

# Descriptive Complexity and Regular Languages

Giovanni Pighizzini

Dipartimento di Informatica e Comunicazione  
Università degli Studi di Milano

Université Nice Sophia Antipolis – May 19, 2011



UNIVERSITÀ DEGLI STUDI  
DI MILANO

# Outline

Introduction: What is Descriptive Complexity?

The Question of Sakoda and Sipser

Unary Automata

Sakoda&Sipser Question vs  $L \stackrel{?}{=} NL$

Context-Free Grammars vs Regular Languages

Conclusion

# A Classical Example: Deterministic vs Nondeterministic Automata

## Formal language point of view:

- ▶ The class of languages recognized by NFAs coincides with the class of languages recognized by DFAs

## Descriptive complexity point of view:

- ▶ Each  $n$ -state NFA can be simulated by a DFA *with  $2^n$  states* [Rabin&Scott '59]
- ▶ For each integer  $n$  there exists a language  $L_n$  s.t.:
  - ▶  $L_n$  is accepted by an  $n$ -state NFA
  - ▶ the minimum DFA for  $L_n$  requires  $2^n$  states[Meyer&Fischer '71]
- ▶ Hence:

The exact cost, in terms of states, of the simulation  
of NFAs by DFAs is  $2^n$

# Descriptive Complexity

Given

- ▶  $\mathcal{C}$ , a class of languages
- ▶  $\mathcal{S}$ , a formal system (e.g., class of devices, class of grammars,...) able to represent all the languages in  $\mathcal{C}$

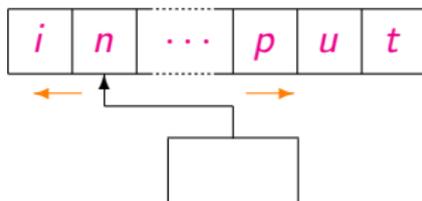
What is the *size* of the representations of the languages in  $\mathcal{C}$  by the system  $\mathcal{S}$ ?

Descriptive complexity compares different descriptions of a same class of languages:

- ▶ given  $\mathcal{S}'$ , another formal system able to represent all the languages in  $\mathcal{C}$ :

Find the *relationships between the sizes* of the representations in the system  $\mathcal{S}$  and in the system  $\mathcal{S}'$  of the languages belonging to  $\mathcal{C}$

# Finite State Automata



Base version:

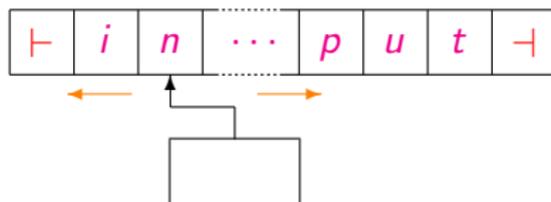
one-way deterministic finite automata (1DFA)

- ▶ one-way input tape
- ▶ deterministic transitions

Possible 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  $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \rightarrow 2^{Q \times \{-1, 0, +1\}}$   
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

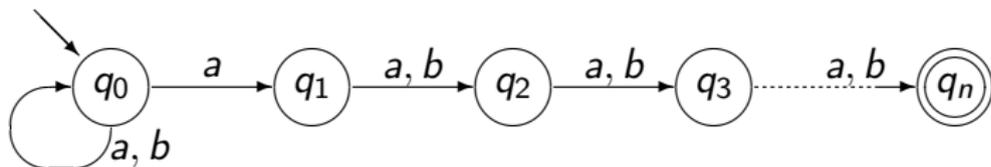
What about the power of these models?

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

...some of them are more succinct

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

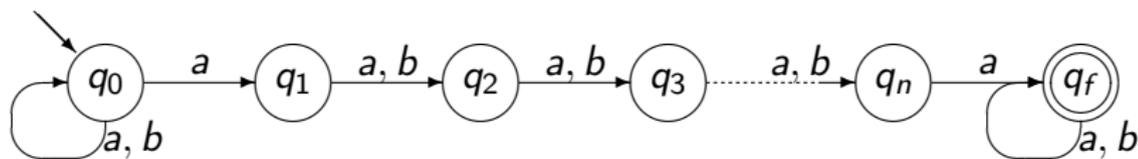
- ▶  $L$  is accepted by a 1NFA with  $n + 1$  states



- ▶ 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

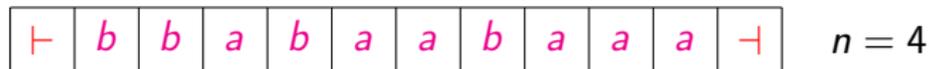
Example:  $L = (a + b)^* a(a + b)^{n-1} a(a + b)^*$

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



**while** input symbol  $\neq a$  **do** move to the right

move  $n$  squares to the right

**if** input symbol =  $a$  **then accept**

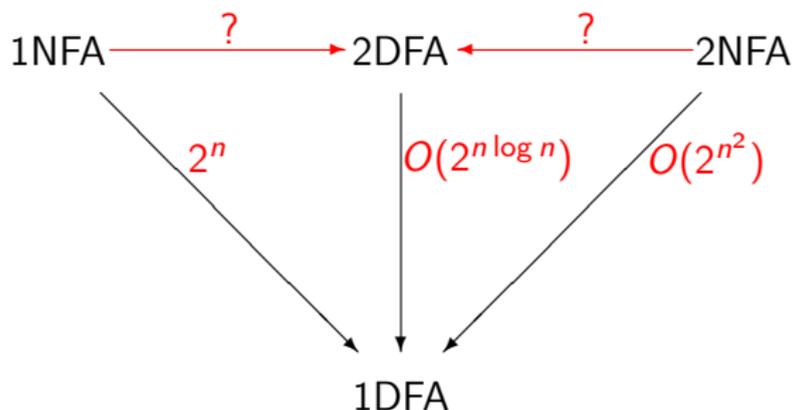
**else** move  $n - 1$  cells to the left

**repeat** from the first step

**Exception:** **if** input symbol =  $\perp$  **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

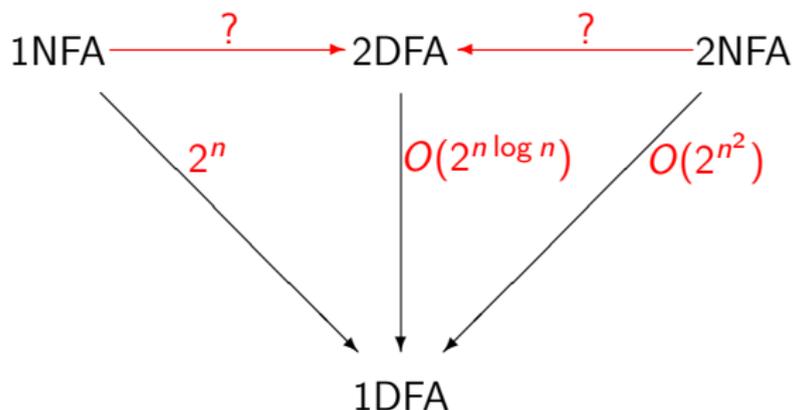


[Rabin&Scott '59, Shepardson '59, Meyer&Fischer '71, ...]

## 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: Lower Bounds

**Polynomial lower bounds** for the cost  $c(n)$  of simulation of 1NFAs by 2DFAs:

- ▶  $c(n) \in \Omega\left(\frac{n^2}{\log n}\right)$  [Berman&Lingas '77]
- ▶  $c(n) \in \Omega(n^2)$  [Chrobak '86]

**Exponential lower bounds** for the simulation of 2NFAs by 2DFAs, for special classes of resulting machines:

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

# “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 ?*

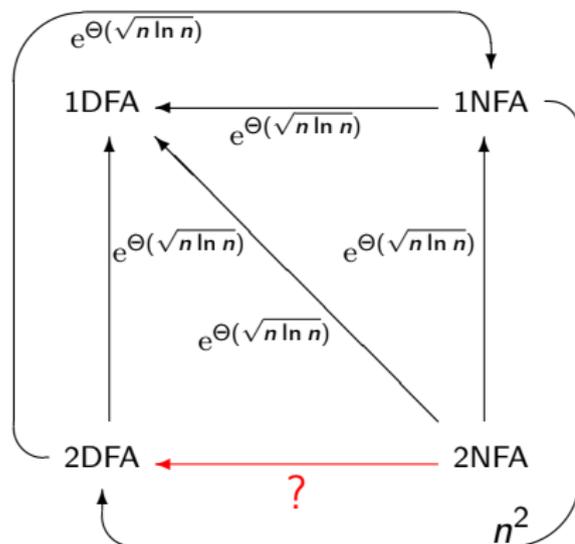
Another possible restriction:

The unary case  $\#\Sigma = 1$



# Optimal Simulation Between Unary Automata

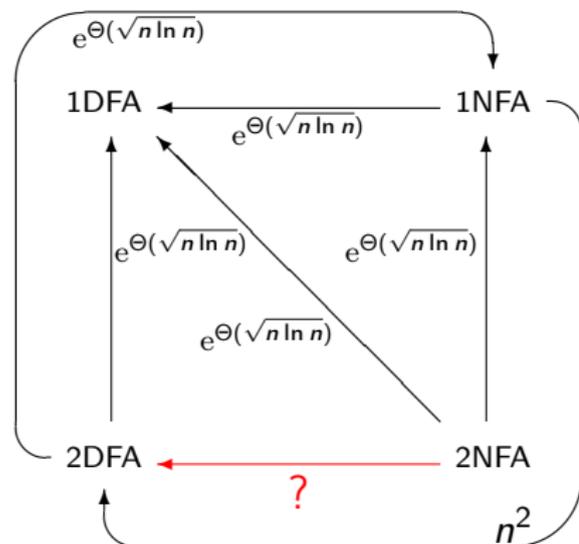
The costs of the optimal simulations between automata are different in the unary and in the general case



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  $\rightarrow$  2DFA

*Even in the unary case  
this question is open!*

- ▶  $e^{\Theta(\sqrt{n \ln n})}$  upper bound  
(from 2NFA  $\rightarrow$  1DFA)
- ▶  $\Omega(n^2)$  lower bound  
(from 1NFA  $\rightarrow$  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&Pighizzini '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&Pighizzini '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&Pighizzini '03]
- (ii) Polynomial complementation of unary 2NFAs  
Inductive counting argument for qsNFAs  
[Geffert Mereghetti&Pighizzini '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&Pighizzini '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

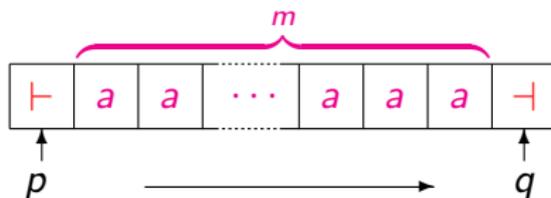
*From now on, we fix an  $n$ -state unary 2NFA  $A$*

- ▶ 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:

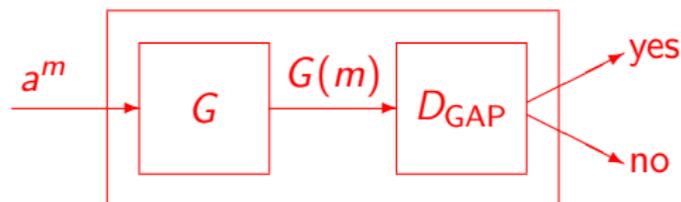
- ▶ the vertices are the states of  $M$
- ▶  $(p, q)$  is an edge iff  $M$  can traverse the input
  - from one endmarker in the state  $p$
  - to the opposite endmarker in the state  $q$
  - without visiting the endmarkers in the meantime

Then

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

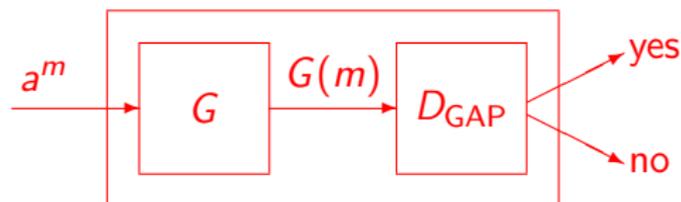
The existence of the edge  $(p, q)$  can be verified by a finite automaton  $A_{p,q}$  with  $N$  states

# 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  $a^m$

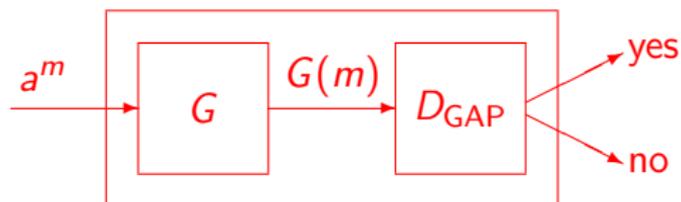
*The worktape of  $D_{GAP}$  can be encoded in a finite control using a number of states polynomial in  $N$*

- ▶ 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”

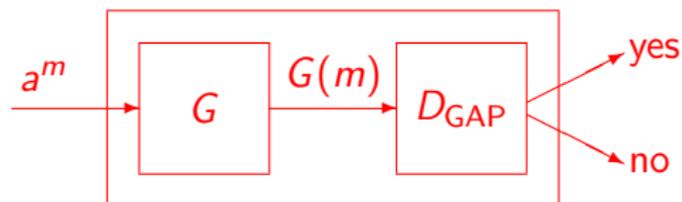
# Polynomial Deterministic Simulation (under $L = NL$ )



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

↓

Quasi Sweeping Simulation

$M$  qsNFA almost equivalent to  $A$   $N \leq 2n + 2$  states

↓

Deterministic Simulation

$M'$  2DFA equivalent to  $M$   $poly(N)$  states

↓

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&Pighizzini '10])

*If  $L = NL$  then each  $n$ -state unary 2NFA can be simulated by an equivalent 2DFA with a polynomial number of states*

Hence

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

Another condition:

## Theorem ([Berman&Lingas '77])

*If  $L = NL$  then there exists a polynomial  $p$  s.t. for each  $m > 0$  and  $k$ -state 2NFA  $A$ , there exists a  $p(mk)$ -state 2DFA  $A'$  s.t.  $L(A') \subseteq L(A)$  and  $L(A) \cap \Sigma^{\leq m} = L(A') \cap \Sigma^{\leq m}$*

Further relationships with logspace complexity in [Kapoutsis '11]

# What About the Converse?

## Question

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

- ▶ 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  $L = NL$*

# Unambiguous Logspace (Nonuniform)

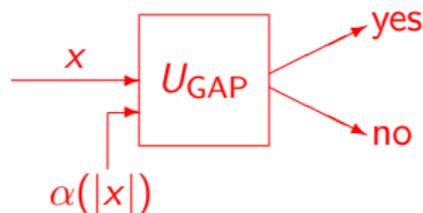
Theorem ([Reinhardt&Allender '00])

$NL \subseteq UL/poly$

- ▶  $UL/poly$   
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&Pighizzini '10])

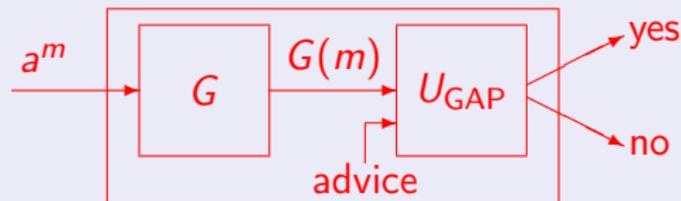
*Each  $n$ -state unary 2NFA can be simulated by an equivalent unambiguous 2NFA with a polynomial number of states*

Proof.

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

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



# Descriptive Complexity of Regular Languages

- ▶ Different variants of finite automata characterize regular languages
- ▶ However, we can describe regular languages using more powerful formalisms or devices, as context-free grammars and pushdown automata

What about the sizes of CFGs or PDAs describing regular languages vs the sizes of finite automata?

# Descriptive Complexity Measures

- ▶ Context-free grammars:

number of variables?

For  $n \geq 1$ , consider the language  $L_n = (a^n)^*$ :

- ▶  $L_n$  requires  $n$  states to be accepted by 1DFAs or 1NFAs
- ▶  $L_n$  is generated by the grammar with one variable  $S$  and the productions

$$S \rightarrow a^n S \quad S \rightarrow \epsilon$$

- ▶ Thus, the number of variables cannot be a descriptive complexity measure for context-free grammars.
- ▶ However, for grammars in *Chomsky Normal Form* the number of variables is a “reasonable” measure of complexity [Gruska '73]

## Context-Free vs Regular: Descriptive Complexity

Given a context-free grammar of size  $n$ , generating a regular language, how much is big an equivalent finite automaton, wrt  $n$  ?

Theorem ([Meyer&Fischer '71])

*For any recursive function  $f$  and arbitrarily large integers  $n$ , there exists a CFG of size  $n$  generating a regular language  $L$ , s.t. any DFA accepting  $L$  must have at least  $f(n)$  states*

Then:

the trade-off between CFG and finite automata is not recursive

However...

the witness language is defined over a binary alphabet

What about unary languages?

# CFGs vs Automata: Unary Case

Theorem ([Ginsburg&Rice, '62])

*Every unary context-free language is regular*

What about descriptonal complexity?

Theorem ([Pighizzini Shallit&Wang '02])

*Given a unary CFG in Chomsky normal form with  $h$  variables, there exist:*

- ▶ *an equivalent 1NFA with at most  $2^{2h-1} + 1$  states*
- ▶ *an equivalent 1DFA with at most  $2^{h^2}$  states*

*These bounds are tight, namely, matching lower bound have been obtained.*

## Final considerations

- ▶ Many results in formal language theory have been revisited and refined considering descriptonal complexity aspects
- ▶ Having descriptions of small size can be very interesting for many applications  
(e.g., small circuits and programs for portable devices)
- ▶ There are strong connections between descriptonal complexity and structural complexity  
(e.g., Sakoda and Sipser question and L vs NL question, machine simulations, similar techniques as crossing sequences, inductive counting, Savitch simulation,...)
- ▶ Other complexity measure deserve further investigation  
(e.g., ambiguity degrees, measures of nondeterminism)

## Final considerations

- ▶ We discussed only a few aspects related to descriptonal complexity of regular languages
- ▶ Many other aspects have been investigated
- ▶ Probably the first paper in the area:  
A.R. Meyer, M.J. Fischer:  
*Economy of Description by Automata, Grammars, and Formal Systems*, FOCS 1971, 188–191
- ▶ An interesting survey:  
J. Goldstine, M. Kappes, C.M.R. Kintala, H. Leung,  
A. Malcher, D. Wotschke:  
*Descriptonal Complexity of Machines with Limited Resources*,  
J. Universal Comp. Science 8(2): 193–234 (2002)
- ▶ Annual international workshop on descriptonal complexity  
**Descriptonal Complexity of Formal Systems (DCFS)**  
Limburg, Germany, July 25-27, 2011.