

1. Considera la seguente funzione da $\{0,1\}^*$ a $\{0,1\}^*$:

$$\text{stutter}(w) = \begin{cases} \varepsilon & \text{se } w = \varepsilon \\ aa.\text{stutter}(x) & \text{se } w = ax \text{ per qualche simbolo } a \text{ e parola } x \end{cases}$$

Dimostra che se L è un linguaggio regolare sull'alfabeto $\{0,1\}$, allora anche il seguente linguaggio è regolare:

$$\text{stutter}(L) = \{\text{stutter}(w) \mid w \in L\}.$$

2. Considera il linguaggio

$$L_2 = \{w0^n \mid w \in \{0,1\}^* \text{ e } n = |w|\}.$$

Dimostra che L_2 non è regolare.

3. Per ogni linguaggio L , sia $\text{suffix}(L) = \{v \mid uv \in L \text{ per qualche stringa } u\}$. Dimostra che se L è un linguaggio context-free, allora anche $\text{suffix}(L)$ è un linguaggio context-free.

Se L è un linguaggio regolare esiste un DFA che riconosca L .

Avremo quindi una grammatica $G (Q, \Sigma, \delta, q_0, S)$ per cui dall'automa sarà riconosciuta similmente $G (Q', \Sigma, \delta', q_0, S')$

Se applicassimo stutter su (0110) avremmo la parola 00111100, letteralmente invertito

Si nota che il linguaggio può essere composto da:

- ε nel caso in cui $w = \varepsilon$
- ricorsivamente $aa.\text{stutter}$, nel caso in cui $w = ax$ nell'alfabeto $\{0,1\}^*$

L'automa quindi procede ripetendosi solamente se lo stato attuale è uguale a quello precedente, in particolare concatenando la parte ricorsiva (aa), la parola (x) e w (stringa attuale).

Ragioniamo quindi creando degli stati A formati da triple (aa, r_x, r_w) .

Avremmo quindi due possibili casi:

- $(aa, r_x, r_w) \rightarrow (aa, r_x)$ nel caso in cui $w = \varepsilon$ (rimane inalterato)
- $(aa, r_x, r_w) \rightarrow (aa, s_x, s_w)$ con $s_w = ax$ nel caso in cui invece si abbia $w = ax$, in questo modo (x) potrà essere parola generica, ci sarà (aa) ripetuto ricorsivamente e l'altro simbolo rappresenta un simbolo nell'alfabeto $\{0,1\}^*$.

Stabilite le possibili transizioni, definiamo gli stati finali dell'automa.

L'idea del linguaggio è avere alternanza di "a" ed "x" in maniera discontinua, quindi gli stati finali saranno idealmente costituiti da alternanza di entrambi i simboli.

Al di là del caso di $w = \varepsilon$, avremmo una situazione in cui avremo:

- $F_A = aa.(s_w, s_x, w)$ in cui appunto si ha la ripetizione di aa concatenato all'alternanza di "w", di "x" e della stessa parola w scelta nell'alfabeto
- $F_A = aa.(s_x, s_w, w)$ in cui si hanno gli stessi stati letteralmente alternati gli uni con gli altri nell'ordine inverso.

L'idea del linguaggio è letteralmente di duplicare stringhe, alternandole. Con questa dimostrazione si nota la regolarità del linguaggio.

2) Assumiamo per assurdo il linguaggio sia regolare.

Per completezza assumo entrambi i casi (a noi serve $|w| = n$)

- se $n \neq w \rightarrow 0^k 0^p$ per $k, p > 0$.
Assumiamo quindi $w = xyz \rightarrow 0^k 0^p$
Con pumping $xy^0z \rightarrow x=0^p \quad y=0^q \quad z=0^{k-p-q}$
Quindi avremo uno sbilanciamento degli 0 nelle parti della parola, infatti:
 $0^p 0^{k-p-q} \rightarrow 0^{k-q}$ che sarebbe assurdo

Dunque il linguaggio non è regolare.

- se $n = w, 1^k 0^p$ per $k, p > 0$.
A questo punto assumiamo (date le solite condizioni $y \neq \varepsilon, |xy| \leq k$):
 $w = xyz \rightarrow 1^k 0^p$
Con pumping xy^0z , il rimanente numero di 0 sarà compensato dagli 1 del resto della parola, quindi:

$$x=1^p \quad y=1^q \quad z=1^{k-p}0^k$$

Quindi avremo più 1 che 0, infatti:

$$1^p 1^{k-p-q} 0^k \rightarrow 1^{k-q} 0^k \quad \text{che sarebbe assurdo}$$

Dunque il linguaggio non è regolare.

3) (fatto così dal prof)

Sia G la grammatica che genera L.

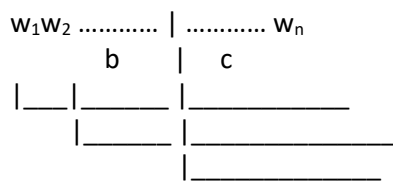
Assumiamo che G sia in forma normale di Chomsky.

Avremo quindi regole del tipo:

$$A \rightarrow BC$$

$$D \rightarrow d$$

Siccome c'è una concatenazione, un pezzo della parola sarà generato da "b" e un altro da "C"



A' suffissi delle parole generate da A

$$A' \rightarrow A|B'C|C'| \epsilon$$

Quindi quello che viene adesso è la variabile iniziale

$$D' \rightarrow d| \epsilon$$

S' è la variabile iniziale sse S è variabile iniziale di G

L'esercizio dei *prefix* inverte la stessa cosa.