cases} \max\{t(\mathcal{C}) \mid \mathcal{C} \text{ is accepting on } x\} & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$$

worst case for acceptance!

$$t(x) = \max\{t(y) \mid x \in \Sigma^*, |x| = |y|\}$$

Complexity Measures

strong measure: costs of *all computations* on x

$$t(x) = \max\{t(\mathcal{C}) \mid \mathcal{C} \text{ is a computation on } x\}$$

worst case!

weak measure: *minimum* cost of *accepting* x

$$t(x) = \begin{cases} \min\{t(\mathcal{C}) \mid \mathcal{C} \text{ is accepting on } x\} & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$$

best case for acceptance!

accept measure: costs of *all accepting* computations on x

$$t(x) = \begin{cases} \max\{t(\mathcal{C}) \mid \mathcal{C} \text{ is accepting on } x\} & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$$

worst case for acceptance!

$$t(n) = \max\{t(x) \mid x \in \Sigma^*, |x| = n\}$$

Complexity Measures

strong measure: costs of *all computations* on x

$$t(x) = \max\{t(\mathcal{C}) \mid \mathcal{C} \text{ is a computation on } x\}$$

worst case!

weak measure: *minimum* cost of *accepting* x

$$t(x) = \begin{cases} \min\{t(\mathcal{C}) \mid \mathcal{C} \text{ is accepting on } x\} & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$$

best case for acceptance!

accept measure: costs of *all accepting* computations on x

$$t(x) = \begin{cases} \max\{t(\mathcal{C}) \mid \mathcal{C} \text{ is accepting on } x\} & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$$

worst case for acceptance!

$$t(n) = \max\{t(x) \mid x \in \Sigma^*, |x| = n\}$$

Complexity Measures

strong measure: costs of *all computations* on x

$$t(x) = \max\{t(\mathcal{C}) \mid \mathcal{C} \text{ is a computation on } x\}$$

worst case!

weak measure: *minimum* cost of *accepting* x

$$t(x) = \begin{cases} \min\{t(\mathcal{C}) \mid \mathcal{C} \text{ is accepting on } x\} & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$$

best case for acceptance!

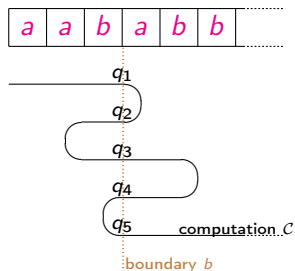
accept measure: costs of *all accepting* computations on x

$$t(x) = \begin{cases} \max\{t(\mathcal{C}) \mid \mathcal{C} \text{ is accepting on } x\} & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$$

worst case for acceptance!

$$t(n) = \max\{t(x) \mid x \in \Sigma^*, |x| = n\}$$

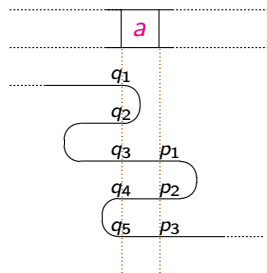
Crossing Sequences



Crossing sequence of a computation C
at a boundary b between two tape squares:

- ▶ (q_1, \dots, q_k)
- ▶ q_i is the state when b is crossed for the i th time

Crossing Sequences: Compatibility



- ▶ $(q_1, \dots, q_k), (p_1, \dots, p_h)$: finite crossing sequence
- ▶ it is possible to verify whether or not they are *compatible* with respect to an input symbol a ,
i.e., (q_1, \dots, q_k) and (p_1, \dots, p_h) could be at the left boundary and at the right boundary of a tape square which initially contains the symbol a

Lower Bounds

One-Tape Machines

Problem:

Find tight lower bounds for

- ▶ the minimum amount of time $t(n)$
- ▶ the length of crossing sequences $c(n)$

for nonregular language recognition

One-Tape Machines: Simple Bounds

Length of the crossing sequences

Theorem

If L is accepted by a n TM such that $c(n) = O(1)$, under the weak measure, then L is regular

Proof idea:

- ▶ Let K be such that $c(n) \leq K$
- ▶ Define a NFA A accepting L s.t.
 - the states are the crossing sequences of length $\leq K$
 - the transition function is defined according to the “compatibility” between crossing sequences

One-Tape Machines: Simple Bounds

Length of the crossing sequences

Theorem

If L is accepted by a n TM such that $c(n) = O(1)$, under the weak measure, then L is regular

Proof idea:

- ▶ Let K be such that $c(n) \leq K$
- ▶ Define a NFA A accepting L s.t.
 - the states are the crossing sequences of length $\leq K$
 - the transition function is defined according to the “compatibility” between crossing sequences

One-Tape Machines: Simple Bounds

Time

Theorem

If L is accepted by a n TM such that $t(n) = o(n)$, under the weak measure, then $t(n) = O(1)$ and L is regular

Proof idea:

- ▶ Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$
- ▶ Given $x \in L$ with $|x| \geq n_0$, there is a computation \mathcal{C} that accepts x just reading a proper prefix x' of length $\leq t(x)$
- ▶ \mathcal{C} should also accept x'
- ▶ Since all x' is read in \mathcal{C} , $t(x') \geq |x'|$ implying $|x'| < n_0$
- ▶ Hence, the membership to L can be decided just testing an input prefix of length at most n_0

Remark: The same argument works for multitape machines

One-Tape Machines: Simple Bounds

Time

Theorem

If L is accepted by a n TM such that $t(n) = o(n)$, under the weak measure, then $t(n) = O(1)$ and L is regular

Proof idea:

- ▶ Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$
- ▶ Given $x \in L$ with $|x| \geq n_0$, there is a computation \mathcal{C} that accepts x just reading a proper prefix x' of length $\leq t(x)$
- ▶ \mathcal{C} should also accept x'
- ▶ Since all x' is read in \mathcal{C} , $t(x') \geq |x'|$ implying $|x'| < n_0$
- ▶ Hence, the membership to L can be decided just testing an input prefix of length at most n_0

Remark: The same argument works for multitape machines

One-Tape Machines: Simple Bounds

Time

Theorem

If L is accepted by a n TM such that $t(n) = o(n)$, under the weak measure, then $t(n) = O(1)$ and L is regular

Proof idea:

- ▶ Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$
- ▶ Given $x \in L$ with $|x| \geq n_0$, there is a computation \mathcal{C} that accepts x just reading a proper prefix x' of length $\leq t(x)$
- ▶ \mathcal{C} should also accept x'
- ▶ Since all x' is read in \mathcal{C} , $t(x') \geq |x'|$ implying $|x'| < n_0$
- ▶ Hence, the membership to L can be decided just testing an input prefix of length at most n_0

Remark: The same argument works for multitape machines

Does it is possible to improve the lower bounds on $c(n)$ and $t(n)$ for nonregular language recognition given in the previous results?

Different bounds have found depending

- ▶ on the measure (strong, accept, weak)
- ▶ on the kind of machines (deterministic, nondeterministic)

Does it is possible to improve the lower bounds
on $c(n)$ and $t(n)$ for nonregular language recognition
given in the previous results?

Different bounds have found depending

- ▶ on the measure (strong, accept, weak)
- ▶ on the kind of machines (deterministic, nondeterministic)

Deterministic Machines: Lower Bounds (strong measure)

- ▶ Hennie (1965) proved that

one-tape *deterministic* machines working in linear time
accept only regular languages

Furthermore, in order to accept nonregular languages

$c(n)$ must grow at least as $\log n$

- ▶ Trakhtenbrot (1964) and Hartmanis (1968), independently, got a better time lower bound:

in order to recognize a nonregular language a dTM needs
time $t(n)$ growing at least as $n \log n$

▶

here are nonregular languages accepted in time $O(n \log n)$

Deterministic Machines: Lower Bounds (strong measure)

- ▶ Hennie (1965) proved that

one-tape *deterministic* machines working in linear time
accept only regular languages

Furthermore, in order to accept nonregular languages

$c(n)$ must grow at least as $\log n$

- ▶ Trakhtenbrot (1964) and Hartmanis (1968), independently, got a better time lower bound:

in order to recognize a nonregular language a dTM needs
time $t(n)$ growing at least as $n \log n$

- ▶ Optimal!

There are nonregular languages accepted in time $O(n \log n)$

Deterministic Machines: Lower Bounds (strong measure)

- ▶ Hennie (1965) proved that

one-tape *deterministic* machines working in linear time
accept only regular languages

Furthermore, in order to accept nonregular languages

$c(n)$ must grow at least as $\log n$

- ▶ Trakhtenbrot (1964) and Hartmanis (1968), independently, got a better time lower bound:

in order to recognize a nonregular language a dTM needs
time $t(n)$ growing at least as $n \log n$

- ▶ **Optimal!**

There are nonregular languages accepted in time $O(n \log n)$

Deterministic Machines: Lower Bounds (strong measure)

- ▶ Hennie (1965) proved that

one-tape *deterministic* machines working in linear time
accept only regular languages

Furthermore, in order to accept nonregular languages

$c(n)$ must grow at least as $\log n$

- ▶ Trakhtenbrot (1964) and Hartmanis (1968), independently, got a better time lower bound:

in order to recognize a nonregular language a dTM needs
time $t(n)$ growing at least as $n \log n$

- ▶ **Optimal!**

There are nonregular languages accepted in time $O(n \log n)$

Nondeterministic Machines

weak measure:

- ▶ There is nonregular language accepted by a nTM
in $o(n \log n)$ time [Wagner&Wechsung '86]
- ▶ There is a NP-complete language accepted by a nTM
in $O(n)$ time [Michel '91]

strong measure:

- ▶ The time lower bound $n \log n$ proved for dTMs
also holds for nTMs [Tadaki&Yamakami&Lin '10]

accept measure:

- ▶ The $n \log n$ lower bound also holds [P.'09]

Nondeterministic Machines

weak measure:

- ▶ There is nonregular language accepted by a nTM
in $O(n \log n)$ time [Wagner&Wechsung '86]
- ▶ There is a NP-complete language accepted by a nTM
in $O(n)$ time [Michel '91]

strong measure:

- ▶ The time lower bound $n \log n$ proved for dTMs
also holds for nTMs [Tadaki&Yamakami&Lin '10]

accept measure:

- ▶ The $n \log n$ lower bound also holds [P.'09]

Nondeterministic Machines

weak measure:

- ▶ There is nonregular language accepted by a nTM
in $o(n \log n)$ time [Wagner&Wechsung '86]
- ▶ There is a NP-complete language accepted by a nTM
in $O(n)$ time [Michel '91]

strong measure:

- ▶ The time lower bound $n \log n$ proved for dTMs
also holds for nTMs [Tadaki&Yamakami&Lin '10]

accept measure:

- ▶ The $n \log n$ lower bound also holds [P.'09]

Nondeterministic Machines

weak measure:

- ▶ There is nonregular language accepted by a nTM
in $o(n \log n)$ time [Wagner&Wechsung '86]
- ▶ There is a NP-complete language accepted by a nTM
in $O(n)$ time [Michel '91]

strong measure:

- ▶ The time lower bound $n \log n$ proved for dTMs
also holds for nTMs [Tadaki&Yamakami&Lin '10]

accept measure:

- ▶ The $n \log n$ lower bound also holds [P.'09]

Nondeterministic Machines

weak measure:

- ▶ There is nonregular language accepted by a nTM in $o(n \log n)$ time [Wagner&Wechsung '86]
- ▶ There is a NP-complete language accepted by a nTM in $O(n)$ time [Michel '91]

strong measure:

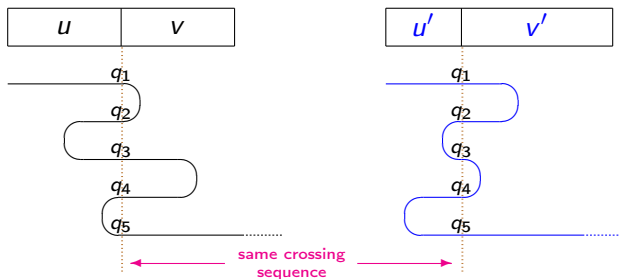
- ▶ The time lower bound $n \log n$ proved for dTMs also holds for nTMs [Tadaki&Yamakami&Lin '10]

accept measure:

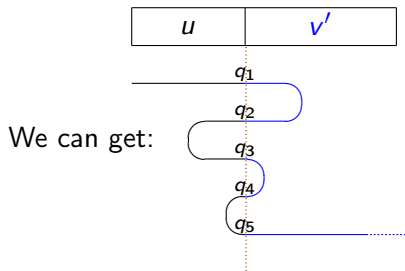
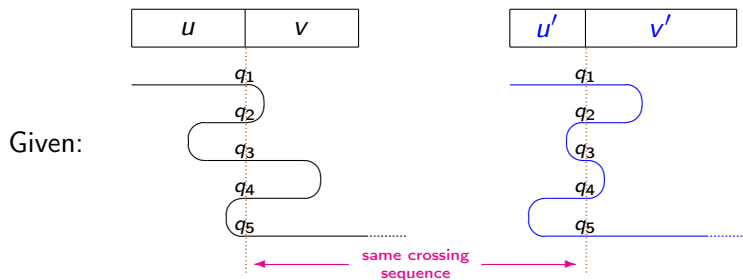
- ▶ The $n \log n$ lower bound also holds [P.'09]

Crossing Sequences: "Cut-and-Paste"

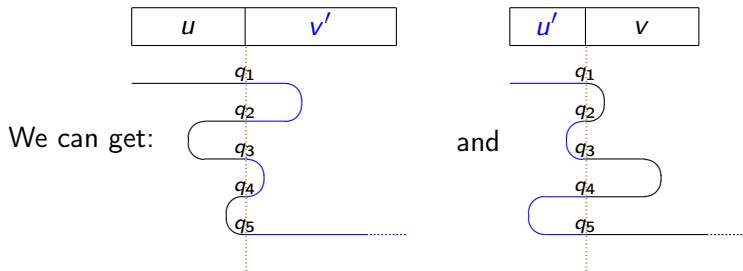
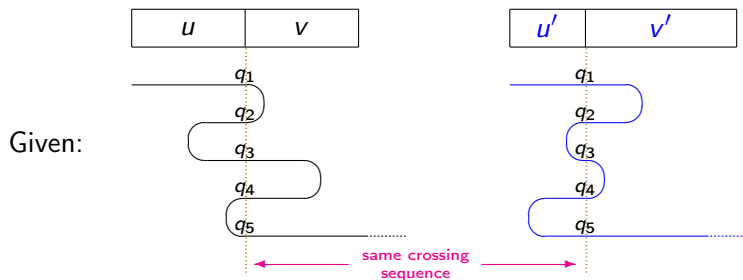
Given:



Crossing Sequences: "Cut-and-Paste"



Crossing Sequences: "Cut-and-Paste"



Lower Bounds for Accept Measure

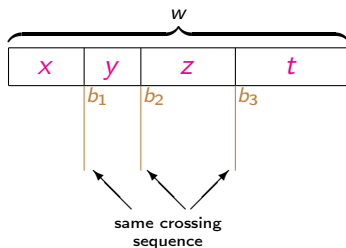
Lemma

If a string w is accepted by a computation C having a same crossing sequence at 3 different boundaries of the input, then there is a computation C' with $c(C') = c(C)$ accepting a shorter string w'

Lower Bounds for Accept Measure

Lemma

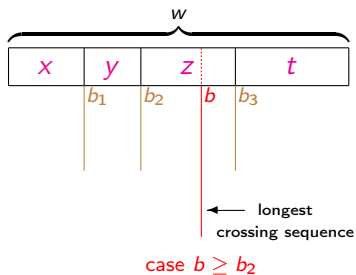
If a string w is accepted by a computation C having a same crossing sequence at 3 different boundaries of the input, then there is a computation C' with $c(C') = c(C)$ accepting a shorter string w'



Lower Bounds for Accept Measure

Lemma

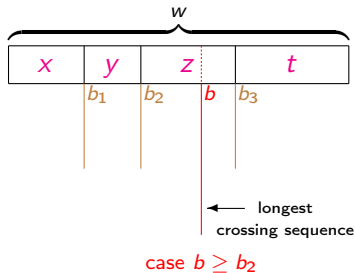
If a string w is accepted by a computation C having a same crossing sequence at 3 different boundaries of the input, then there is a computation C' with $c(C') = c(C)$ accepting a shorter string w'



Lower Bounds for Accept Measure

Lemma

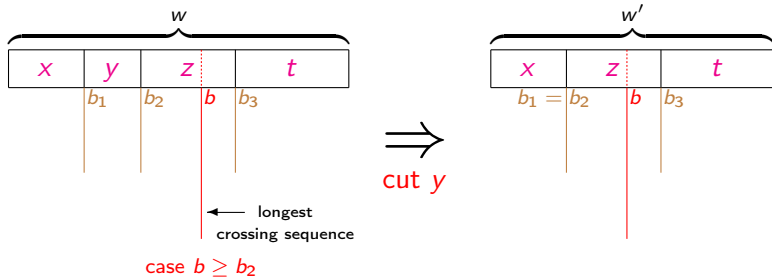
If a string w is accepted by a computation C having a same crossing sequence at 3 different boundaries of the input, then there is a computation C' with $c(C') = c(C)$ accepting a shorter string w'



Lower Bounds for Accept Measure

Lemma

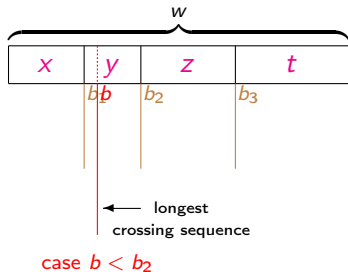
If a string w is accepted by a computation C having a same crossing sequence at 3 different boundaries of the input, then there is a computation C' with $c(C') = c(C)$ accepting a shorter string w'



Lower Bounds for Accept Measure

Lemma

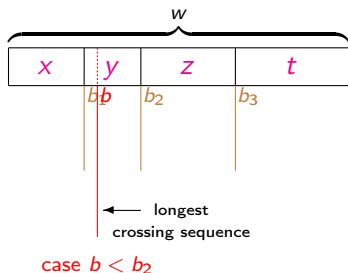
If a string w is accepted by a computation C having a same crossing sequence at 3 different boundaries of the input, then there is a computation C' with $c(C') = c(C)$ accepting a shorter string w'



Lower Bounds for Accept Measure

Lemma

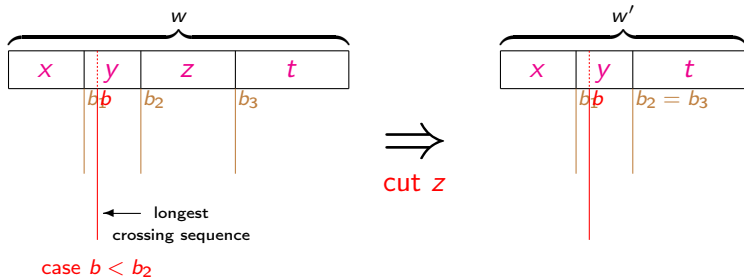
If a string w is accepted by a computation C having a same crossing sequence at 3 different boundaries of the input, then there is a computation C' with $c(C') = c(C)$ accepting a shorter string w'



Lower Bounds for Accept Measure

Lemma

If a string w is accepted by a computation C having a same crossing sequence at 3 different boundaries of the input, then there is a computation C' with $c(C') = c(C)$ accepting a shorter string w'



Accept Measure: Lower Bound for $c(n)$

- ▶ $L[k] :=$ set of strings having an accepting computation \mathcal{C} with $c(\mathcal{C}) = k$
- ▶ $w_k :=$ a shortest string in $L[k]$, for $L[k] \neq \emptyset$
- ▶ $n_k := |w_k|$
- ▶ On w_k each crossing sequence can appear at most twice
- ▶ At least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ Hence $q^{k+1} \geq \lfloor \frac{n_k-1}{2} \rfloor$ ($q :=$ number of states), then:

$$c(n_k) \geq k \geq \log_q \lfloor \frac{n_k-1}{2} \rfloor - 1$$

- ▶ If $c(n)$ is unbounded then $L[k] \neq \emptyset$ for infinitely many k
- ▶ Hence $c(n) \geq d \log n$, for some $d > 0$, infinitely many n

Accept Measure: Lower Bound for $c(n)$

- ▶ $L[k] :=$ set of strings having an accepting computation \mathcal{C} with $c(\mathcal{C}) = k$
- ▶ $w_k :=$ a shortest string in $L[k]$, for $L[k] \neq \emptyset$
- ▶ $n_k := |w_k|$
- ▶ On w_k each crossing sequence can appear at most twice
- ▶ At least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ Hence $q^{k+1} \geq \lfloor \frac{n_k-1}{2} \rfloor$ ($q :=$ number of states), then:

$$c(n_k) \geq k \geq \log_q \lfloor \frac{n_k-1}{2} \rfloor - 1$$

- ▶ If $c(n)$ is unbounded then $L[k] \neq \emptyset$ for infinitely many k
- ▶ Hence $c(n) \geq d \log n$, for some $d > 0$, infinitely many n

Accept Measure: Lower Bound for $c(n)$

- ▶ $L[k] :=$ set of strings having an accepting computation \mathcal{C} with $c(\mathcal{C}) = k$
- ▶ $w_k :=$ a shortest string in $L[k]$, for $L[k] \neq \emptyset$
- ▶ $n_k := |w_k|$
- ▶ On w_k each crossing sequence can appear at most twice
- ▶ At least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ Hence $q^{k+1} \geq \lfloor \frac{n_k-1}{2} \rfloor$ ($q :=$ number of states), then:

$$c(n_k) \geq k \geq \log_q \lfloor \frac{n_k-1}{2} \rfloor - 1$$

- ▶ If $c(n)$ is unbounded then $L[k] \neq \emptyset$ for infinitely many k
- ▶ Hence $c(n) \geq d \log n$, for some $d > 0$, infinitely many n

Accept Measure: Lower Bound for $c(n)$

- ▶ $L[k] :=$ set of strings having an accepting computation \mathcal{C} with $c(\mathcal{C}) = k$
- ▶ $w_k :=$ a shortest string in $L[k]$, for $L[k] \neq \emptyset$
- ▶ $n_k := |w_k|$
- ▶ On w_k each crossing sequence can appear at most twice
- ▶ At least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ Hence $q^{k+1} \geq \lfloor \frac{n_k-1}{2} \rfloor$ ($q :=$ number of states), then:

$$c(n_k) \geq k \geq \log_q \lfloor \frac{n_k-1}{2} \rfloor - 1$$

- ▶ If $c(n)$ is unbounded then $L[k] \neq \emptyset$ for infinitely many k
- ▶ Hence $c(n) \geq d \log n$, for some $d > 0$, infinitely many n

Accept Measure: Lower Bound for $c(n)$

- ▶ $L[k] :=$ set of strings having an accepting computation \mathcal{C} with $c(\mathcal{C}) = k$
- ▶ $w_k :=$ a shortest string in $L[k]$, for $L[k] \neq \emptyset$
- ▶ $n_k := |w_k|$
- ▶ On w_k each crossing sequence can appear at most twice
- ▶ **At least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences**
- ▶ Hence $q^{k+1} \geq \lfloor \frac{n_k-1}{2} \rfloor$ ($q :=$ number of states), then:

$$c(n_k) \geq k \geq \log_q \lfloor \frac{n_k-1}{2} \rfloor - 1$$

- ▶ If $c(n)$ is unbounded then $L[k] \neq \emptyset$ for infinitely many k
- ▶ Hence $c(n) \geq d \log n$, for some $d > 0$, infinitely many n

Accept Measure: Lower Bound for $c(n)$

- ▶ $L[k] :=$ set of strings having an accepting computation \mathcal{C} with $c(\mathcal{C}) = k$
- ▶ $w_k :=$ a shortest string in $L[k]$, for $L[k] \neq \emptyset$
- ▶ $n_k := |w_k|$
- ▶ On w_k each crossing sequence can appear at most twice
- ▶ At least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ Hence $q^{k+1} \geq \lfloor \frac{n_k-1}{2} \rfloor$ ($q :=$ number of states), then:

$$c(n_k) \geq k \geq \log_q \lfloor \frac{n_k-1}{2} \rfloor - 1$$

- ▶ If $c(n)$ is unbounded then $L[k] \neq \emptyset$ for infinitely many k
- ▶ Hence $c(n) \geq d \log n$, for some $d > 0$, infinitely many n

Accept Measure: Lower Bound for $c(n)$

- ▶ $L[k] :=$ set of strings having an accepting computation \mathcal{C} with $c(\mathcal{C}) = k$
- ▶ $w_k :=$ a shortest string in $L[k]$, for $L[k] \neq \emptyset$
- ▶ $n_k := |w_k|$
- ▶ On w_k each crossing sequence can appear at most twice
- ▶ At least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ Hence $q^{k+1} \geq \lfloor \frac{n_k-1}{2} \rfloor$ ($q :=$ number of states), then:

$$c(n_k) \geq k \geq \log_q \lfloor \frac{n_k-1}{2} \rfloor - 1$$

- ▶ If $c(n)$ is unbounded then $L[k] \neq \emptyset$ for infinitely many k
- ▶ Hence $c(n) \geq d \log n$, for some $d > 0$, infinitely many n

Accept Measure: Lower Bound for $c(n)$

- ▶ $L[k] :=$ set of strings having an accepting computation \mathcal{C} with $c(\mathcal{C}) = k$
- ▶ $w_k :=$ a shortest string in $L[k]$, for $L[k] \neq \emptyset$
- ▶ $n_k := |w_k|$
- ▶ On w_k each crossing sequence can appear at most twice
- ▶ At least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ Hence $q^{k+1} \geq \lfloor \frac{n_k-1}{2} \rfloor$ ($q :=$ number of states), then:

$$c(n_k) \geq k \geq \log_q \lfloor \frac{n_k-1}{2} \rfloor - 1$$

- ▶ If $c(n)$ is unbounded then $L[k] \neq \emptyset$ for infinitely many k
- ▶ Hence $c(n) \geq d \log n$, for some $d > 0$, infinitely many n

Accept Measure: Lower Bound for $c(n)$

- ▶ $L[k] :=$ set of strings having an accepting computation \mathcal{C} with $c(\mathcal{C}) = k$
- ▶ $w_k :=$ a shortest string in $L[k]$, for $L[k] \neq \emptyset$
- ▶ $n_k := |w_k|$
- ▶ On w_k each crossing sequence can appear at most twice
- ▶ At least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ Hence $q^{k+1} \geq \lfloor \frac{n_k-1}{2} \rfloor$ ($q :=$ number of states), then:

$$c(n_k) \geq k \geq \log_q \lfloor \frac{n_k-1}{2} \rfloor - 1$$

- ▶ If $c(n)$ is unbounded then $L[k] \neq \emptyset$ for infinitely many k
- ▶ Hence $c(n) \geq d \log n$, for some $d > 0$, infinitely many n

$c(n) = o(\log n)$ implies $c(n) = O(1)$ and L regular

Accept Measure: Lower Bound for $t(n)$

If $c(n) \neq O(1)$:

- ▶ w_k is accepted by a computation C_k
using at least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ C_k has $\geq \lfloor \frac{n_k-1}{4} \rfloor$ crossing sequences of length $\geq \log_q \lfloor \frac{n_k-1}{4} \rfloor$
(combinatorial argument)
- ▶ Hence C_k consists of at least
$$\lfloor \frac{n_k-1}{4} \rfloor \cdot \log_q \lfloor \frac{n_k-1}{4} \rfloor \geq d |w_k| \log |w_k|$$

many steps, for some $d \geq 0$
- ▶ $t(n) \geq d n \log n$, for infinitely many n

Accept Measure: Lower Bound for $t(n)$

If $c(n) \neq O(1)$:

- ▶ w_k is accepted by a computation C_k
using at least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ C_k has $\geq \lfloor \frac{n_k-1}{4} \rfloor$ crossing sequences of length $\geq \log_q \lfloor \frac{n_k-1}{4} \rfloor$
(combinatorial argument)
- ▶ Hence C_k consists of at least
$$\lfloor \frac{n_k-1}{4} \rfloor \cdot \log_q \lfloor \frac{n_k-1}{4} \rfloor \geq d |w_k| \log |w_k|$$
many steps, for some $d \geq 0$
- ▶ $t(n) \geq d n \log n$, for infinitely many n

Accept Measure: Lower Bound for $t(n)$

If $c(n) \neq O(1)$:

- ▶ w_k is accepted by a computation \mathcal{C}_k
using at least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ \mathcal{C}_k has $\geq \lfloor \frac{n_k-1}{4} \rfloor$ crossing sequences of length $\geq \log_q \lfloor \frac{n_k-1}{4} \rfloor$
(combinatorial argument)

▶ Hence \mathcal{C}_k consists of at least

$$\lfloor \frac{n_k-1}{4} \rfloor \cdot \log_q \lfloor \frac{n_k-1}{4} \rfloor \geq d |w_k| \log |w_k|$$

many steps, for some $d \geq 0$

▶ $t(n) \geq d n \log n$, for infinitely many n

Accept Measure: Lower Bound for $t(n)$

If $c(n) \neq O(1)$:

- ▶ w_k is accepted by a computation \mathcal{C}_k
using at least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ \mathcal{C}_k has $\geq \lfloor \frac{n_k-1}{4} \rfloor$ crossing sequences of length $\geq \log_q \lfloor \frac{n_k-1}{4} \rfloor$
(combinatorial argument)
- ▶ Hence \mathcal{C}_k consists of at least

$$\lfloor \frac{n_k-1}{4} \rfloor \cdot \log_q \lfloor \frac{n_k-1}{4} \rfloor \geq d |w_k| \log |w_k|$$

many steps, for some $d \geq 0$

- ▶ $t(n) \geq d n \log n$, for infinitely many n

Accept Measure: Lower Bound for $t(n)$

If $c(n) \neq O(1)$:

- ▶ w_k is accepted by a computation \mathcal{C}_k
using at least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ \mathcal{C}_k has $\geq \lfloor \frac{n_k-1}{4} \rfloor$ crossing sequences of length $\geq \log_q \lfloor \frac{n_k-1}{4} \rfloor$
(combinatorial argument)
- ▶ Hence \mathcal{C}_k consists of at least

$$\lfloor \frac{n_k-1}{4} \rfloor \cdot \log_q \lfloor \frac{n_k-1}{4} \rfloor \geq d |w_k| \log |w_k|$$

many steps, for some $d \geq 0$

- ▶ $t(n) \geq d n \log n$, for infinitely many n

Accept Measure: Lower Bound for $t(n)$

If $c(n) \neq O(1)$:

- ▶ w_k is accepted by a computation \mathcal{C}_k
using at least $\lfloor \frac{n_k-1}{2} \rfloor$ different crossing sequences
- ▶ \mathcal{C}_k has $\geq \lfloor \frac{n_k-1}{4} \rfloor$ crossing sequences of length $\geq \log_q \lfloor \frac{n_k-1}{4} \rfloor$
(combinatorial argument)
- ▶ Hence \mathcal{C}_k consists of at least

$$\lfloor \frac{n_k-1}{4} \rfloor \cdot \log_q \lfloor \frac{n_k-1}{4} \rfloor \geq d |w_k| \log |w_k|$$

many steps, for some $d \geq 0$

- ▶ $t(n) \geq d n \log n$, for infinitely many n

Hence:

$t(n) = o(n \log n)$ implies $c(n) = O(1)$ and, thus,
 L regular and $t(n) = O(n)$

Lower Bounds for Accept Measure

Summing up:

Theorem ([P.'09])

Let M be a n TM accepting a language L such that in each accepting computation

- ▶ *the length of crossing sequences is bounded by $c(n)$*
- ▶ *the time is bounded by $t(n)$*

If $c(n) = o(\log n)$ then $c(n) = O(1)$ and L is regular

If $t(n) = o(n \log n)$ then

- ▶ *$t(n) = O(n)$*
- ▶ *$c(n) = O(1)$*
- ▶ *L is regular*

Lower Bounds for Accept Measure

Summing up:

Theorem ([P.'09])

Let M be a n TM accepting a language L such that in each accepting computation

- ▶ *the length of crossing sequences is bounded by $c(n)$*
- ▶ *the time is bounded by $t(n)$*

If $c(n) = o(\log n)$ then $c(n) = O(1)$ and L is regular

If $t(n) = o(n \log n)$ then

- ▶ *$t(n) = O(n)$*
- ▶ *$c(n) = O(1)$*
- ▶ *L is regular*

Lower Bounds for Accept Measure

Summing up:

Theorem ([P.'09])

Let M be a n TM accepting a language L such that in each accepting computation

- ▶ *the length of crossing sequences is bounded by $c(n)$*
- ▶ *the time is bounded by $t(n)$*

If $c(n) = o(\log n)$ then $c(n) = O(1)$ and L is regular

If $t(n) = o(n \log n)$ then

- ▶ *$t(n) = O(n)$*
- ▶ *$c(n) = O(1)$*
- ▶ *L is regular*

Weak Measure

Does it is possible to extend the lower bounds
from the accept to the weak measure?

Time: negative answer

Theorem ([Michel '91])

*There exists an NP-complete language accepted in time $O(n)$
by a n TM under the weak measure*

- ▶ Linear time is necessary for regular languages

However:

The length of crossing sequences should grow
at least as $\log \log n$ [P.'09]

Does it is possible to extend the lower bounds
from the accept to the weak measure?

Time: negative answer

Theorem ([Michel '91])

*There exists an NP-complete language accepted in time $O(n)$
by a nTM under the weak measure*

- ▶ Linear time is necessary for regular languages

However:

The length of crossing sequences should grow
at least as $\log \log n$ [P.'09]

Does it is possible to extend the lower bounds
from the accept to the weak measure?

Time: negative answer

Theorem ([Michel '91])

*There exists an NP-complete language accepted in time $O(n)$
by a nTM under the weak measure*

- ▶ Linear time is necessary for regular languages

However:

The length of crossing sequences should grow
at least as $\log \log n$ [P.'09]

Weak Measure: Lower Bound for $c(n)$

- ▶ $L :=$ language accepted by the given machine M
- ▶ For each $n \geq 1$:
 - $N_n :=$ NFA with states all crossing sequences of length $\leq c(n)$, transitions defined according to the “compatibility” relation, at most $q^{c(n)+1}$ states
 - N_n agrees with M on strings of length $\leq n$
 - $A_n :=$ DFA equivalent to N_n , at most $2^{q^{c(n)+1}}$ states
 - If L is not regular, then the number of the states of A_n is $\geq \frac{n+3}{2}$ i.o. [Karp '67]
 - Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, implying

$$c(n) \geq d \log \log n$$

for some $d > 0$ and infinitely many n 's

Weak Measure: Lower Bound for $c(n)$

- ▶ $L :=$ language accepted by the given machine M
- ▶ For each $n \geq 1$:
 - $N_n :=$ NFA with states all crossing sequences of length $\leq c(n)$, transitions defined according to the “compatibility” relation, at most $q^{c(n)+1}$ states
- ▶ N_n agrees with M on strings of length $\leq n$
- ▶ $A_n :=$ DFA equivalent to N_n , at most $2^{q^{c(n)+1}}$ states
- ▶ If L is not regular, then the number of the states of A_n is $\geq \frac{n+3}{2}$ i.o. [Karp '67]
- ▶ Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, implying

$$c(n) \geq d \log \log n$$

for some $d > 0$ and infinitely many n 's

Weak Measure: Lower Bound for $c(n)$

- ▶ $L :=$ language accepted by the given machine M
- ▶ For each $n \geq 1$:
 - $N_n :=$ NFA with states all crossing sequences of length $\leq c(n)$, transitions defined according to the “compatibility” relation, at most $q^{c(n)+1}$ states
 - ▶ N_n agrees with M on strings of length $\leq n$
 - ▶ $A_n :=$ DFA equivalent to N_n , at most $2^{q^{c(n)+1}}$ states
 - ▶ If L is not regular, then the number of the states of A_n is $\geq \frac{n+3}{2}$ i.o. [Karp '67]
 - ▶ Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, implying

$$c(n) \geq d \log \log n$$

for some $d > 0$ and infinitely many n 's

Weak Measure: Lower Bound for $c(n)$

- ▶ $L :=$ language accepted by the given machine M
- ▶ For each $n \geq 1$:
 - $N_n :=$ NFA with states all crossing sequences of length $\leq c(n)$, transitions defined according to the “compatibility” relation, at most $q^{c(n)+1}$ states
- ▶ N_n agrees with M on strings of length $\leq n$
- ▶ $A_n :=$ DFA equivalent to N_n , at most $2^{q^{c(n)+1}}$ states
- ▶ If L is not regular, then the number of the states of A_n is $\geq \frac{n+3}{2}$ i.o. [Karp '67]
- ▶ Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, implying

$$c(n) \geq d \log \log n$$

for some $d > 0$ and infinitely many n 's

Weak Measure: Lower Bound for $c(n)$

- ▶ $L :=$ language accepted by the given machine M
- ▶ For each $n \geq 1$:
 - $N_n :=$ NFA with states all crossing sequences of length $\leq c(n)$, transitions defined according to the “compatibility” relation, at most $q^{c(n)+1}$ states
- ▶ N_n agrees with M on strings of length $\leq n$
- ▶ $A_n :=$ DFA equivalent to N_n , at most $2^{q^{c(n)+1}}$ states
- ▶ If L is not regular, then the number of the states of A_n is $\geq \frac{n+3}{2}$ i.o. [Karp '67]
- ▶ Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, implying

$$c(n) \geq d \log \log n$$

for some $d > 0$ and infinitely many n 's

Weak Measure: Lower Bound for $c(n)$

- ▶ $L :=$ language accepted by the given machine M
- ▶ For each $n \geq 1$:
 - $N_n :=$ NFA with states all crossing sequences of length $\leq c(n)$, transitions defined according to the “compatibility” relation, at most $q^{c(n)+1}$ states
- ▶ N_n agrees with M on strings of length $\leq n$
- ▶ $A_n :=$ DFA equivalent to N_n , at most $2^{q^{c(n)+1}}$ states
- ▶ If L is not regular, then the number of the states of A_n is $\geq \frac{n+3}{2}$ i.o. [Karp '67]
- ▶ Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, implying

$$c(n) \geq d \log \log n$$

for some $d > 0$ and infinitely many n 's

Summary of the Lower Bounds

		strong	accept	weak
dTM	$t(n)$			
	$c(n)$			
nTM	$t(n)$			
	$c(n)$			

Summary of the Lower Bounds

		strong	accept	weak
dTM	$t(n)$	$n \log n$		
	$c(n)$	$\log n$		
nTM	$t(n)$			
	$c(n)$			

Trakhtenbrot (1964) and Hartmanis (1968)
Hennie (1965) for $c(n)$

Summary of the Lower Bounds

		strong	accept	weak
dTM	$t(n)$	$n \log n$		
	$c(n)$	$\log n$		
nTM	$t(n)$	$n \log n$		
	$c(n)$	$\log n$		

Tadaki, Yamakami, and Lin (2010)

Summary of the Lower Bounds

		strong	accept	weak
dTM	$t(n)$	$n \log n$		
	$c(n)$	$\log n$		
nTM	$t(n)$	$n \log n$	$n \log n$	
	$c(n)$	$\log n$	$\log n$	

Pighizzini (2009)

Summary of the Lower Bounds

		strong	accept	weak
dTM	$t(n)$	$n \log n$	$n \log n$	
	$c(n)$	$\log n$	$\log n$	
nTM	$t(n)$	$n \log n$	$n \log n$	
	$c(n)$	$\log n$	$\log n$	

Consequence of accept nondeterministic case

Summary of the Lower Bounds

		strong	accept	weak
dTM	$t(n)$	$n \log n$	$n \log n$	$n \log n$
	$c(n)$	$\log n$	$\log n$	$\log n$
nTM	$t(n)$	$n \log n$	$n \log n$	
	$c(n)$	$\log n$	$\log n$	

For deterministic machines, accept and weak is the same

Summary of the Lower Bounds

		strong	accept	weak
dTM	$t(n)$	$n \log n$	$n \log n$	$n \log n$
	$c(n)$	$\log n$	$\log n$	$\log n$
nTM	$t(n)$	$n \log n$	$n \log n$	n
	$c(n)$	$\log n$	$\log n$	$\log \log n$

$t(n)$: simple bound

$c(n)$: Pighizzini (2009)

Summary of the Lower Bounds

		strong	accept	weak
dTM	$t(n)$	$n \log n$	$n \log n$	$n \log n$
	$c(n)$	$\log n$	$\log n$	$\log n$
nTM	$t(n)$	$n \log n$	$n \log n$	n
	$c(n)$	$\log n$	$\log n$	$\log \log n$

Optimality

Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks all the 1st, 3rd, 5th, etc. unmarked squares

Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked



Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked

a	a	a	a	a	a	a	a	a	a	a	a
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----



X	a	X	a	X	a	X	a	X	a	X	a
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

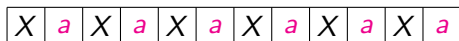
Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked



Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked

X	a	X	a	X	a	X	a	X	a	X	a
---	---	---	---	---	---	---	---	---	---	---	---



X	X	X	a	X	X	X	a	X	X	X	a
---	---	---	---	---	---	---	---	---	---	---	---

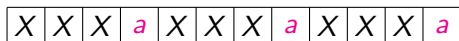
Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked



Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked

X	X	X	a	X	X	X	a	X	X	X	a
---	---	---	---	---	---	---	---	---	---	---	---



X	X	X	X	X	X	X	a	X	X	X	X
---	---	---	---	---	---	---	---	---	---	---	---

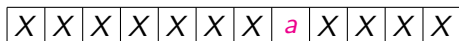
Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked



Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked

X	X	X	X	X	X	X	a	X	X	X	X
---	---	---	---	---	---	---	---	---	---	---	---

 reject!

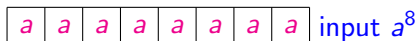
Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked



Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked

a	a	a	a	a	a	a	a	a
-----	-----	-----	-----	-----	-----	-----	-----	-----



X	a	X	a	X	a	X	a	a
-----	-----	-----	-----	-----	-----	-----	-----	-----

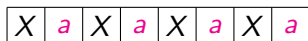
Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked



Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked

X	a	X	a	X	a	X	a
---	---	---	---	---	---	---	---



X	X	X	a	X	X	X	a
---	---	---	---	---	---	---	---

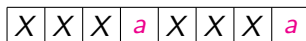
Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked



Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked

X	X	X	a	X	X	X	a
---	---	---	---	---	---	---	---



X	X	X	X	X	X	X	a
---	---	---	---	---	---	---	---

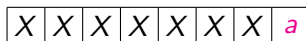
Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked



Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked

X	X	X	X	X	X	X	X	a
---	---	---	---	---	---	---	---	-----



X	X	X	X	X	X	X	X	X
---	---	---	---	---	---	---	---	---

Optimality of the Bounds

$$L = \{a^{2^m} \mid m \geq 0\}$$

[Hartmanis '68]

L is accepted by a dTM M as follows:

- ▶ At the beginning all the input cells are “unmarked”
- ▶ M sweeps from left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares
- ▶ M repeats the previous step until the rightmost square of the input segment becomes marked
- ▶ M accepts if and only if all the input segment is marked

X	X	X	X	X	X	X	X
---	---	---	---	---	---	---	---

 accept!

Complexity

- ▶ On input a^n , M makes $O(\log n)$ sweeps of the tape:
 $c(n) = O(\log n)$ and $t(n) = O(n \log n)$
- ▶ M is *deterministic* and the previous bounds are satisfied by *all* computations: **strong measure**
- ▶ This gives the optimality of all the lower bounds in the table

		strong	weak
DTM	$T(n)$	$\Omega(n \log n)$	$\Omega(n \log n)$
DTM	$C(n)$	$\Omega(\log n)$	$\Omega(\log n)$
NTM	$T(n)$	$\Omega(n \log n)$	$\Omega(n \log n)$
NTM	$C(n)$	$\Omega(\log n)$	$\Omega(\log n)$

with the exception of those for NTM under the weak measure

Complexity

- ▶ On input a^n , M makes $O(\log n)$ sweeps of the tape:
 $c(n) = O(\log n)$ and $t(n) = O(n \log n)$
- ▶ M is *deterministic* and the previous bounds are satisfied by *all* computations: **strong measure**
- ▶ This gives the optimality of all the lower bounds in the table

		strong	accept	weak
dTM	$t(n)$	$n \log n$	$n \log n$	$n \log n$
	$c(n)$	$\log n$	$\log n$	$\log n$
nTM	$t(n)$	$n \log n$	$n \log n$	n
	$c(n)$	$\log n$	$\log n$	$\log \log n$

with the exception of those for nTMs, under the weak measure

- ▶ Unary systems

Complexity

- ▶ On input a^n , M makes $O(\log n)$ sweeps of the tape:
 $c(n) = O(\log n)$ and $t(n) = O(n \log n)$
- ▶ M is *deterministic* and the previous bounds are satisfied by *all* computations: **strong measure**
- ▶ This gives the optimality of all the lower bounds in the table

		strong	accept	weak
dTM	$t(n)$	$n \log n$	$n \log n$	$n \log n$
	$c(n)$	$\log n$	$\log n$	$\log n$
nTM	$t(n)$	$n \log n$	$n \log n$	n
	$c(n)$	$\log n$	$\log n$	$\log \log n$

with the exception of those for nTMs, under the weak measure

- ▶ *Unary witness*

Complexity

- ▶ On input a^n , M makes $O(\log n)$ sweeps of the tape:
 $c(n) = O(\log n)$ and $t(n) = O(n \log n)$
- ▶ M is *deterministic* and the previous bounds are satisfied by *all* computations: **strong measure**
- ▶ This gives the optimality of all the lower bounds in the table

		strong	accept	weak
dTM	$t(n)$	$n \log n$	$n \log n$	$n \log n$
	$c(n)$	$\log n$	$\log n$	$\log n$
nTM	$t(n)$	$n \log n$	$n \log n$	n
	$c(n)$	$\log n$	$\log n$	$\log \log n$

with the exception of those for nTMs, under the weak measure

- ▶ *Unary witness*

Complexity

- ▶ On input a^n , M makes $O(\log n)$ sweeps of the tape:
 $c(n) = O(\log n)$ and $t(n) = O(n \log n)$
- ▶ M is *deterministic* and the previous bounds are satisfied by *all* computations: **strong measure**
- ▶ This gives the optimality of all the lower bounds in the table

		strong	accept	weak
dTM	$t(n)$	$n \log n$	$n \log n$	$n \log n$
	$c(n)$	$\log n$	$\log n$	$\log n$
nTM	$t(n)$	$n \log n$	$n \log n$	n
	$c(n)$	$\log n$	$\log n$	$\log \log n$

with the exception of those for nTMs, under the weak measure

- ▶ *Unary* witness

Weak Measure: Optimality for nTMs

- ▶ There are nonregular languages accepted in time $O(n)$
[Michel '91]
- ▶ Regular languages require time n
- ▶ No “gap” between regular and nonregular languages
- ▶ The example in [Michel '91] strongly relies on an input alphabet with *more than one symbol*
- ▶ Up to now, we do not know any example of unary nonregular language accepted in weak time $O(n)$
- ▶ *Conjecture: If a nTM accepts a unary language L in time $o(n \log \log n)$ under the weak measure then L is regular*
- ▶ We know an example of unary language accepted in time $O(n \log \log n)$ under the weak measure

Weak Measure: Optimality for nTMs

- ▶ There are nonregular languages accepted in time $O(n)$
[Michel '91]
- ▶ Regular languages require time n
- ▶ No “gap” between regular and nonregular languages
- ▶ The example in [Michel '91] strongly relies on an input alphabet with *more than one symbol*
- ▶ Up to now, we do not know any example of unary nonregular language accepted in weak time $O(n)$
- ▶ *Conjecture: If a nTM accepts a unary language L in time $o(n \log \log n)$ under the weak measure then L is regular*
- ▶ We know an example of unary language accepted in time $O(n \log \log n)$ under the weak measure

Weak Measure: Optimality for nTMs

- ▶ There are nonregular languages accepted in time $O(n)$
[Michel '91]
- ▶ Regular languages require time n
- ▶ No “gap” between regular and nonregular languages
- ▶ The example in [Michel '91] strongly relies on an input alphabet with *more than one symbol*
- ▶ Up to now, we do not know any example of unary nonregular language accepted in weak time $O(n)$
- ▶ *Conjecture: If a nTM accepts a unary language L in time $o(n \log \log n)$ under the weak measure then L is regular*
- ▶ We know an example of unary language accepted in time $O(n \log \log n)$ under the weak measure

Weak Measure: Optimality for nTMs

- ▶ There are nonregular languages accepted in time $O(n)$
[Michel '91]
- ▶ Regular languages require time n
- ▶ No “gap” between regular and nonregular languages
- ▶ The example in [Michel '91] strongly relies on an input alphabet with *more than one symbol*
- ▶ Up to now, we do not know any example of unary nonregular language accepted in weak time $O(n)$
- ▶ Conjecture: If a nTM accepts a unary language L in time $o(n \log \log n)$ under the weak measure then L is regular
- ▶ We know an example of unary language accepted in time $O(n \log \log n)$ under the weak measure

Weak Measure: Optimality for nTMs

- ▶ There are nonregular languages accepted in time $O(n)$
[Michel '91]
- ▶ Regular languages require time n
- ▶ No “gap” between regular and nonregular languages
- ▶ The example in [Michel '91] strongly relies on an input alphabet with *more than one symbol*
- ▶ Up to now, we do not know any example of unary nonregular language accepted in weak time $O(n)$
- ▶ Conjecture: If a nTM accepts a unary language L in time $o(n \log \log n)$ under the weak measure then L is regular
- ▶ We know an example of unary language accepted in time $O(n \log \log n)$ under the weak measure

Weak Measure: Optimality for nTMs

- ▶ There are nonregular languages accepted in time $O(n)$
[Michel '91]
- ▶ Regular languages require time n
- ▶ No “gap” between regular and nonregular languages
- ▶ The example in [Michel '91] strongly relies on an input alphabet with *more than one symbol*
- ▶ Up to now, we do not know any example of unary nonregular language accepted in weak time $O(n)$
- ▶ **Conjecture:** If a nTM accepts a *unary* language L in time $o(n \log \log n)$ under the weak measure then L is regular
- ▶ We know an example of unary language accepted in time $O(n \log \log n)$ under the weak measure

Weak Measure: Optimality for nTMs

- ▶ There are nonregular languages accepted in time $O(n)$
[Michel '91]
- ▶ Regular languages require time n
- ▶ No “gap” between regular and nonregular languages
- ▶ The example in [Michel '91] strongly relies on an input alphabet with *more than one symbol*
- ▶ Up to now, we do not know any example of unary nonregular language accepted in weak time $O(n)$
- ▶ *Conjecture:* If a nTM accepts a *unary* language L in time $o(n \log \log n)$ under the weak measure then L is regular
- ▶ We know an example of unary language accepted in time $O(n \log \log n)$ under the weak measure

Fast Recognition of Unary Languages

nTMs and Unary Languages: Basic Techniques

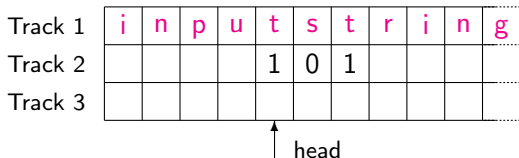
Tape Tracks

- ▶ We can consider a tape divided in a fixed number of tracks
- ▶ The input is written on the first track

Track 1	i	n	p	u	t	s	t	r	i	n	g
Track 2	m	e	m	o	r	y	s	p	a	c	e
Track 3	m	e	m	o	r	y	s	p	a	c	e

nTMs and Unary Languages: Basic Techniques

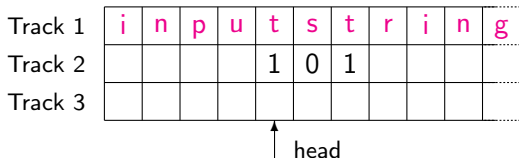
How to count input symbols



- ▶ A counter is kept on track 2, *starting from the position scanned by the tape head*
- ▶ When the head is moved to the right, the counter is incremented to count one more position and it is shifted to the right
- ▶ This is done in $O(\log j)$ steps using track 3 as an auxiliary variable ($j =$ value of the counter)
- ▶ k tape positions are counted in $O(k \log k)$ moves

nTMs and Unary Languages: Basic Techniques

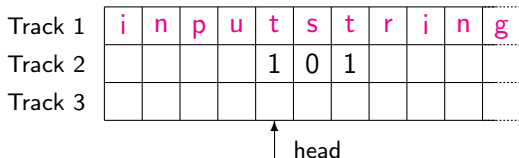
How to count input symbols



- ▶ A counter is kept on track 2, *starting from the position scanned by the tape head*
- ▶ When the head is moved to the right, the counter is incremented to count one more position and it is shifted to the right
- ▶ This is done in $O(\log j)$ steps using track 3 as an auxiliary variable ($j = \text{value of the counter}$)
- ▶ k tape positions are counted in $O(k \log k)$ moves

nTMs and Unary Languages: Basic Techniques

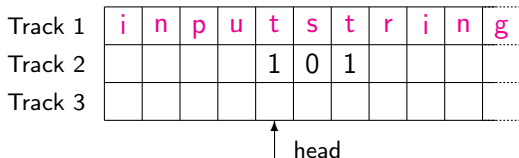
How to count input symbols



- ▶ A counter is kept on track 2, *starting from the position scanned by the tape head*
- ▶ When the head is moved to the right, the counter is incremented to count one more position and it is shifted to the right
- ▶ This is done in $O(\log j)$ steps using track 3 as an auxiliary variable ($j = \text{value of the counter}$)
- ▶ k tape positions are counted in $O(k \log k)$ moves

nTMs and Unary Languages: Basic Techniques

How to count input symbols



- ▶ A counter is kept on track 2, *starting from the position scanned by the tape head*
- ▶ When the head is moved to the right, the counter is incremented to count one more position and it is shifted to the right
- ▶ This is done in $O(\log j)$ steps using track 3 as an auxiliary variable ($j = \text{value of the counter}$)
- ▶ k tape positions are counted in $O(k \log k)$ moves

nTMs and Unary Languages: Basic techniques

How to compute $n \bmod k$

n = input length

k = integer written somewhere

- ▶ Reset the counter on track 2 each time it becomes equal to k
- ▶ When the end of the input is reached, track 2 contains $n \bmod k$
- ▶ To implement the comparison between the counter and k :
 - The value of k is kept on an extra track (track 4)
 - When the input head is moved to the right to count one more position, the representation of k is moved one position to the left, in such a way that it is always aligned with the current head on track 2, to make easy the comparison
- ▶ The total time is $O(n \log k)$

nTMs and Unary Languages: Basic techniques

How to compute $n \bmod k$

n = input length

k = integer written somewhere

- ▶ Reset the counter on track 2 each time it becomes equal to k
- ▶ When the end of the input is reached, track 2 contains $n \bmod k$
- ▶ To implement the comparison between the counter and k :
 - The value of k is kept on an extra track (track 4)
 - When the input head is moved to the right to count one more position, the representation of k is moved one position to the right in such a way that it is always aligned with the counter on track 2, to make easy the comparison
- ▶ The total time is $O(n \log k)$

nTMs and Unary Languages: Basic techniques

How to compute $n \bmod k$

n = input length

k = integer written somewhere

- ▶ Reset the counter on track 2 each time it becomes equal to k
- ▶ When the end of the input is reached, track 2 contains $n \bmod k$
- ▶ To implement the comparison between the counter and k :
 - The value of k is kept on an extra track (track 4)
 - When the input head is moved to the right to count one more position, the representation of k is moved one position to the right in such a way that it is always aligned with the counter on track 2, to make easy the comparison
- ▶ The total time is $O(n \log k)$

nTMs and Unary Languages: Basic techniques

How to compute $n \bmod k$

n = input length

k = integer written somewhere

- ▶ Reset the counter on track 2 each time it becomes equal to k
- ▶ When the end of the input is reached, track 2 contains $n \bmod k$
- ▶ To implement the comparison between the counter and k :
 - The value of k is kept on an extra track (track 4)
 - When the input head is moved to the right to count one more position, the representation of k is moved one position to the right in such a way that it is always aligned with the counter on track 2, to make easy the comparison
- ▶ The total time is $O(n \log k)$

nTMs and Unary Languages: Basic techniques

How to compute $n \bmod k$

n = input length

k = integer written somewhere

- ▶ Reset the counter on track 2 each time it becomes equal to k
- ▶ When the end of the input is reached, track 2 contains $n \bmod k$
- ▶ To implement the comparison between the counter and k :
 - The value of k is kept on an extra track (track 4)
 - When the input head is moved to the right to count one more position, the representation of k is moved one position to the right in such a way that it is always aligned with the counter on track 2, to make easy the comparison
- ▶ The total time is $O(n \log k)$

nTMs and Unary Languages: Basic techniques

How to compute $n \bmod k$

n = input length

k = integer written somewhere

- ▶ Reset the counter on track 2 each time it becomes equal to k
- ▶ When the end of the input is reached, track 2 contains $n \bmod k$
- ▶ To implement the comparison between the counter and k :
 - The value of k is kept on an extra track (track 4)
 - When the input head is moved to the right to count one more position, the representation of k is moved one position to the right in such a way that it is always aligned with the counter on track 2, to make easy the comparison
- ▶ The total time is $O(n \log k)$

nTMs and Unary Languages: Basic techniques

How to compute $n \bmod k$

n = input length

k = integer written somewhere

- ▶ Reset the counter on track 2 each time it becomes equal to k
- ▶ When the end of the input is reached, track 2 contains $n \bmod k$
- ▶ To implement the comparison between the counter and k :
 - The value of k is kept on an extra track (track 4)
 - When the input head is moved to the right to count one more position, the representation of k is moved one position to the right in such a way that it is always aligned with the counter on track 2, to make easy the comparison
- ▶ The total time is $O(n \log k)$

A Unary Language Accepted in Weak Time $O(n \log \log n)$

- ▶ For each integer n let

$q(n) :=$ the smallest integer that does not divide n

- ▶ We consider the language

$$L = \{a^n \mid q(n) \text{ is not a power of } 2\}$$

- ▶ L is recognized by the following nondeterministic algorithm:

[Meregheiti '08]

```
input  $a^n$ 
guess an integer  $s, s > 1$ 
guess an integer  $t, 2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
else reject
```

A Unary Language Accepted in Weak Time $O(n \log \log n)$

- ▶ For each integer n let

$q(n) :=$ the smallest integer that does not divide n

- ▶ We consider the language

$$L = \{a^n \mid q(n) \text{ is not a power of } 2\}$$

- ▶ L is recognized by the following nondeterministic algorithm:

[Meregheiti '08]

input a^n

guess an integer $s, s > 1$

guess an integer $t, 2^s < t < 2^{s+1}$

if $n \bmod 2^s = 0$ and $n \bmod t \neq 0$ then accept
else reject

A Unary Language Accepted in Weak Time $O(n \log \log n)$

- ▶ For each integer n let

$q(n) :=$ the smallest integer that does not divide n

- ▶ We consider the language

$L = \{a^n \mid q(n) \text{ is not a power of } 2\}$

- ▶ L is recognized by the following nondeterministic algorithm:

[Meregghetti '08]

input a^n

guess an integer s , $s > 1$

guess an integer t , $2^s < t < 2^{s+1}$

if $n \bmod 2^s = 0$ and $n \bmod t \neq 0$ then accept
else reject

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
    else reject
```

Implementation and complexity:

- ▶ Two extra tracks (track 5 and 6) are used to guess 2^s and t (linear time)

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
    else reject
```

Implementation and complexity:

- ▶ Two extra tracks (track 5 and 6) are used to guess 2^s and t (linear time)
- ▶ Using the previous technique, $n \bmod 2^s$ and $n \bmod t$ are computed (time $O(n \log t)$)

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
else reject
```

Implementation and complexity:

- ▶ Two extra tracks (track 5 and 6) are used to guess 2^s and t (linear time)
- ▶ Using the previous technique, $n \bmod 2^s$ and $n \bmod t$ are computed (time $O(n \log t)$)
- ▶ Depending on the outcomes, the input is accepted or rejected

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
    else reject
```

Implementation and complexity:

- ▶ Two extra tracks (track 5 and 6) are used to guess 2^s and t (linear time)
- ▶ Using the previous technique, $n \bmod 2^s$ and $n \bmod t$ are computed (time $O(n \log t)$)
- ▶ Depending on the outcomes, the input is accepted or rejected
- ▶ The overall time of a computation is $O(n \log t)$

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
    else reject
```

Implementation and complexity:

- ▶ Two extra tracks (track 5 and 6) are used to guess 2^s and t (linear time)
- ▶ Using the previous technique, $n \bmod 2^s$ and $n \bmod t$ are computed (time $O(n \log t)$)
- ▶ Depending on the outcomes, the input is accepted or rejected
- ▶ The overall time of a computation is $O(n \log t)$
- ▶ Weak measure: it is enough to find a bound for *one accepting* computation, namely for a t which leads to acceptance

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
    else reject
```

Implementation and complexity:

- ▶ Using the previous technique, $n \bmod 2^s$ and $n \bmod t$ are computed (time $O(n \log t)$)
- ▶ Depending on the outcomes, the input is accepted or rejected
- ▶ The overall time of a computation is $O(n \log t)$
- ▶ Weak measure: it is enough to find a bound for *one accepting* computation, namely for a t which leads to acceptance
- ▶ We can take $t = q(n)$

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
    else reject
```

Implementation and complexity:

- ▶ Depending on the outcomes, the input is accepted or rejected
- ▶ The overall time of a computation is $O(n \log t)$
- ▶ Weak measure: it is enough to find a bound for *one accepting* computation, namely for a t which leads to acceptance
- ▶ We can take $t = q(n)$
- ▶ $q(n) = O(\log n)$

[Alt&Mehlhorn '75]

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
    else reject
```

Implementation and complexity:

- ▶ The overall time of a computation is $O(n \log t)$
- ▶ Weak measure: it is enough to find a bound for *one accepting* computation, namely for a t which leads to acceptance
- ▶ We can take $t = q(n)$
- ▶ $q(n) = O(\log n)$ [Alt&Mehlhorn '75]
- ▶ The time is $O(n \log \log n)$

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
else reject
```

Implementation and complexity:

- ▶ The overall time of a computation is $O(n \log t)$
- ▶ Weak measure: it is enough to find a bound for *one accepting* computation, namely for a t which leads to acceptance
- ▶ We can take $t = q(n)$
- ▶ $q(n) = O(\log n)$ [Alt&Mehlhorn '75]
- ▶ The time is $O(n \log \log n)$
- ▶ With a similar argument, we can prove $c(n) = O(\log \log n)$

A Unary Language Accepted in Weak Time $O(n \log \log n)$

```
input  $a^n$ 
guess an integer  $s$ ,  $s > 1$ 
guess an integer  $t$ ,  $2^s < t < 2^{s+1}$ 
if  $n \bmod 2^s = 0$  and  $n \bmod t \neq 0$  then accept
    else reject
```

Implementation and complexity:

- ▶ Weak measure: it is enough to find a bound for *one accepting* computation, namely for a t which leads to acceptance
- ▶ We can take $t = q(n)$
- ▶ $q(n) = O(\log n)$ [Alt&Mehlhorn '75]
- ▶ The time is $O(n \log \log n)$
- ▶ With a similar argument, we can prove $c(n) = O(\log \log n)$

A Unary Language Accepted in Weak Time $O(n \log \log n)$

$L = \{a^n \mid \text{the smallest integer which does not divide } n$
 $\text{is not a power of } 2\}$

We have proved the following:

Theorem ([P.'09])

L is accepted by a one-tape nondeterministic machine with

- ▶ $t(n) = O(n \log \log n)$
- ▶ $c(n) = O(\log \log n)$

under the weak measure

More on L : Space Complexity

The language L and its complement have been widely studied in the literature.

Some results concerning space:

- ▶ L^c is accepted by a dTM with a separate worktape, using the minimum amount of space $O(\log \log n)$ [Alt&Mehlhorn '75]
- ▶ For L we can even do better:
 L is accepted by a *one-way* nTM with a separate worktape, using the minimum amount of space $O(\log \log n)$, under the weak measure [Mereghetti '08]

L seems a good example of nonregular language with "low" complexity

More on L : Space Complexity

The language L and its complement have been widely studied in the literature.

Some results concerning space:

- ▶ L^c is accepted by a dTM with a separate worktape, using the minimum amount of space $O(\log \log n)$
[Alt&Mehlhorn '75]
- ▶ For L we can even do better:
 L is accepted by a *one-way* nTM with a separate worktape, using the minimum amount of space $O(\log \log n)$, under the weak measure [Mereghetti '08]

L seems a good example of nonregular language with “low” complexity

More on L : Space Complexity

The language L and its complement have been widely studied in the literature.

Some results concerning space:

- ▶ L^c is accepted by a dTM with a separate worktape, using the minimum amount of space $O(\log \log n)$
[Alt&Mehlhorn '75]
- ▶ For L we can even do better:
 L is accepted by a *one-way* nTM with a separate worktape, using the minimum amount of space $O(\log \log n)$, under the weak measure [Meregheiti '08]

L seems a good example of nonregular language with “low” complexity

More on L : Space Complexity

The language L and its complement have been widely studied in the literature.

Some results concerning space:

- ▶ L^c is accepted by a dTM with a separate worktape, using the minimum amount of space $O(\log \log n)$
[Alt&Mehlhorn '75]
- ▶ For L we can even do better:
 L is accepted by a *one-way* nTM with a separate worktape, using the minimum amount of space $O(\log \log n)$, under the weak measure [Mereghetti '08]

L seems a good example of nonregular language with “low” complexity

Final Remarks

Closing Remarks

We considered the “border” between regular and nonregular languages, wrt to the time $t(n)$ and the length of crossing sequences $c(n)$

Similar investigations can be have been done (even for different classes of languages) wrt other resources:

- ▶ Space e.g. [Szepietowski '94, Mereghetti '08]
- ▶ Head reversals e.g. [Bertoni&Mereghetti&P.'94]
- ▶ Return complexity or Active visits [Wechsung '75, Chytil '76]
- ▶ Dual return complexity [Hibbard '67]
- ▶ ...

Closing Remarks

We considered the “border” between regular and nonregular languages, wrt to the time $t(n)$ and the length of crossing sequences $c(n)$

Similar investigations can be have been done (even for different classes of languages) wrt other resources:

- ▶ Space e.g. [Szepietowski '94, Mereghetti '08]
- ▶ Head reversals e.g. [Bertoni&Mereghetti&P. '94]
- ▶ Return complexity or Active visits [Wechsung '75, Chytil '76]
- ▶ Dual return complexity [Hibbard '67]
- ▶ ...

Closing Remarks

We considered the “border” between regular and nonregular languages, wrt to the time $t(n)$ and the length of crossing sequences $c(n)$

Similar investigations can be have been done (even for different classes of languages) wrt other resources:

- ▶ **Space** e.g. [Szepietowski '94, Mereghetti '08]
- ▶ Head reversals e.g. [Bertoni&Mereghetti&P.'94]
- ▶ Return complexity or Active visits [Wechsung '75, Chytil '76]
- ▶ Dual return complexity [Hibbard '67]
- ▶ ...

Closing Remarks

We considered the “border” between regular and nonregular languages, wrt to the time $t(n)$ and the length of crossing sequences $c(n)$

Similar investigations can be have been done (even for different classes of languages) wrt other resources:

- ▶ Space e.g. [Szepietowski '94, Mereghetti '08]
- ▶ Head reversals e.g. [Bertoni&Mereghetti&P.'94]
- ▶ Return complexity or Active visits [Wechsung '75, Chytil '76]
- ▶ Dual return complexity [Hibbard '67]
- ▶ ...

Closing Remarks

We considered the “border” between regular and nonregular languages, wrt to the time $t(n)$ and the length of crossing sequences $c(n)$

Similar investigations can be have been done (even for different classes of languages) wrt other resources:

- ▶ Space e.g. [Szepietowski '94, Mereghetti '08]
- ▶ Head reversals e.g. [Bertoni&Mereghetti&P.'94]
- ▶ Return complexity or Active visits [Wechsung '75, Chytil '76]
- ▶ Dual return complexity [Hibbard '67]
- ▶ ...

Closing Remarks

We considered the “border” between regular and nonregular languages, wrt to the time $t(n)$ and the length of crossing sequences $c(n)$

Similar investigations can be have been done (even for different classes of languages) wrt other resources:

- ▶ Space e.g. [Szepietowski '94, Mereghetti '08]
- ▶ Head reversals e.g. [Bertoni&Mereghetti&P.'94]
- ▶ Return complexity or Active visits [Wechsung '75, Chytil '76]
- ▶ Dual return complexity [Hibbard '67]
- ▶ ...

Closing Remarks

We considered the “border” between regular and nonregular languages, wrt to the time $t(n)$ and the length of crossing sequences $c(n)$

Similar investigations can be have been done (even for different classes of languages) wrt other resources:

- ▶ Space e.g. [Szepietowski '94, Mereghetti '08]
- ▶ Head reversals e.g. [Bertoni&Mereghetti&P.'94]
- ▶ Return complexity or Active visits [Wechsung '75, Chytil '76]
- ▶ Dual return complexity [Hibbard '67]
- ▶ ...

Thank you for your attention!