Removing Nondeterminism from Two-Way Automata

Giovanni Pighizzini

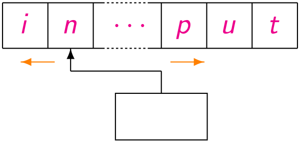
Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano

Porto – June 22, 2010

UNIVERSITÀ DEGLI STUDI
DI MILANO

Outline

Finite State Automata



Base version:

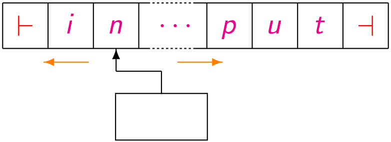
one-way deterministic finite automata (1DFA)

- 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

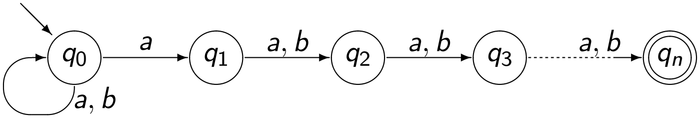
What about the power of these models?

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

...some of them are more succinct

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

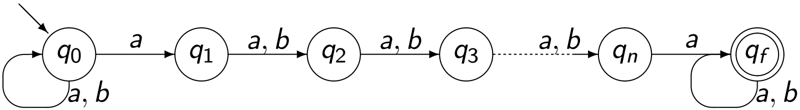
- L is accepted by a 1NFA with $n+1$ states



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

$\vdash$	b	b	a	b	a	a	b	a	a	a	$\dashv$

$n = 4$

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*

Costs of the Optimal Simulations Between Automata

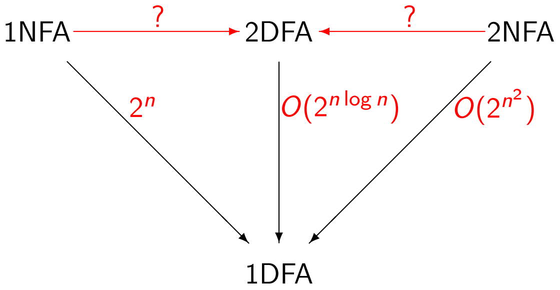
$$
\begin{tikzcd}
\mathrm{1NFA} \arrow[r, "?"] \arrow[dr, "2^n"] & \mathrm{2DFA} \arrow[d, "O(2^{n\log n})"] & \mathrm{2NFA} \arrow[l, "?"'] \arrow[dl, "O(2^{n^2})"] \\
 & \mathrm{1DFA} &
\end{tikzcd}
$$

[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:

- $c(n) \in \Omega(\frac{n^2}{\log n})$ [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]

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]

The Unary Case: $\#\Sigma = 1$

1NFAs vs 2DFAs?

- In the unary case, the cost of the simulation of 1NFAs by 2DFAs is $O(n^2)$ [Chrobak '86]

2NFAs vs 2DFAs?

- Even restricted to the case of unary automata, the problem seems to be difficult
- The unary version of the problem is connected with the important question $\mathsf{L} \stackrel{?}{=} \mathsf{NL}$ in complexity theory
- In the study of unary 2NFA simulations, the notion of *quasi sweeping automata* turns out to be useful. This notion extends sweeping automata with some *restricted nondeterministic capabilities*

Quasi Sweeping Automata

Definition

A 2NFA is quasi sweeping (qsNFA) iff both

- nondeterministic choices and
- head reversals

are possible only at the endmarkers

Computation of a qsNFA:

- sequence of *deterministic* left-to-right and right-to-left traversals of the input
- *nondeterministic* choices at the endmarkers,
  i.e., only immediately before/after input traversals

Quasi Sweeping Simulation of Unary 2NFAs

[Geffert Mereghetti&Pighizzini '03]

Definition

Two finite automata are *almost equivalent* iff they accept the same language, with the possible exception of a finite number of strings

Theorem (Quasi Sweeping Simulation)

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

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

Quasi Sweeping Simulation: Outline

Let A be a given n-state unary 2NFA

- Wlog suppose that A accepts in a fixed state q_f with the input head scanning the left endmarker
- Given an accepting computation:
  $r_0, r_1, \ldots, r_p$ sequence of states reached *at the endmarkers*
  (hence $r_0 = q_0$, $r_p = q_f$)
- A *segment* is the part of computation from r_{i-1} to r_i
- Two kinds of segments:
  - traversals
  - U-turns
- The simulation "simplifies" these segments

Quasi Sweeping Simulation: Traversals

Traversal: segment of computation starting at one endmarker and ending at the opposite one

- Traversals in 2NFAs can be very complicated
- However, in the unary case:

Lemma ([Geffert '91])

For each traversal from a state r_{i-1} to a state r_i there exists another traversal from r_{i-1} to r_i with the following structure:

- *initial and final parts consuming $O(n^2)$ input symbols*
- *in the middle: a "dominant" loop, visiting $\leq n$ cells, is repeated many times*

- Hence, traversals are essentially used to compute the input length modulo one integer
- They can be simulated by *deterministic loops* and *nondeterministic moves at the endmarkers*

Quasi Sweeping Simulation: U-Turns

U-turn: segment of computation starting and ending at the same endmarker

Lemma ([Geffert '91])

For each U-turn from a state r_{i-1} to a state r_i
there exists another U-turn from r_{i-1} to r_i
in which the input head is never moved farther than n^2 cells
from the corresponding endmarker

Hence:

- Each "long" U-turn can be replaced by a shorter U-turn
- For sufficiently long inputs ($> n^2$):
  - the set of possible U-turns can be precomputed
  - U-turns can be replaced by stationary moves at the endmarkers

Quasi Sweeping Simulation

Theorem (Quasi Sweeping Simulation)

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

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

Proof.

From the above arguments:

- Traversals are simulated by *deterministic loops* and *nondeterministic moves at the endmarkers*
- *U*-turns are replaced by stationary moves
- This can be implemented using at most $2n + 2$ states
- Possible "errors" on inputs of length $\leq 5n^2$ $\square$

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 unambiguous 2NFAs

(iv) Relationship with the space complexity question $\mathsf{L} \stackrel{?}{=} \mathsf{NL}$

We are going to discuss (iv) and (iii) [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 Conditional Simulation

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

- We describe how to reduce the membership problem for $L(A)$ to the problem GAP

  i.e, for each input string a^m we define a graph $G(m)$ s.t.

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

- *Under the hypothesis* $\mathrm{L} = \mathrm{NL}$
  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

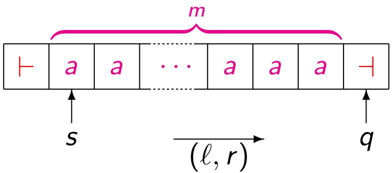
Polynomial Deterministic Conditional Simulation

A	fixed unary 2NFA	n states
$\Downarrow$		Quasi Sweeping Simulation
M	almost equivalent qsNFA	$N \leq 2n+2$ states

From now on, also M is fixed

- $L(M)$ and $L(A)$ can differ only on strings of length $\leq 5n^2$
- The computation of M is a sequence of traversals of the input
- The states used in each traversal form a *deterministic loop*
- Nondeterministic choices possible *only at the endmarkers*
- M has exactly one final state q_F
- q_F can be reached only at the left endmarker

Describing M Computations



Traversal of the input a^m

- starting from the leftmost input symbol in a state s
- moving at each step to the right
- finally reaching the right endmarker in a state q

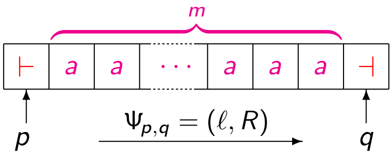
Then:

- s and q must belong to a same deterministic loop
- q depends on $m \bmod \ell$, where ℓ is the length of the loop

IDEA: Associate with (s, q), the pair of integers (ℓ, r) s.t.:

$$\text{there is a traversal of } a^m \text{ from } s \text{ to } q \iff m \bmod \ell = r$$

Describing M Computations



However a traversal starts on the left endmarker

- we consider states p such that $p \xrightarrow{\vdash} s$
- actually, we associate the pair (ℓ, r) with (p, q).

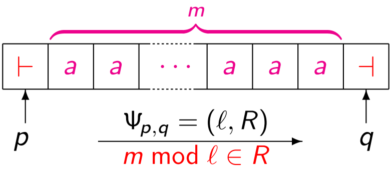
How many pairs (ℓ, r) can be associated with the same (p, q)?

- q belongs to a deterministic loop: only one possible ℓ
- on the left endmarked nondeterministic moves are possible: $p \xrightarrow{\vdash} s'$ and $p \xrightarrow{\vdash} s''$, for different s', s'' in the same loop of q, produce different remainders r: a set of possible remainders

With (p, q) we associate $\Psi_{p,q} = (\ell, R)$, where $R \subseteq \{0, \ldots, \ell - 1\}$

Similar argument for traversals from right to left

Describing M Computations



By summarizing:

Lemma

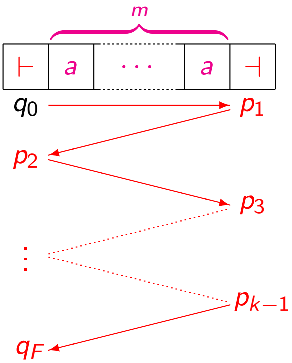
For all states p, q, input a^m, the automaton M

- *starting from one endmarker in the state p*
- *can reach the opposite endmarker in the state q*
- *without any visit of the endmarkers in the meantime*

if and only if

- $\Psi_{p,q} = (\ell, R)$ *and* $m \bmod \ell \in R$

An Accepting Computation



$$
\begin{array}{ll}
m \bmod \ell_1 \in R_1 & \Psi_{q_0,p_1} = (\ell_1, R_1) \\
m \bmod \ell_2 \in R_2 & \Psi_{p_1,p_2} = (\ell_2, R_2) \\
m \bmod \ell_3 \in R_3 & \Psi_{p_2,p_3} = (\ell_3, R_3) \\
\vdots & \vdots \\
m \bmod \ell_k \in R_k & \Psi_{p_{k-1},q_F} = (\ell_k, R_k)
\end{array}
$$

For each accepting computation
all these conditions are satified

Conversely:

- Each sequence of states $q_0 = p_1, p_2, \ldots, p_{k-1}, p_k = q_F$
  s.t. $m \bmod \ell_i \in R_i \quad (i = 1, \ldots, k)$
  describes an accepting computation for a^m

Reducing Membership for $L(M)$ to GAP

With each input a^m we associate the following graph $G(m)$:

- Vertex set Q, the set of states of M
- Edge sets $E(m)$

$$(p, q) \in E(m) \;\text{ iff }\; m \bmod \ell \in R, \text{ where } \Psi_{p,q} = (\ell, R)$$

namely

The graph contains the edge (p, q) if and only if there is a traversal from p to q on input a^m

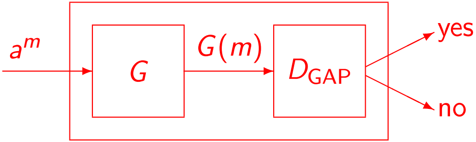
Lemma

The input a^m is accepted if and only if the graph $G(m)$ contains a path from q_0 to q_F

Hence:

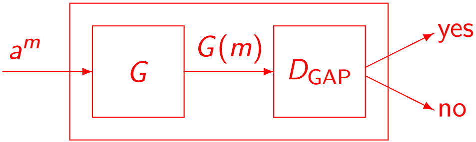
To decide whether or not $a^m \in L(M)$ reduces to decide GAP for $G(m)$

Deterministic simulation



- Suppose $\mathrm{L} = \mathrm{NL}$
- 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 $a^m \in L(M)$

Deterministic simulation



- 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

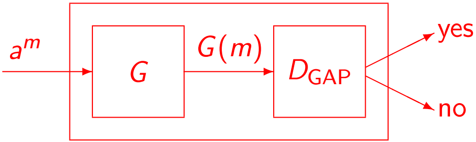
  The worktape of D_{GAP} can be encoded in a finite control using a number of states polynomial in N

- 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 we compute input bits for D_{GAP} "on demand"

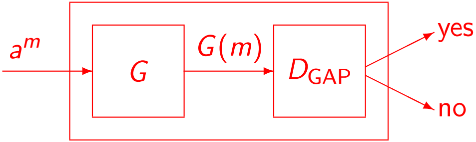
Deterministic simulation



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

Deterministic simulation



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 $A_{p,q}$ computes the input bit corresponding to (p, q)
- $A_{p,q}$ traverses the input a^m to check whether or not the machine M can make a traversal from p to q
- $A_{p,q}$ can be implemented using no more than N states
- 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&Pighizzini '10])

If $\mathsf{L} = \mathsf{NL}$ *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 $\mathsf{L} = \mathsf{NL}$ *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* $L(A) \cap \Sigma^{\leq m} = L(A') \cap \Sigma^{\leq m}$

What About the Converse?

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 $\mathsf{L} = \mathsf{NL}$

Unambiguous Logspace (Nonuniform)

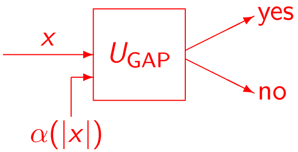
Theorem ([Reinhardt&Allender '00])

$\mathsf{NL} \subseteq \mathsf{UL}/poly$

- $\mathsf{UL}/poly$
  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

$\mathsf{GAP} \in \mathsf{UL}/poly$



Making Unary 2NFAs Unambiguous

Theorem ([Geffert&Pighizzini '10])

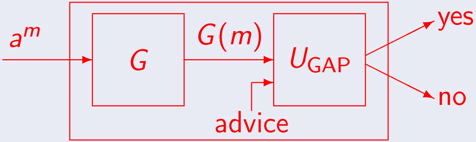
Each n-state unary 2NFA can be simulated by an equivalent unambiguous 2NFA with a polynomial number of states

Proof.

- Similar to the polynomial deterministic conditional simulation
- Hypothetical machine D_{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



Sakoda&Sipser Question: Current Knowledge

- Upper bounds

	1NFA→ 2DFA	2NFA→ 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]

Conclusion

- Unary automata look as very simple computational models
  - finite state control
  - contentless input

  However, their investigation shows strong connections with fundamental questions
- The study of the Sakoda&Sipser question looks interesting and challenging also in the unary case
- Several connections between descriptional complexity and space complexity have been discovered.

  Many techniques from one of the two fields, turn out to be useful in the other one, e.g.,
  - Savitch theorem
  - Inductive counting
  - Pumping arguments
  - Crossing sequences
  - ...