Risultati recenti intorno al problema di Sakoda e Sipser

Giovanni Pighizzini

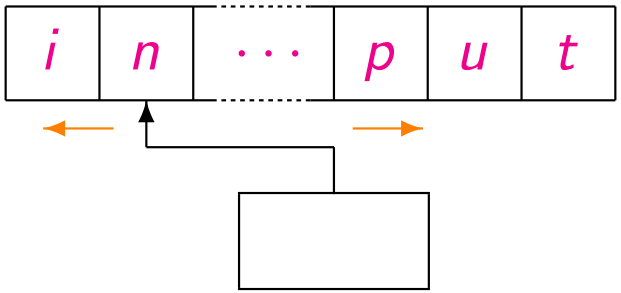
Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano

Milano, 20 ottobre 2011

UNIVERSITÀ DEGLI STUDI
DI MILANO

Outline

Finite State Automata



Base version:

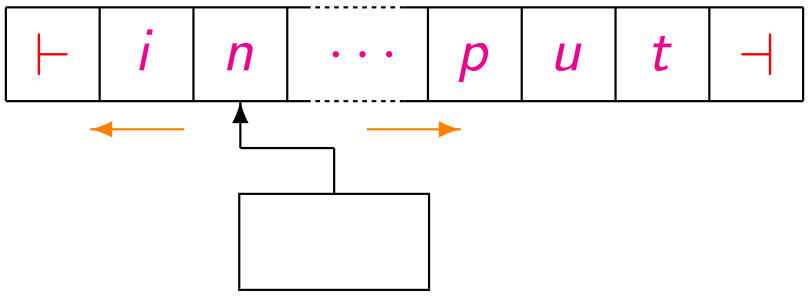
one-way deterministic finite automata (1DFA)

- one-way input tape
- deterministic transitions

Possibile variants allowing:

- nondeterministic transitions
  - one-way nondeterministic finite automata (1NFA)
- input head moving forth and back
  - two-way deterministic finite automata (2DFA)
  - two-way nondeterministic finite automata (2NFA)
- alternation
- ...

Two-Way Automata: Technical Details



- Input surrounded by the endmarkers $\vdash$ and $\dashv$
- Transition function $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to 2^{Q \times \{-1,0,+1\}}$ where $-1, 0, +1$ are the possible movements of the input head
- $w \in \Sigma^*$ accepted iff there is a computation
  - with input tape $\vdash w \dashv$
  - from the initial state q_0, scanning the left endmarker $\vdash$
  - reaching a final state

1DFA, 1NFA, 2DFA, 2NFA

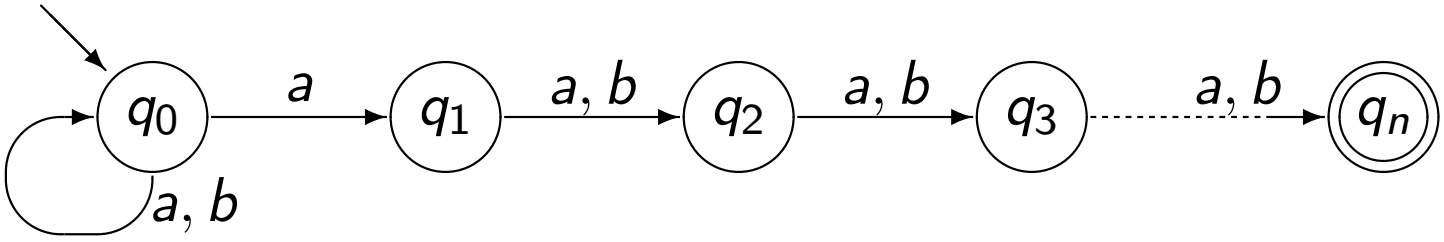
What about the power of these models?

They share the same computational power, namely they characterize the class of *regular languages*, however...

...some of them are more succinct

Example: $L = (a+b)^* a (a+b)^{n-1}$

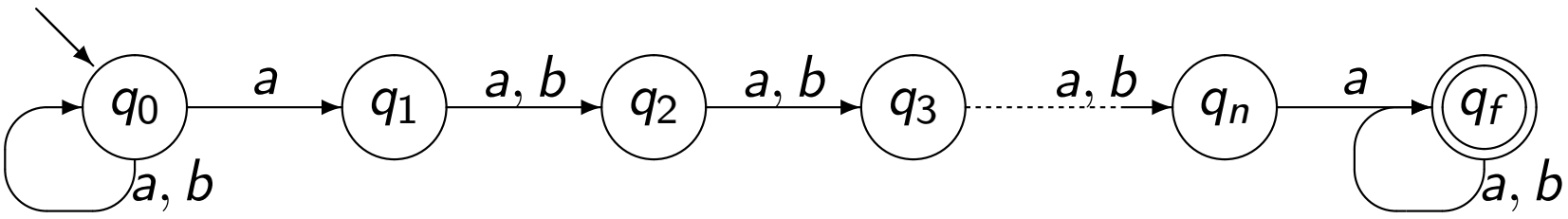
- L is accepted by a 1NFA with $n+1$ states



- The minimum 1DFA accepting L requires 2^n states
- We can get a *deterministic* automaton for L with $n+2$ states, which reverses the input head direction just one time
- Hence L is accepted by
  - a 1NFA and a 2DFA with approx. the same number of states
  - a minimum 1DFA exponentially larger

Example: $L = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

- L is accepted by a 1NFA with $n+2$ states



- The minimum 1DFA accepting L uses $3 \cdot 2^{n-1} + 1$ states
- Using head reversals the number of states becomes linear
- Even in this case L is accepted by
  - a 1NFA and a 2DFA with linearly related numbers of states
  - a minimum 1DFA exponentially larger

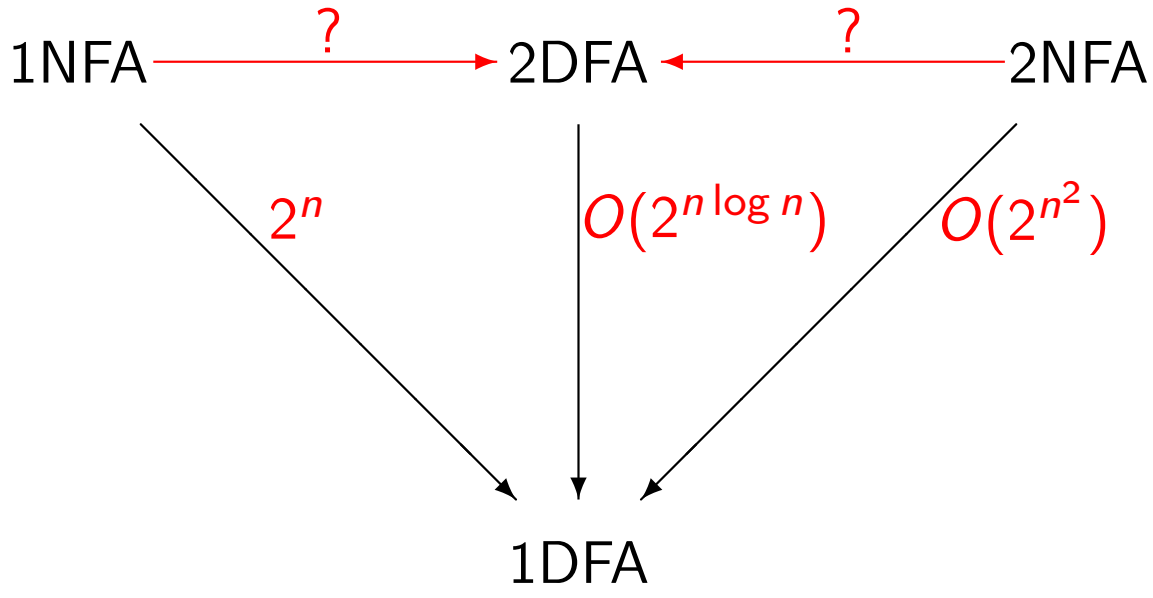
Example: $L = (a+b)^* a (a+b)^{n-1} a (a+b)^*$

$n = 4$

while input symbol $\neq a$ **do** move to the right
move n squares to the right
if input symbol $= a$ **then accept**
else move $n-1$ cells to the left
repeat from the first step
Exception: **if** input symbol $=\dashv$ **then reject**

- This can be implemented by a 2DFA with $O(n)$ states
- By a different algorithm, L can be also accepted by a 2DFA with $O(n)$ states *which changes the direction of its input head only at the endmarkers* (a *sweeping* automaton)

Costs of the Optimal Simulations Between Automata

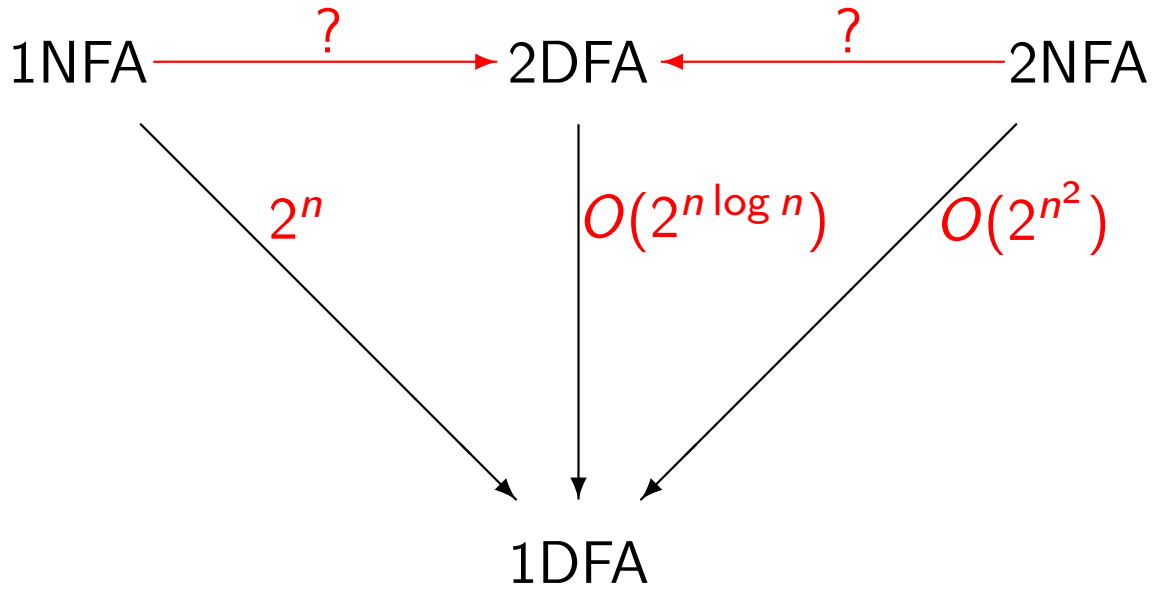


[Rabin&Scott '59, Shepardson '59, Meyer&Fischer '71, . . .]

Question

How much the possibility of moving the input head forth and back is useful to eliminate the nondeterminism?

Costs of the Optimal Simulations Between Automata



Problem ([Sakoda&Sipser '78])

Do there exist polynomial simulations of

- *1NFAs by 2DFAs*
- *2NFAs by 2DFAs ?*

Conjecture

These simulations are not polynomial

Sakoda&Sipser Question: Upper and Lower Bounds

- Exponential upper bounds
  deriving from the simulations of 1NFAs and 2NFAs by 1DFAs

- Polynomial lower bounds
  for the cost $c(n)$ of simulation of 1NFAs by 2DFAs:
  - $c(n) \in \Omega(\frac{n^2}{\log n})$ [Berman&Lingas '77]
  - $c(n) \in \Omega(n^2)$ [Chrobak '86]

- Complete languages
  ...

Sakoda and Sipser question

- Very difficult in its general form
- Not very encouraging obtained results:

  Lower and upper bounds too far
  (Polynomial vs exponential)

- Hence:

  Try to attack restricted versions of the problem!

2NFAs vs 2DFAs: restricted versions

(i) Restrictions on the resulting machines (2DFAs)

- sweeping automata [Sipser '80]
- oblivious automata [Hromkovič&Schnitger '03]
- "few reversal" automata [Kapoutsis '11]

(ii) Restrictions on the languages

- unary regular languages [Geffert Mereghetti&P '03]

(iii) Restrictions on the starting machines (2NFAs)

- outer nondeterministic automata [Geffert Guillon&P '11]

Sweeping Automata

Definition (Sweeping Automata)

A two-way automaton A is said to be sweeping if and only if

- A is deterministic
- the input head of A can change direction only at the endmarkers

Each computation is a sequence of complete traversals of the input

- Sweeping automata can be exponentially larger than 1NFAs [Sipser '80]
- However, they can be also *exponentially larger* than 2DFAs [Berman '81, Micali '81]

Oblivious automata

Definition

A two-way automaton A is said to be *oblivious* if and only if

- A is deterministic, and
- for each integer n, the "trajectory" of the input head is the same for all inputs of length n

Each sweeping automaton can be made oblivious with at most a quadratic growth of the number of the states

Oblivious automata

- Oblivious automata can be exponentially larger than 2NFAs
  [Hromkovič&Schnitger '03]
- Oblivious automata can be exponentially smaller than sweeping automata:
  - $L_k = (\{uv \mid u, v \in \{a, b\}^k \text{ and } u \neq v\}\#)^*$
  - L_k is accepted by an oblivious automaton with $O(k)$ states
    [Kutrib Malcher&P '11]
  - each sweeping automaton for L_k requires at least $2^{\frac{k-1}{2}}$ states
    [Hromkovič&Schnitger '03]
- Oblivious automata can be exponentially larger than 2DFAs
  - Witness: $PAD(L_k) = \bigcup_{a_1 a_2 \ldots a_m \in L_k} \$^* a_1 \$^* a_2 \$^* \cdots \$^* a_m \$^*$
    [Kutrib Malcher&P '11]

"Few Reversal" Automata [Kapoutsis'11]

Definition (Few Reversal Automata)

A two-way automaton A makes few reversals if and only if the number of reversals on input of length n is $o(n)$

Model between sweeping automata ($O(1)$ reversals) and 2NFAs

Theorem ([Kapoutsis'11])

- *Few reversal DFAs can be exponentially larger than few reversal NFAs and, hence, than 2NFAs*
- *Sweeping automata can be exponentially larger than few reversal DFAs*
- *Few reversal DFAs can be exponentially larger than 2DFAs*

Hence, this result really extends Sipser's separation, but does not solve the full problem

Sakoda&Sipser Question

Problem ([Sakoda&Sipser '78])

Do there exist polynomial simulations of

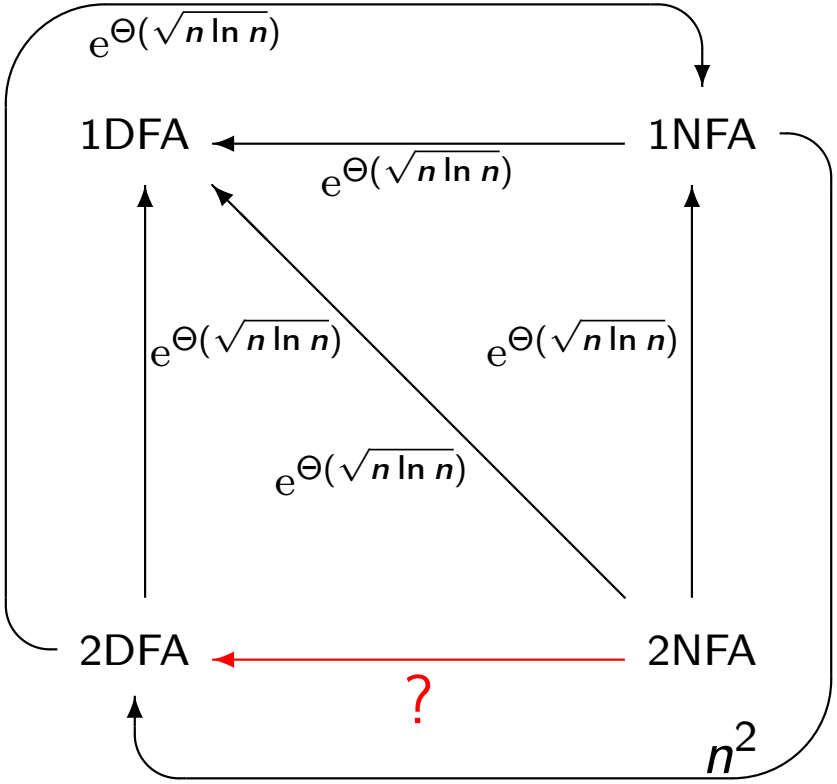
- *1NFAs by 2DFAs*
- *2NFAs by 2DFAs ?*

Another possible restriction:

The unary case $\#\Sigma = 1$

Optimal Simulation Between Unary Automata

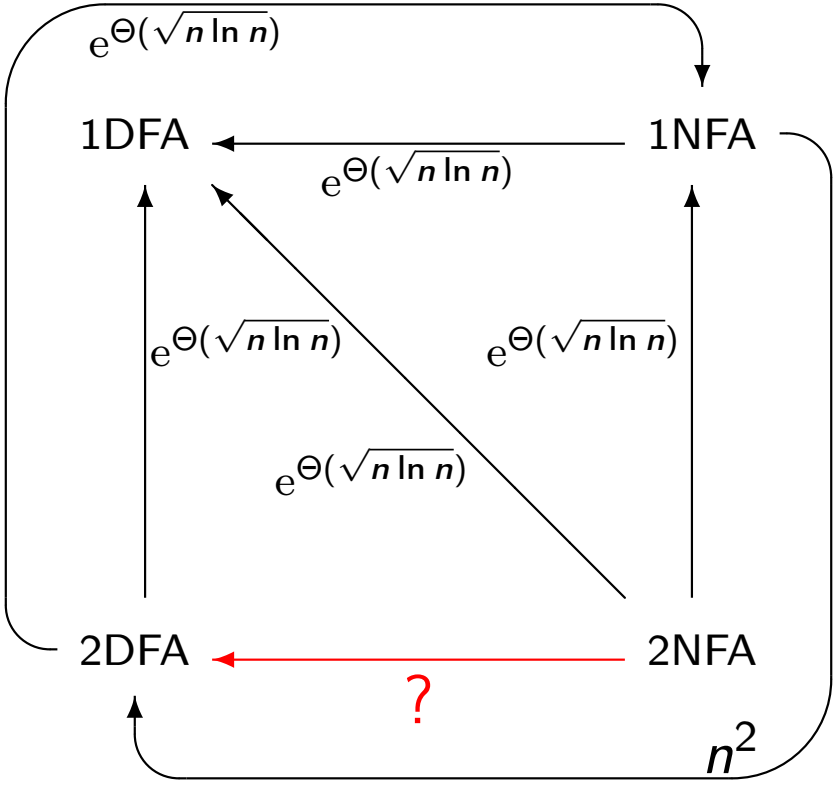
The costs of the optimal simulations between automata are different in the unary and in the general case



[Chrobak '86, Mereghetti&P '01]

Optimal Simulation Between Unary Automata

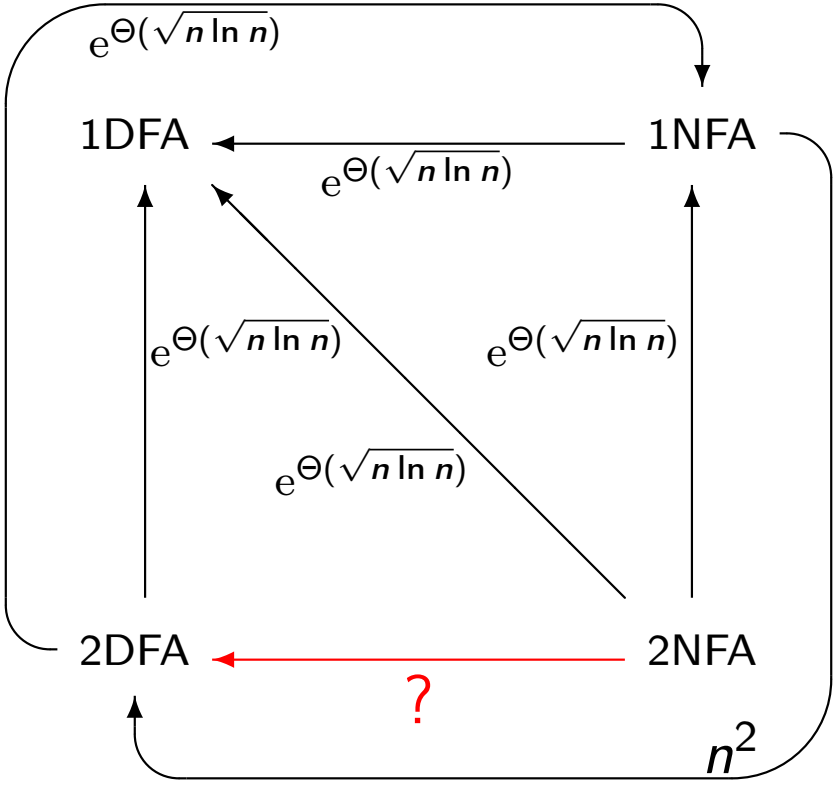
The costs of the optimal simulations between automata are different in the unary and in the general case



1NFA $\rightarrow$ 2DFA
In the unary case
this question is solved!
(polynomial conversion)

Optimal Simulation Between Unary Automata

The costs of the optimal simulations between automata are different in the unary and in the general case



2NFA $\to$ 2DFA

Even in the unary case this question is open!

- $e^{\Theta(\sqrt{n \ln n})}$ upper bound (from 2NFA $\to$ 1DFA)
- $\Omega(n^2)$ lower bound (from 1NFA $\to$ 2DFA)

A better upper bound $e^{O(\ln^2 n)}$ has been proved!

Sakoda&Sipser Question: Current Knowledge

- Upper bounds

	1NFA$\rightarrow$ 2DFA	2NFA$\rightarrow$ 2DFA
unary case	$O(n^2)$ optimal	$e^{O(\ln^2 n)}$
general case	exponential	exponential

Unary case [Chrobak '86, Geffert Mereghetti&P '03]

- Lower Bounds

  In all the cases, the best known lower bound is $\Omega(n^2)$ [Chrobak '86]

Unary Case: Quasi Sweeping Automata

[Geffert Mereghetti&P '03]

In the study of unary 2NFA, sweeping automata with some *restricted nondeterministic capabilities* turn out to be very useful:

Definition

A 2NFA is quasi sweeping (qsNFA) iff both

- nondeterministic choices and head reversals

are possible only at the endmarkers

Theorem (Quasi Sweeping Simulation)

Each n-state unary 2NFA A can be transformed into a 2NFA M s.t.

- *M is quasi sweeping*
- *M has at most* $N \leq 2n + 2$ *states*
- *M and A are "almost equivalent"*
  (differences are possible only for inputs of length $\leq 5n^2$*)*

Quasi Sweeping Simulation: Consequences

Several results using quasi sweeping simulation of unary 2NFAs have been found:

(i) Subexponential simulation of unary 2NFAs by 2DFAs
Each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states [Geffert Mereghetti&P '03]

(ii) Polynomial complementation of unary 2NFAs
Inductive counting argument for qsNFAs
[Geffert Mereghetti&P '07]

(iii) Polynomial simulation of unary 2NFAs by 2DFAs
under the condition L = NL

(iv) Polynomial simulation of unary 2NFAs by unambiguous 2NFAs (unconditional)

We are going to discuss (iii) and (iv) [Geffert&P '10]

Logspace Classes and Graph Accessibility Problem

L: class of languages accepted in logarithmic space by *deterministic* machines

NL: class of languages accepted in logarithmic space by *nondeterministic* machines

Problem

$L \stackrel{?}{=} NL$

Graph Accessibility Problem GAP

- Given $G = (V, E)$ oriented graph, $s, t \in V$
- Decide whether or not G contains a path from s to t

Theorem ([Jones '75])

GAP *is complete for* NL

Hence GAP $\in$ L iff L = NL

Polynomial Deterministic Simulation (under L = NL)

Outline

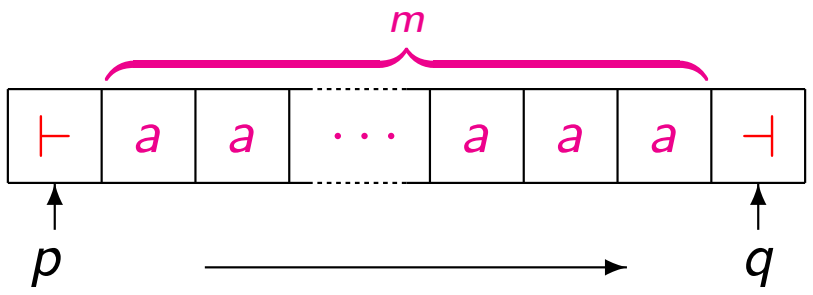
From now on, we fix an n-state unary 2NFA A

- We give a *reduction* from $L(A)$ to GAP
  i.e, for each input string a^m we define a graph $G(m)$ s.t.

$$a^m \in L(A) \iff G(m) \in \text{GAP}$$

- *Under the hypothesis* L = NL
  this reduction will be used to build 2DFA equivalent to A,
  with a number of states polynomial in n
- Actually we do not work directly with A:
  we use the qsNFA M obtained from A
  according to the quasi sweeping simulation

The graph $G(m)$



Given the qsNFA M with N states and an input a^m the graph $G(m)$ is defined as:

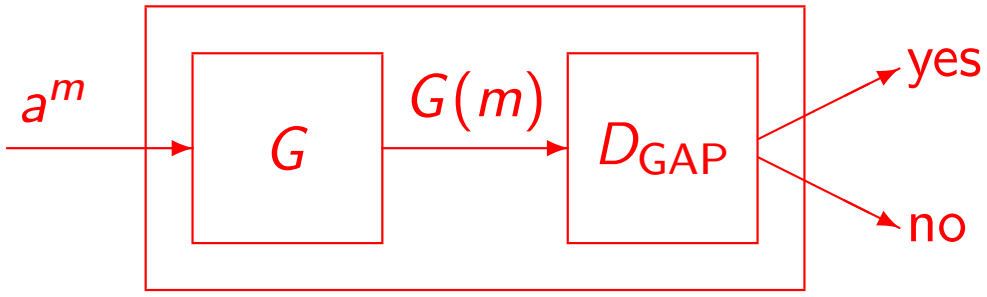
- the vertices are the states of M
- (p, q) is an edge iff M can traverse the input
  - from one endmarker in the state p
  - to the opposite endmarker in the state q
  - without visiting the endmarkers in the meantime

Then

$$a^m \in \mathrm{L(M)} \text{ iff } G(m) \text{ contains a path from } q_0 \text{ to } q_F$$

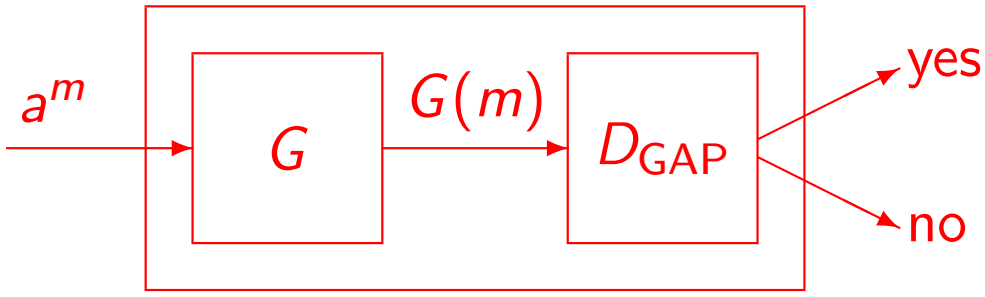
The existence of the edge (p, q) can be verified by a finite automaton $A_{p,q}$ with N states

Polynomial Deterministic Simulation (under L = NL)



- Suppose L = NL
- Let D_{GAP} be a logspace bounded *deterministic* machine solving GAP
- On input a^m, compute $G(m)$ and give the resulting graph as input to D_{GAP}
- This decides whether or not $a^m \in L(M)$

Polynomial Deterministic Simulation (under L = NL)



- The graph $G(m)$ has N vertices, the number of states of M
- D_{GAP} uses space $O(\log N)$
- M is fixed. Hence N is constant, independent on the input a^m

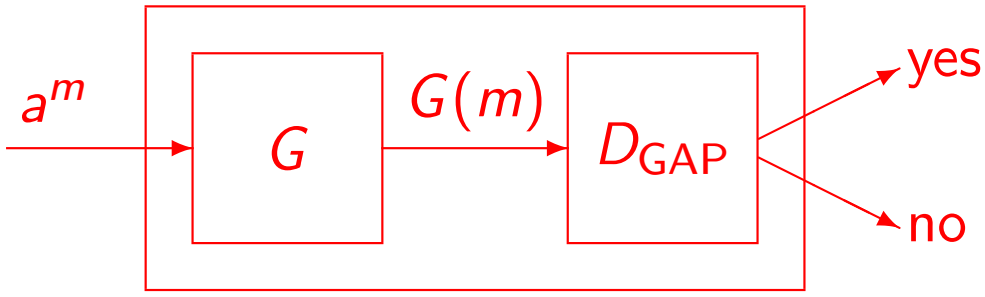
  The worktape of D_{GAP} can be encoded in a finite control using a number of states polynomial in N

- The graph $G(m)$ can be represented with N^2 bits

  Representing the graph in a finite control would require exponentially many states

- To avoid this, input bits for D_{GAP} are computed "on demand"

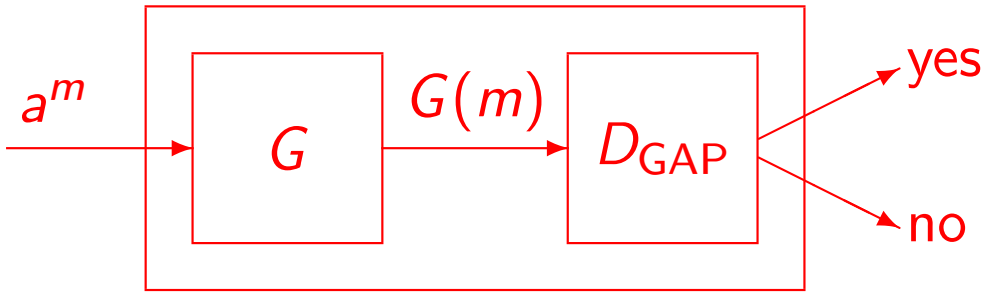
Polynomial Deterministic Simulation (under L = NL)



We define a unary 2DFA M' equivalent to M

- M' keeps in its finite control:
  - The input head position of D_{GAP}
  - The worktape content of D_{GAP}
  - The finite control of D_{GAP}
- This uses a number of states polynomial in N

Polynomial Deterministic Simulation (under L = NL)



We define a unary 2DFA M' equivalent to M

- On input a^m, M' simulates D_{GAP} on input $G(m)$
- Input bits for D_{GAP} are the entries of $G(m)$ adjacency matrix
- Subroutine $A_{p,q}$, using N states, computes the input bit corresponding to (p, q)
- Considering all possible (p, q), this part uses at most N^3 states

Summing Up...

We described the following simulation:

- M is *almost equivalent* to the original 2NFA A
- Hence, M' is *almost equivalent* to A
- Possible differences for input length $\leq 5n^2$
- They can be fixed in a preliminary scan ($5n^2 + 2$ more states)
- The resulting automaton has polynomially many states

A	given unary 2NFA	n states
$\Downarrow$	Quasi Sweeping Simulation	
M	qsNFA almost equivalent to A	$N \leq 2n + 2$ states
$\Downarrow$	Deterministic Simulation	
M'	2DFA equivalent to M	$poly(N)$ states
$\Downarrow$	Preliminary scan to accept/reject inputs of length $\leq 5n^2$ then simulation of M' for longer inputs	
M''	2DFA equivalent to A	$poly(n)$ states

Polynomial Deterministic Conditional Simulation

Theorem ([Geffert&P '10])

If L = NL *then each n-state unary 2NFA can be simulated by an equivalent 2DFA with poly*(n) *many states*

Hence

Proving the Sakoda&Sipser conjecture for unary 2NFAs would separate L *and* NL

Unambiguous Logspace (Nonuniform)

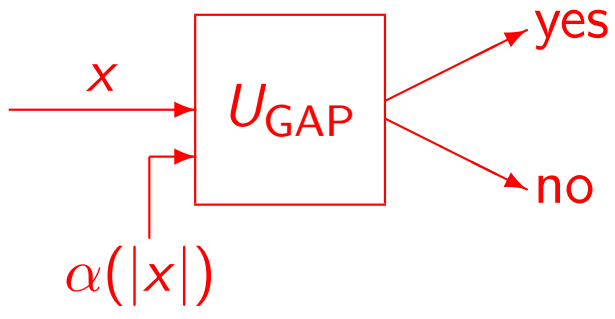
Theorem ([Reinhardt&Allender '00])

NL $\subseteq$ UL/*poly*

- UL/*poly*
  class of languages accepted by *unambiguous* logspace machines with a *polynomial advice*, i.e.,
- A sequence of strings $\{\alpha(n) \mid n \geq 0\}$ of polynomial length
- With each input string x, the machine also receives the advice string $\alpha(|x|)$

Corollary

GAP $\in$ UL/*poly*



Making Unary 2NFAs Unambiguous

Theorem ([Geffert&P '10])

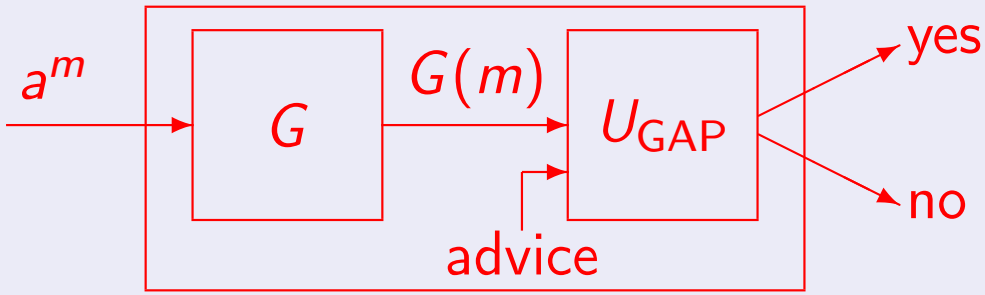
Each n-state unary 2NFA can be simulated by an equivalent unambiguous 2NFA with poly(n) many states

Proof.

- Similar to the polynomial deterministic conditional simulation
- Hypothetical machine D_{GAP} replaced with U_{GAP} and advice

Given a 2NFA the size of $G(m)$ (input of U_{GAP}) is fixed

- Hence the advice is fixed (i.e., it does not depend on a^m)
- Advice encoded in the hardware of the simulating machine



Polynomial Deterministic Conditional Simulation

If L = NL then each n-state unary 2NFA can be simulated by an equivalent 2DFA with $poly(n)$ many states

What about the converse?

Polynomial Deterministic Conditional Simulation

Question

Does a polynomial simulation of unary 2NFAs by 2DFAs imply $\mathsf{L} = \mathsf{NL}$*?*

- The answer is positive, *under an additional assumption*:

  The transformation from unary 2NFAs to 2DFAs must be computable in deterministic logspace

- Under this assumption, the answer is positive *even restricting* to the simulation of unary 1NFAs by 2DFAs:

Theorem

If there exists a deterministic logspace bounded transducer transforming each n-state unary 1NFA into an equivalent $n^{O(1)}$*-state 2DFA then* $\mathsf{L} = \mathsf{NL}$

Polynomial Deterministic Conditional Simulation

If L $\supseteq$ NL then each n-state unary 2NFA can be simulated by an equivalent 2DFA with $poly(n)$ many states

Ch. Kapoutsis observed that the proof works also for the nonuniform version of L:

Theorem

If L/*poly* $\supseteq$ NL *then each n-state unary 2NFA can be simulated by an equivalent 2DFA with* $poly(n)$ *many states*

Note: L $\subseteq$ L/*poly* hence the assumption is weaker

We proved the converse of the last statement. Hence:

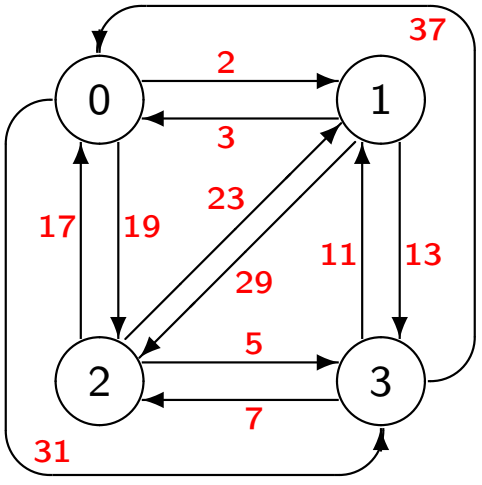
Theorem ([Kapoutsis&P '11])

L/*poly* $\supseteq$ NL *iff*
each n-state unary 2NFA can be simulated by an equivalent 2DFA with $poly(n)$ *many states*

Outline of the proof (1/4)

The "hardest" language

- $K_n :=$ complete directed graph with vertices $\{0, \ldots, n-1\}$
- With each edge (i, j) we associate a different prime number $p_{(i,j)}$



- We define a unary 2NFA A_n s.t.
  - the states visited at the endmarkers are (copies of) $0, \ldots, n-1$
  - an input a^m can be traversed from the state i at one endmarker to the state j at the opposite endmarker iff $p_{(i,j)}$ divides m
  - the initial and the final states are 0 and $n-1$, respectively
- Choosing small different primes, A_n can be implemented with $O(n^4 \log n)$ states

Outline of the proof (2/4)

The "hardest" language

- Let $H_n = L(A_n)$ be the language accepted by A_n. Then:
- $a^m \in H_n$ iff there is a path

$$0 = i_0 \overset{p_{(i_0,i_1)}}{\longrightarrow} i_1 \overset{p_{(i_1,i_2)}}{\longrightarrow} \cdots \overset{p_{(i_{k-1},i_k)}}{\longrightarrow} i_k = n-1$$

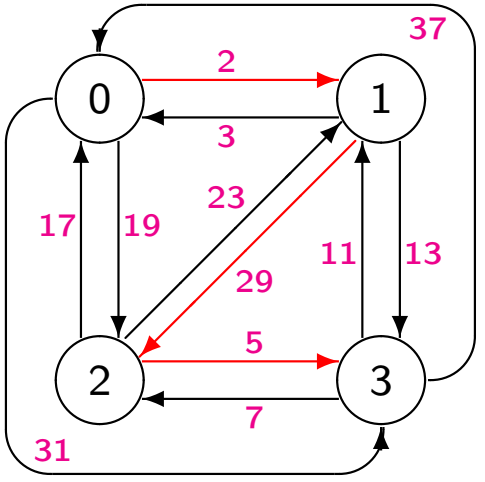
s.t. m is a multiple of all $p_{(i_{j-1},i_j)}$, for $j = 1, \ldots, k$

Example: $m = 580$

- 580 is a multiple of $2, 5, 29$
- a^{580} is accepted along the path

$$0 \overset{2}{\longrightarrow} 1 \overset{29}{\longrightarrow} 2 \overset{5}{\longrightarrow} 3$$

- Even 9570 is a multiple of those numbers, hence a^{9570} is accepted along the same path

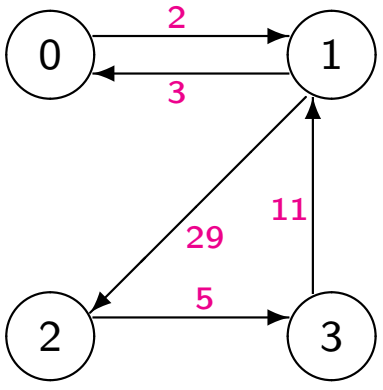


Outline of the proof (3/4)

From GAP to the "hardest" language

Given $G = (\{0, \ldots, n-1\}, E)$,
we encode it by the number:

$$m_G = \prod_{(i,j) \in E} p_{(i,j)}$$



$$m_G = 2 \cdot 3 \cdot 5 \cdot 11 \cdot 29 = 9570$$

- G is a subgraph of K_n
- Then $G \in$ GAP iff $a^{m_G} \in H_n$

Hence, we have defined a reduction from GAP (restricted to graphs with n vertices) to H_n

Outline of the proof (4/4)

The final machine solving GAP

If the conversion of unary 2NFAs by 2DFAs is polynomial:

- H_n is accepted by a 2DFA B_n with $poly(n)$ states

  Furthermore we can suppose that (...more or less...)
  - B_n is sweeping
  - In each sweep B_n verifies the divisibility of the input length by one of the $p_{(i,j)}$
- We define the following machine M_n
  - the input is a graph $G = (\{0, \ldots, n-1\}, E)$
  - M_n simulates B_n on input a^{m_G}, where $m_G = \prod_{(i,j)\in E} p_{(i,j)}$
  - m_G is not explicitly computed:
    to simulate a sweep of B_n checking divisibility by $p_{(i,j)}$,
    M_n checks whether of not $(i,j) \in E$
- M_n solves GAP for graphs with n vertices
- We can prove that M_n works in logarithmic space

Since GAP is complete for NL, we conclude $\mathrm{L}/poly \supseteq \mathrm{NL}$

Outer nondeterministic automata (ONFAs)

Definition

A two-way automaton is said to be *outer nondeterministic* iff nondeterministic choices are allowed *only* when the input head is scanning the endmarkers

Hence:

- No restrictions on the *input alphabet*
- No restrictions on *head reversals*
- *Deterministic transitions* on "real" input symbols
- *Nondeterminstic choices* only at the endmarkers

Outer nondeterministic automata (ONFAs)

All the results we obtained for the unary case
can be extended to ONFAs: [Geffert Guillon&P '11]

(i) Subexponential simulation of 2ONFAs by 2DFAs

(ii) Polynomial complementation of unary 2ONFAs

(iii) Polynomial simulation of 2ONFAs by 2DFAs
under the condition L/*poly* $\supseteq$ NL

(iv) Polynomial simulation of 2ONFAs by unambiguous 2ONFAs

While in the unary case all the proofs rely on the conversion
of 2NFAs into quasi sweeping automata,
in the case of 2ONFAs we do not have a similar tool!

Outer nondeterministic automata (ONFAs)

Main tool: procedure $reach(p, q)$

- The procedure checks the existence of a computation segment
  - from the left endmarker in the state p
  - to the left endmarker in the state q
  - not visiting the left endmarker in between
- Difficult point: possibility of infinite loops
- Implementation:
  - Modification of a technique for the complementation of 2DFAs [Geffert Mereghetti&P '07], which is a refinement of a construction for the complementation of space bounded Turing machines [Sipser '80]
  - Finite control with a *linear number* of states reading a two-way input tape

Loops involving endmarkers can be avoided by observing that for each accepting computation visiting one of the endmarkers more than $|Q|$ times there exists a shorter accepting computation

Final considerations

- The question of Sakoda and Sipser is very challenging
- In the investigation of restricted versions many interesting and not artificial models have been considered
- The results obtained for restricted versions of the problem, even if they do not solve the full problem, are nontrivial and, in many cases, very deep
- Strong connections with open questions in structural complexity
- Many times techniques used in space complexity can be adapted for the investigation of automata and vice versa

Two Further Directions

- The results obtained in the unary case have been extended to the general case for outer nondeterministic automata

Question

Does it is possible to extend the same results (or some of them) to some less restricted models of computation?

- Input head reversals are a critical resource that deserves further investigation

  An example of problem that could be investigated:

Question

Given $k > 0$, does there exists a language L such that each 2DFA accepting L with less than k head reversals is exponentially larger than each 2DFA with k reversals?

A positive answer is known only for $k \leq 2$ [Balcerzac&Niwiński '10]