Universality and Computational Completeness of Controlled Leftist Insertion-Deletion Systems

Sergiu Ivanov   Serghei Verlan

Université Paris-Est Créteil

Université Grenoble-Alpes

February 6, LIFO

Insertion-deletion Systems

$(u, x, v)_{ins}$

Insertion-deletion Systems

$$(u, x, v)_{ins}$$

$$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$$

Insertion-deletion Systems

$(u, x, v)_{ins}$ $\qquad$ $(u, x, v)_{del}$

$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$ $\qquad$ $\cdots u x v \cdots \Longrightarrow \cdots u \quad v \cdots$

Insertion-deletion Systems

$(u, x, v)_{ins}$ $\qquad$ $(u, x, v)_{del}$

$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$ $\qquad$ $\cdots u x v \cdots \Longrightarrow \cdots u \quad v \cdots$

Insertion-deletion system = {insertion rules,
deletion rules,
axioms}

Insertion-deletion Systems

Size:

$\big(\text{len}(x), \text{len}(u), \text{len}(v)\big)$

$(u, x, v)_{ins}$ $\qquad$ $(u, x, v)_{del}$

$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$ $\qquad$ $\cdots u x v \cdots \Longrightarrow \cdots u \quad v \cdots$

Insertion-deletion system = {insertion rules,
deletion rules,
axioms}

Insertion-deletion Systems

Size:

$$(\text{len}(x), \text{len}(u), \text{len}(v))$$

$$(u, x, v)_{ins}$$

$$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$$

Size:

$$(\text{len}(x), \text{len}(u), \text{len}(v))$$

$$(u, x, v)_{del}$$

$$\cdots u x v \cdots \Longrightarrow \cdots u \quad v \cdots$$

Insertion-deletion system = {insertion rules,
deletion rules,
axioms}

Insertion-deletion Systems

Size:

$(\mathrm{len}(x), \mathrm{len}(u), \mathrm{len}(v))$

$(u, x, v)_{ins}$

$\cdots u \quad v \cdots \Longrightarrow \cdots u x v \cdots$

Size:

$(\mathrm{len}(x), \mathrm{len}(u), \mathrm{len}(v))$

$(u, x, v)_{del}$

$\cdots u x v \cdots \Longrightarrow \cdots u \quad v \cdots$

Insertion-deletion system = {insertion rules,
deletion rules,
axioms}

System size = $(\underbrace{n, m, m'}_{\text{max insertion rule size}};\ \underbrace{p, q, q'}_{\text{max deletion rule size}})$

max insertion rule size

max deletion rule size

Leftist Insertion-deletion Systems

$(u, x, \lambda)_{\substack{ins \\ del}}$

https://openclipart.org/

Leftist Insertion-deletion Systems

only left context

$$(u, x, \lambda)_{\substack{ins \\ del}}$$

one symbol inserted / deleted

https://openclipart.org/

Leftist Insertion-deletion Systems

only left context

$$(u,\ x,\ \lambda)_{\substack{ins\\del}}$$

one symbol inserted deleted

Facts:

$(1, 1, 0; 1, 1, 0)$

- $\not\ni (ab)^*$
- generate non-regular context-free languages

https://openclipart.org/

Leftist Insertion-deletion Systems

only left context

$$(u,\ x,\ \lambda)_{\substack{ins\\del}}$$

one symbol inserted / deleted

Facts:

$(1, 1, 0; 1, 1, 0)$

- $\not\ni (ab)^*$
- generate non-regular context-free languages

$(1, m, 0; 1, q, 0), \quad m \cdot q \geq 2$

- generate all REG
- many more Easter eggs 

https://openclipart.org/

Regular Contexts

regular expressions for contexts

$$((ab)^+, x, (cd)^+)_{del}$$

https://openclipart.org/

Regular Contexts

regular expressions for contexts

$$((ab)^+, x, (cd)^+)_{del}$$

matches

$$\ldots\ a\ b\ a\ b\ \ x\ \ c\ d\ c\ d\ \ \ldots$$

The match need not be greedy.

https://openclipart.org/

Regular Contexts

regular expressions for contexts

$$((ab)^+, x, (cd)^+)_{del}$$

matches

$$\ldots\ a\ b\ a\ b\ \ x\ \ c\ d\ c\ d\ \ldots$$

The match need not be greedy.

Size notation: $(1, REG, 2; 1, 2, REG)$

- regular left insertion contexts
- regular right deletion contexts

https://openclipart.org/

Outline

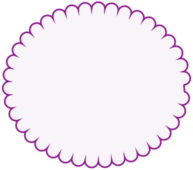
1. Completeness and Universality
2. Universality of Leftist Systems
3. Completeness of Leftist Systems

Outline

Computational Completeness and Universality

Computational Completeness

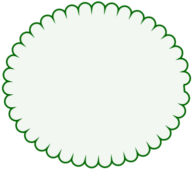
Class of devices



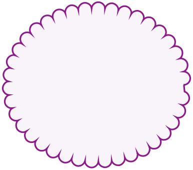
Computational Completeness and Universality

Computational Completeness

Turing machines

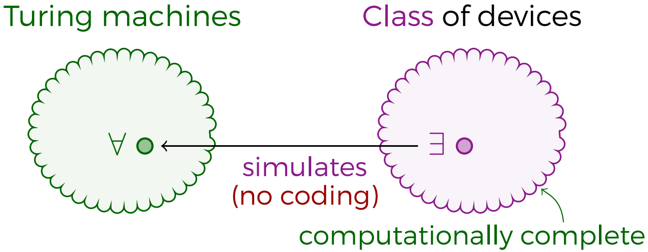


Class of devices



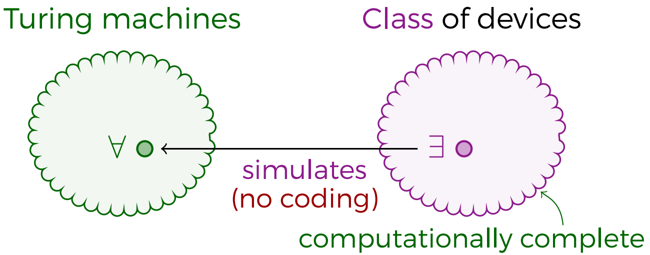
Computational Completeness and Universality

Computational Completeness

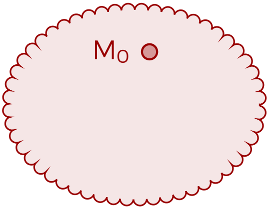


Computational Completeness and Universality

Computational Completeness

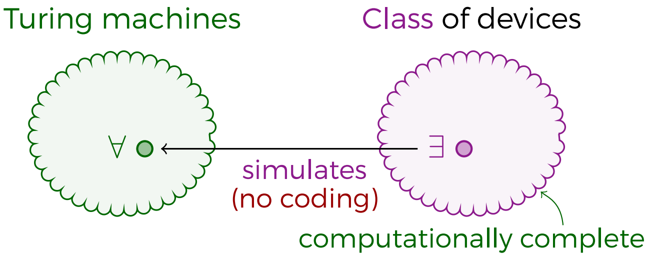


Universality



Computational Completeness and Universality

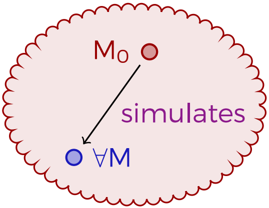
Computational Completeness



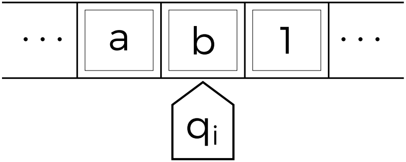
Universality

M_0 is universal if it can simulate any other M from the same class.

- coding allowed



Turing Machines



We use a special (complete!) subclass:

- no state loops
- either move or write
- no read when move

2-Tag Systems

Context-free string rewriting.

erase on the left

append on the right

2-Tag Systems

Context-free string rewriting.

erase on the left

append on the right

$$a \rightarrow x\,y\,z$$

$$\underbrace{a\;b}_{2}\,c\;d \quad\Longrightarrow\quad c\;d\;x\;y\;z$$

2-Tag Systems

Context-free string rewriting.

erase on the left — append on the right

$$a \to x\,y\,z$$

$$\underbrace{a\,b}_{2}\,c\,d \implies c\,d\,x\,y\,z$$

Halt when the string starts with the halting symbol h.

2-Tag Systems

Context-free string rewriting.

erase on the left — append on the right

$$a \to x\,y\,z$$

$$\underbrace{a\,b}_{2}\,c\,d \quad \Longrightarrow \quad c\,d\,x\,y\,z$$

Halt when the string starts with the halting symbol h.

2-tag systems are universal.

- generate any RE language, modulo a coding

Outline

1. Completeness and Universality
2. Universality of Leftist Systems
3. Completeness of Leftist Systems

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) > (1, 2, 0; 1, 1, 0)$?

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) > (1, 2, 0; 1, 1, 0)$?

In fact, $(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) \sim (1, 2, 0; 1, 1, 0)$

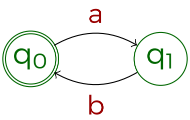
Regular contexts can be checked by "small" rules.

$(1, REG, 0; 1, REG, 0) > (1, 2, 0; 1, 1, 0)$?

In fact, $(1, REG, 0; 1, REG, 0) \sim (1, 2, 0; 1, 1, 0)$

Regular contexts can be checked by "small" rules.

Consider $((ab)^*, c, \lambda)_{ins}$

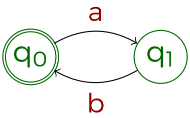


$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) > (1, 2, 0; 1, 1, 0)$?

In fact, $(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) \sim (1, 2, 0; 1, 1, 0)$

Regular contexts can be checked by "small" rules.

Consider $((ab)^*, c, \lambda)_{ins}$



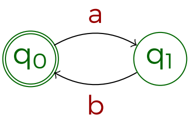
$\ldots$ x a b a y $\ldots$

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) > (1, 2, 0; 1, 1, 0)$?

In fact, $(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) \sim (1, 2, 0; 1, 1, 0)$

Regular contexts can be checked by "small" rules.

Consider $((ab)^*, c, \lambda)_{ins}$



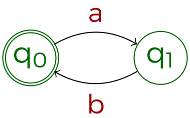
$\ldots$ x q_0 a b a y $\ldots$

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) > (1, 2, 0; 1, 1, 0)$?

In fact, $(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) \sim (1, 2, 0; 1, 1, 0)$

Regular contexts can be checked by "small" rules.

Consider $((ab)^*, c, \lambda)_{ins}$

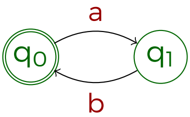


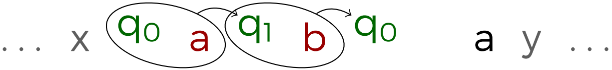
$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) > (1, 2, 0; 1, 1, 0)$?

In fact, $(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) \sim (1, 2, 0; 1, 1, 0)$

Regular contexts can be checked by "small" rules.

Consider $((ab)^*, c, \lambda)_{ins}$



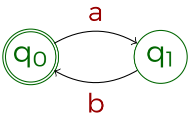


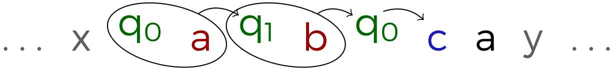
$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) > (1, 2, 0; 1, 1, 0)$?

In fact, $(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) \sim (1, 2, 0; 1, 1, 0)$

Regular contexts can be checked by "small" rules.

Consider $((ab)^*, c, \lambda)_{ins}$



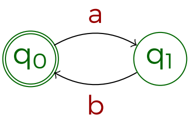


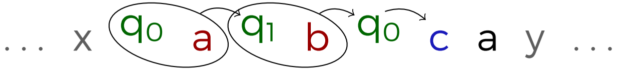
$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) > (1, 2, 0; 1, 1, 0)$?

In fact, $(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) \sim (1, 2, 0; 1, 1, 0)$

Regular contexts can be checked by "small" rules.

Consider $((ab)^*, c, \lambda)_{ins}$





- only final states insert c
- clean-up at any moment

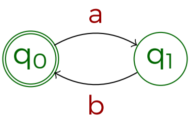
insertion size $= (1, 2, 0)$

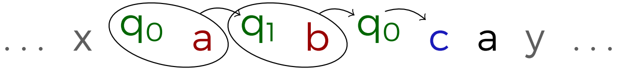
$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) > (1, 2, 0; 1, 1, 0)$?

In fact, $(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) \sim (1, 2, 0; 1, 1, 0)$

Regular contexts can be checked by "small" rules.

Consider $((ab)^*, c, \lambda)_{ins}$





- only final states insert c
- clean-up at any moment

insertion size $= (1, 2, 0)$

Similarly, $(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) \sim (1, 1, 0; 1, 2, 0)$

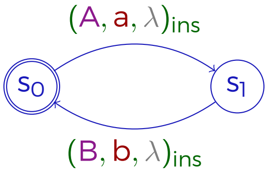
Graph Control

Put insertion/deletion rules on graph edges.

Graph Control

Put insertion/deletion rules on graph edges.

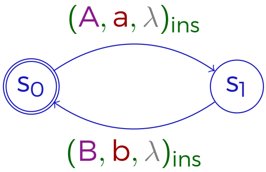
$$A\,B \Longrightarrow^{2n} A\,a^n\,B\,b^n$$



Graph Control

Put insertion/deletion rules on graph edges.

$$A\,B \Longrightarrow^{2n} A\,a^n\,B\,b^n$$



Strictly increases the power of rules.

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) + \mathrm{GC}_2 \sim$ Tag Systems

Form of the string:

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) + GC_2 \sim$ Tag Systems

Form of the string:

1 states 2

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) + GC_2 \sim$ Tag Systems

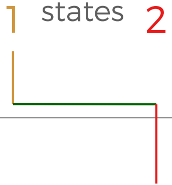
Form of the string:

1 states 2

generate control word

$(1, \text{REG}, 0; 1, \text{REG}, 0) + GC_2 \sim$ Tag Systems

Form of the string:

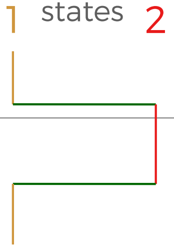


generate control word

insert service symbols (phase I)

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) + GC_2 \sim$ Tag Systems

Form of the string:



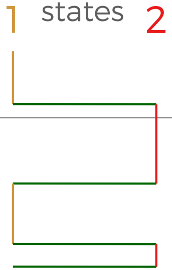
generate control word

insert service symbols (phase I)

erase on the left

$(1, \text{REG}, 0; 1, \text{REG}, 0) + GC_2 \sim$ Tag Systems

Form of the string:



generate control word

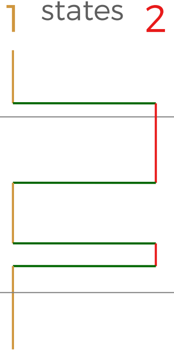
insert service symbols (phase I)

erase on the left

insert service symbols (phase II)

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) + GC_2 \sim$ Tag Systems

Form of the string:



generate control word

---

insert service symbols (phase I)

erase on the left

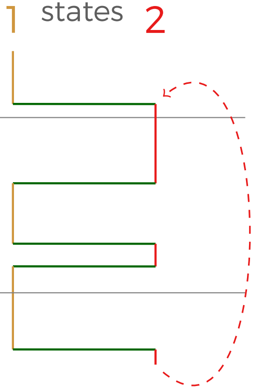
insert service symbols (phase II)

---

insert right-hand side

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) + GC_2 \sim$ Tag Systems

Form of the string:



generate control word

insert service symbols (phase I)

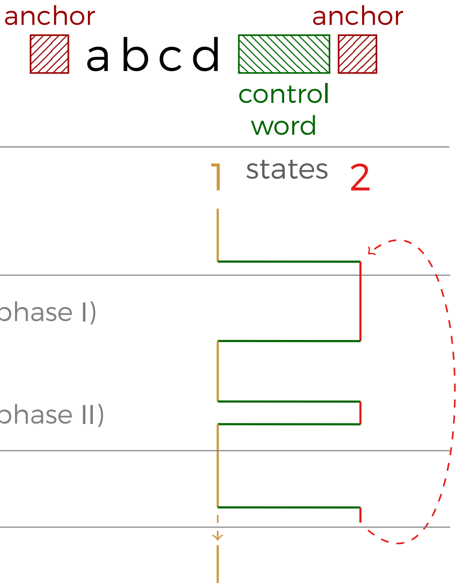
erase on the left

insert service symbols (phase II)

insert right-hand side

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) + GC_2 \sim$ Tag Systems

Form of the string:



generate control word

insert service symbols (phase I)

erase on the left

insert service symbols (phase II)

insert right-hand side

cleanup

$(1, \substack{2\\1}, 0;\ 1, \substack{1\\2}, 0) + GC_2 \sim$ Tag Systems

Simulate the construction for

$(1, REG, 0; 1, REG, 0) + GC_2$

by systems of types

- $(1, 2, 0; 1, 1, 0) + GC_2$
- $(1, 1, 0; 1, 2, 0) + GC_2$

$(1, \frac{2}{1}, 0; 1, \frac{1}{2}, 0) + GC_2 \sim$ Tag Systems

Simulate the construction for

$(1, REG, 0; 1, REG, 0) + GC_2$

by systems of types

- $(1, 2, 0; 1, 1, 0) + GC_2$
- $(1, 1, 0; 1, 2, 0) + GC_2$

https://openclipart.org/

$(1, \frac{2}{1}, 0;\ 1, \frac{1}{2}, 0) + GC_2 \sim$ Tag Systems

Simulate the construction for

$$(1, REG, 0; 1, REG, 0) + GC_2$$

by systems of types

- $(1, 2, 0; 1, 1, 0) + GC_2$
- $(1, 1, 0; 1, 2, 0) + GC_2$

This simulation does not work in general:

$$(1, REG, 0; 1, REG, 0) + GC \succ (1, 2, 0; 1, 1, 0) + GC$$

$$(1, REG, 0; 1, REG, 0) + GC \succ (1, 1, 0; 1, 2, 0) + GC$$

https://openclipart.org/

Outline

Matrix Insertion-deletion Systems

#REG

- group rules into matrices
- all rules in matrices must be applied, in order

$$\Big((BT^*q_i a, q_j, \lambda)_{ins} \, , \; (BT^*, q_i, \lambda)_{del} \Big)$$

Matrix Insertion-deletion Systems

#REG

- group rules into matrices
- all rules in matrices must be applied, in order

$$\Big((BT^* q_i a, q_j, \lambda)_{ins}\ ,\ \ (BT^*, q_i, \lambda)_{del} \Big)$$

$$\ldots\ q_i\ a\ \ \ldots$$

Matrix Insertion-deletion Systems

#REG

- group rules into matrices
- all rules in matrices must be applied, in order

$$\left((BT^{*}q_i a, q_j, \lambda)_{ins} \, , \; (BT^{*}, q_i, \lambda)_{del} \right)$$

$$\ldots \; q_i \; a \; q_j \; \ldots$$

Matrix Insertion-deletion Systems

- group rules into matrices
- all rules in matrices must be applied, in order

$$\Big((BT^* q_i a, q_j, \lambda)_{ins} \,,\; (BT^*, q_i, \lambda)_{del} \Big)$$

$$\ldots \; \not{q}_i \; a \; q_j \; \ldots$$

Matrix Insertion-deletion Systems

- group rules into matrices
- all rules in matrices must be applied, in order

$$\left((BT^* q_i a, q_j, \lambda)_{ins} \,, \quad (BT^*, q_i, \lambda)_{del} \right)$$

$$\ldots \, \not{q}_i \; a \;\; q_j \, \ldots$$

$$(1, REG, 0; 1, REG, 0) + Mat_2 = RE$$

- generate all recurively enumerable languages
- no coding

Random Context Insertion-deletion Systems #REG

Add permitting / forbidding context conditions to rules.

$$\Big((BT^*, \bar{q}_j, \lambda)_{ins} \,,\; \{q_i\} \,,\; \{\bar{q}_j\} \Big)$$

insert $\bar{q}_j$

if q_i is present

if $\bar{q}_j$ is absent

Random Context Insertion-deletion Systems #REG

Add permitting / forbidding context conditions to rules.

$$\left((BT^*, \bar{q}_j, \lambda)_{ins} \,,\; \{q_i\} \,,\; \{\bar{q}_j\} \right)$$

insert $\bar{q}_j$

if q_i is present

if $\bar{q}_j$ is absent

Context conditions give sufficient visibility on the $\overrightarrow{\text{right}}$.

$$(1, REG, 0; 1, REG, 0) + RC = RE$$

Random Context Insertion-deletion Systems #REG

Add permitting / forbidding context conditions to rules.

$$\Big((BT^*, \bar{q}_j, \lambda)_{ins} \, , \; \{q_i\} \, , \; \{\bar{q}_j\} \Big)$$

insert $\bar{q}_j$

if q_i is present

if $\bar{q}_j$ is absent

Context conditions give sufficient visibility on the right.

$$(1, REG, 0; 1, REG, 0) + RC = RE$$

- poor visibility $\Longrightarrow$ more complex proof

Graph Controlled Insertion-deletion Systems #REG

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) + GC_2 \sim$ Tag Systems

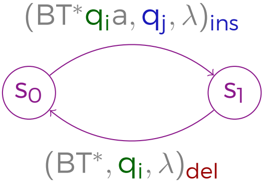
- universality
- coding

Graph Controlled Insertion-deletion Systems #REG

$(1, \mathrm{REG}, 0; 1, \mathrm{REG}, 0) + GC_2 \sim$ Tag Systems

- universality
- coding

But, in fact:

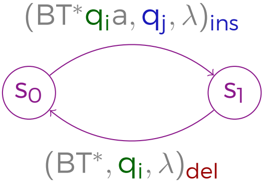


Graph Controlled Insertion-deletion Systems #REG

$(1, \text{REG}, 0; 1, \text{REG}, 0) + GC_2 \sim$ Tag Systems

- universality
- coding

But, in fact:



$+$ some more effort for rewriting

$\Downarrow$

$(1, \text{REG}, 0; 1, \text{REG}, 0) + GC_2 = \text{RE}$

two states

Conclusions and Future Work

- Introduced regular contexts.
- Proved universality for regular contexts + graph control.
- Proved universality for $(1, \frac{2}{1}, 0;\ 1, \frac{1}{2}, 0)$ + graph control.
- Proved computational completeness (RE) of regular contexts + control.

? Power without control mechanisms

? Power with regular contexts + bigger $\frac{\text{inserted}}{\text{deleted}}$ strings