Descriptive Complexity of Non-deterministic Finite Automata of Different Ambiguity

Hing Leung

Department of Computer Science
New Mexico State University

Research was done while visiting University of Frankfurt, Germany, and was supported by Alexander von Humboldt foundation and NSF MII grant EIA-0220590.
Notations:

- DFA - Deterministic Finite Automaton
Notations:

- DFA - Deterministic Finite Automaton
- MDFA - DFA with Multiple Initial States
Notations:

- DFA - Deterministic Finite Automaton
- MDFA - DFA with Multiple Initial States
- UFA - Unambiguous Nondeterministic Finite Automaton

Unique accepting path for each string $x \in L(M)$. 
Notations:

- DFA - Deterministic Finite Automaton
- MDFA - DFA with Multiple Initial States
- UFA - Unambiguous Nondeterministic Finite Automaton
- FNA - Finitely Ambiguous NFA

$\exists k \forall x \in L(M)$, there are $\leq k$ accepting paths for $x$. 
Notations:

- DFA - Deterministic Finite Automaton
- MDFA - DFA with Multiple Initial States
- UFA - Unambiguous Nondeterministic Finite Automaton
- FNA - Finitely Ambiguous NFA
- LNA - Linearly ambiguous NFA
Notations:

- DFA - Deterministic Finite Automaton
- MDFA - DFA with Multiple Initial States
- UFA - Unambiguous Nondeterministic Finite Automaton
- FNA - Finitely Ambiguous NFA
- LNA - Linearly ambiguous NFA
- PNA - Polynomially ambiguous NFA
Notations:

- DFA - Deterministic Finite Automaton
- MDFA - DFA with Multiple Initial States
- UFA - Unambiguous Nondeterministic Finite Automaton
- FNA - Finitely Ambiguous NFA
- LNA - Linearly ambiguous NFA
- PNA - Polynomially ambiguous NFA
Notations:

- DFA - Deterministic Finite Automaton
- MDFA - DFA with Multiple Initial States
- UFA - Unambiguous Nondeterministic Finite Automaton
- FNA - Finitely Ambiguous NFA
- LNA - Linearly ambiguous NFA
- PNA - Polynomially ambiguous NFA
- ENA - Exponentially ambiguous NFA
Notations:

- **DFA** - Deterministic Finite Automaton
- **MDFA** - DFA with Multiple Initial States
- **UFA** - Unambiguous Nondeterministic Finite Automaton
- **FNA** - Finitely Ambiguous NFA
- **LNA** - Linearly ambiguous NFA
- **PNA** - Polynomially ambiguous NFA
- **ENA** - Exponentially ambiguous NFA
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

DFA: one entry, one exit
DFA: one entry, one exit

Reversal of DFA gives a UFA.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

UFA: one entry, one exit
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

- **UFA**: one entry, one exit
- Smallest DFA has $2^{n-1}$ states
UFA: one entry, one exit
- Smallest DFA has $2^{n-1}$ states
- Known since 1970's. [Meyer, Fischer 71]
UFA: one entry, one exit

Smallest DFA has $2^{n-1}$ states

Known since 1970’s. [Meyer, Fischer 71]

First state is always loaded.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

- UFA: one entry, one exit
- Smallest DFA has $2^{n-1}$ states
- Known since 1970’s. [Meyer, Fischer 71]
- First state is always loaded.
  All subsets of remaining states are reachable.
DFA/MDFA/FNA - LNA/PNA - ENA

DFA: one entry, one exit
DFA: one entry, one exit
Reversal of DFA gives a UFA.
**UFA: one entry, one exit**
UFA: one entry, one exit

Smallest DFA has $2^n$ states.
Two different subsets of states can be distinguished.
Two different subsets of states can be distinguished.
Reason: reversal of the automaton is a DFA.
All subsets of states are reachable.
All subsets of states are reachable.

How to reach this subset of states
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

All subsets of states are reachable.

How to reach this subset of states from the initial starting state?
All subsets of states are reachable.

How to reach this subset of states from the initial starting state? Deduce the string in reverse order.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Cannot apply 0. Apply 1.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Next apply 00
DFA/UFA - MDFA/FNA - LNA/PNA - ENA
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100

Next apply 11
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

10011
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

10011

Next apply 00
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

1001100
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

1001100

Next apply 11
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100110011
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100110011

Next apply 00
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100110011100
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

10011001100

Next apply 11
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

1001100110011
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

1001100110011

Next apply 0
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

10011001100110
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

DFA: one entry, one exit
DFA: one entry, one exit

Reversal of DFA gives a UFA.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

UFA: one entry, one exit
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

- UFA: one entry, one exit
- Smallest DFA has $2^n$ states.

Skip Proof
Two different subsets of states can be distinguished.
Two different subsets of states can be distinguished.

Reason: reversal of the automaton is a DFA.
All subsets of states are reachable.
All subsets of states are reachable.

How to reach this subset of states
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

All subsets of states are reachable.

How to reach this subset of states from the initial starting state?
All subsets of states are reachable.

How to reach this subset of states from the initial starting state? Deduce the string in reverse order.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA
Cannot apply 0. Apply 1.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA
1

Next apply 000
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

1000
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

1000

Next apply 11
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100011
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100011

Next apply 000
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100011000
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100011000

Next apply 11
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

10001100011
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Next apply 0000000
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100011000110000000
100011000110000000

Next apply 11
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

1000110001110000000011
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

1000110001110000000011

Next apply 00
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

10001100011000000001100
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Next apply 11

100011000111000000001100
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

10001100011100000000110011
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Next apply 0
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

1000110001110000000001100110
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

10001110001110000000001100110

Next apply 11
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

10001100011100000000110011011
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Next apply 0
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

100011000111000000001100110110
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Leiss 1981
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Leiss 1981

DFA:
Pick one start state.
Pick alternate states as final states.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Leiss 1981

DFA:
Pick one start state.
Pick alternate states as final states.

Reversal of DFA gives a UFA.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Leiss 1981
UFA has multiple start states and one final state.

Leiss 1981
Leiss 1981

UFA has multiple start states and one final state.

Its smallest equivalent DFA has $2^n$ states.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Veloso, Gill 1979
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Veloso, Gill 1979

All states initial. One final state.
Veloso, Gill 1979

- All states initial. One final state.
- MDFA: Transitions deterministic.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Veloso, Gill 1979

- All states initial. One final state.
- MDFA: Transitions deterministic.
- FNA: Finitely ambiguous DFA.
Veloso, Gill 1979

All states initial. One final state.

MDFA: Transitions deterministic.

FNA: Finitely ambiguous DFA.

Smallest equivalent DFA has $2^n - 1$ states.
[Schmidt 78]

$n$-state FNA. Equivalent UFA has $2^\Omega(\sqrt{n})$. 
Pick all states as starting states. 

**Veloso & Gill**: All nonempty subsets are realizable.
Pick all states as starting states.

*Veloso & Gill*: All nonempty subsets are realizable.

Pick alternate states as final states.

*Leiss*: All subsets can be realized by processing backward from final states.
Pick all states as starting states.

**Veloso & Gill**: All nonempty subsets are realizable.

Pick alternate states as final states.

**Leiss**: All subsets can be realized by processing backward from final states.

**MDFA**
A string $xy$ is accepted if the subset reached by processing $x$ intersects with the subset reached by processing $y^R$ backward starting from final states.
A string $xy$ is accepted if the subset reached by processing $x$ intersects with the subset reached by processing $y^R$ backward starting from final states.

[Schmidt 78] Regular language $L$. Pick strings $x_i, y_i$, for $i = 1, \ldots, n$. Define $M$ s.t. $M[x_i, y_j] = 1$ if $x_i y_j \in L$, and 0 otherwise. UFA for $L$ has $\geq \text{rank}(M)$ states.
A string $xy$ is accepted if the subset reached by processing $x$ intersects with the subset reached by processing $y^R$ backward starting from final states.

[Schmidt 78] Regular language $L$. Pick strings $x_i, y_i$, for $i = 1, \ldots, n$. Define $M$ s.t. $M[x_i, y_j] = 1$ if $x_iy_j \in L$, and 0 otherwise. UFA for $L$ has $\geq \text{rank}(M)$ states.

The matrix for our language is indexed by nonempty subsets of states such that entry $[Q_1, Q_2]$ has the value 1 if $Q_1 \cap Q_2 \neq \emptyset$, otherwise the entry has the value 0.
The matrix for our language is indexed by nonempty subsets of states such that entry \([Q_1, Q_2]\) has the value 1 if \(Q_1 \cap Q_2 \neq \emptyset\), otherwise the entry has the value 0.
The matrix for our language is indexed by nonempty subsets of states such that entry \([Q_1, Q_2]\) has the value 1 if \(Q_1 \cap Q_2 \neq \emptyset\), otherwise the entry has the value 0.
The matrix for our language is indexed by nonempty subsets of states such that entry \([Q_1, Q_2]\) has the value 1 if \(Q_1 \cap Q_2 \neq \emptyset\), otherwise the entry has the value 0.

\[
M = \begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

\[\text{Rank}(M) = 7.\]
The matrix for our language is indexed by nonempty subsets of states such that entry \([Q_1, Q_2]\) has the value 1 if \(Q_1 \cap Q_2 \neq \emptyset\), otherwise the entry has the value 0.

The matrix for our language has rank \(2^n - 1\).
The matrix for our language is indexed by nonempty subsets of states such that entry \([Q_1, Q_2]\) has the value 1 if \(Q_1 \cap Q_2 \neq \emptyset\), otherwise the entry has the value 0.

The matrix for our language has rank \(2^n - 1\).

The smallest equivalent UFA has \(2^n - 1\) states.
The matrix for our language is indexed by nonempty subsets of states such that entry \([Q_1, Q_2]\) has the value 1 if \(Q_1 \cap Q_2 \neq \emptyset\), otherwise the entry has the value 0.

The matrix for our language has rank \(2^n - 1\).

The smallest equivalent UFA has \(2^n - 1\) states.

The smallest equivalent DFA has \(2^n - 1\) states.
Reverse the previous MDFA design. FNA.
Reverse the previous MDFA design. FNA.

Pick alternate states as starting states.
Pick all states as final states.
Reverse all arrows.
Reverse the previous MDFA design. FNA.

Pick alternate states as starting states.
Pick all states as final states.
Reverse all arrows.

The smallest equivalent UFA has $2^n - 1$ states.
Reverse the previous MDFA design. FNA.

Pick alternate states as starting states.
Pick all states as final states.
Reverse all arrows.

The smallest equivalent UFA has $2^n - 1$ states.
The smallest equivalent DFA has $2^n$ states.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

101001
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

101001
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

101001
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

1 0 1 0 0 1
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

10_1^001
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

101001
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

101_001
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

101_{001}
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

101₀₀₁
Finite Language

DFA/UFA - MDFA/FNA - LNA/PNA - ENA

101_001
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

\[101_001\]
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Finite Language

101001
Finite Language

Example input is accepted.

Finite Language $L_3$
Finite Language

Example input is accepted.

101001

Finite Language $L_3$

$(3m)$-state MDFA for $L_m$
Finite Language

101001

Example input is accepted.

Finite Language $L_3$

$(3m)$-state MDFA for $L_m$

Same technique as before.
Example input is accepted.

- Finite Language $L_3$
- $(3m)$-state MDFA for $L_m$
- Same technique as before.
- Smallest equivalent UFA has $2^m - 1$ states.
Example input is accepted.

101001

- Finite Language $L_3$
- $(3m)$-state MDFA for $L_m$
- Same technique as before.
- Smallest equivalent UFA has $2^m - 1$ states.
- Consider $L_m \cap \{0, 1\}^{2m}$. 
$O(m^2)$-state MDFA for $L_m \cap \{0, 1\}^{2m}$
$O(m^2)$-state MDFA for $L_m \cap \{0, 1\}^{2m}$

Smallest equivalent UFA has $2^m - 1$ states.
Suppose there are $n$ registers.
Suppose there are \(n\) registers.

Each register holds a value of either 0 or 1 (‘off’ or ’on’).
Suppose there are $n$ registers.

Each register holds a value of either 0 or 1 ('off' or 'on').

Initially, register 1 is on. All other registers are off.
Suppose there are $n$ registers.
Each register holds a value of either 0 or 1 (‘off’ or ’on’).
Initially, register 1 is on. All other registers are off.

Instruction: Copy $i$ to $j$. (In short, $C_{i,j}$)
Action: Copy the value of register $i$ to register $j$. 
Suppose there are \( n \) registers.

Each register holds a value of either 0 or 1 (‘off’ or ’on’).

Initially, register 1 is on. All other registers are off.

Instruction: **Copy** \( i \) **to** \( j \). (In short, \( C_{i,j} \))

Action: Copy the value of register \( i \) to register \( j \).

Input: A sequence of copy instructions.
Suppose there are $n$ registers.

Each register holds a value of either 0 or 1 (‘off’ or ’on’).

Initially, register 1 is on. All other registers are off.

Instruction: $\text{Copy} \ i \ \text{to} \ j$. (In short, $C_{i,j}$)

Action: Copy the value of register $i$ to register $j$.

Input: A sequence of copy instructions.

Example input: $C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}$
Suppose there are \( n \) registers.

Each register holds a value of either 0 or 1 (‘off’ or ’on’).

Initially, register 1 is on. All other registers are off.

Instruction: \texttt{Copy} \( i \) to \( j \). (In short, \( C_{i,j} \))

Action: Copy the value of register \( i \) to register \( j \).

Input: A sequence of copy instructions.

Example input: \( C_{1,4} C_{4,2} C_{3,1} C_{4,3} C_{1,2} \)

An input is accepted if some register is on after the sequence of copy instructions have been performed.
Suppose there are \( n \) registers.

Each register holds a value of either 0 or 1 (‘off’ or ’on’).

Initially, register 1 is on. All other registers are off.

Instruction: copy \( i \) to \( j \). (In short, \( C_{i,j} \))

Action: Copy the value of register \( i \) to register \( j \).

Input: A sequence of copy instructions.

Example input: \( C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2} \) (Accepted)

An input is accepted if some register is on after the sequence of copy instructions have been performed.

Language of "some-register-on". Alphabet: \( n^2 \) letters.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Language of "some-register-on".
Language of "some-register-on".

Transitions: \((k, C_{i,j}, k)\) where \(k \neq j\) and \((i, C_{i,j}, j)\).
Language of "some-register-on".

Transitions \((k, C_{i,j}, k)\) where \(k \neq i, k \neq j\) are not shown.
Language of "some-register-on".

All states final. $n$-state FNA. (Reverse arrows: MDFA)
Language of "some-register-on".

All states final. \( n \)-state FNA. (Reverse arrows: MDFA)

Smallest UFA has \( 2^n - 1 \) states
Language of "some-register-on".
All states final. \( n \)-state FNA. (Reverse arrows: MDFA)
Extend the language. Add a query at the end of input.
Language of "some-register-on".

All states final. $n$-state FNA. (Reverse arrows: MDFA)

Extend the language. Add a query at the end of input.

Assert: Register $i$ is on (In short, $Q_i$)
Language of "some-register-on".

- All states final. $n$-state FNA. (Reverse arrows: MDFA)
- Extend the language. Add a query at the end of input.
- Assert: Register $i$ is on (In short, $Q_i$)
- The input is accepted if the register queried is on.
Language of "some-register-on".

All states final. $n$-state FNA. (Reverse arrows: MDFA)

Extend the language. Add a query at the end of input.

Assert: Register $i$ is on (In short, $Q_i$)

The input is accepted if the register queried is on.

Example: $C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}Q_3$ is accepted.
Language of "some-register-on".

All states final. \( n \)-state FNA. (Reverse arrows: MDFA)

Extend the language. Add a query at the end of input.

Assert: Register \( i \) is on (In short, \( Q_i \))

The input is accepted if the register queried is on.

Example: \( C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}Q_1 \) is not accepted.
Add state $f$, the only final state. $(n + 1)$-state UFA.
Add state \( f \), the only final state. \((n + 1)\)-state UFA.

Minimum size UFA.
Add state \( f \), the only final state. \((n + 1)\)-state UFA.

Minimum size UFA.

refute a conjecture by Hromkovič et al. about minimum UFA computation.
Add state $f$, the only final state. $(n + 1)$-state UFA.

Minimum size UFA.

Extend the language. Allow multiple queries.
Add state $f$, the only final state. $(n + 1)$-state UFA.

Minimum size UFA.

Extend the language. Allow multiple queries.

Accept if at least one query is positively answered.
Add state $f$, the only final state. $(n + 1)$-state UFA.

Minimum size UFA.

Extend the language. Allow multiple queries.

Accept if at least one query is positively answered.

Example: $C_{1,4}C_{4,2}Q_3C_{3,1}C_{4,3}Q_1C_{1,2}$ is not accepted.
Add state $f$, the only final state. $(n + 1)$-state UFA.

Minimum size UFA.

Extend the language. Allow multiple queries.

Accept if at least one query is positively answered.

Example: $C_{1,4}C_{4,2}Q_2C_{3,1}C_{4,3}Q_1C_{1,2}$ is accepted.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

\( (n+1) \)-state LNA.
(n+1)-state LNA.

Remark: 2-DFA easy. Idea: DFS processing.
DFA/UFA - MDFA/FNA - LNA/PNA - ENA

- (n+1)-state LNA.
- With input $C_{1,3}$, a thread currently at State 1 splits.
(n+1)-state LNA.

With input $C_{1,3}$, a thread currently at State 3 aborts.
(n+1)-state LNA.

A thread may split/abort.
(n+1)-state LNA.

A thread may split/abort.

Intuitions/Sketch
From a FNA for the language with multiple queries, we can deduce a FNA for the language of some-register-on with the same or smaller number of states.
Consider the design of a FNA for the language of "some-register-on" with the number of states less than $2^n - 1$. 
Consider the design of a FNA for the language of "some-register-on" with the number of states less than $2^n - 1$.

The current subset of states is maintained by several threads. Each thread has partial information of the current subset, and may split/abort.
Consider the design of a FNA for the language of "some-register-on" with the number of states less than $2^n - 1$.

The current subset of states is maintained by several threads. Each thread has partial information of the current subset, and may split_abort.

A thread aborts when it knows that the current subset is empty, or it has no knowledge of whether the current subset is non-empty.
Consider the design of a FNA for the language of "some-register-on" with the number of states less than \( 2^n - 1 \).

The current subset of states is maintained by several threads. Each thread has partial information of the current subset, and may split/abort.

A thread aborts when it knows that the current subset is empty, or it has no knowledge of whether the current subset is non-empty.

We want to argue that there is no bound on the number of aborts for processing an input string.
An adversary can force a thread to split, and next force the threads generated from splitting to abort until it is down to two threads, each maintaining a different single piece of knowledge. That is, one knows that $q_i$ is on, whereas the other thread knows that $q_j$ is on. But $i$ differs from $j$. By issuing the copy operation $C_{k,i}$ where $k \neq i$ and $k \neq j$, one thread will abort. We are now down to one thread (which knows that $q_j$ is on). We can repeat the technique to cause an unbounded number of aborts.
Back to the problem of designing a FNA for the language with multiple queries. We extend the adversary tricks given before. Immediately before the copy operation $C_{k,i}$ that causes a thread to abort, we apply a query $Q_i$. Such query will be answered positively by one thread before it aborts. On the other hand, the other thread has no knowledge that the query is positive (or, the query has been answered positively). Thus, we have infinite ambiguity. We conclude that the smallest equivalent FNA has size $2^n - 1$. 
Separation results:

DFA/UFA - MDFA/FNA - LNA/PNA - ENA
Separation results:

- DFA/UFA - MDFA/FNA - LNA/PNA - ENA

- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
Separation results:

- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
Separation results:

- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
Separation results:

1. DFA/UFA - MDFA/FNA - LNA/PNA - ENA
2. DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
3. DFA/UFA - MDFA/FNA - LNA/PNA - ENA
4. DFA/UFA - MDFA/FNA - LNA/PNA - ENA
5. DFA/UFA - MDFA/FNA - LNA/PNA - ENA  Finite Language
Separation results:

- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (Finite Language)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
Separation results:

- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA

Open problem:

- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (conjecture)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
Separation results:

- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (old result)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA
- Finite Language

Open problem:

- DFA/UFA - MDFA/FNA - LNA/PNA - ENA (conjecture)
- DFA/UFA - MDFA/FNA - LNA/PNA - ENA