Tutorato di Automi e Linguaggi Formali

Homework 3: GNFA–ER e Conversioni, Pumping Lemma, Minimum Pumping Length

Gabriel Rovesti
Corso di Laurea in Informatica - Università degli Studi di Padova

Tutorato 3 - 24-03-2025

1 Conversione (GNFA, NFA, DFA, ER)

Esercizio 1. Convertire le seguenti espressioni regolari in NFA utilizzando le costruzioni viste a lezione (unione, concatenazione, stella):

a) $(ab)^* + (ba)^*$

b) $a(a+b)^*b$

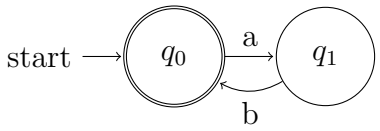
c) $(a+\varepsilon)(b+c)^*$

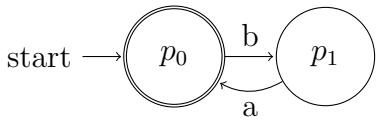
Per ogni NFA ottenuto, fornire il **diagramma degli stati** e la **tabella di transizione**.

Soluzione. Costruiamo gli NFA seguendo le regole di conversione per unione, concatenazione e stella di Kleene.

a) $(ab)^* + (ba)^*$

Costruiamo separatamente gli NFA per $(ab)^*$ e $(ba)^*$, poi li uniamo:





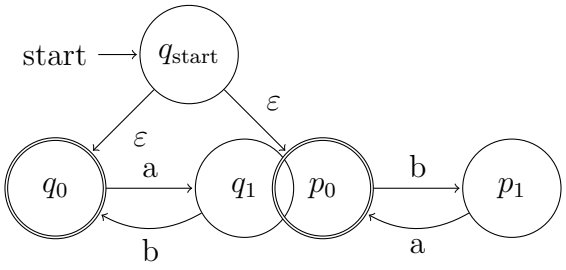
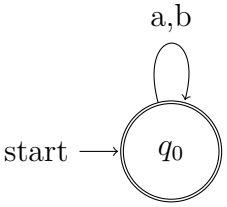


Tabella di transizione:

Stato	**a**	**b**	ε
q_{start}	$\emptyset$	$\emptyset$	$\{q_0, p_0\}$
q_0	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\{q_0\}$	$\emptyset$
p_0	$\emptyset$	$\{p_1\}$	$\emptyset$
p_1	$\{p_0\}$	$\emptyset$	$\emptyset$

b) $a(a+b)^*b$

Costruiamo prima l'NFA per $(a+b)^*$, poi lo combiniamo con 'a' all'inizio e 'b' alla fine:



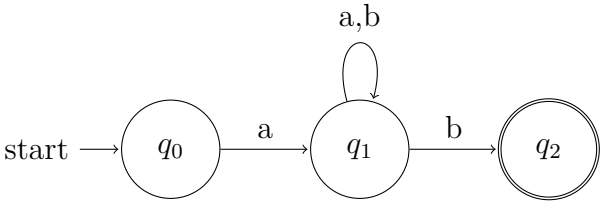
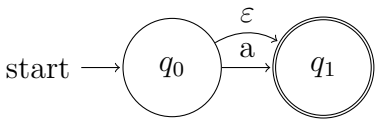


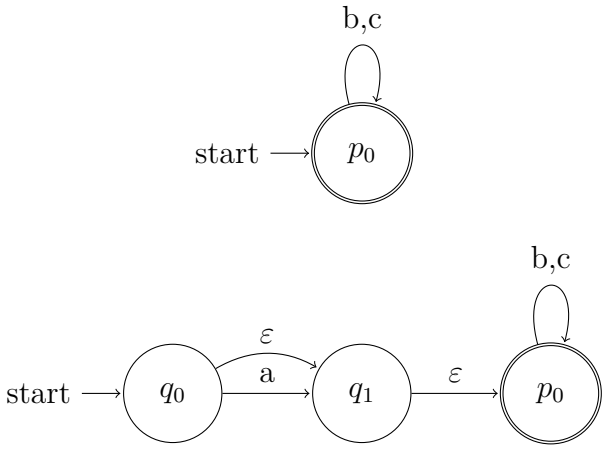
Tabella di transizione:

Stato	**a**	**b**
q_0	$\{q_1\}$	$\emptyset$
q_1	$\{q_1\}$	$\{q_1, q_2\}$
q_2	$\emptyset$	$\emptyset$

c) $(a+\varepsilon)(b+c)^*$

Costruiamo prima gli NFA per $(a+\varepsilon)$ e $(b+c)^*$, poi li concateniamo:





Possiamo semplificare questo NFA eliminando alcune ε-transizioni:

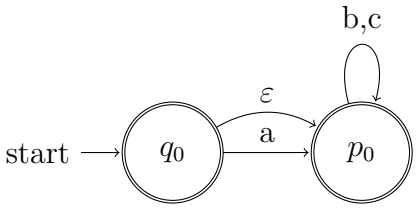


Tabella di transizione:

Stato ε	**a**	**b**	**c**
q_0 $\{p_0\}$	$\{p_0\}$	$\emptyset$	$\emptyset$
p_0 $\emptyset$	$\emptyset$	$\{p_0\}$	$\{p_0\}$

Esercizio 2. Considerare l'espressione regolare $R = (0+1)^*\,0\,(0+1)$ sull'alfabeto $\{0,1\}$.

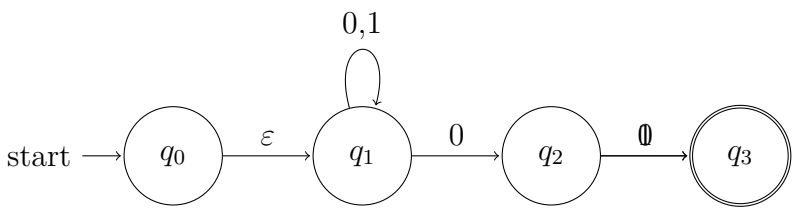
a) Costruire un ε-NFA che riconosca $\mathcal{L}(R)$.

b) Convertire l'ε-NFA in un NFA privo di ε-transizioni.

c) Convertire quest'ultimo NFA in un DFA (costruzione per sottoinsiemi).

Soluzione. a) ε-NFA per $R = (0+1)^*\,0\,(0+1)$

Costruiamo l'ε-NFA componendo le parti dell'espressione regolare:



b) NFA senza ε-transizioni

Eliminiamo l'ε-transizione da q_0 a q_1 propagando le transizioni uscenti da q_1 a q_0:

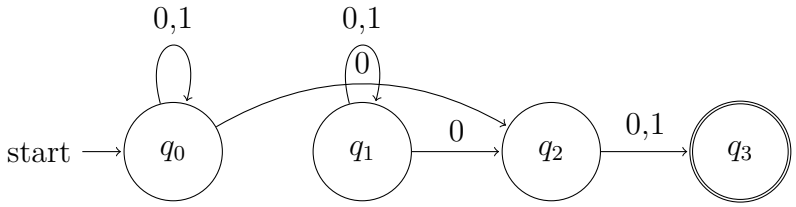


Tabella di transizione per l'NFA senza ε-transizioni:

Stato	**0**	**1**
q_0	$\{q_0, q_2\}$	$\{q_0\}$
q_1	$\{q_1, q_2\}$	$\{q_1\}$
q_2	$\{q_3\}$	$\{q_3\}$
q_3	$\emptyset$	$\emptyset$

c) Conversione in DFA

Applichiamo la costruzione per sottoinsiemi:

1. Stato iniziale: $\{q_0\}$ 2. Calcoliamo le transizioni: - $\delta(\{q_0\}, 0) = \{q_0, q_2\}$ - $\delta(\{q_0\}, 1) = \{q_0\}$ - $\delta(\{q_0, q_2\}, 0) = \{q_0, q_2, q_3\}$ - $\delta(\{q_0, q_2\}, 1) = \{q_0, q_3\}$ - $\delta(\{q_0, q_2, q_3\}, 0) = \{q_0, q_2, q_3\}$ - $\delta(\{q_0, q_2, q_3\}, 1) = \{q_0, q_3\}$ - $\delta(\{q_0, q_3\}, 0) = \{q_0, q_2\}$ - $\delta(\{q_0, q_3\}, 1) = \{q_0\}$

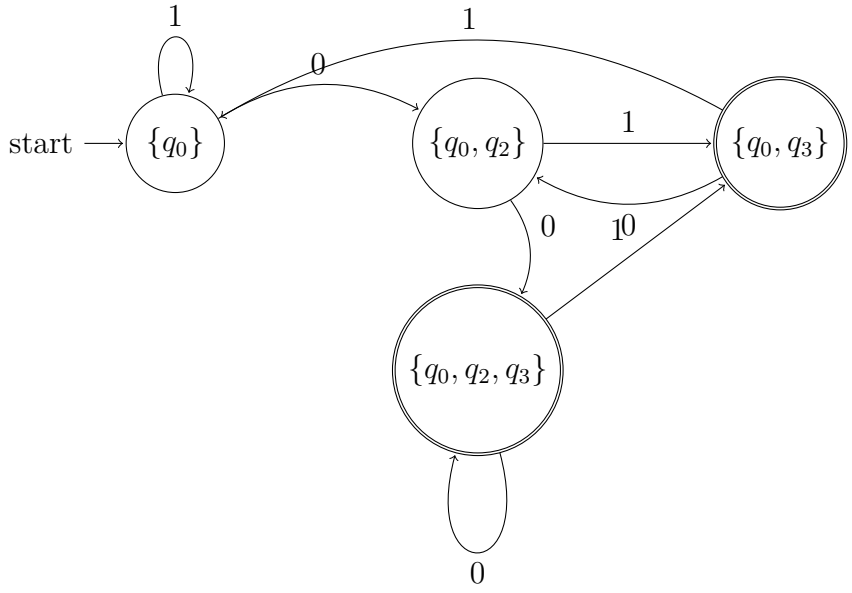
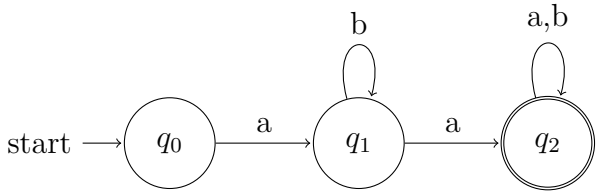


Tabella di transizione del DFA:

Stato	**0**	**1**
$\{q_0\}$	$\{q_0, q_2\}$	$\{q_0\}$
$\{q_0, q_2\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_3\}$
$\{q_0, q_3\}$	$\{q_0, q_2\}$	$\{q_0\}$
$\{q_0, q_2, q_3\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_3\}$

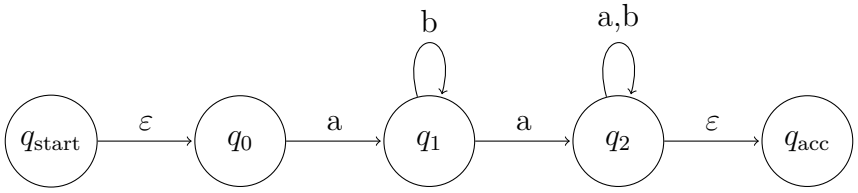
Gli stati finali sono $\{q_0, q_3\}$ e $\{q_0, q_2, q_3\}$ poiché contengono lo stato accettante q_3 dell'NFA originale.

Esercizio 3. Convertire il seguente NFA in un'espressione regolare equivalente tramite *eliminazione degli stati* (GNFA e rimozione graduale):

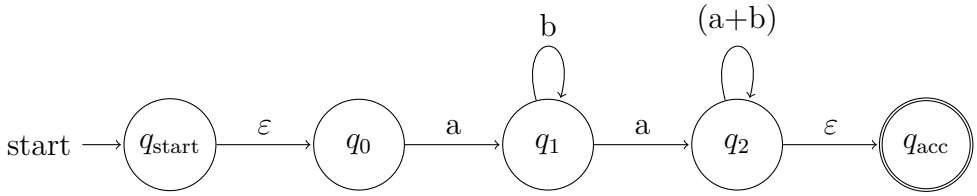


Mostrare tutti i passaggi fondamentali dell'algoritmo e infine l'ER ottenuta.

Soluzione. Trasformiamo prima l'NFA in un GNFA (Generalized NFA) aggiungendo un nuovo stato iniziale q_{start} e un nuovo stato finale q_{acc}, con transizioni opportune:

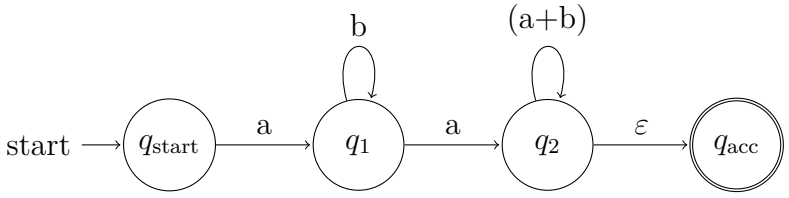


Possiamo trasformarlo in questa forma più compatta con le ε-transizioni già rimosse:



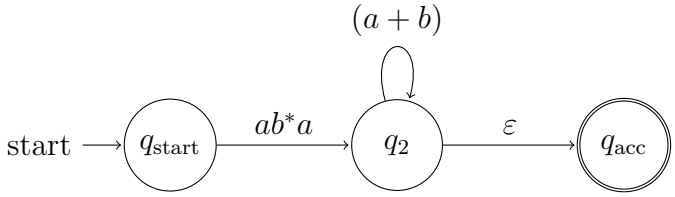
Ora procediamo con l'eliminazione degli stati:

1. Eliminazione di q_0: Aggiorniamo le transizioni $q_{\text{start}} \to q_1$ attraverso q_0: $R_{start,1} = R_{start,0} \cdot R_{0,1} \cdot R_{0,0}^* = \varepsilon \cdot a \cdot \emptyset^* = a$



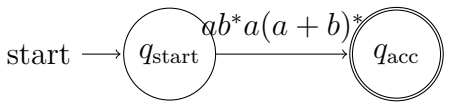
2. Eliminazione di q_1:

Aggiorniamo le transizioni $q_{\text{start}} \to q_2$ attraverso q_1: $R_{start,2} = R_{start,1} \cdot R_{1,2} \cdot R_{1,1}^* = a \cdot a \cdot b^* = ab^*a$



3. Eliminazione di q_2:

Aggiorniamo le transizioni $q_{\text{start}} \to q_{\text{acc}}$ attraverso q_2: $R_{start,acc} = R_{start,2} \cdot R_{2,acc} \cdot R_{2,2}^* = ab^*a \cdot \varepsilon \cdot (a+b)^* = ab^*a(a+b)^*$



L'espressione regolare finale è quindi:

$$R = ab^*a(a+b)^*$$

Questa espressione rappresenta tutte le stringhe che iniziano con 'a', seguite da zero o più 'b', poi un'altra 'a', e infine zero o più simboli 'a' o 'b'.

2 Pumping Lemma, Gioco e Minimum Pumping Length

Esercizio 4. (**Gioco del Pumping Lemma**) Dimostrare via *Pumping Lemma* che i linguaggi seguenti *non* sono regolari, rappresentandolo come una "partita" tra Giocatore 1 (difende la regolarità) e Giocatore 2 (nega la regolarità):

a) $L = \{ w \in \{a,b\}^* \mid \#a(w) = \#b(w)\}$

b) $L = \{ w \in \{a,b\}^* \mid \#a(w) > \#b(w)\}$

c) $L = \{ 0^{n^2} \mid n \geq 0\}$

Per ciascun linguaggio, illustrare:

- La "mossa" con cui Giocatore 2 sceglie la stringa w di lunghezza $\geq p$,
- Come Giocatore 1 è costretto a suddividere $w = xyz$,
- Perché Giocatore 2 può scegliere un i che rende $xy^iz \notin L$.

Soluzione. Il gioco del Pumping Lemma si svolge così:

- Giocatore 1 propone un qualsiasi numero $p > 0$ (pumping length)
- Giocatore 2 sceglie una stringa $w \in L$ con $|w| \geq p$
- Giocatore 1 suddivide $w = xyz$ con $|xy| \leq p$, $|y| > 0$
- Giocatore 2 deve trovare un $i \geq 0$ tale che $xy^iz \notin L$

a) $L = \{ w \in \{a,b\}^* \mid \#a(w) = \#b(w)\}$
Mossa di Giocatore 2: Sceglie $w = a^pb^p$.
Questa stringa ha esattamente p caratteri 'a' e p caratteri 'b', quindi $\#a(w) = \#b(w) = p$ e $w \in L$. Inoltre $|w| = 2p \geq p$.

Suddivisione di Giocatore 1: Deve scegliere $w = xyz$ con $|xy| \leq p$ e $|y| > 0$.
Poiché $|xy| \leq p$, l'intera sottostringa xy cade all'interno della prima parte di w, cioè nel blocco di 'a'. Quindi y contiene solo caratteri 'a'.

Strategia vincente di Giocatore 2: Sceglie $i = 0$ o $i = 2$.

Se $i = 0$, eliminiamo y (che contiene solo 'a') e otteniamo xz che ha meno 'a' che 'b'. Formalmente, se $y = a^k$ con $k > 0$, allora xz contiene $(p - k)$ caratteri 'a' e p caratteri 'b', quindi $\#a(xz) < \#b(xz)$ e $xz \notin L$.

Se $i = 2$, aggiungiamo una copia extra di y (solo 'a') e otteniamo xy^2z che ha più 'a' che 'b'. Formalmente, xy^2z contiene $(p + k)$ caratteri 'a' e p caratteri 'b', quindi $\#a(xy^2z) > \#b(xy^2z)$ e $xy^2z \notin L$.

b) $L = \{ w \in \{a, b\}^* \mid \#a(w) > \#b(w)\}$

Mossa di Giocatore 2: Sceglie $w = a^{p+1}b^p$.

In questa stringa, $\#a(w) = p + 1$ e $\#b(w) = p$, quindi $\#a(w) > \#b(w)$ e $w \in L$. Inoltre, $|w| = 2p + 1 \geq p$.

Suddivisione di Giocatore 1: Deve scegliere $w = xyz$ con $|xy| \leq p$ e $|y| > 0$.

Dato che $|xy| \leq p$ e i primi $p+1$ caratteri di w sono tutti 'a', la sottostringa y contiene solo caratteri 'a'.

Strategia vincente di Giocatore 2: Sceglie $i = 0$.

Rimuovendo y (che contiene solo 'a'), otteniamo xz con meno 'a'. Se $y = a^k$ con $k > 0$, allora xz contiene $(p + 1 - k)$ caratteri 'a' e p caratteri 'b'.

Se $k = 1$, allora $\#a(xz) = p = \#b(xz)$ e quindi $xz \notin L$.

Se $k > 1$, allora $\#a(xz) < p < \#b(xz)$ e quindi $xz \notin L$.

c) $L = \{ 0^{n^2} \mid n \geq 0\}$

Mossa di Giocatore 2: Sceglie $w = 0^{p^2}$.

Questa stringa è in L perché è della forma 0^{n^2} con $n = p$. Inoltre, $|w| = p^2 \geq p$ per $p \geq 1$.

Suddivisione di Giocatore 1: Deve scegliere $w = xyz$ con $|xy| \leq p$ e $|y| > 0$.

Dato che w contiene solo il carattere '0', la sottostringa y sarà della forma $y = 0^k$ per qualche $k > 0$.

Strategia vincente di Giocatore 2: Sceglie $i = 2$ (anche altri valori potrebbero funzionare).

Con $i = 2$, la stringa pompata è $xy^2z = 0^{p^2+k}$.

Dobbiamo dimostrare che non esiste alcun numero intero m tale che $p^2 + k = m^2$. Osserviamo che:

$$p^2 < p^2 + k < p^2 + p \tag{1}$$

Se $k \leq p$, allora $p^2 + k \leq p^2 + p = p(p + 1)$. Inoltre, $p^2 + k > p^2 = p \cdot p$.

Quindi $p^2 < p^2 + k < (p + 1)^2 = p^2 + 2p + 1$, poiché $k \leq p < 2p + 1$ per $p \geq 1$.

Questo significa che $p^2 + k$ è strettamente compreso tra due quadrati consecutivi, p^2 e $(p + 1)^2$, e quindi non può essere un quadrato perfetto.

Pertanto, $xy^2z = 0^{p^2+k} \notin L$ e Giocatore 2 vince.

Esercizio 5. (**Minimum Pumping Length**, rif. Sipser Esercizio 1.55) Per i linguaggi seguenti, determinare la *minima* pumping length (ovvero il più piccolo p per cui "tutte le stringhe di lunghezza $\geq p$ in L sono pompabili"):

a) $000\,1^*$ (alfabeto $\{0, 1\}$)

b) $0^*\,1^*$

c) $\{\texttt{1011}\}$ (linguaggio con un'unica stringa lunga 4)

d) $(01)^*$ (tutte le concatenazioni di "01")

e) Σ^* (tutte le stringhe)

Giustificare brevemente la scelta di p in ciascun caso.

Soluzione. **a)** $000\,1^*$

Questo linguaggio contiene le stringhe della forma "000" seguite da zero o più caratteri '1'.

Per trovare la minima pumping length, esaminiamo le stringhe più corte del linguaggio:

- "000" (lunghezza 3): Non è pompabile. Se prendiamo $y =$ "0", rimuovendolo otteniamo "00" $\notin L$.
- "0001" (lunghezza 4): È pompabile. Possiamo prendere $y =$ "1" e pompare.

Dobbiamo verificare che tutte le stringhe di lunghezza ≥ 4 siano pompabili. Le stringhe di lunghezza ≥ 4 in questo linguaggio sono tutte della forma "000" + almeno un '1', e possiamo sempre scegliere y come uno dei caratteri '1'.

Quindi, la minima pumping length è $p = 4$.

b) $0^*\,1^*$

Questo linguaggio contiene stringhe composte da zero o più '0' seguiti da zero o più '1'.

Consideriamo le stringhe più corte:

- ε (lunghezza 0): Non dobbiamo considerarla per il pumping lemma.
- "0" (lunghezza 1): È pompabile. Possiamo prendere $y =$ "0" e pompare.
- "1" (lunghezza 1): È pompabile. Possiamo prendere $y =$ "1" e pompare.

Tutte le stringhe di lunghezza ≥ 1 sono pompabili, poiché possiamo sempre scegliere y come uno dei caratteri.

Quindi, la minima pumping length è $p = 1$.

c) $\{\texttt{1011}\}$

Questo linguaggio contiene un'unica stringa "1011".

Per un linguaggio finito con una sola stringa, se scegliessimo $p \leq 4$, dovremmo poter pompare la stringa "1011". Tuttavia, qualunque scelta di y (con $|y| > 0$) porterebbe a una stringa che non è in L quando cambiamo il numero di ripetizioni di y.

Pertanto, dobbiamo scegliere $p > 4$, e poiché non ci sono stringhe di lunghezza ≥ 5 nel linguaggio, la condizione del pumping lemma è soddisfatta vacuamente.

Quindi, la minima pumping length è $p = 5$.

d) $(01)^*$

Questo linguaggio contiene concatenazioni di "01", ovvero ε, "01", "0101", "010101", ...

Esaminiamo le stringhe più corte:

- ε (lunghezza 0): Non dobbiamo considerarla per il pumping lemma.
- "01" (lunghezza 2): Non è pompabile. Se prendiamo $y =$ "0" o $y =$ "1", rimuovendolo otteniamo una stringa che non è in L.
- "0101" (lunghezza 4): È pompabile. Possiamo prendere $y =$ "01" e pompare.

Dobbiamo ora verificare che $p = 3$ funzioni. Le stringhe di lunghezza ≥ 3 in questo linguaggio sono "0101...", tutte di lunghezza pari ≥ 4 (poiché ogni blocco "01" aggiunge 2 alla lunghezza). Quindi tutte le stringhe di lunghezza ≥ 3 sono effettivamente di lunghezza ≥ 4 e sono pompabili.

Quindi, la minima pumping length è $p = 3$.

e) Σ^*

Questo linguaggio contiene tutte le stringhe possibili sull'alfabeto.

Qualsiasi stringa in questo linguaggio, indipendentemente dalla sua lunghezza, può essere pompata e il risultato rimane in Σ^*. Anche le stringhe di lunghezza 1 sono pompabili.

Quindi, la minima pumping length è $p = 1$.

3 Conclusioni

In sintesi, le minimum pumping length sono:

- $000\,1^*$: $p = 4$
- $0^*\,1^*$: $p = 1$
- $\{\texttt{1011}\}$: $p = 5$
- $(01)^*$: $p = 3$
- Σ^*: $p = 1$

Questi risultati evidenziano come la struttura di un linguaggio regolare influenzi la sua minimum pumping length. Linguaggi con vincoli più rigidi sulla struttura tendono ad avere pumping length maggiori, mentre linguaggi più "flessibili" possono avere pumping length minori.