Linguaggi liberi e regolari a confronto

Prof. A. Morzenti

aa 2008-2009

LE GRAMMATICHE DEI LINGUAGGI REGOLARI

I linguaggi regolari sono un caso particolare dei linguaggi liberi.

I linguaggi regolari sono generati da grammatiche che presentano una forte restrizione nella forma delle regole.

Le frasi dei linguaggi regolari, al crescere della lunghezza, presentano necessariamente certe ripetitività.

Dimostreremo che certi linguaggi liberi non sono generabili da alcuna espressione regolare.

DIFFERENZA FONDAMENTALE:
il riconoscimento dei LINGUAGGI REGOLARI richiede memoria finita

il riconoscimento dei LINGUAGGI LIBERI richiede memoria illimitata

DALL'ESPRESSIONE REGOLARE ALLA GRAMMATICA LIBERA

Sostituzione di regole ricorsive al posto degli operatori iterativi (stella e croce).

Si decompone ripetutamente la e.r. *r* nelle sue sottoespressioni, numerandole progressivamente. Dalla definizione stessa di e.r. derivano i seguenti casi (Le lettere maiuscole indicano simboli n.t.
Se E_i è un terminale $a_i \in \Sigma$, si scrive ai al posto di E_i)

1. $r = r_1.r_2....r_k$	1. $E = E_1 E_2 ... E_k$
2. $r = r_1 \cup r_2 \cup .... \cup r_k$	2. $E = E_1 \bigcup E_2 \bigcup ... \bigcup E_k$
3. $r = (r_1)^*$	3. $E = EE_1 \mid \varepsilon$ oppure $E = E_1 E \mid \varepsilon$
4. $r = (r_1)^+$	4. $E = EE_1 \mid E_1$ oppure $E = E_1 E \mid E_1$
5. $r = b \in \Sigma$	5. $E = b$
6. $r = \varepsilon$	6. $E = \varepsilon$

ESEMPIO

$$E = (abc)^{*} \bigcup (ff)^{+}$$

$E = E_1 \cup E_2$	quindi	$E \rightarrow E1 \mid E2$
$E_1 = (E_3)^*$	quindi	$E_1 \rightarrow E_1\, E_3 \mid \varepsilon$
$E_3 = abc$	quindi	$E_3 \rightarrow abc$
$E_2 = (E_4)^+$	quindi	$E_2 \rightarrow E_2\, E_4 \mid E_4$
$E_4 = ff$	quindi	$E_4 \rightarrow ff$

Se l’espressione regolare è ambigua anche la grammatica così ottenuta è ambigua.

Quindi ogni ling. regolare è anche libero
Ma vi sono linguaggi liberi non regolari (e.g. palindromi)

Quindi $\quad$ REG $\subset$ LIB

Cerchiamo ora sottoclasse delle grammatiche libere equivalente alle espr. regolari

GRAMMATICHE LINEARI

Una grammatica è detta LINEARE se ogni regola ha al più un nt nella parte dx

$$\boxed{A \to uBv \quad \text{dove} \quad u, v \in \Sigma^*, B \in (V \bigcup \varepsilon)}$$

La famiglia dei linguaggi lineari è ancora troppo potente per i linguaggi regolari.

ESEMPIO: linguaggio lineare non regolare

$$L_1 = \left\{a^n b^n \mid n \geq 1\right\} = \{ac, aacc, \ldots\}$$

$$S \to aSc \mid ac$$

GRAMMATICHE UNILINEARI (di tipo 3)

LINEARE A DESTRA: $\boxed{A \rightarrow uB \quad \text{dove} \quad u \in \Sigma^*, B \in (V \bigcup \varepsilon)}$

LINEARE A SINISTRA: $\boxed{A \rightarrow Bv \quad \text{dove} \quad v \in \Sigma^*, B \in (V \bigcup \varepsilon)}$

Alberi crescono in modo sbilanciato (destro o sinistro).

grammatica lineare a destra (sinistra) $\Rightarrow$ derivazioni ricorsive solo a destra (sinistra)

Senza perdita in generalità si può anche imporre che le regole terminali siano nulle:
e.g., regola $B \rightarrow b$ viene sostituita con regole $B \rightarrow bB'$ e $B' \rightarrow \varepsilon$

Si può mostrare (passando per gli automi finiti, cfr. Cap.3) che espressioni regolari traducibili in grammatiche unilineari

Quindi **REG $\subseteq$ UNILIN**

ESEMPIO: frasi contenenti la sottostringa *aa* e terminanti per *b* sono definite dalla e.r. (ambigua)

$$(a \mid b)^* aa (a \mid b)^* b$$

1. gramm. lineare a destra G_d

$$S \rightarrow aS \mid bS \mid aaA \quad A \rightarrow aA \mid bA \mid b$$

2. gramm. lineare a sinistra G_s

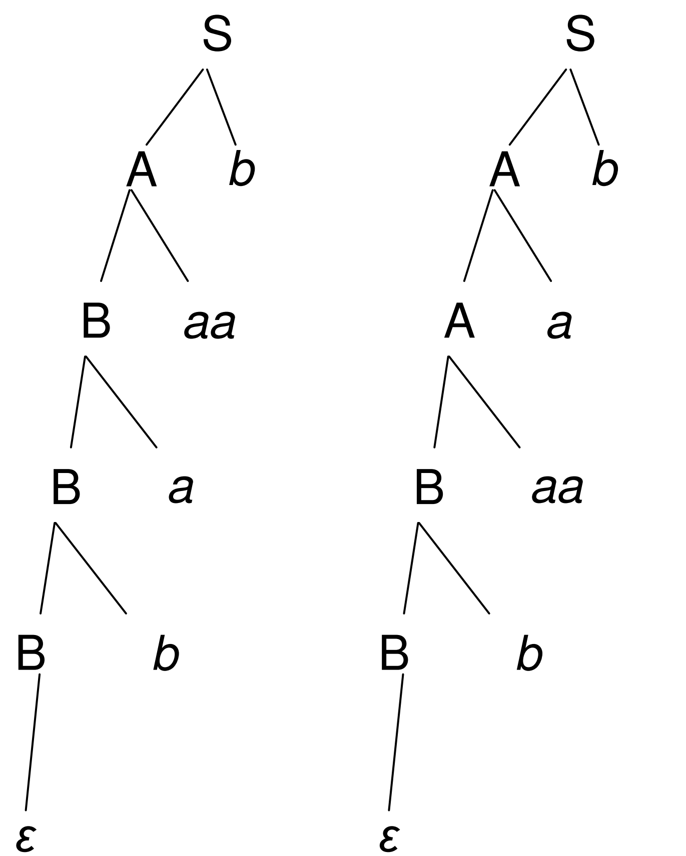
$$S \rightarrow Ab \quad A \rightarrow Aa \mid Ab \mid Baa$$
$$B \rightarrow Ba \mid Bb \mid \varepsilon$$

3. gramm. equiv non unilineare

$$E_1 \rightarrow E_2 aa E_2 b \quad E_2 \rightarrow E_2 a \mid E_2 b \mid \varepsilon$$

NB:anche non ambigua

G_s – alberi sintattici della frase ambigua *baaab*



DALLA GRAMMATICA UNILINEARE ALL'ESPRESSIONE REGOLARE: EQUAZIONI LINEARI

Mostriamo che da ogni grammatica unilineare si ricava un'E.R. equivalente

Quindi UNILIN $\subseteq$ REG

Quindi (per la proprieta` **REG $\subseteq$ UNILIN** mostrata prima) UNILN = REG

regole di grammatica lineare a destra come equazioni

Incognite: i linguaggi generati da ogni n.t.

Sia G - strettamente lineare (e.g.) a destra e
- (senza perdere gen.) con regole terminali nulle A → ε

$$G = (V, \Sigma, P, S)$$

$$L_A = \left\{ x \in \Sigma^* \mid A \overset{+}{\Rightarrow} x \right\} \quad L(G) \equiv L_S$$

$x \in \Sigma^*$ appartiene a L_A nei seguenti casi:

° x è la stringa vuota (in P: $A \rightarrow \varepsilon$);

° $x = ay$ (in P: $A \rightarrow aB$ e $y \in L_B$)

per ogni nt A_0 definito da $A_0 \rightarrow a_1A_1 \mid a_2A_2 \mid \ldots \mid a_kA_k \mid \varepsilon$

$L_{A_0} = a_1L_{A_1} \bigcup a_2L_{A_2} \bigcup \ldots \bigcup a_kL_{A_k} \bigcup \varepsilon$

... otteniamo così un sistema di $n = |V|$ equazioni a n incognite risolvibile con metodo di sostituzione / appl. *identità di Arden*

1

L'equazione $X = KX \bigcup L$ (con K ling non vuoto ed L ling qualsiasi) ha una e <u>una sola</u> soluzione $X = K^*L$ $(K^*L = KK^*L \bigcup L....)$

NB: non dimostraimo l'unicita`

2

2 è soluzione: questa espr. ottenuta sostituendo 2 in 1

ESEMPIO: Equazioni insiemistiche

Grammatica per lista di elementi ***e*** (anche mancanti) separati dal carattere *s.*

$$\boxed{S \rightarrow sS \mid eA \quad A \rightarrow sS \mid \varepsilon}$$

Si trasforma in sistema di equazioni:

$$\begin{cases} L_s = sL_S \bigcup eL_A \\ L_A = sL_S \bigcup \varepsilon \end{cases}$$

sost. seconda eq. nella prima

$$\begin{cases} L_s = sL_S \bigcup e(sL_S \bigcup \varepsilon) \\ L_A = sL_S \bigcup \varepsilon \end{cases}$$

propr. Distr. concat. risp. $\cup$
poi L_s a suffisso comune

$$\begin{cases} L_s = (s \bigcup es)L_S \bigcup e \\ L_A = sL_S \bigcup \varepsilon \end{cases}$$

con identità di Arden

$$\begin{cases} L_s = (s \bigcup es)^* e \\ L_A = sL_S \bigcup \varepsilon \qquad L_A = s(s \bigcup es)^* e \bigcup \varepsilon \end{cases}$$

LINGUAGGI REGOLARI E LIBERI A CONFRONTO

ANALISI DEI LIMITI DELLA FAMIGLIA DEI LINGUAGGI REGOLARI:
quali costrutti linguistici sono modellabili con e.r. e quali invece necessitano della maggiore potenza delle grammatiche ?

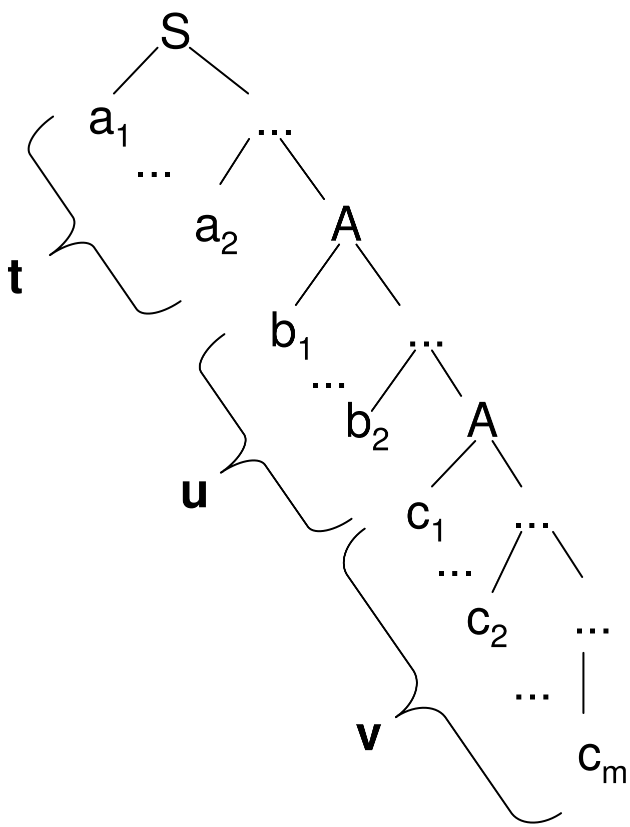
Introduciamo una proprietà che utilizziamo per mostrare come alcuni linguaggi liberi non siano regolari.

SI basa sull'idea che nei linguaggi regolari (quindi nelle grammatiche unilineari) ci siano delle ***RIPETIZIONI INEVITABILI***

PROPRIETÀ: Sia data una grammatica unilineare G. Ogni frase x sufficientemente lunga (di lungh. superiore a una costante k dipendente da G), può essere fattorizzata come $x = t\ u\ v$ - con u non vuota – in modo tale che per ogni $n \geq 1$, $t\ u^n\ v$ appartenga al linguaggio (la frase può essere POMPATA iniettando un numero arbitrario di volte la stringa u).

SI NOTI l'analogia col ***pumping lemma*** degli automi finiti…

DIMOSTRAZIONE: Considerando una grammatica strettamente lineare a destra avente k simboli n.t. Nella derivazione di una frase x lunga k o più caratteri, vi è necessariamente un n.t. *A* che compare almeno due volte.



$$t = a_1 a_2 \ldots, \quad u = b_1 b_2 \ldots \quad v = c_1 c_2 \ldots c_m$$

$$S \overset{+}{\Rightarrow} tA \overset{+}{\Rightarrow} tuA \overset{+}{\Rightarrow} tuv$$

consente di derivare anche *tuuv* e *tv*

USIAMO LA PROPRIETÀ PER MOSTRARE CHE NON È REGOLARE:

$$\boxed{L_1 = \{a^n c^n \mid n \geq 1\}}$$

Usiamo la proprieta` per dimostrare che iinguaggio a due esponenti uguali $L_1 = \{ a^n c^n \mid n \geq 1\}$ NON e` regolare. Supponiamo per assurdo che lo sia: allora

NB: questo ***k*** e` quello della dimostrazione (lucido prec.)

$$x = a^k c^k = tuv \text{ con } u \text{ non vuota}$$

ci sono tre modi di prendere la stringa *u*

	t	u	v	*stringa pompata*
u fatta tutta di a ⟶	1. $t = a^h$	$u = a^i$	$v = a^j c^k$	$a^h a^i a^i a^j c^k$
		$1 \leq i \leq k$	$h + i + j = k$	$h + 2i + j > k$
u fatta di a e di **c** ⟶	2. $t = a^h$	$u = a^i c^j$	$v = c^m$	$a^h a^i c^j a^i c^j c^m \notin a^+ c^+$
		$1 \leq i, j \leq k$	$h + i = j + m = k$	
u fatta tutta di a ⟶	3. $t = a^k c^h$	$u = c^i$	$v = c^j$	$a^k c^h c^i c^i c^j$
		$1 \leq i \leq k$	$h + i + j = k$	$h + 2i + j > k$

In tutti i casi la stringa pompata (che apparterrebbe al linguaggio se questo fosse regolare), non ha la forma richiesta *a^n c^n .*

RUOLO DELLE DERIVAZIONI AUTOINCLUSIVE

$$\boxed{A \overset{+}{\Rightarrow} uAv \quad u \neq \varepsilon \wedge v \neq \varepsilon}$$

Le derivazioni autoinclusive sono assenti dalle grammatiche unilineari dei linguaggi regolari. Non è così per i linguaggi liberi non regolari (palindromi, linguaggio di Dyck, linguaggio a esponenti uguali, ...).

È l'assenza di derivazioni autoiclusive che consente di risolvere le equazioni insiemistiche associate alle grammatiche unilineari e conferisce struttura semplice ai linguaggi derivati. Quindi …

GRAMMATICA SENZA DERIVAZIONI AUTOINCLUSIVE $\Rightarrow$ LINGUAGGIO REGOLARE.

NB: non vale necessariamente il converso, cioe`
una gr. con derivazioni autoinclusive puo` generare un linguaggio regolare

ESEMPIO: Grammatica non autoinclusiva e linguaggio regolare

$$G: \quad S \to AS \mid bA \quad A \to aA \mid \varepsilon$$

$$\begin{cases} L_s = L_A L_S \bigcup bL_A \\ L_A = aL_A \bigcup \varepsilon \end{cases}$$

$$\begin{cases} L_s = L_A L_S \bigcup bL_A \\ L_A = a^* \end{cases}$$

$$L_s = a^* L_S \bigcup ba^*$$

$$L_S = (a^*)^* ba^* \qquad L(G) = a^* ba^*$$

(CONTRO)ESEMPIO: Grammatica autoinclusiva, linguaggio regolare

$$G = \{S \to aSa \mid \varepsilon\}$$

$$S \Rightarrow aSa \qquad L(G) = (aa)^*$$

può essere definito

$$G_d = \{S \to aaS \mid \varepsilon\}$$

curiosa proprieta` dei linguaggi con alfabeto di un solo simbolo: linguaggi definiti da grammatica libera sono regolari se $|\Sigma|=1$

LIMITI DEI LINGUAGGI LIBERI DA CONTESTO

Nei linguaggi liberi da contesto le frasi abbastanza lunghe contengono necessariamente ***due*** sottostringhe che si possono ripetere quante volte si voglia, dando così luogo a strutture autoincassate. Tale vincolo impedisce alle grammatiche libere di generare costrutti in cui ***tre o più*** parti siano ripetute il medesimo numero di volte.

PROPRIETÀ: Sia G una grammatica libera. Ogni sua frase x di lunghezza superiore a una costante dipendente solo da G, può essere fattorizzata come X = t u v w z, dove almeno una delle stringhe u e w non sia vuota, in modo tale che per ogni $n \geq 0$, la stringa $t\, u^n\, v\, w^n\, z$ appartenga al linguaggio.

NB: non la dimostriamo: cfr. Libro di testo

LIMITI DEI LINGUAGGI LIBERI DA CONTESTO: ESEMPI (senza dimostrazioni, cfr. testo)

ESEMPIO: IL LINGUAGGIO A TRE ESPONENTI NON È LIBERO.
(La dimostrazione sfrutta la possibilità di pompare le frasi del linguaggio mediante la ripetizione di una derivazione ricorsiva).

$$\boxed{L = \{a^n b^n c^n \mid n \geq 1\}}$$

ESEMPIO: LINGUAGGIO DELLE REPLICHE
Si incontra nei linguaggi tecnici ogni volta che gli elementi di due liste devono essere in corrispondenza (es. par. formali e attuali)

$$\boxed{\begin{array}{l} L_{replica} = \{uu \mid u \in \Sigma^+\} \\ \Sigma = \{a, b\} \quad x = abbbabbb \end{array}}$$

PROPRIETÀ DI CHIUSURA DI REG E DI LIB

Terminiamo il confronto tra linguaggi liberi e regolari analizzando le proprietà di chiusura rispetto alle operazioni più comuni.

Sia L un linguaggio di LIB e R un linguaggio di REG:

$R^R \in REG$ $\quad\quad$ $L^R \in LIB$

$R^* \in REG$ $\quad\quad$ $L^* \in LIB$

$\neg R \in REG$ $\quad\quad$ $\neg L \notin LIB$ ← LIB non chiuso risp.compl. perche\` chiuso risp. unione ma non risp. intersezione

$\oplus$ sta per $\cup$ o $\cdot$ → $R_1 \oplus R_2 \in REG$ $\quad\quad$ $L_1 \oplus L_2 \in LIB$

$R_1 \cap R_2 \in REG$ $\quad\quad$ $L_1 \cap L_2 \notin LIB$ ← $\{a^n b^n c^n\} = \{a^n b^n c^+\} \cap \{a^+ b^n c^n\}$

$L \cap R \in LIB$ ← Dimostrata piu\` avanti con automi

Proprieta\` di R dimostrate (facilmente) mediante automi finiti

COMMENTI ED ESEMPI

1) Il linguaggio riflesso di L(G) è generato dalla *grammatica riflessa*, quella ottenuta riflettendo specularmente le parti destre delle regole (G lin dx diventa G lin sin, ...).

2) Stella, unione e concatenamento di linguaggi liberi sono liberi.

G_1 e G_2, L_1 L_2, assiomi S_1 S_2, $V_1 \cap V_2 = \emptyset$	
stella:	$S \to SS_1 \mid \varepsilon$
unione:	$S \to S_1 \mid S_2$
concatenamento:	$S \to S_1 S_2$

3) (GIA VISTO) Se le grammatiche hanno n.t. in comune, l'unione delle loro regole produce una grammatica che in generale definisce un linguaggio più ampio dell'unione dei due linguaggi:

$$G_1 = \{S_1 \to aA, A \to bA \mid b\}, \quad G_2 = \{S_2 \to aA, A \to cA \mid c\}$$

$$G_{1\cup 2} = \{S \to S_1 \mid S_2, S_1 \to aA, A \to bA \mid b, S_2 \to aA, A \to cA \mid c\}$$

$$a(b \mid c)^+ \supset L(G_1) \cup L(G_2) = ab^+ \cup ac^+$$

4) Il simbolo di non appartenenza significa che esistono dei ling. che non appartengono alla famiglia.

5) L'intersezione di due linguaggi liberi non è in generale libera: ecco (contro)esempio.

$$\boxed{\left\{a^n b^n c^n \mid n \geq 1\right\} = \left\{a^n b^n c^+ \mid n \geq 1\right\} \bigcap \left\{a^+ b^n c^n \mid n \geq 1\right\}}$$

6) Esistono dei linguaggi liberi il cui complemento non è libero.
Conseguenza di (5) e DeMorgan: $L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$ essendo LIB chiuso per $\cup$, se fosse chiuso per $\neg$ dovrebbe essere chiuso anche per $\cap$

ATTENZIONE: questo non significa che il complemento di ogni ling libero non e` libero: esempio banale $\{a^n \mid n \text{ pari}\}$ e` libero cosi` come $\{a^n \mid n \text{ dispari}\}$

INTERSEZIONE CON LINGUAGGI REGOLARI

Per rendere una grammatica più selettiva la si può filtrare attraverso (i.e., fare intersezione con) un linguaggio regolare.

ESEMPIO: Filtri regolari sul linguaggio di Dyck

frasi prive della sottostringa *cc*, ossia prive di parentesi annidate

$$L_1 = L_D \cap \neg(\Sigma^* cc \Sigma^*) = (ac)^*$$

frasi con un solo nido di parentesi

$$L_2 = L_D \cap \neg(\Sigma^* ca \Sigma^*) = \left\{ a^n c^n \mid n \geq 0 \right\}$$

Entrambi sono linguaggi liberi, e il primo è anche regolare.

TRASFORMAZIONI ALFABETICHE

Linguaggi molto simili differiscono solo per alcuni simboli terminali.
TRASLITTERAZIONE o OMOMORFISMO ALFABETICO è l'operazione linguistica che sostituisce a certi caratteri terminali altri caratteri terminali (conserva la "struttura", cambia la "facciata")

TRASLITTERAZIONE (o OMOMORFISMO) ALFABETICO è una funzione:

$$\boxed{h: \quad \Sigma \to \Delta \bigcup \{\varepsilon\}}$$

Traslitterazione priva di cancellazioni: per ogni $c \in \Sigma$ $\boxed{h(c) \neq \varepsilon}$

Traslitterazione di una stringa sorgente:

$$a_1 a_2 \ldots a_n \in \Sigma^*$$

$$h(a_1)h(a_2)\ldots h(a_n)$$

$$h(v.w) = h(v).h(w)$$

ESEMPIO: Stampante

$$h(c) = c \quad \text{se } c \in \{a, b, ..., z, 0, 1, ..., 9\};$$

$$h(c) = c \quad \text{se } c \text{ è punteggiatura o s.b.}$$

$$h(c) = \square \quad \text{se } c \in \{\alpha, \beta, ..., \omega\};$$

$$h(\text{start-text}) = h(\text{end-text}) = \varepsilon$$

h(start-text la cost. π vale 3.14 end-text)=la cost. □ vale 3.14

TRASLITTERAZIONE A PAROLE: nella traslitterazione l’immagine di almeno un carattere sorgente è una stringa, non unitaria, dell’alfabeto pozzo

$$h(\leftarrow) = ':=', \quad h(c) = c \;\; per\, ogni\, altro \;\; c \in \Sigma$$

SOSTITUZIONE (già vista) – a un carattere sorgente viene sostituita una stringa presa da un linguaggio specificato. Dato un alfabeto sorgente: $\Sigma = a, b, ...$

una sostituzione *h* associa a ogni lettera un linguaggio di alfabeto pozzo Δ

$$\boxed{h(a) = L_a, \quad h(b) = L_b}$$

L'applicazione della sostituzione *h* a una stringa sorgente

$$x = a_1 a_2 ... a_n \in \Sigma^*$$

produce un insieme di stringhe $h(x) = \{ y_1 y_2 ... y_n \mid y_i \in L_{a_i} \}$

Una traslitterazione a parole è una sostituzione dove ogni linguaggio immagine contiene una sola stringa; se poi la stringa è di lunghezza uno o zero, la traslitterazione è alfabetica.

CHIUSURA RISPETTO ALLE TRASFORMAZIONI ALFABETICHE

PROPRIETÀ: Le famiglie LIB e REG sono chiuse rispetto alla traslitterazione a parole

DIMOSTRAZIONE: Sia G la grammatica di L e h una traslitterazione a parole dall'alfabeto Σ a stringhe di alfabeto Δ. Si applica la traslitterazione a ogni regola $A \rightarrow \alpha$ (lasciando invariati la freccia e i n.t).

$$\boxed{h(A \rightarrow \alpha) \equiv A \rightarrow h(\alpha)}$$

Le regole generate hanno forma di sintassi libera da contesto e sono dello stesso tipo delle regole di partenza. La grammatica ottenuta genera il linguaggio *h(L(G)).*

NB: la costruzione (quindi la dimostrazione) vale anche per i linguaggi regolari: le grammatiche rimangono unilineari applicando un omomorfismo a parole alle parti destre delle regole

CHIUSURA RISPETTO ALLE TRASFORMAZIONI ALFABETICHE

PROPRIETÀ. Le famiglie LIB e REG sono chiuse rispetto alla sostituzione di linguaggi della stessa famiglia.

DIMOSTRAZIONE: Sia G la grammatica libera di L e *h* una sostituzione tale che, per ogni $c \in \Sigma$, L_c sia linguaggio libero di grammatica G_c e di assioma S_c con alfabeti n.t. di G, G_a, G_b, ... disgiunti.

Per ottenere la grammatica G' del linguaggio *h(L)*, applichiamo alle regole di G la traslitterazione alfabetica *f* definita come $f(c) = S_c$ per ogni terminale $c \in \Sigma$; *lasciamo* $f(A) = A$ per ogni altro simbolo di G.

Per ottenere G':

° a ciascuna regola $A \rightarrow \alpha$ di G applica la traslitterazione *f* che cambia ogni carattere terminale nell'assioma della grammatica corrispondente

° alle regole così ottenute si aggiungono quelle delle grammatiche G, G_a, G_b, ...

La grammatica così ottenuta genera il linguaggio *h(L(G))…*