Ambiguity and Communication

Juraj Hromkovič [1] Georg Schnitger [2]

[1] Department of Computer Science
ETH Zürich

[2] Institut für Informatik
Goethe Universität Frankfurt am Main

The Goal

By how much does the size of NFA's increase, if the number of accepting computations is restricted?

Ambiguity

$\text{ambig}_N(x)$, the ambiguity of NFA N for input x, is the number of accepting computations of N on x.

Ambiguity

$\text{ambig}_N(x)$, the ambiguity of NFA N for input x, is the number of accepting computations of N on x.
$\text{ambig}_N(n) = \max\{\text{ambig}_N(x) : x \in \Sigma^n\}$ is the ambiguity of NFA N for input length n.

Ambiguity

$\text{ambig}_N(x)$, the ambiguity of NFA N for input x, is the number of accepting computations of N on x.
$\text{ambig}_N(n) = \max\{\text{ambig}_N(x) : x \in \Sigma^n\}$ is the ambiguity of NFA N for input length n.

Why are automata with small ambiguity of interest? Basic decision problems are efficiently solvable:

Ambiguity

$\text{ambig}_N(x)$, the ambiguity of NFA N for input x, is the number of accepting computations of N on x.
$\text{ambig}_N(n) = \max\{\text{ambig}_N(x) : x \in \Sigma^n\}$ is the ambiguity of NFA N for input length n.

Why are automata with small ambiguity of interest? Basic decision problems are efficiently solvable:

- Given an NFA N and a fixed constant c, is the ambiguity of N bounded by c? (Stearns and Hunt III, 1985).

Ambiguity

$\text{ambig}_N(x)$, the ambiguity of NFA N for input x, is the number of accepting computations of N on x.
$\text{ambig}_N(n) = \max\{\text{ambig}_N(x) : x \in \Sigma^n\}$ is the ambiguity of NFA N for input length n.

Why are automata with small ambiguity of interest? Basic decision problems are efficiently solvable:

- Given an NFA N and a fixed constant c, is the ambiguity of N bounded by c? (Stearns and Hunt III, 1985).
- Given an NFA N, is the ambiguity of N bounded, polynomial or exponential? (Weber and Seidl, 1991).

Ambiguity

$\text{ambig}_N(x)$, the ambiguity of NFA N for input x, is the number of accepting computations of N on x.
$\text{ambig}_N(n) = \max\{\text{ambig}_N(x) : x \in \Sigma^n\}$ is the ambiguity of NFA N for input length n.

Why are automata with small ambiguity of interest? Basic decision problems are efficiently solvable:

- Given an NFA N and a fixed constant c, is the ambiguity of N bounded by c? (Stearns and Hunt III, 1985).
- Given an NFA N, is the ambiguity of N bounded, polynomial or exponential? (Weber and Seidl, 1991).
- Given two NFA N_1 and N_2 with ambiguity at most c, are N_1 and N_2 equivalent? (Stearns and Hunt III, 1985).

What is Known?

- Ambiguity is either bounded by a constant or bounded by a polynomial or at least exponential.

What is Known?

- Ambiguity is either bounded by a constant or bounded by a polynomial or at least exponential.
- There are NFA's with exponential ambiguity and size n such that equivalent NFA's with polynomial ambiguity require $2^n - 1$ states (Leung98).

What is Known?

- Ambiguity is either bounded by a constant or bounded by a polynomial or at least exponential.
- There are NFA's with exponential ambiguity and size n such that equivalent NFA's with polynomial ambiguity require $2^n - 1$ states (Leung98).
- Open for almost twenty years: can NFA's with polynomial ambiguity be simulated by NFA's with bounded ambiguity, if size is only allowed to increase polynomially?

The Result

Languages with small automata of ambiguity $O(n^k)$

Let L be an arbitrary language. Define
$\exists_k(L) = \{w_1\$w_2\$\cdots\$w_m : m \in \mathbb{N} \text{ and } w_i \in L \text{ for at least } k \text{ positions}\}$.

The Result

Languages with small automata of ambiguity $O(n^k)$

Let L be an arbitrary language. Define
$\exists_k(L) = \{w_1\$w_2\$\cdots\$w_m : m \in \mathbb{N}$ and $w_i \in L$ for at least k positions$\}$.

Let Σ_r be the alphabet of all r-element subsets of $\{1, \ldots, r^{32}\}$. Then $L_r = \{xy \in \Sigma_r^2 \mid x \cap y \neq \emptyset\}$ is the language of non-disjointness.

The Result

Languages with small automata of ambiguity $O(n^k)$

Let L be an arbitrary language. Define
$\exists_k(L) = \{w_1\$w_2\$\cdots\$w_m : m \in \mathbb{N}$ and $w_i \in L$ for at least k positions$\}$.

Let Σ_r be the alphabet of all r-element subsets of $\{1, \ldots, r^{32}\}$. Then $L_r = \{xy \in \Sigma_r^2 \mid x \cap y \neq \emptyset\}$ is the language of non-disjointness.

A hierarchy of polynomial ambiguity

Set $t = r^{1/3}$. Then $\exists_k((L_r)^t)$ has NFA's with ambiguity $O(n^k)$ and $k \cdot \mathrm{poly}(r)$ states,

The Result

Languages with small automata of ambiguity $O(n^k)$

Let L be an arbitrary language. Define
$\exists_k(L) = \{w_1\$w_2\$\cdots\$w_m : m \in \mathbb{N} \text{ and } w_i \in L \text{ for at least } k \text{ positions}\}$.

Let Σ_r be the alphabet of all r-element subsets of $\{1, \ldots, r^{32}\}$. Then $L_r = \{xy \in \Sigma_r^2 \mid x \cap y \neq \emptyset\}$ is the language of non-disjointness.

A hierarchy of polynomial ambiguity

Set $t = r^{1/3}$. Then $\exists_k((L_r)^t)$ has NFA's with ambiguity $O(n^k)$ and $k \cdot \mathrm{poly}(r)$ states, but any equivalent NFA with ambiguity $o(n^k)$ has at least $2^{(r/k^2)^{1/3}}$ states.

Why Product Languages $(L_r)^t$?

Choose $L = \{uv \mid u, v \in \{0,1\}^r, u \neq v\}$: the language of inequality between r-bit strings. How large are NFA's for $\exists_1(L)$, if bounded ambiguity is required?

Why Product Languages $(L_r)^t$?

Choose $L = \{uv \mid u, v \in \{0,1\}^r, u \neq v\}$: the language of inequality between r-bit strings. How large are NFA's for $\exists_1(L)$, if bounded ambiguity is required?

- Guess a position $i \in \{1, \ldots, r\}$ and accept $u^1 v^1 \$ \cdots \$ u^m v^m$ if $u^j_i \neq v^j_i$ for some $1 \leq j \leq r$.

Why Product Languages $(L_r)^t$?

Choose $L = \{uv \mid u, v \in \{0,1\}^r, u \neq v\}$: the language of inequality between r-bit strings. How large are NFA's for $\exists_1(L)$, if bounded ambiguity is required?

- Guess a position $i \in \{1, \ldots, r\}$ and accept $u^1 v^1 \$ \cdots \$ u^m v^m$ if $u^j_i \neq v^j_i$ for some $1 \leq j \leq r$.
  $\exists_1(L)$ is recognizable with $\mathrm{poly}(r)$ states and ambiguity r.

Why Product Languages $(L_r)^t$?

Choose $L = \{uv \mid u, v \in \{0,1\}^r, u \neq v\}$: the language of inequality between r-bit strings. How large are NFA's for $\exists_1(L)$, if bounded ambiguity is required?

- Guess a position $i \in \{1, \ldots, r\}$ and accept $u^1 v^1 \$ \cdots \$ u^m v^m$ if $u^j_i \neq v^j_i$ for some $1 \leq j \leq r$.
  $\exists_1(L)$ is recognizable with $\mathrm{poly}(r)$ states and ambiguity r.
- What went wrong?

Why Product Languages $(L_r)^t$?

Choose $L = \{uv \mid u, v \in \{0,1\}^r, u \neq v\}$: the language of inequality between r-bit strings. How large are NFA's for $\exists_1(L)$, if bounded ambiguity is required?

- Guess a position $i \in \{1, \ldots, r\}$ and accept $u^1 v^1 \$ \cdots \$ u^m v^m$ if $u^j_i \neq v^j_i$ for some $1 \leq j \leq r$.
  $\exists_1(L)$ is recognizable with $\mathrm{poly}(r)$ states and ambiguity r.
- What went wrong? Few advice bits suffice and these advice bits can be remembered with few states.

Why Product Languages $(L_r)^t$?

Choose $L = \{uv \mid u, v \in \{0,1\}^r, u \neq v\}$: the language of inequality between r-bit strings. How large are NFA's for $\exists_1(L)$, if bounded ambiguity is required?

- Guess a position $i \in \{1, \ldots, r\}$ and accept $u^1 v^1 \$ \cdots \$ u^m v^m$ if $u_i^j \neq v_i^j$ for some $1 \leq j \leq r$.
  $\exists_1(L)$ is recognizable with $\mathrm{poly}(r)$ states and ambiguity r.
- What went wrong? Few advice bits suffice and these advice bits can be remembered with few states.
- Advantages, when working with $L = (L_r)^t$:
  - L has (small) NFA's with size poly$(r + t)$ with linear ambiguity.

Why Product Languages $(L_r)^t$?

Choose $L = \{uv \mid u, v \in \{0,1\}^r, u \neq v\}$: the language of inequality between r-bit strings. How large are NFA's for $\exists_1(L)$, if bounded ambiguity is required?

- Guess a position $i \in \{1, \ldots, r\}$ and accept $u^1 v^1 \$ \cdots \$ u^m v^m$ if $u^j_i \neq v^j_i$ for some $1 \leq j \leq r$.
  $\exists_1(L)$ is recognizable with $\mathrm{poly}(r)$ states and ambiguity r.
- What went wrong? Few advice bits suffice and these advice bits can be remembered with few states.
- Advantages, when working with $L = (L_r)^t$:
  - L has (small) NFA's with size poly$(r + t)$ with linear ambiguity.
  - The required number of guesses increases exponentially with t and these guesses cannot be remembered by small NFA.

A Proof Sketch for Sublinear Ambiguity

- $L = (L_r)^t$ is the language of t-fold non-disjointness.

A Proof Sketch for Sublinear Ambiguity

- $L = (L_r)^t$ is the language of t-fold non-disjointness. Set $\exists_{=0}(L) = \{w_1\$w_2\$\cdots\$w_m : m \in \mathbb{N}$ and $w_i \notin L$ for all positions$\}$.

A Proof Sketch for Sublinear Ambiguity

- $L = (L_r)^t$ is the language of t-fold non-disjointness. Set $\exists_{=0}(L) = \{w_1\$w_2\$\cdots\$w_m : m \in \mathbb{N}$ and $w_i \notin L$ for all positions$\}$.
- Let N be an NFA for $\exists_1(L)$ with sublinear ambiguity and let Q be its set of states.

A Proof Sketch for Sublinear Ambiguity

- $L = (L_r)^t$ is the language of t-fold non-disjointness. Set $\exists_{=0}(L) = \{w_1\$w_2\$\cdots\$w_m : m \in \mathbb{N}$ and $w_i \notin L$ for all positions$\}$.
- Let N be an NFA for $\exists_1(L)$ with sublinear ambiguity and let Q be its set of states.

A first step: show

- There are states $p_0, p_1 \in Q$ such that at least $|L|/|Q|^2$ strings in L have a computation starting in p_0 and ending in p_1.

A Proof Sketch for Sublinear Ambiguity

- $L = (L_r)^t$ is the language of t-fold non-disjointness. Set $\exists_{=0}(L) = \{w_1\$w_2\$\cdots\$w_m : m \in \mathbb{N}$ and $w_i \notin L$ for all positions$\}$.
- Let N be an NFA for $\exists_1(L)$ with sublinear ambiguity and let Q be its set of states.

A first step: show

- There are states $p_0, p_1 \in Q$ such that at least $|L|/|Q|^2$ strings in L have a computation starting in p_0 and ending in p_1.
- For **any** string $z' \in \overline{L}$ there is a string $u \in \exists_{=0}(L)$ with a "**launching cycle**" $r \xrightarrow{(z'u)^a} r \xrightarrow{(z'u)^b} p_0$ before reaching p_0

A Proof Sketch for Sublinear Ambiguity

- $L = (L_r)^t$ is the language of t-fold non-disjointness. Set $\exists_{=0}(L) = \{w_1\$w_2\$\cdots\$w_m : m \in \mathbb{N}$ and $w_i \notin L$ for all positions$\}$.
- Let N be an NFA for $\exists_1(L)$ with sublinear ambiguity and let Q be its set of states.

A first step: show

- There are states $p_0, p_1 \in Q$ such that at least $|L|/|Q|^2$ strings in L have a computation starting in p_0 and ending in p_1.
- For **any** string $z' \in \overline{L}$ there is a string $u \in \exists_{=0}(L)$ with a "**launching cycle**" $r \overset{(z'u)^a}{\longrightarrow} r \overset{(z'u)^b}{\longrightarrow} p_0$ before reaching p_0 and a "**storage cycle**" $p_1 \overset{(uz')^c}{\longrightarrow} s \overset{(uz')^d}{\longrightarrow} s$ after leaving p_1.

How to Exploit Sublinear Ambiguity?

The launching cycle delivers a power of $z'u$ to state p_0 and the storage cycle of p_1 stores a power of uz'.

How to Exploit Sublinear Ambiguity?

The launching cycle delivers a power of $z'u$ to state p_0 and the storage cycle of p_1 stores a power of uz'.

When is N forced into at least linear ambiguity?

How to Exploit Sublinear Ambiguity?

The launching cycle delivers a power of $z'u$ to state p_0 and the storage cycle of p_1 stores a power of uz'.

When is N forced into at least linear ambiguity?

- If a single occurrence of z' within the launching cycle is replaced by an impostor string $z \in L$ without N noticing **and**

How to Exploit Sublinear Ambiguity?

The launching cycle delivers a power of $z'u$ to state p_0 and the storage cycle of p_1 stores a power of uz'.

When is N forced into at least linear ambiguity?

- If a single occurrence of z' within the launching cycle is replaced by an impostor string $z \in L$ without N noticing **and**
- if z also hides unnoticed in a matching position within the storage cycle **and**

How to Exploit Sublinear Ambiguity?

The launching cycle delivers a power of $z'u$ to state p_0 and the storage cycle of p_1 stores a power of uz'.

When is N forced into at least linear ambiguity?

- If a single occurrence of z' within the launching cycle is replaced by an impostor string $z \in L$ without N noticing **and**
- if z also hides unnoticed in a matching position within the storage cycle **and**
- if $z \in L$ has a computation starting in p_0 and ending in p_1.

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The detection problem:

- all potential impostor strings $z \in L$ with a computation $p_0 \xrightarrow{z} p_1$ are to be accepted.

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The detection problem:

- all potential impostor strings $z \in L$ with a computation $p_0 \xrightarrow{z} p_1$ are to be accepted. At least $|L|/|Q|^2$ strings!

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The detection problem:

- all potential impostor strings $z \in L$ with a computation $p_0 \xrightarrow{z} p_1$ are to be accepted. At least $|L|/|Q|^2$ strings!
- **All** strings z surviving in matching positions within both cycles are to be rejected:

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The detection problem:

- all potential impostor strings $z \in L$ with a computation $p_0 \xrightarrow{z} p_1$ are to be accepted. At least $|L|/|Q|^2$ strings!
- **All** strings z surviving in matching positions within both cycles are to be rejected: all strings in the "complement" of L are rejected.

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The detection problem:

- all potential impostor strings $z \in L$ with a computation $p_0 \xrightarrow{z} p_1$ are to be accepted. At least $|L|/|Q|^2$ strings!
- **All** strings z surviving in matching positions within both cycles are to be rejected: all strings in the "complement" of L are rejected.
- No string may be accepted as well as rejected:

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The detection problem:

- all potential impostor strings $z \in L$ with a computation $p_0 \xrightarrow{z} p_1$ are to be accepted. At least $|L|/|Q|^2$ strings!
- **All** strings z surviving in matching positions within both cycles are to be rejected: all strings in the “complement” of L are rejected.
- No string may be accepted as well as rejected: no impostor may survive in both cycles.

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The detection problem:

- all potential impostor strings $z \in L$ with a computation $p_0 \xrightarrow{z} p_1$ are to be accepted. At least $|L|/|Q|^2$ strings!
- **All** strings z surviving in matching positions within both cycles are to be rejected: all strings in the "complement" of L are rejected.
- No string may be accepted as well as rejected: no impostor may survive in both cycles.
- The remaining strings from L can be treated either way.

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The detection problem:

- all potential impostor strings $z \in L$ with a computation $p_0 \xrightarrow{z} p_1$ are to be accepted. At least $|L|/|Q|^2$ strings!
- **All** strings z surviving in matching positions within both cycles are to be rejected: all strings in the "complement" of L are rejected.
- No string may be accepted as well as rejected: no impostor may survive in both cycles.
- The remaining strings from L can be treated either way.

A small, but significant minority of strings in L is accepted.

The Detection Problem

An NFA N with sublinear ambiguity has to detect impostors.

The detection problem:

- all potential impostor strings $z \in L$ with a computation $p_0 \xrightarrow{z} p_1$ are to be accepted. At least $|L|/|Q|^2$ strings!
- **All** strings z surviving in matching positions within both cycles are to be rejected: all strings in the "complement" of L are rejected.
- No string may be accepted as well as rejected: no impostor may survive in both cycles.
- The remaining strings from L can be treated either way.

A small, but significant minority of strings in L is accepted.
All strings in the "complement" of L are rejected and no string is accepted as well as rejected.

The Perspective of Communication

Let N be an NFA for $(L_r)^t$ with state set Q. If $z = (x_1 y_1, \ldots, x_t y_t)$ is input for N, then assign $(x_1, \ldots, x_s)$ to Alice and $(y_1, \ldots, y_s)$ to Bob.

The Perspective of Communication

Let N be an NFA for $(L_r)^t$ with state set Q. If $z = (x_1 y_1, \ldots, x_t y_t)$ is input for N, then assign $(x_1, \ldots, x_s)$ to Alice and $(y_1, \ldots, y_s)$ to Bob. Alice and Bob can simulate N by exchanging at most $|Q|^t$ messages.

The Perspective of Communication

Let N be an NFA for $(L_r)^t$ with state set Q. If $z = (x_1 y_1, \ldots, x_t y_t)$ is input for N, then assign $(x_1, \ldots, x_s)$ to Alice and $(y_1, \ldots, y_s)$ to Bob. Alice and Bob can simulate N by exchanging at most $|Q|^t$ messages.

There is a nondeterministic communication protocol which on input $z = (x_1 y_1, \ldots, x_t y_t)$ exchanges at most $|Q|^{O(t)}$ messages,

The Perspective of Communication

Let N be an NFA for $(L_r)^t$ with state set Q. If $z = (x_1 y_1, \ldots, x_t y_t)$ is input for N, then assign $(x_1, \ldots, x_s)$ to Alice and $(y_1, \ldots, y_s)$ to Bob. Alice and Bob can simulate N by exchanging at most $|Q|^t$ messages.

There is a nondeterministic communication protocol which on input $z = (x_1 y_1, \ldots, x_t y_t)$ exchanges at most $|Q|^{O(t)}$ messages,

- accepts at least $|(L_r)^t|/|Q|^2$ strings z from $(L_r)^t$,

The Perspective of Communication

Let N be an NFA for $(L_r)^t$ with state set Q. If $z = (x_1 y_1, \ldots, x_t y_t)$ is input for N, then assign $(x_1, \ldots, x_s)$ to Alice and $(y_1, \ldots, y_s)$ to Bob. Alice and Bob can simulate N by exchanging at most $|Q|^t$ messages.

There is a nondeterministic communication protocol which on input $z = (x_1 y_1, \ldots, x_t y_t)$ exchanges at most $|Q|^{O(t)}$ messages,

- accepts at least $|(L_r)^t|/|Q|^2$ strings z from $(L_r)^t$,
- rejects all strings z in the "complement" of $(L_r)^t$ and

The Perspective of Communication

Let N be an NFA for $(L_r)^t$ with state set Q. If $z = (x_1 y_1, \ldots, x_t y_t)$ is input for N, then assign $(x_1, \ldots, x_s)$ to Alice and $(y_1, \ldots, y_s)$ to Bob. Alice and Bob can simulate N by exchanging at most $|Q|^t$ messages.

There is a nondeterministic communication protocol which on input $z = (x_1 y_1, \ldots, x_t y_t)$ exchanges at most $|Q|^{O(t)}$ messages,

- accepts at least $|(L_r)^t|/|Q|^2$ strings z from $(L_r)^t$,
- rejects all strings z in the "complement" of $(L_r)^t$ and
- treats the many remaining strings in $(L_r)^t$ arbitrarily.

The Perspective of Communication

Let N be an NFA for $(L_r)^t$ with state set Q. If $z = (x_1 y_1, \dots, x_t y_t)$ is input for N, then assign $(x_1, \dots, x_s)$ to Alice and $(y_1, \dots, y_s)$ to Bob. Alice and Bob can simulate N by exchanging at most $|Q|^t$ messages.

There is a nondeterministic communication protocol which on input $z = (x_1 y_1, \dots, x_t y_t)$ exchanges at most $|Q|^{O(t)}$ messages,

- accepts at least $|(L_r)^t|/|Q|^2$ strings z from $(L_r)^t$,
- rejects all strings z in the "complement" of $(L_r)^t$ and
- treats the many remaining strings in $(L_r)^t$ arbitrarily.

How to analyze the nondeterministic communication protocol?

Utilize the above properties to obtain a deterministic protocol!

Analyzing the Nondeterministic Protocol

There is a deterministic protocol which

- accepts at least $|(L_r)^t|/|Q|^2$ strings z from $(L_r)^t$,

Analyzing the Nondeterministic Protocol

There is a deterministic protocol which

- accepts at least $|(L_r)^t|/|Q|^2$ strings z from $(L_r)^t$,
- rejects all strings z in the “complement” of $(L_r)^t$

Analyzing the Nondeterministic Protocol

There is a deterministic protocol which

- accepts at least $|(L_r)^t|/|Q|^2$ strings z from $(L_r)^t$,
- rejects all strings z in the "complement" of $(L_r)^t$
- and exchanges at most $|Q|^{t^2 \cdot \log_2 |Q|}$ messages.

Analyzing the Nondeterministic Protocol

There is a deterministic protocol which

- accepts at least $|(L_r)^t|/|Q|^2$ strings z from $(L_r)^t$,
- rejects all strings z in the "complement" of $(L_r)^t$
- and exchanges at most $|Q|^{t^2 \cdot \log_2 |Q|}$ messages.

Let α be sufficiently small. If such a deterministic protocol exchanges at most $2^{\alpha \cdot r \cdot t}$ messages, then D accepts at most $|(L_r)^t|/2^{\alpha \cdot t}$ strings from $(L_r)^t$. (Hromkovic and Schnitger, 2003)

Analyzing the Nondeterministic Protocol

There is a deterministic protocol which

- accepts at least $|(L_r)^t|/|Q|^2$ strings z from $(L_r)^t$,
- rejects all strings z in the "complement" of $(L_r)^t$
- and exchanges at most $|Q|^{t^2 \cdot \log_2 |Q|}$ messages.

Let α be sufficiently small. If such a deterministic protocol exchanges at most $2^{\alpha \cdot r \cdot t}$ messages, then D accepts at most $|(L_r)^t|/2^{\alpha \cdot t}$ strings from $(L_r)^t$. (Hromkovic and Schnitger, 2003)

Let N be an NFA with sublinear ambiguity recognizing $\exists_1((L_r)^t)$. Then N has at least $2^{\Omega(r^{1/3})}$ states.

Conclusions

- The detection problem allows to investigate NFA's of restricted ambiguity with the help of communication arguments.

Conclusions

- The detection problem allows to investigate NFA's of restricted ambiguity with the help of communication arguments.
- Showing that an NFA for $\exists_k(L)$ solves an appropriately defined detection problem for $k > 1$ proceeds similarly, but requires further work.