Limited Automata and Descriptional Complexity

Giovanni Pighizzini
(joint papers with Andrea Pisoni – DCFS 2013, NCMA 2013)

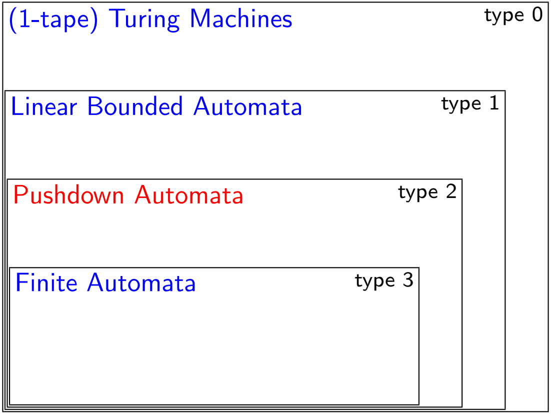
Dipartimento di Informatica
Università degli Studi di Milano, Italy

Riunione PRIN 2013

Milano, 28–29 novembre 2013

UNIVERSITÀ DEGLI STUDI
DI MILANO

The Chomsky Hierarchy



(1-tape) Turing Machines

type 0

Linear Bounded Automata

type 1

Pushdown Automata

type 2

Finite Automata

type 3

Limited Automata [Hibbard'67]

One-tape Turing machines with restricted rewritings

Definition

Fixed an integer $d \geq 1$, a *d-limited automaton* is

- a one-tape Turing machine
- which is allowed to rewrite the content of each tape cell *only in the first d visits*

Limited Automata [Hibbard'67]

One-tape Turing machines with restricted rewritings

Definition

Fixed an integer $d \geq 1$, a *d-limited automaton* is

- a one-tape Turing machine
- which is allowed to rewrite the content of each tape cell *only in the first d visits*

Limited Automata [Hibbard'67]

One-tape Turing machines with restricted rewritings

Definition

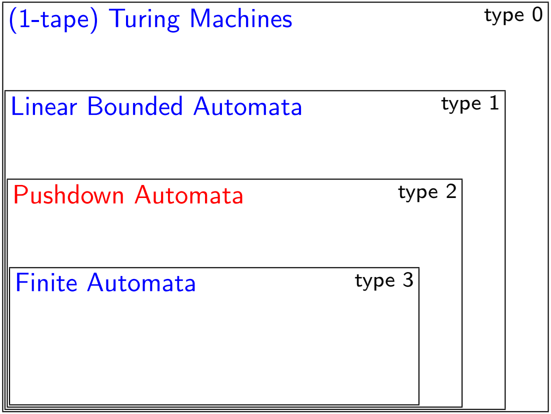
Fixed an integer $d \geq 1$, a *d-limited automaton* is

- a one-tape Turing machine
- which is allowed to rewrite the content of each tape cell *only in the first d visits*

Computational power

- For each $d \geq 2$, d-limited automata characterize context-free languages [Hibbard'67]
- 1-limited automata characterize regular languages [Wagner&Wechsung'86]

The Chomsky Hierarchy



(1-tape) Turing Machines

type 0

Linear Bounded Automata

type 1

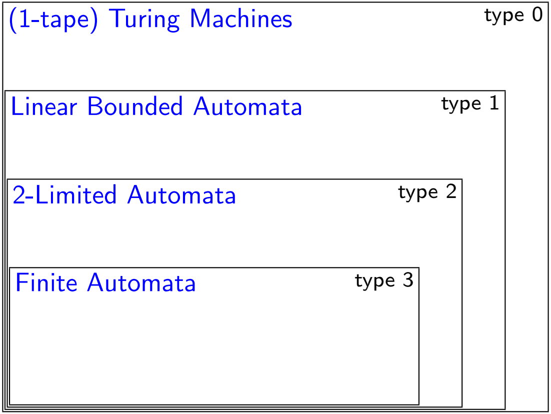
Pushdown Automata

type 2

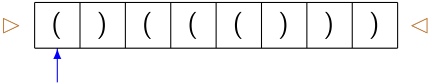
Finite Automata

type 3

The Chomsky Hierarchy



Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

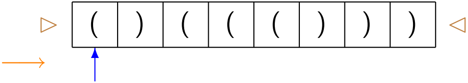
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



▷ ( ) ((( ))) ◁

(i) Move to the right to search a closed parenthesis

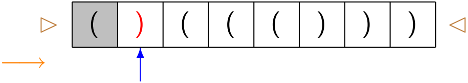
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

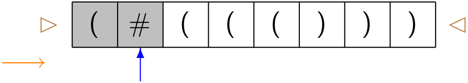
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

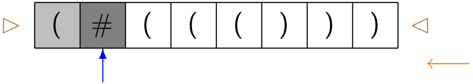
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

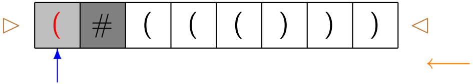
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

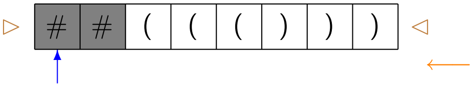
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



▷	#	#	(	(	(	)	)	)	◁

(i) Move to the right to search a closed parenthesis

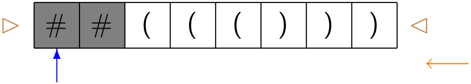
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

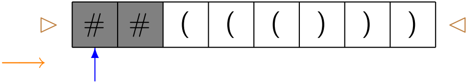
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



▷	#	#	(	(	(	)	)	)	◁

(i) Move to the right to search a closed parenthesis

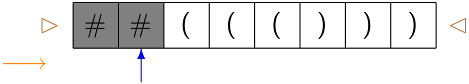
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

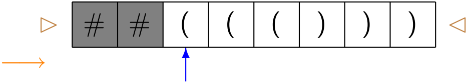
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

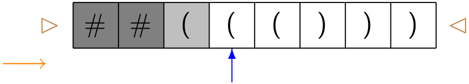
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

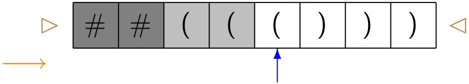
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

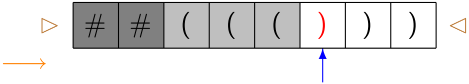
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

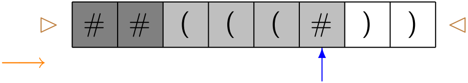
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

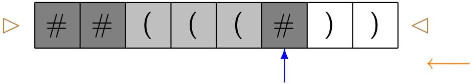
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

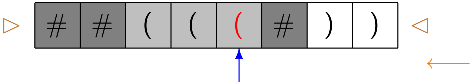
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

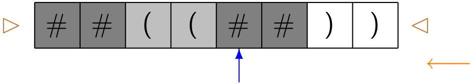
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



▷	#	#	(	(	#	#	)	)	◁

(i) Move to the right to search a closed parenthesis

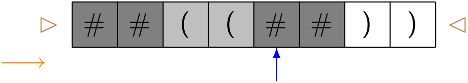
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

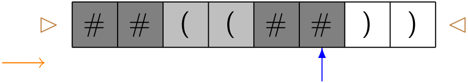
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

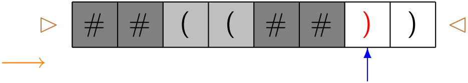
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

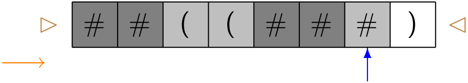
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



▷	#	#	(	(	#	#	#	)	◁

(i) Move to the right to search a closed parenthesis

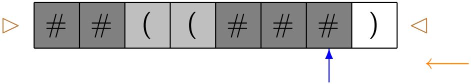
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

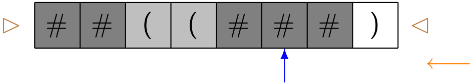
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

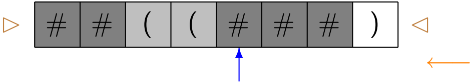
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

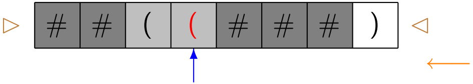
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

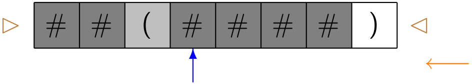
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



▷	#	#	(	#	#	#	#	)	◁

(i) Move to the right to search a closed parenthesis

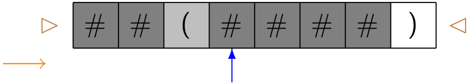
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

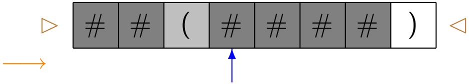
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

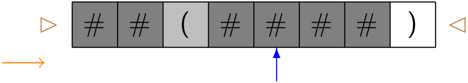
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

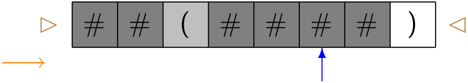
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

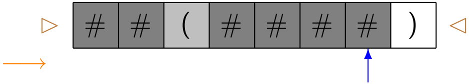
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



#	#	(	#	#	#	#	)

(i) Move to the right to search a closed parenthesis

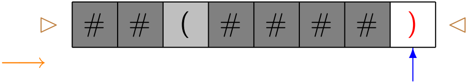
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

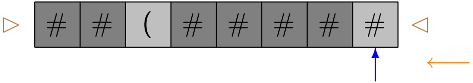
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



▷ # # (# # # # # ◁

(i) Move to the right to search a closed parenthesis

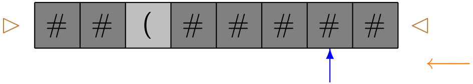
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

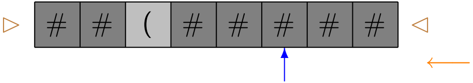
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

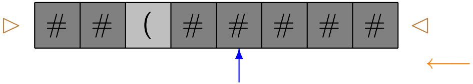
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

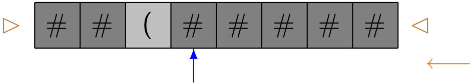
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

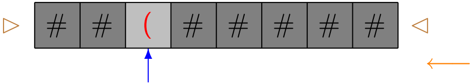
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



▷ # # (# # # # # ◁

(i) Move to the right to search a closed parenthesis

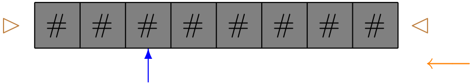
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

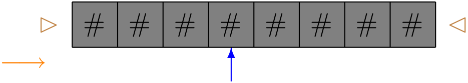
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

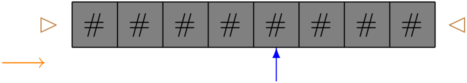
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



▷ # # # # # # # # ◁

(i) Move to the right to search a closed parenthesis

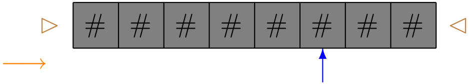
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

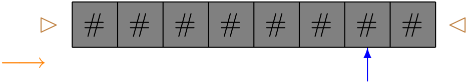
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

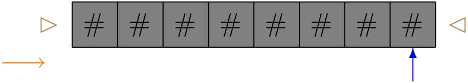
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

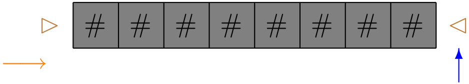
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

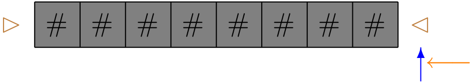
(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

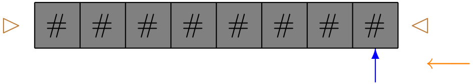
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

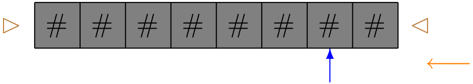
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

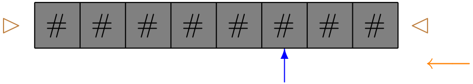
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

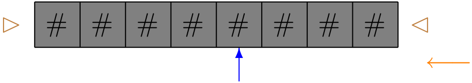
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

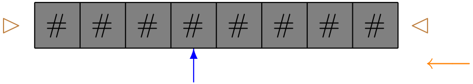
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

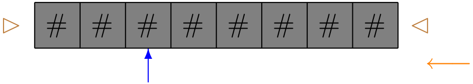
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

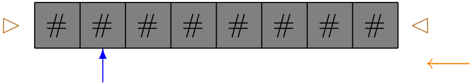
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

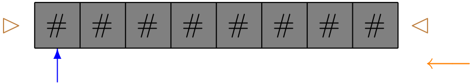
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



▷ # # # # # # # # ◁

(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

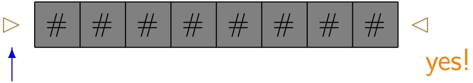
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

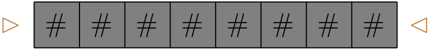
(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses



▷ # # # # # # # # ◁

(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Example: Balanced Parentheses

▷	#	#	#	#	#	#	#	#	◁

(i) Move to the right to search a closed parenthesis

(ii) Rewrite it by #

(iii) Move to the left to search an open parenthesis

(iv) Rewrite it by #

(v) Repeat from the beginning

Special cases:

(i') If in (i) the right end of the tape is reached then scan all the tape and *accept* iff all tape cells contain #

(iii') If in (iii) the left end of the tape is reached then *reject*

Each cell is rewritten only in the first 2 visits!

2-Limited Automata → Pushdown Automata

Problem

How much it costs, in the description size, the simulation of 2-LAs *by* PDAs*?*

Our result

Exponential cost!
(optimal)

2-Limited Automata → Pushdown Automata

Problem

How much it costs, in the description size, the simulation of 2-LAs *by* PDAs*?*

Our result

Exponential cost!
(optimal)

2-Limited Automata → Pushdown Automata

Problem

How much it costs, in the description size, the simulation of 2-LAs *by* PDAs*?*

Our result

Exponential cost!
(optimal)

Deterministic case

- Determinism is preserved by the simulation *provided that the input of the* PDA *is right end-marked*
- Without end-marker: *double exponential* size for the simulation of D2-LAs by DPDA
- Conjecture: this cost cannot be reduced

2-Limited Automata → Pushdown Automata

Problem

How much it costs, in the description size, the simulation of 2-LAs *by* PDAs*?*

Our result

Exponential cost!
(optimal)

Deterministic case

- Determinism is preserved by the simulation *provided that the input of the* PDA *is right end-marked*
- Without end-marker: *double exponential* size for the simulation of D2-LAs by DPDA
- Conjecture: this cost cannot be reduced

2-Limited Automata → Pushdown Automata

Problem

How much it costs, in the description size, the simulation of 2-LAs *by* PDAs*?*

Our result

Exponential cost!
(optimal)

Deterministic case

- Determinism is preserved by the simulation
provided that the input of the PDA *is right end-marked*
- Without end-marker: *double exponential* size for the simulation of D2-LAs by DPDA
- Conjecture: this cost cannot be reduced

CFLs → 2-Limited Automata

New trasformation based on:

Theorem ([Chomsky&Schützenberger'63])

Every context-free language $L \subseteq \Sigma^*$ can be expressed as

$$L = h(D_k \cap R)$$

where, for $\Omega_k = \{(_1,)_1, (_2,)_2, \ldots, (_k,)_k\}$:

- $D_k \subseteq \Omega_k^*$ *is a Dyck language*
- $R \subseteq \Omega_k^*$ *is a regular language*
- $h : \Omega_k \to \Sigma^*$ *is an homomorphism*

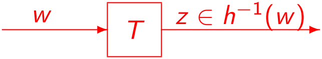
Furthermore, it is possible to restrict to *non-erasing* homomorphisms [Okhotin'12]

CFLs → 2-Limited Automata

L context-free language, with $L = h(D_k \cap R)$

- T nondeterministic transducer computing h^{-1}
- A_D 2-LA accepting the Dyck language D_k
- A_R finite automaton accepting R

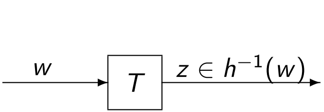
CFLs → 2-Limited Automata



L context-free language, with $L = h(D_k \cap R)$

- T nondeterministic transducer computing h^{-1}
- A_D 2-LA accepting the Dyck language D_k
- A_R finite automaton accepting R

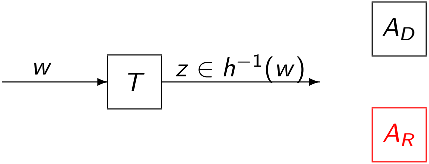
CFLs → 2-Limited Automata



L context-free language, with $L = h(D_k \cap R)$

- T nondeterministic transducer computing h^{-1}
- A_D 2-LA accepting the Dyck language D_k
- A_R finite automaton accepting R

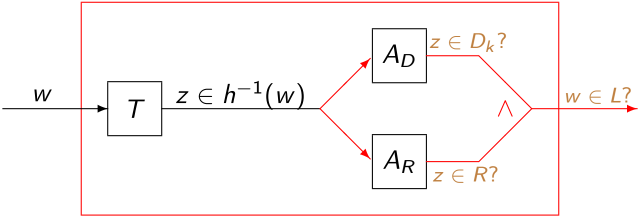
CFLs → 2-Limited Automata



L context-free language, with $L = h(D_k \cap R)$

- T nondeterministic transducer computing h^{-1}
- A_D 2-LA accepting the Dyck language D_k
- A_R finite automaton accepting R

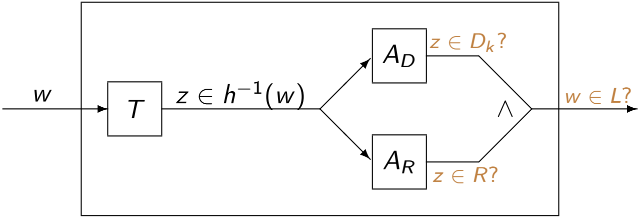
CFLs → 2-Limited Automata



L context-free language, with $L = h(D_k \cap R)$

- T nondeterministic transducer computing h^{-1}
- A_D 2-LA accepting the Dyck language D_k
- A_R finite automaton accepting R

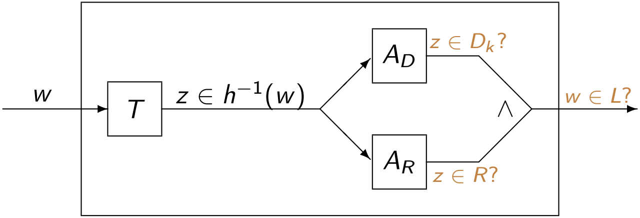
CFLs → 2-Limited Automata



$$\underbrace{\boxed{\quad w \quad}}_{\text{input of } T}$$

$$z = \sigma_1\sigma_2 \cdots \sigma_k \in h^{-1}(w)$$

CFLs $\longrightarrow$ 2-Limited Automata



$$\underbrace{\begin{array}{|c|c|c|c|}\hline u_1 & u_2 & \cdots & u_k \\ \hline\end{array}}_{\text{input of } T}$$

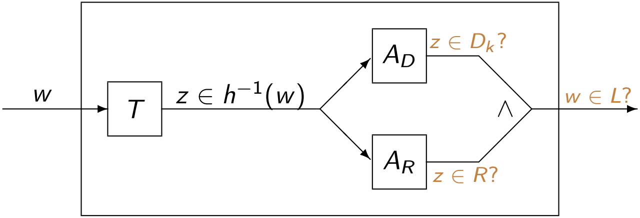
$$z = \sigma_1\sigma_2\cdots\sigma_k \in h^{-1}(w)$$

$$h(\sigma_i) = u_i$$

$$\begin{array}{|c|c|c|c|}\hline \#\#\#\#\sigma_1 & \#\#\sigma_2 & \cdots & \#\#\#\sigma_k \\ \hline\end{array}$$

Non erasing homomorphism!

CFLs → 2-Limited Automata



$$\underbrace{\begin{array}{|c|c|c|c|}\hline \quad u_1 \quad & \quad u_2 \quad & \cdots & \quad u_k \quad \\ \hline \end{array}}_{\text{input of } T}$$

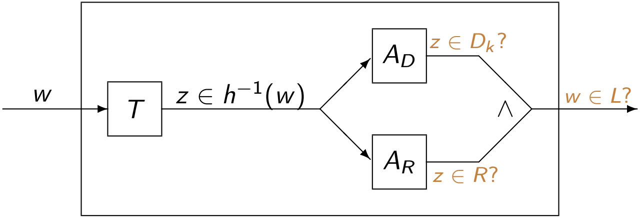
$$z = \sigma_1 \sigma_2 \cdots \sigma_k \in h^{-1}(w)$$

$$h(\sigma_i) = u_i$$

$$\underbrace{\begin{array}{|c|c|c|c|}\hline \#\#\#\#\sigma_1 & \#\#\sigma_2 & \cdots & \#\#\#\sigma_k \\ \hline \end{array}}_{\text{(padded) input of } A_D \text{ and } A_R}$$

Non erasing homomorphism!

CFLs → 2-Limited Automata



$$\underbrace{\begin{array}{|c|c|c|c|}\hline u_1 & u_2 & \cdots & u_k \\ \hline\end{array}}_{\text{input of } T}$$

$$z = \sigma_1\sigma_2\cdots\sigma_k \in h^{-1}(w)$$

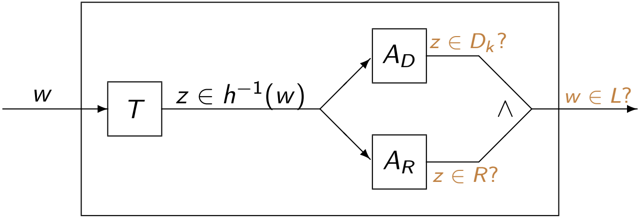
$$h(\sigma_i) = u_i$$

$$\underbrace{\begin{array}{|c|c|c|c|}\hline \#\#\#\#\sigma_1 & \#\#\sigma_2 & \cdots & \#\#\#\sigma_k \\ \hline\end{array}}_{\text{(padded) input of } A_D \text{ and } A_R}$$

Not stored into the tape!

Non erasing homomorphism!

CFLs → 2-Limited Automata



$$\underbrace{\begin{array}{|c|c|c|c|}\hline u_1 & u_2 & \cdots & u_k \\ \hline\end{array}}_{\text{input of } T}$$

$$z = \sigma_1\sigma_2 \cdots \sigma_k \in h^{-1}(w)$$

$$h(\sigma_i) = u_i$$

$$\underbrace{\begin{array}{|c|c|c|c|}\hline \#\#\#\#\sigma_1 & \#\#\sigma_2 & \cdots & \#\#\#\sigma_k \\ \hline\end{array}}_{\text{(padded) input of } A_D \text{ and } A_R}$$

Non erasing homomorphism!

Not stored into the tape!

$\boxed{\#\#\#\#\gamma_1}$ . . .

Each σ_i is produced "on the fly" and replaced by γ_i, its first rewriting by A_D

Pushdown Automata → 2-Limited Automata

PDAs → 2-LAs

Polynomial cost!

(in the description size)

Pushdown Automata → 2-Limited Automata

PDAs → 2-LAs

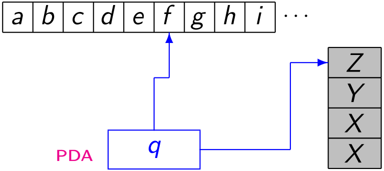
Polynomial cost!

DPDAs → D2-LAs

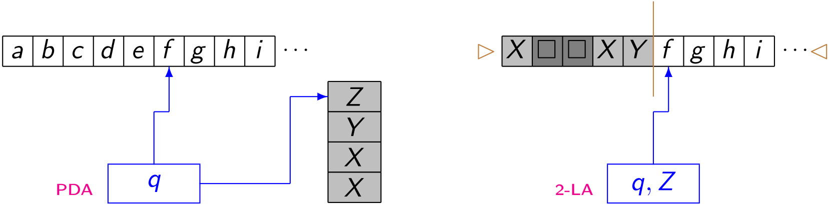
Polynomial cost!

(in the description size)

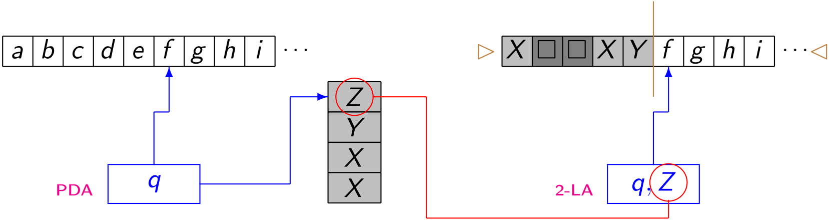
Pushdown Automata → 2-Limited Automata



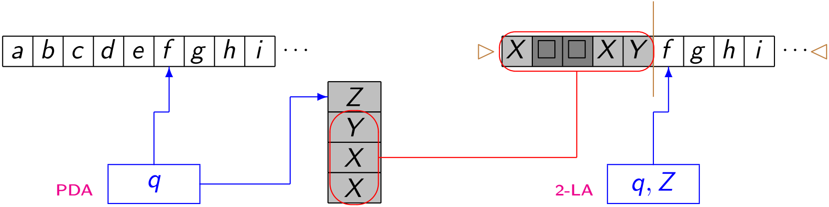
Pushdown Automata → 2-Limited Automata



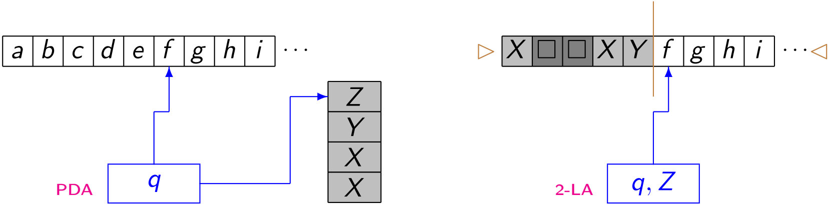
Pushdown Automata → 2-Limited Automata



Pushdown Automata → 2-Limited Automata



Pushdown Automata → 2-Limited Automata



Normal form for (D)PDAs:

- at each step, the stack height increases at most by 1
- ϵ-moves cannot push on the stack

Each (D)PDA can be simulated by an equivalent (D)2-LA of polynomial size

2-Limited Automata ≡ Pushdown Automata

Summing up...

- Descriptional complexity point of view

2-LAs → PDAs

Exponential gap

PDAs → 2-LAs

Polynomial upper bound

2-Limited Automata ≡ Pushdown Automata

Summing up...

- Descriptional complexity point of view

2-LAs → PDAs

Exponential gap

PDAs → 2-LAs

Polynomial upper bound

- Determinism vs Nondeterminism

Deterministic Context-Free Languages ≡ Deterministic 2-LAs

2-Limited Automata $\equiv$ Pushdown Automata

Summing up...

- Descriptional complexity point of view

2-LAs → PDAs

Exponential gap

PDAs → 2-LAs

Polynomial upper bound

- Determinism vs Nondeterminism

Deterministic Context-Free Languages $\equiv$ Deterministic 2-LAs

On the other hand:

$$L = \{a^n b^n c \mid n \geq 0\} \cup \{a^n b^{2n} d \mid n \geq 0\} \in det\text{-}3\text{-LA} - \text{DCFL}$$

2-Limited Automata $\equiv$ Pushdown Automata

Summing up...

- Descriptional complexity point of view

2-LAs → PDAs

Exponential gap

PDAs → 2-LAs

Polynomial upper bound

- Determinism vs Nondeterminism

Deterministic Context-Free Languages $\equiv$ *Deterministic* 2-LAs

On the other hand:

$$L = \{a^n b^n c \mid n \geq 0\} \cup \{a^n b^{2n} d \mid n \geq 0\} \in \mathit{det}\text{-3-LA} - \text{DCFL}$$

Infinite hierarchy [Hibbard'67]:

$$\mathit{det}\text{-}d\text{-LA} \supset \mathit{det}\text{-}(d-1)\text{-LA}, \text{ for each } d \geq 2$$

1-Limited Automata → Finite Automata

Costs in states of the optimal simulations of n-state 1-LAs by finite automata:

	DFA	NFA
nondet. 1-LA		
det. 1-LA		

These upper bounds do not depend on the alphabet size of M!

1-Limited Automata → Finite Automata

Costs in states of the optimal simulations of n-state 1-LAs by finite automata:

	DFA	NFA
nondet. 1-LA	$2^{n \cdot 2^{n^2}}$	
det. 1-LA		

These upper bounds do not depend on the alphabet size of M!

1-Limited Automata → Finite Automata

Costs in states of the optimal simulations of n-state 1-LAs by finite automata:

	DFA	NFA
nondet. 1-LA	$2^{n \cdot 2^{n^2}}$	$n \cdot 2^{n^2}$
det. 1-LA		

These upper bounds do not depend on the alphabet size of M!

1-Limited Automata → Finite Automata

Costs in states of the optimal simulations of n-state 1-LAs by finite automata:

	DFA	NFA
nondet. 1-LA	$2^{n \cdot 2^{n^2}}$	$n \cdot 2^{n^2}$
det. 1-LA	$n \cdot (n+1)^n$	$n \cdot (n+1)^n$

These upper bounds do not depend on the alphabet size of M!

1-Limited Automata → Finite Automata

Costs in states of the optimal simulations of n-state 1-LAs by finite automata:

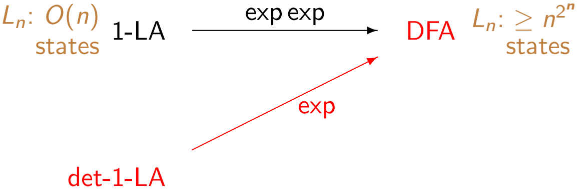
	DFA	NFA
nondet. 1-LA	$2^{n \cdot 2^{n^2}}$	$n \cdot 2^{n^2}$
det. 1-LA	$n \cdot (n+1)^n$	$n \cdot (n+1)^n$

These upper bounds do not depend on the alphabet size of M!

Nondetermism vs Determinism in 1-LAs

L_n: $O(n)$ states 1-LA $\xrightarrow{\text{exp exp}}$ DFA L_n: $\geq n^{2^n}$ states

Nondetermism vs Determinism in 1-LAs



L_n: $O(n)$ states

1-LA

exp exp

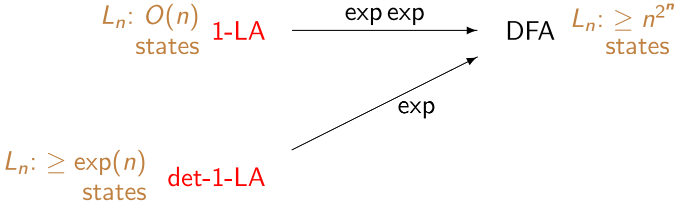
DFA

L_n: $\geq n^{2^n}$ states

exp

det-1-LA

Nondetermism vs Determinism in 1-LAs



L_n: $O(n)$ states 1-LA

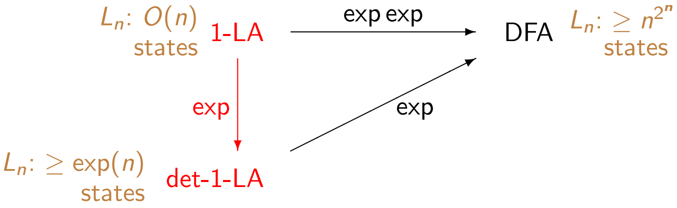
exp exp

DFA L_n: $\geq n^{2^n}$ states

exp

L_n: $\geq \exp(n)$ states det-1-LA

Nondetermism vs Determinism in 1-LAs



L_n: $O(n)$ states

1-LA

exp exp

DFA

L_n: $\geq n^{2^n}$ states

exp

exp

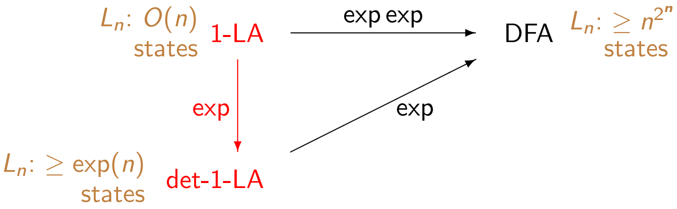
L_n: $\geq \exp(n)$ states

det-1-LA

Corollary

Removing nondeterminism from 1-LAs requires exponentially many states

Nondetermism vs Determinism in 1-LAs



L_n: $O(n)$ states

1-LA

exp exp

DFA

L_n: $\geq n^{2^n}$ states

exp

exp

L_n: $\geq \exp(n)$ states

det-1-LA

Corollary

Removing nondeterminism from 1-LAs *requires exponentially many states*

Cfr. Sakoda and Sipser question [Sakoda&Sipser'78]:

How much it costs in states to remove nondeterminism from two-way finite automata?

Futher Investigations

- Descriptional complexity aspects for $d > 2$

  We conjecture that for $d > 2$ the size gap from d-limited automata to PDAs remains exponential

- Descriptional complexity aspects in the unary case
  - Unary context-free language are regular [Ginbsurg&Rice'62]
  - Ex: $L_n = (a^{2^n})^*$

	size
2-LA	$O(n)$
DPDA	$O(n)$
minimal DFA	2^n
minimal 2NFA	2^n

Futher Investigations

- Descriptional complexity aspects for $d > 2$

  We conjecture that for $d > 2$ the size gap from d-limited automata to PDAs remains exponential

- Descriptional complexity aspects in the unary case
  - Unary context-free language are regular [Ginbsurg&Rice'62]
  - Ex: $L_n = (a^{2^n})^*$

	size
2-LA	$O(n)$
DPDA	$O(n)$
minimal DFA	2^n
minimal 2NFA	2^n

Futher Investigations

- Descriptional complexity aspects for $d > 2$

  We conjecture that for $d > 2$ the size gap from d-limited automata to PDAs remains exponential

- Descriptional complexity aspects in the unary case
  - Unary context-free language are regular [Ginbsurg&Rice'62]
  - Ex: $L_n = (a^{2^n})^*$

	size
2-LA	$O(n)$
DPDA	$O(n)$
minimal DFA	2^n
minimal 2NFA	2^n

Futher Investigations

- Descriptional complexity aspects for $d > 2$

  We conjecture that for $d > 2$ the size gap from d-limited automata to PDAs remains exponential

- Descriptional complexity aspects in the unary case
  - Unary context-free language are regular [Ginbsurg&Rice'62]
  - Ex: $L_n = (a^{2^n})^*$

	size
2-LA	$O(n)$
DPDA	$O(n)$
minimal DFA	2^n
minimal 2NFA	2^n