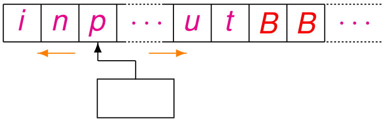
Nondeterministic One-Tape Off-Line Turing Machines

Giovanni Pighizzini

Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano
ITALY

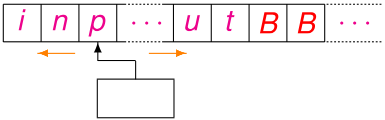
Prague – March 28th, 2009

Machine model



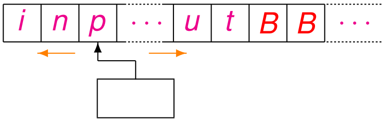
- *Finite state control*
- *Semi-infinite tape* which contains, at the beginning of the computation:
  - the input string, on its part of the tape
  - the blank symbol, in the remaining squares

Machine model



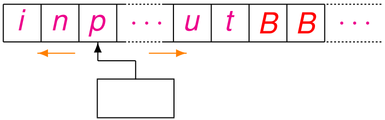
- *Finite state control*
- *Semi-infinite tape* which contains, at the beginning of the computation:
  - the input string, on its part of the tape
  - the blank symbol, in the remaining squares
- According to the *transition function* at each step the machine:
  - changes its internal state
  - writes a nonblank symbol on the scanned tape square
  - moves the head either to the left, or to the right, or keeps it on the same square
- In *accepting* and *rejecting* states the computation stops

Machine model



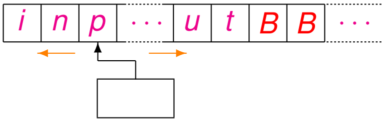
- *Finite state control*
- *Semi-infinite tape* which contains, at the beginning of the computation:
  - the input string, on its part of the tape
  - the blank symbol, in the remaining squares
- According to the *transition function* at each step the machine:
  - changes its internal state
  - writes a nonblank symbol on the scanned tape square
  - moves the head either to the left, or to the right, or keeps it on the same square
- In *accepting* and *rejecting states* the computation stops.

Machine model



- *Finite state control*
- *Semi-infinite tape* which contains, at the beginning of the computation:
  - the input string, on its part of the tape
  - the blank symbol, in the remaining squares
- According to the *transition function* at each step the machine:
  - changes its internal state
  - writes a nonblank symbol on the scanned tape square
  - moves the head either to the left, or to the right, or keeps it on the same square
- In *accepting* and *rejecting states* the computation stops.

Machine model



- *Finite state control*
- *Semi-infinite tape* which contains, at the beginning of the computation:
  - the input string, on its part of the tape
  - the blank symbol, in the remaining squares
- According to the *transition function* at each step the machine:
  - changes its internal state
  - writes a nonblank symbol on the scanned tape square
  - moves the head either to the left, or to the right, or keeps it on the same square
- In *accepting* and *rejecting states* the computation stops.

Machine model

- All the machines we consider in the following are one-tape off-line
- dTM means *one-tape off-line deterministic Turing machine*
- nTM means *one-tape off-line nondeterministic Turing machine*

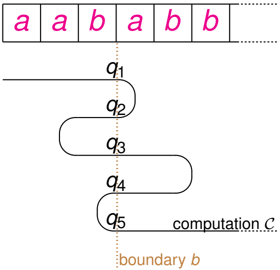
Machine model

- All the machines we consider in the following are one-tape off-line
- dTM means *one-tape off-line deterministic Turing machine*
- nTM means *one-tape off-line nondeterministic Turing machine*

Machine model

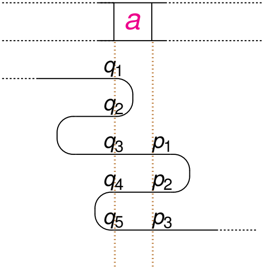
- All the machines we consider in the following are one-tape off-line
- dTM means *one-tape off-line deterministic Turing machine*
- nTM means *one-tape off-line nondeterministic Turing machine*

Crossing sequences



The *crossing sequence* of a computation $\mathcal{C}$ at a boundary b between two tape squares is the sequence of the states $(q_1, \ldots, q_k)$ s.t. q_i is the state when the boundary b is crossed for the ith time.

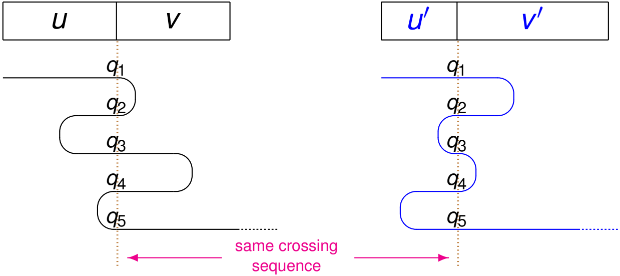
Crossing sequences: compatibility



Given two finite crossing sequences $(q_1, \ldots, q_k)$ and $(p_1, \ldots, p_h)$, it is possible to verify whether or not they are *compatible* with respect to an input symbol a,
i.e., $(q_1, \ldots, q_k)$ and $(p_1, \ldots, p_h)$ can be, in some computation, the crossing sequence at the left boundary and at the right boundary of a tape square which initially contains the symbol a.

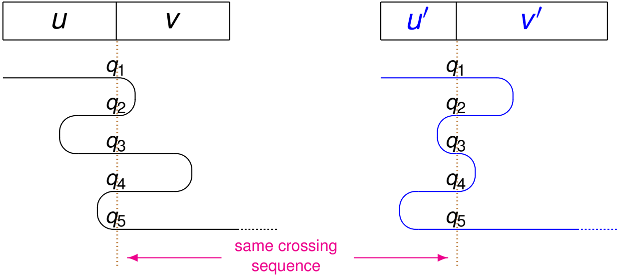
Crossing sequences: “cut–and–paste”

Given:

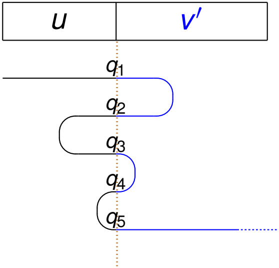


Crossing sequences: “cut–and–paste”

Given:

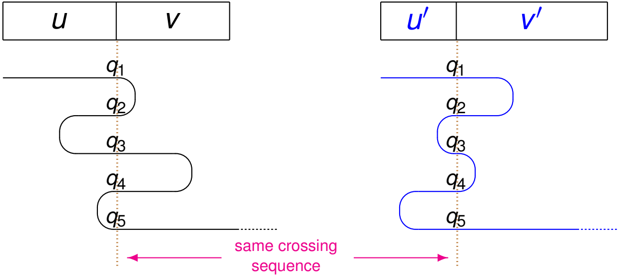


We get:

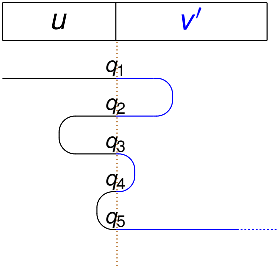


Crossing sequences: “cut–and–paste”

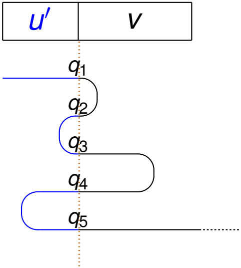
Given:



We get:



and



Complexity measures

Let $\mathcal{C}$ be a computation of a TM. We consider:

- The *time* $t(\mathcal{C})$, namely the number of moves in $\mathcal{C}$.
- The *length of the crossing sequences* $c(\mathcal{C})$, namely the maximal length of the crossing sequences defined by $\mathcal{C}$.

nTMs can have many different computations for a same input string

How to define

- $t(x)$ and $c(x)$ for an input x and,
- $t(n)$ and $c(n)$ for an input length n?

Let $\mathcal{C}$ be a computation of a TM. We consider:

- The *time* $t(\mathcal{C})$, namely the number of moves in $\mathcal{C}$.
- The *length of the crossing sequences* $c(\mathcal{C})$, namely the maximal length of the crossing sequences defined by $\mathcal{C}$.

nTMs can have many different computations for a same input string

How to define

- $t(x)$ and $c(x)$ for an input x and,
- $t(n)$ and $c(n)$ for an input length n?

Complexity measures

Let $\mathcal{C}$ be a computation of a TM. We consider:

- The *time* $t(\mathcal{C})$, namely the number of moves in $\mathcal{C}$.
- The *length of the crossing sequences* $c(\mathcal{C})$, namely the maximal length of the crossing sequences defined by $\mathcal{C}$.

nTMs can have many different computations for a same input string

How to define

- $t(x)$ and $c(x)$ for an input x and,
- $t(n)$ and $c(n)$ for an input length n?

Complexity measures

Let $\mathcal{C}$ be a computation of a TM. We consider:

- The *time* $t(\mathcal{C})$, namely the number of moves in $\mathcal{C}$.
- The *length of the crossing sequences* $c(\mathcal{C})$, namely the maximal length of the crossing sequences defined by $\mathcal{C}$.

nTMs can have many different computations for a same input string

How to define

- $t(x)$ and $c(x)$ for an input x and,
- $t(n)$ and $c(n)$ for an input length n?

Complexity measures: strong vs. weak measures

Let $r \in \{t, c\}$ (time or length of crossing sequences)

- **strong measure:** costs of *all computations* on x

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

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

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

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

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

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

Complexity measures: strong vs. weak measures

Let $r \in \{t, c\}$ (time or length of crossing sequences)

- strong measure: costs of *all computations* on x

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

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

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

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

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

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

Complexity measures: strong vs. weak measures

Let $r \in \{t, c\}$ (time or length of crossing sequences)

- strong measure: costs of *all computations* on x

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

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

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

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

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

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

Complexity measures: strong vs. weak measures

Let $r \in \{t, c\}$ (time or length of crossing sequences)

- strong measure: costs of *all computations* on x

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

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

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

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

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

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

Complexity measures: strong vs. weak measures

Let $r \in \{t, c\}$ (time or length of crossing sequences)

- strong measure: costs of *all computations* on x

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

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

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

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

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

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

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 Off-Line TMs: Simple bounds

For the *length of the crossing sequences*, the following result can be easily proved:

Theorem

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

Idea of the proof:

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

One-Tape Off-Line TMs: Simple bounds

For the *length of the crossing sequences*, the following result can be easily proved:

Theorem

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

Idea of the proof:

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

One-Tape Off-Line TMs: Simple bounds

For the *time*, the following result can be easily proved:

Theorem

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

Idea of the proof:

- Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$.
- Given $x \in L$ and $|x| \geq n_0$, there is a computation $\mathcal{C}$ that accepts x just reading at most the first $t(|x|)$ symbols of x.

One-Tape Off-Line TMs: Simple bounds

For the *time*, the following result can be easily proved:

Theorem

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

Idea of the proof:

- Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$.
- Given $x \in L$ and $|x| \geq n_0$, there is a computation $\mathcal{C}$ that accepts x just reading at most the first $t(x)$ symbols of x.
- $\mathcal{C}$ should accept the prefix x' of x of length $t(x)$.
-

One-Tape Off-Line TMs: Simple bounds

For the *time*, the following result can be easily proved:

Theorem

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

Idea of the proof:

- Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$.
- Given $x \in L$ and $|x| \geq n_0$, there is a computation $\mathcal{C}$ that accepts x just reading at most the first $t(x)$ symbols of x.
- $\mathcal{C}$ should accept the prefix x' of x of length $t(x)$.
- We can prove that $|x'| < n_0$
-

One-Tape Off-Line TMs: Simple bounds

For the *time*, the following result can be easily proved:

Theorem

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

Idea of the proof:

- Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$.
- Given $x \in L$ and $|x| \geq n_0$, there is a computation $\mathcal{C}$ that accepts x just reading at most the first $t(x)$ symbols of x.
- $\mathcal{C}$ should accept the prefix x' of x of length $t(x)$.
- We can prove that $|x'| < n_0$
- Hence, the membership to L can be decided just reading an input prefix of length at most n_0

One-Tape Off-Line TMs: Simple bounds

For the *time*, the following result can be easily proved:

Theorem

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

Idea of the proof:

- Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$.
- Given $x \in L$ and $|x| \geq n_0$, there is a computation $\mathcal{C}$ that accepts x just reading at most the first $t(x)$ symbols of x.
- $\mathcal{C}$ should accept the prefix x' of x of length $t(x)$.
- We can prove that $|x'| < n_0$
- Hence, the membership to L can be decided just testing an input prefix of length at most n_0

One-Tape Off-Line TMs: Simple bounds

For the *time*, the following result can be easily proved:

Theorem

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

Idea of the proof:

- Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$.
- Given $x \in L$ and $|x| \geq n_0$, there is a computation $\mathcal{C}$ that accepts x just reading at most the first $t(x)$ symbols of x.
- $\mathcal{C}$ should accept the prefix x' of x of length $t(x)$.
- We can prove that $|x'| < n_0$
- Hence, the membership to L can be decided just testing an input prefix of length at most n_0

One-Tape Off-Line TMs: Simple bounds

For the *time*, the following result can be easily proved:

Theorem

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

Idea of the proof:

- Let n_0 s.t. $t(n) < n$, for each $n \geq n_0$.
- Given $x \in L$ and $|x| \geq n_0$, there is a computation $\mathcal{C}$ that accepts x just reading at most the first $t(x)$ symbols of x.
- $\mathcal{C}$ should accept the prefix x' of x of length $t(x)$.
- We can prove that $|x'| < n_0$
- Hence, the membership to L can be decided just testing an input prefix of length at most n_0

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 machine (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 machine (deterministic, nondeterministic)

Deterministic machines (strong measure)

- Hennie (1965) proved that

  one-tape off-line *deterministic* machines working in linear time accept 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 nonregular languages the time $t(n)$ must grow at least as $n \log n$.

Deterministic machines (strong measure)

- Hennie (1965) proved that

  one-tape off-line *deterministic* machines working in linear time accept 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 nonregular languages the time $t(n)$ must grow at least as $n \log n$.

- This is optimal because there are languages matching this bound

Deterministic machines (strong measure)

- Hennie (1965) proved that

  one-tape off-line *deterministic* machines working in linear time accept 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 nonregular languages the time $t(n)$ must grow at least as $n \log n$.

- This is optimal because there are languages matching this bound.

Deterministic machines (strong measure)

- Hennie (1965) proved that

  one-tape off-line *deterministic* machines working in linear time accept 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 nonregular languages the time $t(n)$ must grow at least as $n \log n$.

- This is optimal because there are languages matching this bound.

Deterministic machines (strong measure)

- Hennie (1965) proved that

  one-tape off-line *deterministic* machines working in linear time accept 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 nonregular languages the time $t(n)$ must grow at least as $n \log n$.

- This is optimal because there are languages matching this bound.

Nondeterministic machines

- Wagner and Wechsung (1986) gave an example of nonregular language accepted by a nondeterministic machine in time $o(n \log n)$
- Michel (1991) showed that

  there exists a nonregular language accepted in linear time by a nondeterministic machine

- However, both there results refer to the weak measure

Nondeterministic machines

- Wagner and Wechsung (1986) gave an example of nonregular language accepted by a nondeterministic machine in time $o(n \log n)$
- Michel (1991) showed that

  there exists a nonregular language accepted in linear time by a nondeterministic machine

- However, both there results refer to the weak measure
- For the strong measure, Tadaki, Yamakami and Lin (2004) proved the that the $n \log n$ time bound for nonregular language recognition holds *even in the case of nondeterministic machines*

Nondeterministic machines

- Wagner and Wechsung (1986) gave an example of nonregular language accepted by a nondeterministic machine in time $o(n \log n)$
- Michel (1991) showed that

  there exists a nonregular language accepted in linear time by a nondeterministic machine

- However, both there results refer to the weak measure
- For the strong measure, Tadaki, Yamakami and Lin (2004) proved the that the $n \log n$ time bound for nonregular language recognition holds *even in the case of nondeterministic machines*
- We recently extended the last result to the accept measure

Nondeterministic machines

- Wagner and Wechsung (1986) gave an example of nonregular language accepted by a nondeterministic machine in time $o(n \log n)$
- Michel (1991) showed that

  there exists a nonregular language accepted in linear time by a nondeterministic machine

- However, both there results refer to the weak measure
- For the strong measure, Tadaki, Yamakami and Lin (2004) proved the that the $n \log n$ time bound for nonregular language recognition holds *even in the case of nondeterministic machines*
- We recently extended the last result to the accept measure.

Nondeterministic machines

- Wagner and Wechsung (1986) gave an example of nonregular language accepted by a nondeterministic machine in time $o(n \log n)$
- Michel (1991) showed that

  there exists a nonregular language accepted in linear time by a nondeterministic machine

- However, both there results refer to the weak measure
- For the strong measure, Tadaki, Yamakami and Lin (2004) proved the that the $n \log n$ time bound for nonregular language recognition holds *even in the case of nondeterministic machines*
- We recently extended the last result to the accept measure.

Nondeterministic machines

- Wagner and Wechsung (1986) gave an example of nonregular language accepted by a nondeterministic machine in time $o(n \log n)$
- Michel (1991) showed that

  there exists a nonregular language accepted in linear time by a nondeterministic machine
- However, both there results refer to the weak measure
- For the strong measure, Tadaki, Yamakami and Lin (2004) proved the that the $n \log n$ time bound for nonregular language recognition holds *even in the case of nondeterministic machines*
- We recently extended the last result to the accept measure.

Lower bounds for the accept measure

For the accept measure we proved [Pighizzini, 2009]:

Let M be a nTM accepting a language L using

- crossing sequences of length bounded by $c(n)$
- time $t(n)$

under the accept measure. Then:

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

Lower bounds for the accept measure

For the accept measure we proved [Pighizzini, 2009]:

Let M be a nTM accepting a language L using

- crossing sequences of length bounded by $c(n)$
- time $t(n)$

under the accept measure. Then:

- If $c(n) = o(\log n)$ then $c(n) = O(1)$ and then 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 the accept measure

For the accept measure we proved [Pighizzini, 2009]:

Let M be a nTM accepting a language L using

- crossing sequences of length bounded by $c(n)$
- time $t(n)$

under the accept measure. Then:

- If $c(n) = o(\log n)$ then $c(n) = O(1)$ and then 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 the accept measure

For the accept measure we proved [Pighizzini, 2009]:

Let M be a nTM accepting a language L using

- crossing sequences of length bounded by $c(n)$
- time $t(n)$

under the accept measure. Then:

- If $c(n) = o(\log n)$ then $c(n) = O(1)$ and then 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 the accept measure

For the accept measure we proved [Pighizzini, 2009]:

Let M be a nTM accepting a language L using

- crossing sequences of length bounded by $c(n)$
- time $t(n)$

under the accept measure. Then:

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

The proof uses counting arguments based on the following lemma:

Lemma

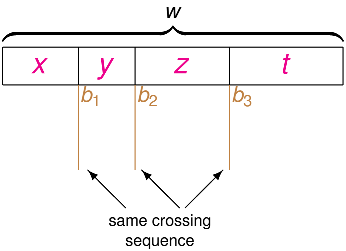
Given

- *an integer k*
- *an input string w accepted by a computation $\mathcal{C}$ with $c(\mathcal{C}) = k$.*

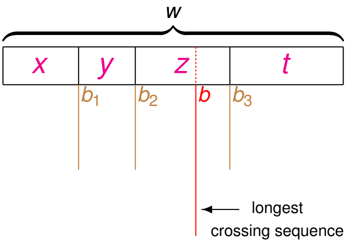
If a same crossing sequence occurs in $\mathcal{C}$ at three different boundaries of the input zone of the tape, then there is another string w' s.t.

- $|w| < |w'|$
- *w' is accepted by a computation $\mathcal{C}'$ with $c(\mathcal{C}') = k$*

Sketch of the proof

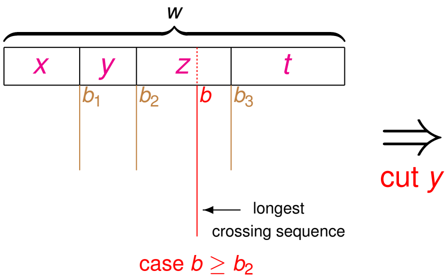


Sketch of the proof

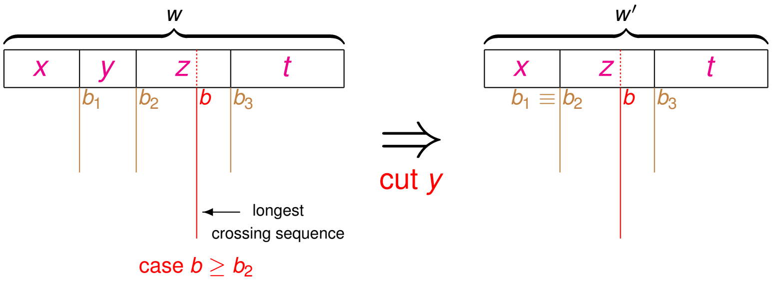


case $b \geq b_2$

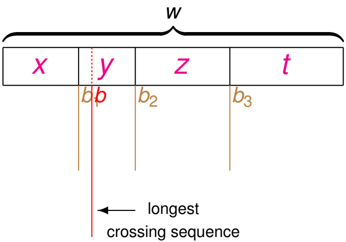
Sketch of the proof



Sketch of the proof

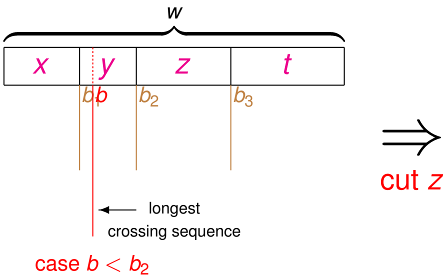


Sketch of the proof

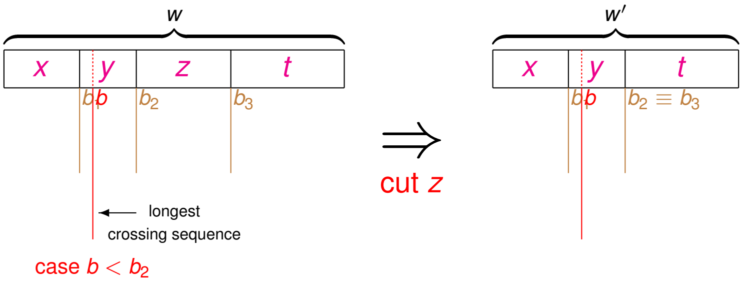


case $b < b_2$

Sketch of the proof



Sketch of the proof



Does it is possible to extend the lower bound for the accept measure to the weak one?

- By the above mentioned result of Michel (1991), for the time the answer is negative
- For the length of the crossing sequences a $\log \log n$ lower bound has been proved

The weak measure

Does it is possible to extend the lower bound for the accept measure to the weak one?

- By the above mentioned result of Michel (1991), for the time the answer is negative
- For the length of the crossing sequences a $\log\log n$ lower bound has been proved [Pighizzini, 2009]

Does it is possible to extend the lower bound for the accept measure to the weak one?

- By the above mentioned result of Michel (1991), for the time the answer is negative
- For the length of the crossing sequences a $\log\log n$ lower bound has been proved [Pighizzini, 2009]

$c(n)$ grows at least as $\log\log n$, weak measure

Sketch of the proof

- $L :=$ language accepted by the given machine M
- $q :=$ number of states of M
- for $n \geq 1$:
  $N_n :=$ nfa whose states are the crossing sequences of length $\leq c(n)$ and whose transitions are defined according to the "compatibility" relation
-

$c(n)$ grows at least as $\log\log n$, weak measure

Sketch of the proof

- $L :=$ language accepted by the given machine M
- $q :=$ number of states of M
- for $n \geq 1$:
  $N_n :=$ nfa whose states are the crossing sequences of length $\leq c(n)$ and whose transitions are defined according to the "compatibility" relation
- N_n agrees with M on strings of length $\leq n$

$c(n)$ grows at least as $\log\log n$, weak measure

Sketch of the proof

- $L :=$ language accepted by the given machine M
- $q :=$ number of states of M
- for $n \geq 1$:
  $N_n :=$ nfa whose states are the crossing sequences of length $\leq c(n)$ and whose transitions are defined according to the "compatibility" relation
- N_n agrees with M on strings of length $\leq n$
- $A_n :=$ dfa equivalent to N_n; it has $\leq 2^{q^{c(n)+1}}$ states

$c(n)$ grows at least as $\log \log n$, weak measure

Sketch of the proof

- $L :=$ language accepted by the given machine M
- $q :=$ number of states of M
- for $n \geq 1$:
  $N_n :=$ nfa whose states are the crossing sequences of length $\leq c(n)$ and whose transitions are defined according to the "compatibility" relation
- N_n agrees with M on strings of length $\leq n$
- $A_n :=$ dfa equivalent to N_n, it has $\leq 2^{q^{c(n)+1}}$ states

$c(n)$ grows at least as $\log\log n$, weak measure

Sketch of the proof

- $L :=$ language accepted by the given machine M
- $q :=$ number of states of M
- for $n \geq 1$:
  $N_n :=$ nfa whose states are the crossing sequences of length $\leq c(n)$ and whose transitions are defined according to the "compatibility" relation
- N_n agrees with M on strings of length $\leq n$
- $A_n :=$ dfa equivalent to N_n, it has $\leq 2^{q^{c(n)+1}}$ states
- By a result of Karp (1967), if L is not regular, then the number of the states of A_n must be $\geq \frac{n+3}{2}$, i.o.
- Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, i.o., implying that

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

for some constant d and infinitely many n

$c(n)$ grows at least as $\log \log n$, weak measure

Sketch of the proof

- $L :=$ language accepted by the given machine M
- $q :=$ number of states of M
- for $n \geq 1$:
  $N_n :=$ nfa whose states are the crossing sequences of length $\leq c(n)$ and whose transitions are defined according to the “compatibility” relation
- N_n agrees with M on strings of length $\leq n$
- $A_n :=$ dfa equivalent to N_n, it has $\leq 2^{q^{c(n)+1}}$ states
- By a result of Karp (1967), if L is not regular, then the number of the states of A_n must be $\geq \frac{n+3}{2}$, i.o.
- Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, i.o., implying that

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

for some constant d and infinitely many n.

$c(n)$ grows at least as $\log\log n$, weak measure

Sketch of the proof

- $L :=$ language accepted by the given machine M
- $q :=$ number of states of M
- for $n \geq 1$:
  $N_n :=$ nfa whose states are the crossing sequences of length $\leq c(n)$ and whose transitions are defined according to the "compatibility" relation
- N_n agrees with M on strings of length $\leq n$
- $A_n :=$ dfa equivalent to N_n, it has $\leq 2^{q^{c(n)+1}}$ states
- By a result of Karp (1967), if L is not regular, then the number of the states of A_n must be $\geq \frac{n+3}{2}$, i.o.
- Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, i.o., implying that

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

for some constant d and infinitely many n.

$c(n)$ grows at least as $\log \log n$, weak measure

Sketch of the proof

- $L :=$ language accepted by the given machine M
- $q :=$ number of states of M
- for $n \geq 1$:
  $N_n :=$ nfa whose states are the crossing sequences of length $\leq c(n)$ and whose transitions are defined according to the "compatibility" relation
- N_n agrees with M on strings of length $\leq n$
- $A_n :=$ dfa equivalent to N_n, it has $\leq 2^{q^{c(n)+1}}$ states
- By a result of Karp (1967), if L is not regular, then the number of the states of A_n must be $\geq \frac{n+3}{2}$, i.o.
- Hence $2^{q^{c(n)+1}} \geq \frac{n+3}{2}$, i.o., implying that

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

for some constant d and infinitely many n.

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 (2004)

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 nTM

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 dTM, 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 of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form left to right over the input segment and marks off the 1st, 3th, 5th, etc. unmarked squares

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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	input a^{12}

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM *M* accepting *L* which works as follows:

- At the beginning all the input cells are "unmarked"
- *M* sweeps form 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

$$\Downarrow$$

X	a	X	a	X	a	X	a	X	a	X	a

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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

$$\Downarrow$$

X	X	X	a	X	X	X	a	X	X	X	a

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are “unmarked”
- M sweeps form 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

$$\Downarrow$$

X	X	X	X	X	X	X	a	X	X	X	X

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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

input a^8

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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

$$\Downarrow$$

X	a	X	a	X	a	X	a

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are “unmarked”
- M sweeps form 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

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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

$$\Downarrow$$

X	X	X	a	X	X	X	a

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are "unmarked"
- M sweeps form 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

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are “unmarked”
- M sweeps form 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

$$\Downarrow$$

X	X	X	X	X	X	X	a

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are “unmarked”
- M sweeps form 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

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are “unmarked”
- M sweeps form 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

$$\Downarrow$$

X	X	X	X	X	X	X	X

Optimality of the bounds

Consider the following unary language (Hartmanis, 1968)

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

We can build a dTM M accepting L which works as follows:

- At the beginning all the input cells are “unmarked”
- M sweeps form 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!

- On input a^n, the machine M makes $O(\log n)$ sweeps of the part of the tape which contains the input. Hence:

$$c(n) = O(\log n) \text{ and } t(n) = O(n \log n).$$

- M is *deterministic* and the previous bounds hold for accepting and rejecting computations: strong measure.

- This gives the optimality of all the lower bounds in the table, with the only exception of those for nTMs under the weak measure:

		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
	$c(n)$	$\log n$	$\log n$	$\log \log n$

Complexity

- On input a^n, the machine M makes $O(\log n)$ sweeps of the part of the tape which contains the input. Hence:

$$c(n) = O(\log n) \text{ and } t(n) = O(n \log n).$$

- M is *deterministic* and the previous bounds hold for accepting and rejecting computations: strong measure.

- This gives the optimality of all the lower bounds in the table, with the only exception of those for nTMs, under the weak measure:

		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
	$c(n)$	$\log n$	$\log n$	$\log \log n$

Complexity

- On input a^n, the machine M makes $O(\log n)$ sweeps of the part of the tape which contains the input. Hence:

$$c(n) = O(\log n) \text{ and } t(n) = O(n \log n).$$

- M is *deterministic* and the previous bounds hold for accepting and rejecting computations: strong measure.
- This gives the optimality of all the lower bounds in the table, with the only exception of those for nTMs, under the weak measure:

		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
	$c(n)$	$\log n$	$\log n$	$\log \log n$

- The optimality was proved by using a *unary language*.

Complexity

- On input a^n, the machine M makes $O(\log n)$ sweeps of the part of the tape which contains the input. Hence:
$$c(n) = O(\log n) \text{ and } t(n) = O(n \log n).$$
- M is *deterministic* and the previous bounds hold for accepting and rejecting computations: strong measure.
- This gives the optimality of all the lower bounds in the table, with the only exception of those for nTMs, under the weak measure:

		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
	$c(n)$	$\log n$	$\log n$	$\log \log n$

- The optimality was proved by using a *unary language*.

Complexity

- On input a^n, the machine M makes $O(\log n)$ sweeps of the part of the tape which contains the input. Hence:

$$c(n) = O(\log n) \text{ and } t(n) = O(n \log n).$$

- M is *deterministic* and the previous bounds hold for accepting and rejecting computations: strong measure.
- This gives the optimality of all the lower bounds in the table, with the only exception of those for nTMs, under the weak measure:

		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
	$c(n)$	$\log n$	$\log n$	$\log \log n$

- The optimality was proved by using a *unary language*.

nTMs and weak measure

What about the bounds for the weak measure for nTMs?

- There are nonregular languages accepted in time $O(n)$.
- Hence, in this case there is no a "gap" between regular and nonregular languages.
- The example provided by Michel (1991) strongly relies on the use of an input alphabet with more than one symbol.

nTMs and weak measure

What about the bounds for the weak measure for nTMs?

- There are nonregular languages accepted in time $O(n)$.
- Hence, in this case there is no a “gap” between regular and nonregular languages.
- The example provided by Michel (1991) strongly relies on the use of 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 $O(n)$

nTMs and weak measure

What about the bounds for the weak measure for nTMs?

- There are nonregular languages accepted in time $O(n)$.
- Hence, in this case there is no a “gap” between regular and nonregular languages.
- The example provided by Michel (1991) strongly relies on the use of 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 $O(n)$.
- We believe that such a language does not exists:

  Conjecture: If a nTM accepts a *unary* language L in time $o(n \log \log n)$ under the weak measure then L is regular.

nTMs and weak measure

What about the bounds for the weak measure for nTMs?

- There are nonregular languages accepted in time $O(n)$.
- Hence, in this case there is no a "gap" between regular and nonregular languages.
- The example provided by Michel (1991) strongly relies on the use of 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 $O(n)$.
- We believe that such a language does not exists:

  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 now show an example of unary nonregular language accepted by a nTM in weak time $O(n \log \log n)$

nTMs and weak measure

What about the bounds for the weak measure for nTMs?

- There are nonregular languages accepted in time $O(n)$.
- Hence, in this case there is no a “gap” between regular and nonregular languages.
- The example provided by Michel (1991) strongly relies on the use of 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 $O(n)$.
- We believe that such a language does not exists:

  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 now show an example of unary nonregular language accepted by a nTM in weak time $O(n \log \log n)$.

What about the bounds for the weak measure for nTMs?

- There are nonregular languages accepted in time $O(n)$.
- Hence, in this case there is no a "gap" between regular and nonregular languages.
- The example provided by Michel (1991) strongly relies on the use of 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 $O(n)$.
- We believe that such a language does not exists:

  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 now show an example of unary nonregular language accepted by a nTM in weak time $O(n \log \log n)$.

nTMs and weak measure

What about the bounds for the weak measure for nTMs?

- There are nonregular languages accepted in time $O(n)$.
- Hence, in this case there is no a "gap" between regular and nonregular languages.
- The example provided by Michel (1991) strongly relies on the use of 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 $O(n)$.
- We believe that such a language does not exists:

  Conjecture: If a nTM accepts a *unary* language L in time $o(n \log \log n)$ under the weak measure then L is regular.

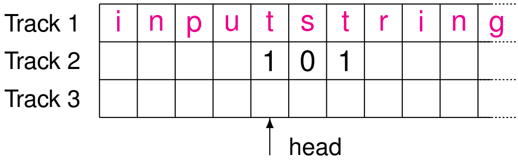
- We now show an example of unary nonregular language accepted by a nTM in weak time $O(n \log \log n)$.

Basic techniques

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

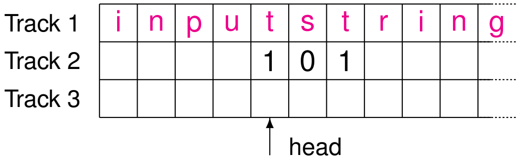
Track 1	i	n	p	u	t	s	t	r	i	n	g
Track 2	m	e	m	o	r	y	s	p	a	c	e
Track 3	m	e	m	o	r	y	s	p	a	c	e

How to count input symbols



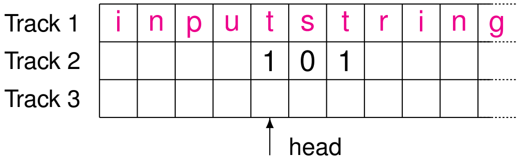
- The counter is kept on track 2, *starting from the position scanned by the tape head*
- When the head must be moved to the right, counting one more input position, the counter is incremented and shifted to the right
- This is done in $O(\log j)$ steps (where j is the value of the counter) using track 3 as an auxiliary variable

How to count input symbols



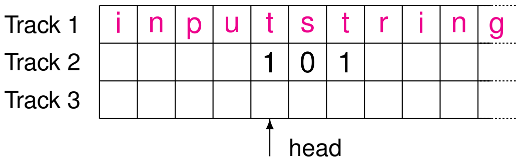
- The counter is kept on track 2, *starting from the position scanned by the tape head*
- When the head must be moved to the right, counting one more input position, the counter is incremented and shifted to the right
- This is done in $O(\log j)$ steps (where j is the value of the counter) using track 3 as an auxiliary variable.

How to count input symbols



- The counter is kept on track 2, *starting from the position scanned by the tape head*
- When the head must be moved to the right, counting one more input position, the counter is incremented and shifted to the right
- This is done in $O(\log j)$ steps (where j is the value of the counter) using track 3 as an auxiliary variable.
- In this way, k tape positions can be counted in $O(k \log k)$ moves.

How to count input symbols



- The counter is kept on track 2, *starting from the position scanned by the tape head*
- When the head must be moved to the right, counting one more input position, the counter is incremented and shifted to the right
- This is done in $O(\log j)$ steps (where j is the value of the counter) using track 3 as an auxiliary variable.
- In this way, k tape positions can be counted in $O(k \log k)$ moves.

How to count input symbols

Track 1	i	n	p	u	t	s	t	r	i	n	g
Track 2					1	0	1				
Track 3											

head

- The counter is kept on track 2, *starting from the position scanned by the tape head*
- When the head must be moved to the right, counting one more input position, the counter is incremented and shifted to the right
- This is done in $O(\log j)$ steps (where j is the value of the counter) using track 3 as an auxiliary variable.
- In this way, k tape positions can be counted in $O(k \log k)$ moves.

How to compute n mod k

($n =$ input length, $k =$ an integer written somewhere)

We adapt the previous technique:

- The counter on track 2 is reset each time it becomes equal to k.
- In this way, track 2 will finally contain n MOD k.
-

How to compute n mod k

($n =$ input length, $k =$ an integer written somewhere)

We adapt the previous technique:

- The counter on track 2 is reset each time it becomes equal to k.
- In this way, track 2 will finally contain n MOD k.
- To implement the comparison between the counter and k

How to compute n mod k

($n =$ input length, $k =$ an integer written somewhere)

We adapt the previous technique:

- The counter on track 2 is reset each time it becomes equal to k.
- In this way, track 2 will finally contain $n \bmod k$.
- To implement the comparison between the counter and k:
  - The value of k is kept on one extra track (track 4)
- The total time is $O(n \log k)$

How to compute $n \bmod k$

($n =$ input length, $k =$ an integer written somewhere)

We adapt the previous technique:

- The counter on track 2 is reset each time it becomes equal to k.
- In this way, track 2 will finally contain $n \text{ MOD } k$.
- To implement the comparison between the counter and k:
  - The value of k is kept on one extra track (track 4)
  - Its representation is shifted to the right, when the input head is moved to the right to count one more position, in such a way that the counter and k are always on the same tape segment.
- The total time is $O(n \log k)$

How to compute n mod k

($n =$ input length, $k =$ an integer written somewhere)

We adapt the previous technique:

- The counter on track 2 is reset each time it becomes equal to k.
- In this way, track 2 will finally contain $n \text{ MOD } k$.
- To implement the comparison between the counter and k:
  - The value of k is kept on one extra track (track 4)
  - Its representation is shifted to the right, when the input head is moved to the right to count one more position, in such a way that the counter and k are always on the same tape segment.
- The total time is $O(n \log k)$

How to compute n mod k

($n =$ input length, $k =$ an integer written somewhere)

We adapt the previous technique:

- The counter on track 2 is reset each time it becomes equal to k.
- In this way, track 2 will finally contain $n \text{ MOD } k$.
- To implement the comparison between the counter and k:
  - The value of k is kept on one extra track (track 4)
  - Its representation is shifted to the right, when the input head is moved to the right to count one more position, in such a way that the counter and k are always on the same tape segment.
- The total time is $O(n \log k)$

How to compute n mod k

($n =$ input length, $k =$ an integer written somewhere)

We adapt the previous technique:

- The counter on track 2 is reset each time it becomes equal to k.
- In this way, track 2 will finally contain $n \text{ MOD } k$.
- To implement the comparison between the counter and k:
  - The value of k is kept on one extra track (track 4)
  - Its representation is shifted to the right, when the input head is moved to the right to count one more position, in such a way that the counter and k are always on the same tape segment.
- The total time is $O(n \log k)$

How to compute n mod k

($n =$ input length, $k =$ an integer written somewhere)

We adapt the previous technique:

- The counter on track 2 is reset each time it becomes equal to k.
- In this way, track 2 will finally contain $n \text{ MOD } k$.
- To implement the comparison between the counter and k:
  - The value of k is kept on one extra track (track 4)
  - Its representation is shifted to the right, when the input head is moved to the right to count one more position, in such a way that the counter and k are always on the same tape segment.
- 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

$$q(n) := \text{ the smallest integer that does not divide } n$$

- We consider the language

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

- L can be recognized using the following nondeterministic algorithm (Mereghetti, 2008):

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

$$q(n) := \text{ the smallest integer that does not divide } n$$

- We consider the language

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

- L can be recognized using the following nondeterministic algorithm (Mereghetti, 2008):

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

$$q(n) := \text{ the smallest integer that does not divide } n$$

- We consider the language

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

- L can be recognized using the following nondeterministic algorithm (Mereghetti, 2008):

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

  $$q(n) := \text{ the smallest integer that does not divide } n$$

- We consider the language

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

- L can be recognized using the following nondeterministic algorithm (Mereghetti, 2008):

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

Implementation of the algorithm

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

Implementation and complexity:

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

Implementation of the algorithm

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 $O(n \log t)$)

Implementation of the algorithm

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

Implementation of the algorithm

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
- Hence, the overall time of a computation which guesses t is $O(n \log t)$

Implementation of the algorithm

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
- Hence, the overall time of a computation which guesses t is $O(n \log t)$
- Since we are using the weak measure, it is enough to find a bound for *one accepting* computation, namely for a value

Implementation of the algorithm

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
- Hence, the overall time of a computation which guesses t is $O(n \log t)$
- Since we are using the weak measure, it is enough to find a bound for *one accepting* computation, namely for a value of t which leads to the acceptance
- We can take $t = q(n)$

Implementation of the algorithm

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
- Hence, the overall time of a computation which guesses t is $O(n \log t)$
- Since we are using the weak measure, it is enough to find a bound for *one accepting* computation, namely for a value of t which leads to the acceptance
- We can take $t = q(n)$
- By a result of Alt and Mehlhorn (1975), $q(n) = O(\log n)$

Implementation of the algorithm

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:

- Hence, the overall time of a computation which guesses t is $O(n \log t)$
- Since we are using the weak measure, it is enough to find a bound for *one accepting* computation, namely for a value of t which leads to the acceptance
- We can take $t = q(n)$
- By a result of Alt and Mehlhorn (1975), $q(n) = O(\log n)$
- Hence, the time is $O(n \log \log n)$

Implementation of the algorithm

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:

- Hence, the overall time of a computation which guesses t is $O(n \log t)$
- Since we are using the weak measure, it is enough to find a bound for *one accepting* computation, namely for a value of t which leads to the acceptance
- We can take $t = q(n)$
- By a result of Alt and Mehlhorn (1975), $q(n) = O(\log n)$
- Hence, the time is $O(n \log \log n)$
- With a similar argument, we can also prove that $c(n) = O(\log \log n)$

Implementation of the algorithm

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:

- Since we are using the weak measure, it is enough to find a bound for *one accepting* computation, namely for a value of t which leads to the acceptance
- We can take $t = q(n)$
- By a result of Alt and Mehlhorn (1975), $q(n) = O(\log n)$
- Hence, the time is $O(n \log \log n)$
- With a similar argument, we can also prove that $c(n) = O(\log \log n)$

A unary language accepted in weak time $O(n \log \log n)$

Hence, we have proved the following:

Theorem ([Pighizzini, 2009])

The language L is accepted by a nTM with

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

under the weak *measure*

More on L

The language L and its complement have been widely studied in the literature. These are some results:

- L^c is accepted by a dTM with a separate worktape, using the minimum amount of space $O(\log \log n)$ [Alt and Mehlhron, 1975]
- The same space complexity can be achieved using the smallest possible number of input head reversals $O(\frac{\log n}{\log \log n})$ [Bertoni, Mereghetti, and Pighizzini, 1994]
- 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

Hence, L seems to be a good example of nonregular language with “low” complexity.

More on L

The language L and its complement have been widely studied in the literature. These are some results:

- L^c is accepted by a dTM with a separate worktape, using the minimum amount of space $O(\log\log n)$ [Alt and Mehlhron, 1975]
- The same space complexity can be achieved using the smallest possible number of input head reversals $O(\frac{\log n}{\log\log n})$ [Bertoni, Mereghetti, and Pighizzini, 1994]
- 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, 2008]

Hence, L seems to be a good example of nonregular language with "low" complexity.

More on L

The language L and its complement have been widely studied in the literature. These are some results:

- L^c is accepted by a dTM with a separate worktape, using the minimum amount of space $O(\log \log n)$ [Alt and Mehlhron, 1975]
- The same space complexity can be achieved using the smallest possible number of input head reversals $O(\frac{\log n}{\log \log n})$ [Bertoni, Mereghetti, and Pighizzini, 1994]
- 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, 2008]

Hence, L seems to be a good example of nonregular language with “low” complexity.

More on L

The language L and its complement have been widely studied in the literature. These are some results:

- L^c is accepted by a dTM with a separate worktape, using the minimum amount of space $O(\log \log n)$ [Alt and Mehlhron, 1975]
- The same space complexity can be achieved using the smallest possible number of input head reversals $O(\frac{\log n}{\log \log n})$ [Bertoni, Mereghetti, and Pighizzini, 1994]
- 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, 2008]

Hence, L seems to be a good example of nonregular language with “low” complexity.

More on L

The language L and its complement have been widely studied in the literature. These are some results:

- L^c is accepted by a dTM with a separate worktape, using the minimum amount of space $O(\log\log n)$ [Alt and Mehlhron, 1975]
- The same space complexity can be achieved using the smallest possible number of input head reversals $O(\frac{\log n}{\log\log n})$ [Bertoni, Mereghetti, and Pighizzini, 1994]
- 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, 2008]

Hence, L seems to be a good example of nonregular language with "low" complexity.

Final 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 (or have been) done (even for different classes of languages) wrt other resources:

- Space (e.g., [Szepietowski, 1994], [Mereghetti, 2008])
- Head reversals
  (for the input head)
-

Final 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 (or have been) done (even for different classes of languages) wrt other resources:

- Space (e.g., [Sziepietowski, 1994], [Mereghetti, 2008])
- Head reversals
  (for the input head [Bertoni, Mereghetti, Pighizzini, 1994])
- Return complexity or Active visit

Final 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 (or have been) done (even for different classes of languages) wrt other resources:

- Space (e.g., [Sziepietowski, 1994], [Mereghetti, 2008])
- Head reversals
  (for the input head [Bertoni, Mereghetti, Pighizzini, 1994])
- Return complexity or Active visit
  [Wechsung 1975 – Chytil, 1976]
- Dual return complexity
- ...

Final 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 (or have been) done (even for different classes of languages) wrt other resources:

- Space (e.g., [Sziepietowski, 1994], [Mereghetti, 2008])
- Head reversals
  (for the input head [Bertoni, Mereghetti, Pighizzini, 1994])
- Return complexity or Active visit
  [Wechsung 1975 – Chytil, 1976]
- Dual return complexity [Hibbard, 1968]
- ...

Final 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 (or have been) done (even for different classes of languages) wrt other resources:

- Space (e.g., [Sziepietowski, 1994], [Mereghetti, 2008])
- Head reversals
  (for the input head [Bertoni, Mereghetti, Pighizzini, 1994])
- Return complexity or Active visit
  [Wechsung 1975 – Chytil, 1976]
- Dual return complexity [Hibbard, 1968]
- ...

Final 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 (or have been) done (even for different classes of languages) wrt other resources:

- Space (e.g., [Sziepietowski, 1994], [Mereghetti, 2008])
- Head reversals
  (for the input head [Bertoni, Mereghetti, Pighizzini, 1994])
- Return complexity or Active visit
  [Wechsung 1975 – Chytil, 1976]
- Dual return complexity [Hibbard, 1968]
- ...

Final remarks

We used of tape tracks, namely large alphabets, big-Oh notation, etc.

It is interesting to investigate what happens when we put stronger restrictions on the resources.

An interesting result in this line has been recently proved by Hemaspaandra, Mukherji and Tantau (2005):

> *Context-free languages are accepted by Turing machines with absolutely no space overhead*

The work space of the machine is:

- the finite state control
- the space that initially contains the input, with the restriction that *only a binary alphabet can be used on it*

Final remarks

We used of tape tracks, namely large alphabets, big-Oh notation, etc.

It is interesting to investigate what happens when we put stronger restrictions on the resources.

An interesting result in this line has been recently proved by Hemaspaandra, Mukherji and Tantau (2005):

> *Context-free languages are accepted by Turing machines with absolutely no space overhead*

The work space of the machine is:

- the finite state control
- the space that initially contains the input, with the restriction that *only a binary alphabet can be used on it.*

Final remarks

We used of tape tracks, namely large alphabets, big-Oh notation, etc.

It is interesting to investigate what happens when we put stronger restrictions on the resources.

An interesting result in this line has been recently proved by Hemaspaandra, Mukherji and Tantau (2005):

> *Context-free languages are accepted by Turing machines with absolutely no space overhead*

The work space of the machine is:

- the finite state control
- the space that initially contains the input, with the restriction that *only a binary alphabet can be used on it*.

Final remarks

We used of tape tracks, namely large alphabets, big-Oh notation, etc.

It is interesting to investigate what happens when we put stronger restrictions on the resources.

An interesting result in this line has been recently proved by Hemaspaandra, Mukherji and Tantau (2005):

> *Context-free languages are accepted by Turing machines with absolutely no space overhead*

The work space of the machine is:

- the finite state control
- the space that initially contains the input, with the restriction that *only a binary alphabet can be used on it*.