Regulated Insertion-Deletion Systems

Dr. Lakshmanan Kuppusamy
VIT, Vellore, INDIA.

December 2, 2019

Outline of my previous talk

1. Can we simulate Type-0 grammars by Type-2 grammars if we regulate the rule applications in some manner?

Outline of my previous talk

1. Can we simulate Type-0 grammars by Type-2 grammars if we regulate the rule applications in some manner?
2. YES !! but with certain regulations on the contexts of application like

Outline of my previous talk

1. Can we simulate Type-0 grammars by Type-2 grammars if we regulate the rule applications in some manner?
2. YES !! but with certain regulations on the contexts of application like
3. Semi-Conditional grammars
4. Simple Semi-Conditional grammars
5. Generalised Forbidding grammars
6. Matrix grammars (we did not discuss this)
7. Graph-Controlled grammars (we did not discuss this)

Insertion-Deletion Systems

Insertion-Deletion Systems

A counterpart of Rewriting Systems

Theoretical meaning of ins-del

- Insertion (Deletion) means appending (removing) a (sub)string to (from) a given string with specific contexts.
- This is not Rewriting and motivation comes from DNA.
- If a string α is inserted between two parts w_1 and w_2 of a string $w_1 w_2$ to get $w_1 \alpha w_2$, the operation is *insertion*.
- Notation: $\boxed{(w_1, \alpha, w_2)_{ins}}$: means $(w_1 w_2 \Longrightarrow w_1 \alpha w_2)$

Theoretical meaning of ins-del

- Insertion (Deletion) means appending (removing) a (sub)string to (from) a given string with specific contexts.
- This is not Rewriting and motivation comes from DNA.
- If a string α is inserted between two parts w_1 and w_2 of a string w_1w_2 to get $w_1\alpha w_2$, the operation is *insertion*.
- Notation: $\boxed{(w_1, \alpha, w_2)_{ins}}$: means $(w_1w_2 \Longrightarrow w_1\alpha w_2)$
- If a substring β is deleted from a string $w_1\beta w_2$ to get w_1w_2, the operation is *deletion*.
- Notation: $\boxed{(w_1, \beta, w_2)_{del}}$: means $(w_1\beta w_2 \Longrightarrow w_1w_2\ )$
- Suffixes of w_1 and prefixes of w_2 are called the left and right context of α or β.
- Starting with axioms and iterating the ins-del operations, we get a set of terminal strings (language of ins-del system).

Definition

An insertion-deletion system is a construct $G = (V, T, A, R)$

- V is an alphabet, $T \subseteq V$, $A \subseteq V^*$
- R is a finite set of n rules of the form $\boxed{(u_i, \alpha_i, v_i)_t}$
  $t \in \{ins, del\}, 1 \leq i \leq n, \ \ u_i, v_i \in V^*, \ \alpha_i \in V^+.$

Definition

An insertion-deletion system is a construct $G = (V, T, A, R)$

- V is an alphabet, $T \subseteq V$, $A \subseteq V^*$
- R is a finite set of n rules of the form $\boxed{(u_i, \alpha_i, v_i)_t}$
  $t \in \{ins, del\}, 1 \leq i \leq n,\ \ u_i, v_i \in V^*,\ \ \alpha_i \in V^+.$

Size of an Ins-Del (ID) system

Notation: $(\boxed{n, i', i''}; \boxed{m, j', j''})$ where

1. n = the maximal length of the insertion string
2. i' = maximal length of left contexts used in insertion rules
3. i'' = maximal length of right contexts used in insertion rules
4. m, j', j'' denote similar maximal lengths among deletion rules.

Ins-del systems for $\{a^n b^n \mid n \geq 1\}$

$G_1 =$
$(\{a, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, ab, b)_{ins}$

Size $= (2, 1, 1; 0, 0, 0)$.

Ins-del systems for $\{a^n b^n \mid n \geq 1\}$

$G_1 = (\{a, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, ab, b)_{ins}$

Size $= (2, 1, 1; 0, 0, 0)$.

Can generate more grammars for the same language?

Ins-del systems for $\{a^n b^n \mid n \geq 1\}$

$G_1 =$
$(\{a, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, ab, b)_{ins}$

Size $= (2, 1, 1; 0, 0, 0)$.

Can generate more grammars for the same language?

$G_2 =$
$(\{a, X, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, X, b)_{ins}$
- $r_2 : (X, ab, b)_{ins}$
- $r_3 : (\lambda, X, \lambda)_{del}$

Size $= (2, 1, 1; 1, 0, 0)$.

Ins-del systems for $\{a^n b^n \mid n \geq 1\}$

$G_1 = (\{a, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, ab, b)_{ins}$

Size $= (2, 1, 1; 0, 0, 0)$.

Can generate more grammars for the same language?

$G_2 = (\{a, X, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, X, b)_{ins}$
- $r_2 : (X, ab, b)_{ins}$
- $r_3 : (\lambda, X, \lambda)_{del}$

Size $= (2, 1, 1; 1, 0, 0)$.

$G_3 = (\{a, C, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, aC, b)_{ins}$
- $r_2 : (a, b, C)_{ins}$
- $r_3 : (b, C, b)_{del}$

Size $= (2, 1, 1; 1, 1, 1)$.

Ins-del systems for $\{a^n b^n \mid n \geq 1\}$

$G_1 = (\{a, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, ab, b)_{ins}$

Size $= (2, 1, 1; 0, 0, 0)$.

Can generate more grammars for the same language?

$G_2 = (\{a, X, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, X, b)_{ins}$
- $r_2 : (X, ab, b)_{ins}$
- $r_3 : (\lambda, X, \lambda)_{del}$

Size $= (2, 1, 1; 1, 0, 0)$.

$G_3 = (\{a, C, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, aC, b)_{ins}$
- $r_2 : (a, b, C)_{ins}$
- $r_3 : (b, C, b)_{del}$

Size $= (2, 1, 1; 1, 1, 1)$.

$G_4 = (\{a, \$, Y, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, aY, b)_{ins}$
- $r_2 : (a, b\$, Y)_{ins}$
- $r_3 : (b, \$Y, b)_{del}$

Size $= (2, 1, 1; 2, 1, 1)$.

Ins-del systems for $\{a^n b^n \mid n \geq 1\}$

$G_1 = (\{a, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, ab, b)_{ins}$

Size $= (2, 1, 1; 0, 0, 0)$.

Can generate more grammars for the same language?

$G_2 = (\{a, X, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, X, b)_{ins}$
- $r_2 : (X, ab, b)_{ins}$
- $r_3 : (\lambda, X, \lambda)_{del}$

Size $= (2, 1, 1; 1, 0, 0)$.

$G_3 = (\{a, C, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, aC, b)_{ins}$
- $r_2 : (a, b, C)_{ins}$
- $r_3 : (b, C, b)_{del}$

Size $= (2, 1, 1; 1, 1, 1)$.

$G_4 = (\{a, \$, Y, b\}, \{a, b\}, \{ab\}, R)$

- $r_1 : (a, aY, b)_{ins}$
- $r_2 : (a, b\$, Y)_{ins}$
- $r_3 : (b, \$Y, b)_{del}$

Size $= (2, 1, 1; 2, 1, 1)$.

$\{a^n b^n\} \in ID(2, 1, 1; 0, 0, 0)$.

Trivial yet important result

- If $L \in ID(s_1, s_2, s_3; s_4, s_5, s_6)$, then $L \in ID(t_1, t_2, t_3; t_4, t_5, t_6)$ for every $t_i \geq s_i$. Objective: Minimize the s_i's.
- If $L \in ID(s_1, s_2, s_3; s_4, s_5, s_6)$, then $L^r \in ID(s_1, s_3, s_2; s_4, s_6, s_5)$.
- If $\mathcal{L}$ is a language class that is closed under reversal and $\mathcal{L} = ID(s_1, s_2, s_3; s_4, s_5, s_6)$, then $\mathcal{L} = ID(s_1, s_3, s_2; s_4, s_6, s_5)$.
- Implication: If $RE = ID(1, 1, 0; 1, 0, 1)$ implies $RE = ID(1, 0, 1; 1, 1, 0)$.

With what sizes does an ID system (not known to) **characterize RE** ?

- $\boxed{(1, 1, 1; 1, 1, 1)}$
- $(1, 1, 1; 2, 0, 0)$
- $(2, 0, 0; 1, 1, 1)$
- $(2, 0, 0; 3, 0, 0)$
- $(3, 0, 0; 2, 0, 0)$

Classic Result 2017

For $i' + i'', j' + j'' \neq 0$,
ID$(2, i', i''; 2, j', j'')$=RE

$\boxed{\text{ID}(2, 0, 0; 2, 0, 0) \neq \mathrm{RE}}$

- $\boxed{(1, 1, 0; 1, 1, 1)}$
- $\boxed{(1, 1, 1; 1, 1, 0)}$
- $\boxed{(1, 1, 0; 1, 1, 0)}$
- $\boxed{(1, 1, 1; 1, 0, 0)}$
- $\boxed{(1, 0, 0; 1, 1, 1)}$
- $(1, 1, 0; 2, 0, 0)$
- $(2, 0, 0; 1, 1, 0)$
- and so on...

Variants of ins-del system

- Ins-del P systems by Krishna and Rama (2001)
- Tissue P systems with ins-del rules by Lakshmanan and Rama (2003)
- Graph-controlled ins-del systems by R Freund *et al* (2010).
- Matrix ins-del systems by Lakshmanan and Anand Mahendran (2011) and independently by I Petre and S Verlan (2012)
- Semi-conditional and Random Context ins-del systems by S Ivanov and S Verlan (2011)
- Generalized forbidding ins-del systems by S Ivanov and S Verlan (2011)

Variants of ins-del system

- Ins-del P systems by Krishna and Rama (2001)
- Tissue P systems with ins-del rules by Lakshmanan and Rama (2003)
- Graph-controlled ins-del systems by R Freund *et al* (2010).
- Matrix ins-del systems by Lakshmanan and Anand Mahendran (2011) and independently by I Petre and S Verlan (2012)
- Semi-conditional and Random Context ins-del systems by S Ivanov and S Verlan (2011)
- Generalized forbidding ins-del systems by S Ivanov and S Verlan (2011)

Common objective

To characterize recursively enumerable languages using any of the above regulated system with as minimal size/resource as possible.
To do so, we use Special Geffert Normal Form of type-0 grammars.

Special Geffert Normal Form (SGNF)

Definition

A type-0 grammar $G = (N, T, P, S)$ is in SGNF if

- N is partitioned into $N = N_1 \cup N_2$, where $N_2 = \{A, B, C, D\}$ and N_1 contains at least the two non-terminals S and S',
- The rules in P are of the form :

$\boxed{p : X \rightarrow bY,\ q : X \rightarrow Yb,\ h : S' \rightarrow \lambda,\ f : AB \rightarrow \lambda,\ g : CD \rightarrow \lambda.}$ where $X, Y \in N_1$, $X \neq Y$, $b \in T \cup N_2$ and p,q,h,f,g are labels.

Special Geffert Normal Form (SGNF)

Definition

A type-0 grammar $G = (N, T, P, S)$ is in SGNF if

- N is partitioned into $N = N_1 \cup N_2$, where $N_2 = \{A, B, C, D\}$ and N_1 contains at least the two non-terminals S and S',
- The rules in P are of the form :

$\boxed{p : X \to bY,\ q : X \to Yb,\ h : S' \to \lambda,\ f : AB \to \lambda,\ g : CD \to \lambda.}$ where $X, Y \in N_1$, $X \neq Y$, $b \in T \cup N_2$ and p,q,h,f,g are labels.

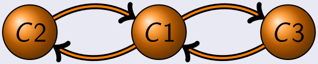
- In Phase I, the (linear-like) CF rules are applied and completed by applying $S' \to \lambda$.
- Adv. At any instant of string in the sentential form, there is only ONE variable from N_1 (No confusion of twins!).
- In Phase II, only $AB \to \lambda,\ CD \to \lambda$ rules are applied.

Graph-Controlled Insertion-Deletion (GCID)

Definition

- A GCID system is $\Pi = (k, V, T, A, H, i_0, i_f, R)$
- k is the number of components
- V is an alphabet, $T \subseteq V$, A is an axiom set, H is a label set.
- i_0 is the initial component and i_f is the final component.
- A rule in R is of the form $\ell : (i, (w_1, \alpha, w_2)_t, j)$, $t \in \{I, D\}$.
  - $\ell \in H$ is a label for the ins-del rule,
  - i: current component, j: target component

Starting with $\#\$$ we generate $\{ww \mid w \in \{a, b\}^*\} \notin CF$

$r_{11} : (1, (\#, a, \lambda)_{ins}, 2)$	$r_{21} : (2, (\$, a, \lambda)_{ins}, 1)$	Size is $(3; 1, 1, 0; 1, 0, 0)$
$r_{12} : (1, (\#, b, \lambda)_{ins}, 3)$	$r_{22} : (2, (\lambda, \#, \lambda)_{del}, 1)$	
$r_{13} : (1, (\lambda, \$, \lambda)_{del}, 2)$	$r_{31} : (3, (\$, b, \lambda)_{ins}, 1)$	

Size of GCID

The size of a GCID system is given by $(k; n, i', i''; m, j', j'')$ where

- $\boxed{k \text{ : Number of Components } (k \geq 1)}$
- n : Maximal length of the insertion string
- i' : Maximal length of the left context used in insertion rules
- i'' : Maximal length of the right context used in insertion rules
- m : Maximal length of the deletion string
- j' : Maximal length of the left context used in deletion rules
- j'' : Maximal length of the right context used in deletion rules

Size of GCID

The size of a GCID system is given by $(k; n, i', i''; m, j', j'')$ where

- $\boxed{k : \text{Number of Components } (k \geq 1)}$
- n : Maximal length of the insertion string
- i' : Maximal length of the left context used in insertion rules
- i'' : Maximal length of the right context used in insertion rules
- m : Maximal length of the deletion string
- j' : Maximal length of the left context used in deletion rules
- j'' : Maximal length of the right context used in deletion rules

Objective

1. With what size does a GCID system (with $n + m \in \{2, 3\}$) characterize RE?
2. Is the underlying control graph, a path?

Computational completeness of *GCID* for $n = 1$, $m = 1$

No.	Size of the system $(k; 1, i', i''; 1, j', j'')$	No.of Comps	Control graph type
1.	$(k; 1, 0, 0; 1, 1, 1)$ or $(k; 1, 1, 1; 1, 0, 0)$	5	*path*
2.	$(k; 1, 1, 0; 1, 1, 0)$ or $(k; 1, 0, 1; 1, 0, 1)$	4 3 4	*Non − tree* *Non − tree* *path*
3.	$(k; 1, 1, 0; 1, 0, 1)$ or $(k; 1, 0, 1; 1, 1, 0)$	4 3 4	*Non − tree* *Non − tree* *path*
4.	$(k; 1, 1, 0; 1, 1, 1)$ or $(k; 1, 0, 1; 1, 1, 1)$	3	*path*
5.	$(k; 1, 1, 1; 1, 1, 0)$ or $(k; 1, 1, 1; 1, 0, 1)$	3	*path*
6.	$(k; 1, 1, 1; 1, 1, 1)$	1	Null

5

Computational completeness of $GCID$ for $n + m = 3$

No.	Size $(k; 1, i', i''; 2, j', j'')$	No. of Comps	Graph type
1.	$(k; 1, 0, 0; 2, 1, 1)$ or $(k; 2, 1, 1; 1, 0, 0)$	5	*path*
2.	$(k; 1, 1, 0; 2, 0, 0)$ or $(k; 1, 0, 1; 2, 0, 0)$ or $(k; 1, 1, 0; 2, 1, 0)$ or $(k; 1, 0, 1; 2, 0, 1)$ or $(k; 1, 1, 0; 2, 0, 1)$ or $(k; 1, 0, 1; 2, 1, 0)$	3 4	$Non - tree$ path
3.	$(k; 2, 0, 0; 1, 1, 0)$ or $(k; 2, 0, 0; 1, 0, 1)$	3 3	$Non - tree$ *path*
4.	$(k; 2, 1, 0; 1, 1, 0)$ or $(k; 2, 0, 1; 1, 0, 1)$ or $(k; 2, 1, 0; 1, 0, 1)$ or $(k; 2, 0, 1; 1, 1, 0)$ or $(k; 2, 1, 1; 1, 1, 0)$ or $(k; 2, 1, 1; 1, 0, 1)$ or $(k; 1, 1, 0; 2, 1, 1)$ or $(k; 1, 0, 1; 2, 1, 1)$	3	*path*
5.	$(k; 1, 1, 1; 2, 0, 0)$ or $(k; 1, 1, 1; 2, 1, 0)$ or $(k; 1, 1, 1; 2, 0, 1)$ or $(k; 1, 1, 1; 2, 1, 1)$ or $(k; 2, 0, 0; 1, 1, 1)$ or $(k; 2, 1, 0; 1, 1, 1)$ or $(k; 2, 0, 1; 1, 1, 1)$ or $(k; 2, 1, 1; 1, 1, 1)$	1	Null

RE $= \mathrm{GCID}_P(3;1,1,0;1,1,1)$ Axiom $= \kappa S \kappa'$

We simulate $r : X \to Y_1 Y_2$, $f : AB \to \lambda \mid CD \to \lambda$, $h : S' \to \lambda$ as:

Lesson learnt

- More contexts does not imply simple simulation

Component 1

$r1.1 : (1, (X, r, \lambda)_I, 2)$
$r1.2 : (1, (r, \Delta, \lambda)_I, 1)$
$r1.3 : (1, (r, Y_2, \lambda)_I, 2)$
$f1.1 : (1, (\lambda, f, \lambda)_I, 2)$
$h1.1 : (1, (\lambda, S', \lambda)_D, 1)$
$\kappa 1.1 : (1, (\lambda, \kappa, \lambda)_D, 1)$
$\kappa' 1.1 : (1, (\lambda, \kappa', \lambda)_D, 1)$

Component 2

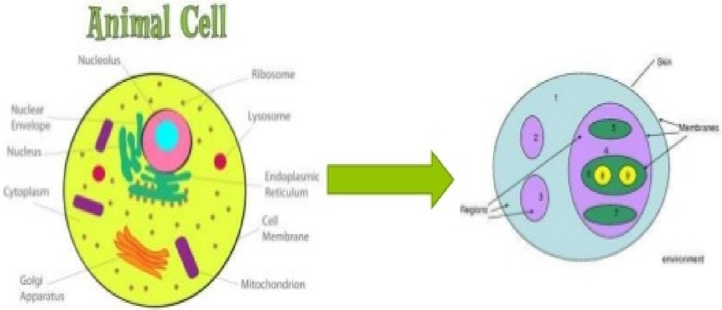
$r2.1 : (2, (\lambda, X, r)_D, 1)$
$r2.2c : (2, (Y_2, \Delta, c)_D, 3), c \neq \Delta$
$r2.3c' : (2, (c', r, Y_1)_D, 1)$
$f2.1 : (2, (f, A, B)_D, 3)$
$f2.2 : (2, (\lambda, f, \lambda)_D, 1)$

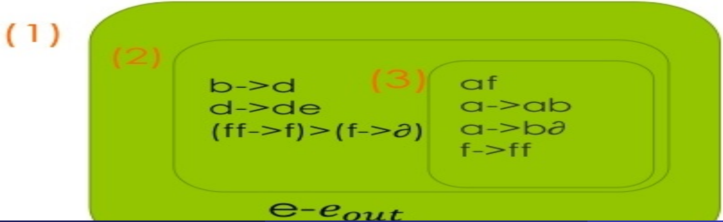
Component 3

$r3.1 : (3, (r, Y_1, \lambda)_I, 2)$
$r3.2 : (3, (f, B, \lambda)_D, 2)$

Why we prefer path?

It has applications in Membrane Computing.





Bridging the gap between LIN and CFL

- The systems GCID(k;1,1,0;1,0,0) and GCID(k;2,1,0;1,0,0) are not known to characterize RE (not even CFL) for any $k \geq 1$.
- However the systems GCID(k;1,1,0;1,0,0) and GCID(k;2,1,0;1,0,0) characterize LIN for $k \geq 3$.

Bridging the gap between LIN and CFL

- The systems GCID(k;1,1,0;1,0,0) and GCID(k;2,1,0;1,0,0) are not known to characterize RE (not even CFL) for any $k \geq 1$.
- However the systems GCID(k;1,1,0;1,0,0) and GCID(k;2,1,0;1,0,0) characterize LIN for $k \geq 3$.
- We aim to show that these systems characterize several classes between LIN and CFL for $k \geq 5$.
- To do so, we first introduce/look into some closure classes of LIN and we term them as super-linear languages.

Closure classes of linear Languages

Note: LIN is not closed under Kleene star and concatenation.

- $\mathcal{L}_{op}(LIN)$ = smallest class containing linear languages and is closed under the operation *op* (Kutrib, Malcher (2007))
- $MLIN := \mathcal{L}_{\circ}(LIN)$ (Metalinear languages)
- $SLIN := \mathcal{L}_{*}(LIN)$ (Starlinear languages)
- $SMLIN := \mathcal{L}_{*}(MLIN) = \mathcal{L}_{*}(\mathcal{L}_{\circ}(LIN))$ (containing MLIN...)
- $MSLIN := \mathcal{L}_{\circ}(SLIN) = \mathcal{L}_{\circ}(\mathcal{L}_{*}(LIN))$
- $SMSLIN := \mathcal{L}_{*}(MSLIN) = \mathcal{L}_{*}(\mathcal{L}_{\circ}(\mathcal{L}_{*}(LIN)))$
- $MSMLIN := \mathcal{L}_{\circ}(SMLIN) = \mathcal{L}_{\circ}(\mathcal{L}_{*}(\mathcal{L}_{\circ}(LIN)))$
- $RATLIN := \mathcal{L}_{\circ,*,\cup}(LIN)$
  The smallest class containing LIN and is closed under the 3 regular operations: concatenation, Kleene star and union.

Languages in closure classes

- $L \in MLIN$ iff $L = L_1 L_2 \dots L_k$ for some $k \geq 1$ and $L_i \in LIN$.
- $L \in SLIN$ iff $L = L_1^*$ for $L_1 \in LIN$.
- $L \in MSLIN$ iff $L = L_1^* L_2^* \dots L_k^*$ for some $k \geq 1$ and $L_i \in LIN$.
- $L \in SMLIN$ iff $L = (L_1 L_2 \dots L_k)^*$ for $k \geq 1$ and $L_i \in LIN$.
- $L \in SMSLIN$ iff $L = (M)^*$ for some $M = L_1^* \dots L_k^* \in MSLIN$.
- $L \in MSMLIN$ iff $L = M_1 M_2 \dots M_k$ for each $M_i \in SMLIN$,
  $M_i = (L_{i,1} L_{i,2} \dots L_{i,t_i})^*$ where $L_{i,j} \in LIN$.

Closure under reversal

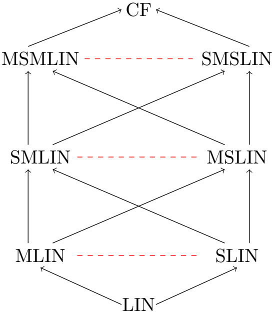
The classes MLIN, SLIN, MSLIN, SMLIN, MSMLIN and SMSLIN are all closed under reversal.

We use the fact that LIN is closed under reversal

- MLIN: $(L_1 L_2 \dots L_k)^R = L_k^R L_{k-1}^R \dots L_1^R$.
- SLIN: $(L_1^*)^R = (L_1^R)^*$.
- SMLIN: $((L_1 L_2 \dots L_k)^*)^R = ((L_1 \dots L_k)^R)^* = (L_k^R \dots L_2^R L_1^R)^*$.
- MSLIN: $(L_1^* L_2^* \dots L_k^*)^R = (L_k^R)^*(L_{k-1}^R)^* \dots (L_2^R)^*(L_1^R)^*$.

Similarly we can extend to MSMLIN and SMSLIN.

Inter-relationship



Solid arrow from A to B indicates $A \subseteq B$. Dashed line between A and B indicates A and B are incomparable.

1. $\mathrm{SLIN} \subseteq \mathrm{MSLIN} \cap \mathrm{SMLIN}$.
2. $\mathrm{MLIN} \subseteq \mathrm{MSLIN} \cap \mathrm{SMLIN}$.
3. $\mathrm{MSLIN} \subseteq \mathrm{MSMLIN} \cap \mathrm{SMSLIN}$.
4. $\mathrm{SMLIN} \subseteq \mathrm{MSMLIN} \cap \mathrm{SMSLIN}$.
5. Incomparable
   - MLIN and SLIN.
   - MSLIN and SMLIN.
   - MSMLIN and SMSLIN.

Sa(i)mple proofs

$\mathrm{MSLIN} \subseteq \mathrm{SMSLIN} \cap \mathrm{MSMLIN}$

- $MSLIN \subseteq SMSLIN$ and
  since $LIN \subseteq MLIN$, $MSLIN \subseteq MSMLIN$.

MSLIN and SMLIN are incomparable

- Let $L_1 = \{a^n b^n \mid n \geq 0\}$ and $L_2 = \{c^m d^m \mid m \geq 0\}$
- $(L_1 L_2)^* \in SMLIN \setminus MSLIN$
  1. $L = L_1 L_2 \in MLIN$ implies $L^* = (L_1 L_2)^* \in SMLIN$.
  2. $L = L_1 L_2 \notin LIN$ implies $L^* \notin SLIN$ and hence $L^* \notin MSLIN$.
- $L_1^* L_2^* \in MSLIN \setminus SMLIN$
- Important: $(L_1 L_2)^* \neq L_1^* L_2^*$ (check yourself!!)

Rewriting grammar for SLIN

Recall: $L \in SLIN$ iff $L = (L_1)^*$

1. Let $G_1 = (N_1, T, S_1, P_1)$ be linear grammar for L_1.
2. A language of SLIN is generated by a grammar $G = (N, T, S, P)$ where
   - $N = N_1 \cup \{S\}$
   - P includes the conventional LIN rules of P_1 and $X \rightarrow Ya$, $X \rightarrow aY$, $X \rightarrow \lambda$
   - The additional CF rules : $S \rightarrow SS_1 \mid \lambda$.

Rewriting grammar for MLIN and SMLIN

Recall: $L \in MLIN$ iff $L = L_1 L_2 \ldots L_k$

1. Let $G_i = (N_i, T, S_i, P_i)$ be linear grammar for L_i.
2. A language of MLIN is generated by a grammar $G = (N, T, S, P)$ where
   - $N = \bigcup_{i=1}^{k} N_i \cup \{S, S_2', S_3', \ldots S_{k+1}'\}$
   - P includes the conventional LIN rules of P_i and $X \to Ya$, $X \to aY$, $X \to \lambda$
   - The additional following CF rules.
     $S \to S_1 S_2'$
     $S_i' \to S_i S_{i+1}'$ for $2 \leq i \leq k$
     $S_{k+1}' \to \lambda$

Rewriting grammar for MLIN and SMLIN

Recall: $L \in MLIN$ iff $L = L_1 L_2 \dots L_k$

1. Let $G_i = (N_i, T, S_i, P_i)$ be linear grammar for L_i.
2. A language of MLIN is generated by a grammar $G = (N, T, S, P)$ where
   - $N = \bigcup_{i=1}^{k} N_i \cup \{S, S_2', S_3', \dots S_{k+1}'\}$
   - P includes the conventional LIN rules of P_i and $X \to Ya$, $X \to aY$, $X \to \lambda$
   - The additional following CF rules.
     $S \to S_1 S_2'$
     $S_i' \to S_i S_{i+1}'$ for $2 \le i \le k$
     $S_{k+1}' \to \lambda \quad | \ S_1 S_2'$ (Additional rule for **SMLIN**)

Rewriting grammar for MLIN and SMLIN

Recall: $L \in MLIN$ iff $L = L_1 L_2 \ldots L_k$

1. Let $G_i = (N_i, T, S_i, P_i)$ be linear grammar for L_i.
2. A language of MLIN is generated by a grammar $G = (N, T, S, P)$ where
   - $N = \bigcup_{i=1}^{k} N_i \cup \{S, S_2', S_3', \ldots S_{k+1}'\}$
   - P includes the conventional LIN rules of P_i and
     $X \to Ya$, $X \to aY$, $X \to \lambda$
   - The additional following CF rules.
     $S \to S_1 S_2'$
     $S_i' \to S_i S_{i+1}'$ for $2 \leq i \leq k$
     $S_{k+1}' \to \lambda \quad | \; S_1 S_2'$ (Additional rule for **SMLIN**)

Sample derivation for MLIN is

$$S \Longrightarrow S_1 S_2' \Longrightarrow^* L_1 S_2' \Longrightarrow L_1 S_2 S_3' \Longrightarrow^* L_1 L_2 S_3' \Longrightarrow^* L_1 L_2 L_3 S_4'$$

Rewriting grammar for MSLIN

Recall: $L \in MSLIN$ iff $L = L_1^* L_2^* \dots L_k^*$

1. Let $G_i = (N_i, T, S_i, P_i)$ be linear grammar for L_i.
2. A language of MSLIN is generated by a grammar $G = (N, T, S, P)$ where
   - $N = \bigcup_{i=1}^{k} N_i \cup \{S, S_2', S_3', \dots S_{k+1}'\}$
   - P includes the conventional LIN rules of P_i and
     $X \to Ya$, $X \to aY$, $X \to \lambda$, $S_i \to \lambda$
   - The additional following CF rules.
     $S \to S_1 S_2'$
     $S_{i+1}' \to S_i S_{i+1}' \mid S_{i+1} S_{i+2}'$ for $1 \leq i \leq k-1$
     The first rule to stay in L_i and second rule to pass to L_{i+1}
     $S_{k+1}' \to \lambda$

Rewriting grammar for MSLIN

Recall: $L \in MSLIN$ iff $L = L_1^* L_2^* \dots L_k^*$

1. Let $G_i = (N_i, T, S_i, P_i)$ be linear grammar for L_i.
2. A language of MSLIN is generated by a grammar $G = (N, T, S, P)$ where
   - $N = \bigcup_{i=1}^{k} N_i \cup \{S, S_2', S_3', \dots S_{k+1}'\}$
   - P includes the conventional LIN rules of P_i and $X \to Ya$, $X \to aY$, $X \to \lambda$, $S_i \to \lambda$
   - The additional following CF rules.
     $S \to S_1 S_2'$
     $S_{i+1}' \to S_i S_{i+1}' \mid S_{i+1} S_{i+2}'$ for $1 \leq i \leq k-1$
     The first rule to stay in L_i and second rule to pass to L_{i+1}
     $S_{k+1}' \to \lambda \ \mid S_1 S_2'$, $S \to \lambda$ (Additional rule for **SMSLIN**)

Rewriting grammar for MSMLIN

Recall: $L \in MSMLIN$ iff $L = M_1 M_2 \ldots M_k$ for each $M_i \in SMLIN$.
$M_i = (L_{i,1} L_{i,2} \ldots L_{i,t_i})^*$ where $L_{i,j} \in LIN$.

1. Let $G_{i,j} = (N_{i,j}, T, S_{i,j}, P_{i,j})$ be linear grammar for $L_{i,j}$.
2. The grammar rules of MSMLIN include the conventional LIN rules of $P_{i,j}$ and P'.

Recalling SMLIN

$S \to S_1 S_2'$

for $2 \le j \le t$

$S_j' \to S_j S_{j+1}'$

$S_{t+1}' \to \lambda \mid S_1 S_2'$

Rules of P' for MSMLIN

$S \to S_{1,1} S_{1,2}'$

For $1 \le i \le k$ and $2 \le j \le t_i$

$S_{i,j}' \to S_{i,j} S_{i,j+1}'$

$S_{i,t_i+1}' \to S_{i,1} S_{i,2}' \mid \underbrace{S_{i+1,1} S_{i+1,2}'}_{for\ \ i \neq k} \mid \underbrace{\lambda}_{if\ \ i=k}$

$LIN \subsetneq GCID(3; 1, 1, 0; 1, 0, 0)$

We simulate the rules $p : X \to Ya$, $q : X \to aY$ and $h : X \to \lambda$ as:

Component 1

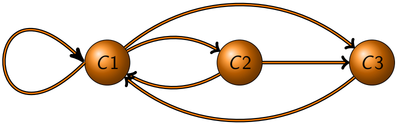
$p1.1 : (1, (X, p, \lambda)_{ins}, 3)$
p1.2: $(1, (p, a, \lambda)_{ins}, 2)$
p1.3: $(1, (p', Y, \lambda)_{ins}, 2)$
$q1.1 : (1, (X, q, \lambda)_{ins}, 3)$
$q1.2 : (1, (q, q', \lambda)_{ins}, 2)$
$q1.3 : (1, (q', Y, \lambda)_{ins}, 2)$
$h1.1 : (1, (\lambda, X, \lambda)_{ins}, 1)$

Component 2

$p2.1 : (2, (p, p', \lambda)_{ins}, 3)$
$p2.2 : (2, (\lambda, p', \lambda)_{del}, 1)$
$q2.1 : (2, (q, a, \lambda)_{ins}, 3)$
$q2.2 : (2, (\lambda, q', \lambda)_{del}, 1)$

Component 3

$p3.1 : (3, (\lambda, X, \lambda)_{del}, 1)$
$p3.2 : (3, (\lambda, p, \lambda)_{del}, 1)$
$q3.1 : (3, (\lambda, X, \lambda)_{del}, 1)$
$q3.2 : (3, (\lambda, q, \lambda)_{del}, 1)$



$LIN \subsetneq GCID(3; 2, 1, 0; 1, 0, 0)$

We simulate the rules $p : X \to aY$, $q : X \to Ya$, $h : X \to \lambda$ as:

Component 1

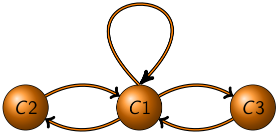
$p1.1 : (1, (X, p, \lambda)_{ins}, 2)$
p1.2: $(1, (p, aY, \lambda)_{ins}, 3)$
$q1.1 : (1, (X, q, \lambda)_{ins}, 2)$
$q1.2 : (1, (q, Ya, \lambda)_{ins}, 3)$
$h1.1 : (1, (\lambda, X, \lambda)_{del}, 1)$

Component 2

$p2.1 : (2, (\lambda, X, \lambda)_{del}, 1)$
$q2.1 : (2, (\lambda, X, \lambda)_{del}, 1)$

Component 3

$p3.1 : (3, (\lambda, p, \lambda)_{del}, 1)$
$q3.1 : (3, (\lambda, q, \lambda)_{del}, 1)$



Simulating Transition rules of MLIN

Recall: $S'_{i+1} \rightarrow S_{i+1}S_{i+2}$ for $1 \leq i \leq k-1$ and $S'_{k+1} \rightarrow \lambda$
$MLIN \subseteq GCID(5;2,1,0;1,0,0)$. For each $1 \leq i \leq k$,

Component 1

$p_i 1.1 : (1, (X_i, p_i, \lambda)_{ins}, 2)$
$p_i 1.2 : (1, (p_i, aY_i, \lambda)_{ins}, 3)$
$q_i 1.1 : (1, (X_i, q_i, \lambda)_{ins}, 2)$
$q_i 1.2 : (1, (q_i, Y_i a, \lambda)_{ins}, 3)$
$h_i 1.1 : (1, (\lambda, X_i, \lambda)_{del}, 4)$

Component 2

$p_i 2.1 : (2, (\lambda, X_i, \lambda)_{del}, 1)$
$q_i 2.1 : (2, (\lambda, X_i, \lambda)_{del}, 1)$

Component 3

$p_i 3.1 : (3, (\lambda, p_i, \lambda)_{del}, 1)$
$q_i 3.1 : (3, (\lambda, q_i, \lambda)_{del}, 1)$

Component 4

For $i \neq k$
$r_i 4.1 : (4, (S'_{i+1}, S_{i+1}, \lambda)_{ins}, 5)$
$r_i 4.2 : (4, (S_{i+1}, S'_{i+2}, \lambda)_{ins}, 1)$
For $i = k$
$r_i 4.1 : (4, (\lambda, S'_{i+1}, \lambda)_{del}, 1)$

Component 5

For $i \neq k$
$r_i 5.1 : (5, (\lambda, S'_{i+1}, \lambda)_{del}, 4)$

Simulating Transition rules of MLIN

Recall: $S'_{i+1} \rightarrow S_{i+1}S_{i+2}$ for $1 \leq i \leq k-1$ and $S'_{k+1} \rightarrow \lambda$
$MLIN \subseteq GCID(5;2,1,0;1,0,0)$. For each $1 \leq i \leq k$,

Component 1

$p_i 1.1 : (1, (X_i, p_i, \lambda)_{ins}, 2)$
$p_i 1.2 : (1, (p_i, aY_i, \lambda)_{ins}, 3)$
$q_i 1.1 : (1, (X_i, q_i, \lambda)_{ins}, 2)$
$q_i 1.2 : (1, (q_i, Y_i a, \lambda)_{ins}, 3)$
$h_i 1.1 : (1, (\lambda, X_i, \lambda)_{del}, 4)$

Component 2

$p_i 2.1 : (2, (\lambda, X_i, \lambda)_{del}, 1)$
$q_i 2.1 : (2, (\lambda, X_i, \lambda)_{del}, 1)$

Component 3

$p_i 3.1 : (3, (\lambda, p_i, \lambda)_{del}, 1)$
$q_i 3.1 : (3, (\lambda, q_i, \lambda)_{del}, 1)$

Component 4

For $i \neq k$
$r_i 4.1 : (4, (S'_{i+1}, S_{i+1}, \lambda)_{ins}, 5)$
$r_i 4.2 : (4, (S_{i+1}, S'_{i+2}, \lambda)_{ins}, 1)$
For $i = k$
$r_i 4.1 : (4, (\lambda, S'_{i+1}, \lambda)_{del}, 1)$

Component 5

For $i \neq k$
$r_i 5.1 : (5, (\lambda, S'_{i+1}, \lambda)_{del}, 4)$

$$(S_1 S'_2)_1 \Longrightarrow^* (L_1 S'_2)_4 \Longrightarrow (L_1 S'_2 S_2)_5 \Longrightarrow (L_1 S_2)_4 \Longrightarrow (L_1 S_2 S'_3)_1$$

$MSLIN \subseteq GCID(5;2,1,0;1,0,0)$

Recall: $S'_{i+1} \to S_{i+1}S'_{i+1} \mid S_{i+1}S'_{i+2}$ for $1 \le i \le k-1$ and $S'_{k+1} \to \lambda$.
For each $1 \le i \le k$,

Component 1

$p_i1.1 : (1,(X_i,p_i,\lambda)_{ins},2)$
$p_i1.2 : (1,(p_i,aY_i,\lambda)_{ins},3)$
$q_i1.1 : (1,(X_i,q_i,\lambda)_{ins},2)$
$q_i1.2 : (1,(q_i,Y_ia,\lambda)_{ins},3)$
$h_i1.1 : (1,(\lambda,X_i,\lambda)_{del},4)$

Component 2

$p_i2.1 : (2,(\lambda,X_i,\lambda)_{del},1)$
$q_i2.1 : (2,(\lambda,X_i,\lambda)_{del},1)$

Component 3

$p_i3.1 : (3,(\lambda,p_i,\lambda)_{del},1)$
$q_i3.1 : (3,(\lambda,q_i,\lambda)_{del},1)$

Component 4

For $i \neq k$
$r_i4.1 : (4,(S'_{i+1},S_{i+1},\lambda)_{ins},5)$
$r_i4.2 : (4,(S_{i+1},S'_{i+2},\lambda)_{ins},1)$
$r_i4.3 : (4,(S_{i+1},S'_{i+1},\lambda)_{ins},1)$
For $i = k$
$r_i4.1 : (4,(\lambda,S'_{i+1},\lambda)_{del},1)$

Component 5

For $i \neq k$
$r_i5.1 : (5,(\lambda,S'_{i+1},\lambda)_{del},4)$

Summary of the results

Each of SLIN, MLIN, SMLIN, MSLIN, SMSLIN, MSMLIN is a subset of each of the following.

- GCID(5;2,1,0;1,0,0) with tree as a control graph
- GCID(5;1,1,0;1,0,0) with non-tree as a control graph

Summary of the results

Each of SLIN, MLIN, SMLIN, MSLIN, SMSLIN, MSMLIN is a subset of each of the following.

- GCID(5;2,1,0;1,0,0) with tree as a control graph
- GCID(5;1,1,0;1,0,0) with non-tree as a control graph

The obtained results can be stated as a general theorem.

Generic Theorem

For integers $t, n, m \geq 1$ and $i', i'', j', j'' \geq 0$ with $i' + i'' \geq 1$ and $X \in \{NTr, Tr\}$, if $\mathrm{LIN} \subseteq \mathrm{GCID}_X(t; n, i', i''; m, j', j'')$, then $\mathrm{F} \subseteq \mathrm{GCID}_X(t+2; n, i', i''; m, j', j'')$ where F $\in$ {SLIN, MLIN, SMLIN, MSLIN, SMSLIN, MSMLIN}.

Extending the results

RATLIN: smallest family containing LIN and closed under union, concatenation and Kleene star.

- Let $L = (L_1 L_2)^* L_3^* L_4 L_5^*$
- Continuation points

$i =$	1	2	3	4	5
$\mathrm{cont}(i)$	2	$1, 3, 4$	$3, 4$	5	$5, 6$

- Assumption: $i + 1 \in cont(i)$

Extending the results

RATLIN: smallest family containing LIN and closed under union, concatenation and Kleene star.

- Let $L = (L_1 L_2)^* L_3^* L_4 L_5^*$
- Continuation points

$i =$	1	2	3	4	5
$\mathrm{cont}(i)$	2	$1, 3, 4$	$3, 4$	5	$5, 6$

- Assumption: $i + 1 \in cont(i)$

Transition rules: Axiom $= S'_1$

$S'_i \rightarrow S_i S'_c$ for all $c \in cont(i)$ and $1 \leq i \leq k$

$S'_{k+1} \rightarrow \lambda$

Matrix Ins-del system

Definition

A matrix insertion-deletion system is a construct $\Gamma = (V, T, A, R)$

- V is an alphabet, $T \subseteq V$, A is a finite language over V
- R is a finite set of matrices $\{m_1, m_2, \ldots m_l\}$
- $\boxed{m_i = [(u_1, \alpha_1, v_1)_{t_1}, (u_2, \alpha_2, v_2)_{t_2}, \ldots, (u_k, \alpha_k, v_k)_{t_k}]}$

Notes to remember:

- On choosing a matrix m_i, all rules in m_i are applied in order.
- If a rule in m_i cannot be applied, then m_i itself is not applied.

Matrix Ins-del system

Definition

A matrix insertion-deletion system is a construct $\Gamma = (V, T, A, R)$

- V is an alphabet, $T \subseteq V$, A is a finite language over V
- R is a finite set of matrices $\{m_1, m_2, \ldots m_l\}$
- $\boxed{m_i = [(u_1, \alpha_1, v_1)_{t_1}, (u_2, \alpha_2, v_2)_{t_2}, \ldots, (u_k, \alpha_k, v_k)_{t_k}]}$

Notes to remember:

- On choosing a matrix m_i, all rules in m_i are applied in order.
- If a rule in m_i cannot be applied, then m_i itself is not applied.

Size

Size of a matrix ins-del system is $(k; n, i', i''; m, j', j'')$ where

$\boxed{k : \text{Maximum number of ins-del rules in a matrix}}$

$n, i', i''; m, j', j''$ are same as in ID size.

Examples

Language generated by the following matrix ins-del systems?

Axiom: $\#\$$

r1 $= [(\#, a, \lambda)_{ins}, (\$, a, \lambda)_{ins}]$
r2 $= [(\#, b, \lambda)_{ins}, (\$, b, \lambda)_{ins}]$
r3 $= [(\lambda, \#, \lambda)_{del}, (\lambda, \$, \lambda)_{del}]$

Examples

Language generated by the following matrix ins-del systems?

Axiom: $\#\$$

r1 $= [(\#, a, \lambda)_{ins}, (\$, a, \lambda)_{ins}]$
r2 $= [(\#, b, \lambda)_{ins}, (\$, b, \lambda)_{ins}]$
r3 $= [(\lambda, \#, \lambda)_{del}, (\lambda, \$, \lambda)_{del}]$

Language $= \{ww \mid w \in \{a, b\}^*\}$
Size of the system is
$(2; 1, 1, 0; 1, 0, 0)$.

Examples

Language generated by the following matrix ins-del systems?

Axiom: $\#\$$

$r1 = [(\#, a, \lambda)_{ins}, (\$, a, \lambda)_{ins}]$
$r2 = [(\#, b, \lambda)_{ins}, (\$, b, \lambda)_{ins}]$
$r3 = [(\lambda, \#, \lambda)_{del}, (\lambda, \$, \lambda)_{del}]$

Language $= \{ww \mid w \in \{a, b\}^*\}$
Size of the system is
$(2; 1, 1, 0; 1, 0, 0)$.

Axiom: $\#$

$r1 = [(\lambda, a, \#)_{ins}, (\#, b, \lambda)_{ins}]$
$r2 = [(\lambda, \#, \lambda)_{del}]$

Examples

Language generated by the following matrix ins-del systems?

Axiom: $\#\$$

r1 $= [(\#, a, \lambda)_{ins}, (\$, a, \lambda)_{ins}]$
r2 $= [(\#, b, \lambda)_{ins}, (\$, b, \lambda)_{ins}]$
r3 $= [(\lambda, \#, \lambda)_{del}, (\lambda, \$, \lambda)_{del}]$

Language $= \{ww \mid w \in \{a, b\}^*\}$
Size of the system is $(2; 1, 1, 0; 1, 0, 0)$.

Axiom: $\#$

r1 $= [(\lambda, a, \#)_{ins}, (\#, b, \lambda)_{ins}]$
r2 $= [(\lambda, \#, \lambda)_{del}]$

Language $= \{a^n b^n \mid n \geq 0\}$
Size of the system is $(2; 1, 1, 1; 1, 0, 0)$.

Examples

Language generated by the following matrix ins-del systems?

Axiom: $\#\$$

$r1 = [(\#, a, \lambda)_{ins}, (\$, a, \lambda)_{ins}]$
$r2 = [(\#, b, \lambda)_{ins}, (\$, b, \lambda)_{ins}]$
$r3 = [(\lambda, \#, \lambda)_{del}, (\lambda, \$, \lambda)_{del}]$

Language $= \{ww \mid w \in \{a, b\}^*\}$
Size of the system is $(2; 1, 1, 0; 1, 0, 0)$.

Axiom: $\#$

$r1 = [(\lambda, a, \#)_{ins}, (\#, b, \lambda)_{ins}]$
$r2 = [(\lambda, \#, \lambda)_{del}]$

Language $= \{a^n b^n \mid n \geq 0\}$
Size of the system is $(2; 1, 1, 1; 1, 0, 0)$.

Helpful Results

- $\boxed{MAT(k; n, i', i''; m, j', j'') = [MAT(k; n, i'', i'; m, j'', j')]^R}$
- Since RE is closed under reversal,
  $MAT(k; n, i', i''; m, j', j'') = RE = MAT(k; n, i'', i'; m, j'', j')$.

Exhaustive Analysis for $n = |Ins| = 1$, $m = |Del| = 1$

Size $(k; 1, i', i''; 1, j', j'')$; $i', i'', j', j'' \in \{0, 1\}$	Reference	k	Language Family Relation
$(k; 1, 0, 0; 1, 0, 0)$	S.Verlan 2007	1	$\subset REG$
$(k; 1, 0, 0; 1, 1, 0), (k; 1, 0, 0; 1, 0, 1)$		≥ 1	OPEN
$(k; 1, 0, 0; 1, 1, 1)$	HLI 2018 HLI 2019	3 2	$= \mathrm{RE}$ $= \mathrm{RE}$
$(k; 1, 1, 0; 1, 0, 0)$, $(k; 1, 0, 0; 1, 0, 0)$	HLI 2019	3	$\supset \mathcal{L}_{reg}(\mathrm{LIN})$
$(k; 1, 1, 1; 1, 0, 0)$	HLI 2018 HLI 2019	3 2	$= \mathrm{RE}$ $\supset \mathcal{L}_{reg}(\mathrm{LIN})$
$(k; 1, 1, 0; 1, 1, 0)$, $(k; 1, 1, 0; 1, 0, 1)$ $(k; 1, 0, 1; 1, 0, 1)$, $(k; 1, 0, 1; 1, 1, 0)$	S.Verlan 2012 HLI 2019	3 2	$= \mathrm{RE}$ $= \mathrm{RE}$
$(k; 1, 1, 0; 1, 1, 1)$, $(k; 1, 0, 1; 1, 1, 1)$	HLI 2018	2	$= \mathrm{RE}$
$(k; 1, 1, 1; 1, 1, 0)$, $(k; 1, 1, 1; 1, 0, 1)$	HLI 2018	2	$= \mathrm{RE}$
$(k; 1, 1, 1; 1, 1, 1)$	Takahari 2003	1	$= \mathrm{RE}$

Power of MID systems of size $(k; 1, i', i''; 1, j', j'')$

HLI 2018: H Fernau, Lakshmanan, Indhumathi, Investigations on the Power of Matrix Insertion-Deletion Systems of Small Sizes, Natural Computing, 2018, 17(2), 249 - 269.

HLI 2019: -do-, On Matrix Ins-Del Systems of Small Sum-Norm, SOFSEM 2019, LNCS 11376, 192-205.

Exhaustive Analysis for $n + m = 3$

Size $(k; 1, i', i''; 2, j', j'')$; $i', i'', j', j'' \in \{0, 1\}$ or $(k; 2, i', i''; 1, j', j'')$; $i', i'', j', j'' \in \{0, 1\}$	Reference	k	Language Family Relation
$(k; 1, 0, 0; 2, 0, 0)$, $(k; 2, 0, 0; 1, 0, 0)$	Verlan 2007	1	$\subset REG$
$(k; 1, 0, 0; 2, 1, 0)$, $(k; 1, 0, 0; 2, 0, 1)$		≥ 1	OPEN
$(k; 1, 1, 0; 2, 0, 0)$, $(k; 1, 1, 0; 2, 1, 0)$, $(k; 1, 1, 0; 2, 0, 1)$ $(k; 2, 0, 0; 1, 1, 0)$, $(k; 2, 1, 0; 1, 1, 0)$, $(k; 2, 0, 1; 1, 1, 0)$	Verlan 2012	2	$= \mathrm{RE}$
$(k; 1, 0, 0; 2, 1, 1), (k; 2, 1, 1; 1, 0, 0)$	HLI 2018	3	$= \mathrm{RE}$
$(k; 1, 1, 0; 2, 1, 1)$, $(k; 1, 0, 1; 2, 0, 0)$, $(k; 1, 0, 1; 2, 1, 1)$ $(k; 1, 0, 1; 2, 1, 0)$, $(k; 1, 0, 1; 2, 0, 1)$	HLI 2018	2	$= \mathrm{RE}$
$(k; 2, 0, 0; 1, 0, 1)$, $(k; 2, 1, 0; 1, 0, 1)$, $(k; 2, 0, 1; 1, 0, 1)$ $(k; 2, 1, 1; 1, 1, 0)$, $(k; 2, 1, 1; 1, 0, 1)$	HLI 2018	2	$= \mathrm{RE}$
$(k; 2, 1, 0; 1, 0, 0)$, $(k; 2, 0, 1; 1, 0, 0)$	HLI 2019	2	$\supset \mathcal{L}_{reg}(\mathrm{LIN})$
$(k; 2, 0, 0; 1, 1, 1)$, $(k; 2, 1, 0; 1, 1, 1)$, $(k; 2, 0, 1; 1, 1, 1)$	Krassovitskiy 2008	1	$= \mathrm{RE}$
$(k; 1, 1, 1; 2, 0, 0)$, $(k; 1, 1, 1; 2, 1, 0)$, $(k; 1, 1, 1; 2, 0, 1)$	Paun 1998	1	$= \mathrm{RE}$
$(k; 1, 1, 1; 2, 1, 1)$, $(k; 2, 1, 1; 1, 1, 1)$	Takahari 2003	1	$= \mathrm{RE}$

Power of MID systems of size $(k; 1, i', i''; 2, j', j'')$ or $(k; 2, i', i''; 1, j', j'')$

MAT(3;1,0,0;1,1,1) = RE

Consider a type-0 grammar $G = (N, T, P, S)$ in SGNF.

Simulating p: $X \rightarrow bY$

$p1 = [(\lambda, p, \lambda)_{ins},\ (\lambda, p', \lambda)_{ins}, (p', X, p)_{del}]$
$p2 = [(\lambda, b, \lambda)_{ins}, (\lambda, Y, \lambda)_{ins},\ (b, p, Y)_{del}]$
$p3 = [(\lambda, p', b)_{del}]$ (right context is required to ensure p3 is applied after p2)

MAT(3;1,0,0;1,1,1) = RE

Consider a type-0 grammar $G = (N, T, P, S)$ in SGNF.

Simulating p: $X \to bY$

$p1 = [(\lambda, p, \lambda)_{ins},\ (\lambda, p', \lambda)_{ins}, (p', X, p)_{del}]$
$p2 = [(\lambda, b, \lambda)_{ins}, (\lambda, Y, \lambda)_{ins},\ (b, p, Y)_{del}]$
$p3 = [(\lambda, p', b)_{del}]$ (right context is required to ensure p3 is applied after p2)

Simulating q: $X \to Yb$

$q1 = [(\lambda, q, \lambda)_{ins},\ (\lambda, q', \lambda)_{ins}, (q', X, q)_{del}]$
$q2 = [(\lambda, b, \lambda)_{ins}, (\lambda, Y, \lambda)_{ins},\ (Y, q', b)_{del}]$
$q3 = [(b, q, \lambda)_{del}]$ (left context is required to ensure p3 is applied after p2)

MAT(3;1,0,0;1,1,1) = RE

Consider a type-0 grammar $G = (N, T, P, S)$ in SGNF.

Simulating p: $X \rightarrow bY$

$p1 = [(\lambda, p, \lambda)_{ins},\ (\lambda, p', \lambda)_{ins}, (p', X, p)_{del}]$
$p2 = [(\lambda, b, \lambda)_{ins}, (\lambda, Y, \lambda)_{ins},\ (b, p, Y)_{del}]$
$p3 = [(\lambda, p', b)_{del}]$ (right context is required to ensure p3 is applied after p2)

Simulating q: $X \rightarrow Yb$

$q1 = [(\lambda, q, \lambda)_{ins},\ (\lambda, q', \lambda)_{ins}, (q', X, q)_{del}]$
$q2 = [(\lambda, b, \lambda)_{ins}, (\lambda, Y, \lambda)_{ins},\ (Y, q', b)_{del}]$
$q3 = [(b, q, \lambda)_{del}]$ (left context is required to ensure p3 is applied after p2)

Simulating f: $AB \rightarrow \lambda$

$f1 = [(\lambda, f, \lambda)_{ins}, (\lambda, f', \lambda)_{ins}, (f, A, B)_{del}]$
$f2 = [(f, B, f')_{del}, (\lambda, f', \lambda)_{del}, (\lambda, f, \lambda)_{del}]$

MAT(2;1,1,0;1,1,1) = RE

Simulating p: $X \to bY$: Axiom$= S\#\$$

$$
\begin{aligned}
p1 &= [(X,p,\lambda)_{ins},\ (\#,p',\lambda)_{ins}] & p4 &= [(p,b,\lambda)_{ins},\ (p''',p'',p')_{del}]\\
p2 &= [(\lambda,X,p)_{del},\ (\#,p'',\lambda)_{ins}] & p5 &= [(\lambda,p,b)_{del},\ (p''',p',\$)_{del}]\\
p3 &= [(p,Y,\lambda)_{ins},\ (\#,p''',\lambda)_{ins}] & p6 &= [(\#,p''',\$)_{del}]
\end{aligned}
$$

Simulating f: $AB \to \lambda$

$$
\begin{aligned}
f1 &= [(B,f,\lambda)_{ins},\ (\#,f',\lambda)_{ins}] & f1' &= [(B,f,\lambda)_{ins}]\\
f2 &= [(\lambda,B,f)_{del},\ (\lambda,A,f)_{del}] & f2' &= [(\lambda,B,f)_{del},\ (\lambda,A,f)_{del}]\\
f3 &= [(\lambda,f,\lambda)_{del},\ (\#,f',\$)_{del}] & f3' &= [(\lambda,f,\lambda)_{del}]
\end{aligned}
$$

Malicious derivation for $f : AB \to \lambda$

$$
AAB\delta B\#\$ \Rightarrow^2_{f1'} AABf\delta Bf\#\$ \Rightarrow^2_{f2'}
$$

$$
\underline{AAB}f\delta\underline{B}f\#f'f'\$ = f\delta f\#\$ \Rightarrow_{f3'} \delta\#\$
$$

Note: $[(\lambda,\#,\lambda),(\lambda,\$,\lambda)]$ is applied at the end of the derivation.

MAT(2;1,1,0;1,1,0) = RE

Simulating p: $X \rightarrow bY$

$$p1 = [(X, p, \lambda)_{ins},\ (\lambda, p', \lambda)_{ins}]$$
$$p2 = [(p', X, \lambda)_{del},\ (p', p'', \lambda)_{ins}]$$
$$p3 = [(p'', p, \lambda)_{del},\ (p'', Y, \lambda)_{ins}]$$
$$p4 = [(p', b, \lambda)_{ins},\ (b, p'', \lambda)_{del}]$$
$$p5 = [(\lambda, p', \lambda)_{del}]$$

MAT(2;1,1,0;1,1,0) = RE

Simulating p: $X \to bY$

$$
\begin{aligned}
p1 &= [(X, p, \lambda)_{ins},\ (\lambda, p', \lambda)_{ins}] \\
p2 &= [(p', X, \lambda)_{del},\ (p', p'', \lambda)_{ins}] \\
p3 &= [(p'', p, \lambda)_{del},\ (p'', Y, \lambda)_{ins}]
\end{aligned}
\qquad
\begin{aligned}
p4 &= [(p', b, \lambda)_{ins},\ (b, p'', \lambda)_{del}] \\
p5 &= [(\lambda, p', \lambda)_{del}]
\end{aligned}
$$

Applying p1 twice??

$X \Rightarrow_{p1} p'Xpp\ldots p' \Rightarrow_{p2} p'p''pp\ldots p' \Rightarrow_{p3} p'p''Yp\ldots p' \Rightarrow_{p4}$
$p'bYp\ldots p' \Rightarrow^2_{p5} bYp$. Cannot reapply p3 to get rid of the second p.

Simulating f: $AB \to \lambda$

A new idea of moving in a Z.

$$
\begin{aligned}
h1 &= [(\lambda, S', \lambda)_{del},\ (\lambda, Z, \lambda)_{ins}] \\
f1 &= [(Z, A, \lambda)_{del},\ (Z, B, \lambda)_{del}]
\end{aligned}
\qquad
\begin{aligned}
moveZ &= [(\lambda, Z, \lambda)_{del},\ (\lambda, Z, \lambda)_{ins}] \\
delZ &= [(\lambda, Z, \lambda)_{del}]
\end{aligned}
$$

MAT rules for Super-linear grammars

Each of SLIN, MLIN, SMLIN, MSLIN, SMSLIN, MSMLIN is a subset of each of the following.

- MAT(3;1,1,0;1,0,0)
- MAT(2;2,1,0;1,0,0)
- MAT(2;1,1,1;1,0,0)

MAT rules for Super-linear grammars

Each of SLIN, MLIN, SMLIN, MSLIN, SMSLIN, MSMLIN is a subset of each of the following.

- MAT(3;1,1,0;1,0,0)
- MAT(2;2,1,0;1,0,0)
- MAT(2;1,1,1;1,0,0)

Generic Theorem

For integers $t, n, m \geq 1$ and $i', i'', j', j'' \geq 0$ with $i' + i'' \geq 1$, if $\mathrm{LIN} \subseteq \mathrm{MAT}(t; n, i', i''; m, j', j'')$, then $\mathrm{F} \subseteq \mathrm{MAT}(t; n, i', i''; m, j', j'')$ where F $\in$ {SLIN, MLIN, SMLIN, MSLIN, SMSLIN, MSMLIN, RATLIN}.

Simulation of MLIN

Recall: Apart from the usual LIN rules, the transition rules in MLIN are $S'_{i+1} \to S_{i+1}S_{i+2}$ for $1 \leq i \leq k-1$ and $S'_{k+1} \to \lambda$, for each $1 \leq i \leq k$,

MLIN $\subseteq$ MAT$(3; 1, 1, 0; 1, 0, 0)$: Axiom $= S_1 S'_2$

$p1 = [(X_i, p_i, \lambda)_{ins},\ (p_i, p'_i, \lambda)_{ins}, (\lambda, X_i, \lambda)_{del}]$
$p2 = [(p_i, a_i, \lambda)_{ins}, (p'_i, Y_i, \lambda)_{ins},\ (\lambda, p_i, \lambda)_{del}]$
$p3 = [(\lambda, p'_i, \lambda)_{del}]$
$p4 = [(S'_{i+1}, S'_{i+2}, \lambda)_{ins}, (S'_{i+1}, S_{i+1}, \lambda)_{ins},\ (\lambda, S'_{i+1}, \lambda)_{del}]$ (for each $1 \leq i \leq k-1$)
$p5 = [(\lambda, S'_{k+1}, \lambda)_{del}]$

Simulation of MLIN

Recall: Apart from the usual LIN rules, the transition rules in MLIN are $S'_{i+1} \to S_{i+1}S_{i+2}$ for $1 \le i \le k-1$ and $S'_{k+1} \to \lambda$, for each $1 \le i \le k$,

MLIN $\subseteq$ MAT$(3; 1, 1, 0; 1, 0, 0)$: Axiom $= S_1 S'_2$

$p1 = [(X_i, p_i, \lambda)_{ins},\ (p_i, p'_i, \lambda)_{ins}, (\lambda, X_i, \lambda)_{del}]$
$p2 = [(p_i, a_i, \lambda)_{ins}, (p'_i, Y_i, \lambda)_{ins},\ (\lambda, p_i, \lambda)_{del}]$
$p3 = [(\lambda, p'_i, \lambda)_{del}]$
$p4 = [(S'_{i+1}, S'_{i+2}, \lambda)_{ins}, (S'_{i+1}, S_{i+1}, \lambda)_{ins},\ (\lambda, S'_{i+1}, \lambda)_{del}]$ (for each $1 \le i \le k-1$)
$p5 = [(\lambda, S'_{k+1}, \lambda)_{del}]$

MLIN $\subseteq$ MAT$(2; 1, 1, 1; 1, 0, 0)$: Axiom $= S_1 S'_2$

$p1 = [(X_i, p_i, \lambda)_{ins},\ (\lambda, X_i, \lambda)_{del}]$
$p2 = [(p_i, p'_i, \lambda)_{ins},\ (p_i, a_i, p'_i)_{ins}]$ (cannot reuse due to second rule)
$p3 = [(a_i, Y_i, p'_i)_{ins},\ (\lambda, p_i, \lambda)_{del}]$
$p4 = [(\lambda, p'_i, \lambda)_{del}]$
$p5 = [(S'_{i+1}, S'_{i+2}, \lambda)_{ins}, (S'_{i+1}, S_{i+1}, S'_{i+2})_{ins}]$ (for each $1 \le i \le k-1$)
$p5 = [(\lambda, S'_{i+1}, \lambda)_{del}]$(for each $1 \le i \le k$)

Semi-conditional ins-del system

Definition

A semi-conditional ins-del system (SCID) of degree (i, j) is $G = (V, T, A, P)$, where P is a finite set of rules of the form $((u, x, v)_t, \alpha, \beta)$, where

- $(u, x, v)_t$ is an ins-del rule, $t \in \{ins, del\}$,
- $\alpha, \beta = \phi$ or $\alpha, \beta \subset (N \cup T)^*$ (finite languages) and
- $|\alpha_r| \leq i$ for $\alpha_r \in \alpha$, and $|\beta_s| \leq j$ for $\beta_s \in \beta$.

Semi-conditional ins-del system

Definition

A semi-conditional ins-del system (SCID) of degree (i, j) is $G = (V, T, A, P)$, where P is a finite set of rules of the form $((u, x, v)_t, \alpha, \beta)$, where

- $(u, x, v)_t$ is an ins-del rule, $t \in \{ins, del\}$,
- $\alpha, \beta = \phi$ or $\alpha, \beta \subset (N \cup T)^*$ (finite languages) and
- $|\alpha_r| \leq i$ for $\alpha_r \in \alpha$, and $|\beta_s| \leq j$ for $\beta_s \in \beta$.

Rule application in derivation

$((u, x, v)_t, \alpha, \beta)$ is applied to a string w iff every string in

- [Permitting set] α (when $\alpha \neq \phi$) is a substring of w and
- [Forbidding set] β (when $\beta \neq \phi$) is not a substring of w.
- If $\alpha = \phi$, $\beta = \phi$, the rule is applied without any restriction.

SSCID and an Example

Variants

A semi-conditional grammar is called

- Random Context: if degree $(i, j) = (1, 1)$.
- Simple: If either $\alpha = \phi$ or $\beta = \phi$ in every rule of P.

Example: $L_1 = \{a^n b^n c^n \mid n \geq 1\} \notin CF$

Consider $G_1 = (\{a, b, c, A, B\}, \{a, b, c\}, abc, R)$ where R is

- $[(a, aAb, b)_{ins}, \emptyset, B]$
- $[(b, Bc, c)_{ins}, A, \emptyset]$
- $[(\lambda, A, \lambda)_{del}, B, \emptyset]$
- $[(\lambda, B, \lambda)_{del}, \emptyset, A]$

SSCID and an Example

Variants

A semi-conditional grammar is called

- Random Context: if degree $(i, j) = (1, 1)$.
- Simple: If either $\alpha = \phi$ or $\beta = \phi$ in every rule of P.

Example: $L_1 = \{a^n b^n c^n \mid n \geq 1\} \notin CF$

Consider $G_1 = (\{a, b, c, A, B\}, \{a, b, c\}, abc, R)$ where R is

- $[(a, aAb, b)_{ins}, \emptyset, B]$
- $[(b, Bc, c)_{ins}, A, \emptyset]$
- $[(\lambda, A, \lambda)_{del}, B, \emptyset]$
- $[(\lambda, B, \lambda)_{del}, \emptyset, A]$
- Simple and Random Context
- Size $= (3, 1, 1; 1, 0, 0)$
- Degree $= (1, 1)$

Existing vs Our Results

Semi-conditional Ins-del systems of following sizes (do not) describe the class of RE languages

Existing Results (S.Ivanov, S.Verlan, Fund.Inf., 2012)

- $SCID_{2,2}(1,0,0;1,0,0)$
- $SCID_{1,1}(2,0,0;1,1,0)$
- $SCID_{1,1}(1,1,0;1,1,1)$
- $SCID_{1,1}(1,1,0;2,0,0)$
- None is simple

Results of UCNC 2018

- $\boxed{SSCID_{2,1}(2,0,0;2,0,0)}$
- $SSCID_{3,1}(1,1,0;1,1,0)$
- $SSCID_{2,1}(1,1,0;1,1,1)$
- $SSCID_{2,1}(1,1,0;2,0,0)$
- All are simple

$\text{SSCID}_{2,1}(2,0,0;2,0,0)$=RE

Simulation of $\boxed{f : AB \to \lambda \text{ by } (\lambda, AB, \lambda, \phi, \phi)}$ is direct.

$\text{SSCID}_{2,1}(2,0,0;2,0,0)$=RE

Simulation of $\boxed{f : AB \to \lambda \text{ by } (\lambda, AB, \lambda, \phi, \phi)}$ is direct.

Simulations of $\boxed{p : X \to bY}$ and $\boxed{q : X \to Yb}$ are similar.

Simulating $q : X \to Yb$

$q1 : [(\lambda, qq', \lambda)_{ins}, \emptyset, \{q, q', q'', q'''\}]$
$q2 : [(\lambda, q'X, \lambda)_{del}, \{qq'\}, \emptyset]$
$q3 : [(\lambda, q''b, \lambda)_{ins}, \emptyset, \{q', q'', q'''\}]$
$q4 : [(\lambda, q'''Y, \lambda)_{ins}, \emptyset, N' \cup \{q', q'''\}]$
$q5 : [(\lambda, q''', \lambda)_{del}, \{q''q'''\}, \emptyset]$
$q6 : [(\lambda, qq'', \lambda)_{del}, \{Yb\}, \emptyset]$

SSCID$_{2,1}(2,0,0;2,0,0)$=RE

Simulation of $\boxed{f : AB \to \lambda \text{ by } (\lambda, AB, \lambda, \phi, \phi)}$ is direct.

Simulations of $\boxed{p : X \to bY}$ and $\boxed{q : X \to Yb}$ are similar.

Simulating $q : X \to Yb$

$q1 : [(\lambda, qq', \lambda)_{ins}, \emptyset, \{q, q', q'', q'''\}]$
$q2 : [(\lambda, q'X, \lambda)_{del}, \{qq'\}, \emptyset]$
$q3 : [(\lambda, q''b, \lambda)_{ins}, \emptyset, \{q', q'', q'''\}]$
$q4 : [(\lambda, q'''Y, \lambda)_{ins}, \emptyset, N' \cup \{q', q'''\}]$
$q5 : [(\lambda, q''', \lambda)_{del}, \{q''q'''\}, \emptyset]$
$q6 : [(\lambda, qq'', \lambda)_{del}, \{Yb\}, \emptyset]$

Another simulation?

$q1 : [(\lambda, qq', \lambda)_{ins}, \emptyset, \{q, q'', q'''\}]$
$q2 : [(\lambda, q'X, \lambda)_{del}, \{qq'\}, \emptyset]$
$\hat{q3} : [(\lambda, Yq'', \lambda)_{ins}, \emptyset, N' \cup \{q'', q'''\}]$
$\hat{q4} : [(\lambda, bq''', \lambda)_{ins}, \emptyset, N' \cup \{q', q'''\}]$
$\hat{q5} : [(\lambda, q''q, \lambda)_{del}, \{q'''q''\}, \emptyset]$
$\hat{q6} : [(\lambda, q''', \lambda)_{del}, \emptyset, \{q, q''\}]$

$\mathrm{SSCID}_{2,1}(2,0,0;2,0,0)$=RE

Simulation of $\boxed{f : AB \to \lambda \text{ by } (\lambda, AB, \lambda, \phi, \phi)}$ is direct.

Simulations of $\boxed{p : X \to bY}$ and $\boxed{q : X \to Yb}$ are similar.

Simulating $q : X \to Yb$	Another simulation?
$q1 : [(\lambda, qq', \lambda)_{ins}, \emptyset, \{q, q', q'', q'''\}]$	$q1 : [(\lambda, qq', \lambda)_{ins}, \emptyset, \{q, q'', q'''\}]$
$q2 : [(\lambda, q'X, \lambda)_{del}, \{qq'\}, \emptyset]$	$q2 : [(\lambda, q'X, \lambda)_{del}, \{qq'\}, \emptyset]$
$q3 : [(\lambda, q''b, \lambda)_{ins}, \emptyset, \{q', q'', q'''\}]$	$\hat{q}3 : [(\lambda, Yq'', \lambda)_{ins}, \emptyset, N' \cup \{q'', q'''\}]$
$q4 : [(\lambda, q'''Y, \lambda)_{ins}, \emptyset, N' \cup \{q', q'''\}]$	$\hat{q}4 : [(\lambda, bq''', \lambda)_{ins}, \emptyset, N' \cup \{q', q'''\}]$
$q5 : [(\lambda, q''', \lambda)_{del}, \{q''q'''\}, \emptyset]$	$\hat{q}5 : [(\lambda, q''q, \lambda)_{del}, \{q'''q''\}, \emptyset]$
$q6 : [(\lambda, qq'', \lambda)_{del}, \{Yb\}, \emptyset]$	$\hat{q}6 : [(\lambda, q''', \lambda)_{del}, \emptyset, \{q, q''\}]$

Suppose we have a terminal string α,
$\alpha \Rightarrow_{\hat{q}4} \alpha bq''' \Rightarrow_{\hat{q}6} \alpha b = \alpha' \in T^*$
We get another terminal string without any reason.
(ERROR with right side rules!!)

Forbidding Ins-del systems

Definition

A forbidding ins-del system (FID) of degree k is $G = (V, T, A, P)$, where P is a finite set of rules of the form $((u, x, v)_t, F)$, where

- $(u, x, v)_t$ is an ins-del rule, $t \in \{ins, del\}$,
- $F = \phi$ or $F \subset (N \cup T)^*$ (finite languages) and
- $|f_r| \leq k$ for $f_r \in F$.

Forbidding Ins-del systems

Definition

A forbidding ins-del system (FID) of degree k is $G = (V, T, A, P)$, where P is a finite set of rules of the form $((u, x, v)_t, F)$, where

- $(u, x, v)_t$ is an ins-del rule, $t \in \{ins, del\}$,
- $F = \phi$ or $F \subset (N \cup T)^*$ (finite languages) and
- $|f_r| \leq k$ for $f_r \in F$.

Points to note

- $((u, x, v)_t, F)$ is applied to a string w iff every string in [Forbidding set] F $(\neq \phi)$ is not a substring of w.
- If $F = \phi$, then the rule $((u, x, v)_t, \phi)$ can be applied without any restriction.
- (S)SCID$_{0,k}(s)$=FID$_k(s)$.

Computational Completeness

Recall: (S)SCID results (= RE)

- $SSCID_{2,1}(2,0,0;2,0,0)$
- $SSCID_{2,1}(1,1,0;2,0,0)$
- $SSCID_{2,1}(1,1,0;1,1,1)$
- $SSCID_{3,1}(1,1,0;1,1,0)$
- $SSCID_{3,1}(1,1,0;1,0,1)$

Following systems = RE

- $FID_2(2,0,0;2,0,0)$
- $FID_2(1,1,0;2,0,0)$, $FID_2(1,0,1;2,0,0)$
- $FID_2(2,0,0;1,1,0)$, $FID_2(2,0,0;1,0,1)$
- $FID_2(1,1,0;1,1,0)$, $FID_2(1,0,1;1,0,1)$
- $FID_2(1,1,0;1,0,1)$, $FID_2(1,0,1;1,1,0)$

$\mathrm{FID}_2(2,0,0;2,0,0)$=RE

Simulating $X \to Yb$ by $\mathrm{FID}_2(2,0,0;2,0,0)$

$q1 = [(qq')_{ins}, \{\mathcal{M} \cup (N' \setminus \{X\})\}]$
$q2 = [(q'X)_{del}, \{\mathcal{M} \setminus \{q, q'\} \cup (N' \setminus \{X\})\}]$
$q3 = [(q''b)_{ins}, \{\mathcal{M} \setminus \{q\} \cup N'\}]$
$q4 = [(q'''Y)_{ins}, (\mathcal{M} \setminus \{q, q''\}) \cup N' \cup \{Zq'' \mid Z \neq q\} \cup \{qZ \mid Z \neq q''\}]$
$q5 = [(q^{iv}q^{v})_{ins}, (\mathcal{M} \setminus \{q, q'', q'''\}) \cup (N' \setminus \{Y\}) \cup \{q''b\} \cup \{qZ \mid Z \neq q''\}]$
$q6 = [(q'''q^{iv})_{del}, \{\mathcal{M}\{q, q'', q''', q^{iv}, q^{v}\} \cup (N' \setminus \{Y\}) \cup \{q'''Y, q''b\} \cup \{Zq''' \mid Z \neq q''\} \cup \{qZ \mid Z \neq q''\}]$
$q7 = [(q'')_{del}, \{\mathcal{M}\{q, q'', q^{v}\} \cup (N' \setminus \{Y\}) \cup \{q'''Y, q''b\} \cup \{q''Z \mid Z \neq q^{v}\}]$
$q8 = [(qq^{v})_{del}, \{\mathcal{M}\{q, q^{v}\} \cup (N' \setminus \{Y\})]$

$\mathrm{FID}_2(2,0,0;2,0,0)$=RE

Simulating $X \to Yb$ by $\mathrm{FID}_2(2,0,0;2,0,0)$

$q1 = [(qq')_{ins}, \{\mathcal{M} \cup (N' \setminus \{X\})\}]$
$q2 = [(q'X)_{del}, \{\mathcal{M} \setminus \{q, q'\} \cup (N' \setminus \{X\})\}]$
$q3 = [(q''b)_{ins}, \{\mathcal{M} \setminus \{q\} \cup N'\}]$
$q4 = [(q'''Y)_{ins}, (\mathcal{M} \setminus \{q, q''\}) \cup N' \cup \{Zq'' \mid Z \neq q\} \cup \{qZ \mid Z \neq q''\}]$
$q5 = [(q^{iv}q^{v})_{ins}, (\mathcal{M} \setminus \{q, q'', q'''\}) \cup (N' \setminus \{Y\}) \cup \{q''b\} \cup \{qZ \mid Z \neq q''\}]$
$q6 = [(q'''q^{iv})_{del}, \{\mathcal{M}\{q, q'', q''', q^{iv}, q^{v}\} \cup (N' \setminus \{Y\}) \cup \{q'''Y, q''b\} \cup \{Zq''' \mid Z \neq q''\} \cup \{qZ \mid Z \neq q''\}]$
$q7 = [(q'')_{del}, \{\mathcal{M}\{q, q'', q^{v}\} \cup (N' \setminus \{Y\}) \cup \{q'''Y, q''b\} \cup \{q''Z \mid Z \neq q^{v}\}]$
$q8 = [(qq^{v})_{del}, \{\mathcal{M}\{q, q^{v}\} \cup (N' \setminus \{Y\})]$

$X \Rightarrow_{q1} qq'X \Rightarrow_{q2} q \Rightarrow_{q3} qq''b \Rightarrow_{q4} qq''q'''Yb \Rightarrow_{q5} qq''q'''q^{iv}q^{v}Yb \Rightarrow_{q6} qq''q^{v}Yb \Rightarrow_{q7} qq^{v}Yb \Rightarrow_{q8} Yb$

$FID_2(1,1,0;1,0,1) = RE$

Simulating $X \rightarrow bY$ by $FID_2(1,1,0;1,0,1)$

$p1 = [(X, p, \lambda)_{ins}, \mathcal{M} \cup (N' \setminus \{X\})]$
$p2 = [(\lambda, X, p)_{del}, (\mathcal{M} \setminus \{p\}) \cup (N' \setminus \{X\})]$
$p3 = [(p, Y, \lambda)_{ins}, (\mathcal{M} \setminus \{p\}) \cup N']$
$p4 = [(p, p', \lambda)_{ins}, (\mathcal{M} \setminus \{p\}) \cup (N' \setminus \{Y\}) \cup (\{pZ \mid Z \neq Y\})]$
$p5 = [(p', b, \lambda)_{ins}, (\mathcal{M} \setminus \{p, p'\}) \cup (N' \setminus \{Y\}) \cup (\{p'Z \mid Z \neq Y\})]$
$p6 = [(\lambda, p, p')_{del}, \{p'Y\}]$
$p7 = [(\lambda, p', \lambda)_{del}, \{p\}]$

$FID_2(1,1,0;1,0,1) = RE$

Simulating $X \to bY$ by $FID_2(1,1,0;1,0,1)$

$p1 = [(X, p, \lambda)_{ins}, \mathcal{M} \cup (N' \setminus \{X\})]$
$p2 = [(\lambda, X, p)_{del}, (\mathcal{M} \setminus \{p\}) \cup (N' \setminus \{X\})]$
$p3 = [(p, Y, \lambda)_{ins}, (\mathcal{M} \setminus \{p\}) \cup N']$
$p4 = [(p, p', \lambda)_{ins}, (\mathcal{M} \setminus \{p\}) \cup (N' \setminus \{Y\}) \cup (\{pZ \mid Z \neq Y\})]$
$p5 = [(p', b, \lambda)_{ins}, (\mathcal{M} \setminus \{p, p'\}) \cup (N' \setminus \{Y\}) \cup (\{p'Z \mid Z \neq Y\})]$
$p6 = [(\lambda, p, p')_{del}, \{p'Y\}]$
$p7 = [(\lambda, p', \lambda)_{del}, \{p\}]$

Optimizing the rules - Does the following work?? WHY??

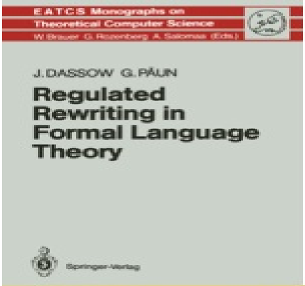
$p1 = [(X, p, \lambda)_{ins}, \mathcal{M} \cup (N' \setminus \{X\})]$
$p2 = [(\lambda, X, p)_{del}, (\mathcal{M} \setminus \{p\}) \cup (N' \setminus \{X\})]$
$p3 = [(p, Y, \lambda)_{ins}, (\mathcal{M} \setminus \{p\}) \cup N']$
$p5 = [(p, b, \lambda)_{ins}, (\mathcal{M} \setminus \{p\}) \cup (N' \setminus \{Y\}) \cup (\{pZ \mid Z \neq Y\})]$
$p7 = [(\lambda, p, \lambda)_{del}, \{\alpha Y \mid \alpha \neq b\}]$

Summary

Outcome of the talk

- Defined Insertion-Deletion systems
- Variants of Ins-del systems
  1. Matrix
  2. Graph-Controlled
  3. (Simple) Semi-conditional
  4. Forbidding
- Showed how these systems can simulate RE with certain sizes.

References



H. Fernau, Lakshmanan and Indhumathi

- On path-controlled insertion-deletion systems. Accepted with Acta Informatica, 2017.
- Investigations on the power of matrix insertion-deletion systems with small sizes.Natural Computing, 17(2):249–269, 2018.
- On the computational completeness of graph-controlled insertion-deletion systems with binary sizes, Theoretical Computer Science, 682, 100–121 (2017). Special Issue on Languages and Combinatorics in Theory and Nature : Dedication to J. Dasssow

THANK YOU