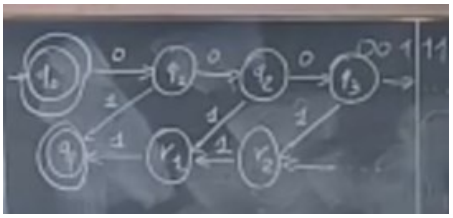
Consideriamo il linguaggio $L = \{0^n 1^n \mid n \geq 0\}$. Il linguaggio non è regolare.

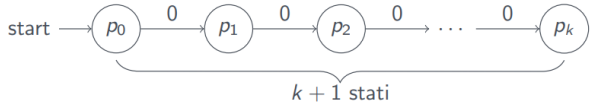


Let's take the language $B = \{0^n 1^n \mid n \geq 0\}$. If we attempt to find a DFA that recognizes B, we discover that the machine seems to need to remember how many 0s have been seen so far as it reads the input. Because the number of 0s isn't limited, the machine will have to keep track of an unlimited number of possibilities. But it cannot do so with any finite number of states.

Costruire un automa e vedere che il suo numero di stati non è finito non è sufficiente.

'Si usa normalmente la dimostrazione per assurdo.

- Supponiamo che $L_{01} = \{0^n 1^n : n \geq 0\}$ sia regolare
- Allora deve essere accettato da un DFA A con un certo numero k di stati
- Cosa succede quando A legge 0^k?
- Seguirà una qualche sequenza di transizioni:



- Siccome ci sono $k+1$ stati nella sequenza, esiste uno stato che si ripete: esistono $i < j$ tali che $p_i = p_j$
- Chiamiamo q questo stato

- Cosa succede quando l'automa A legge 1^i partendo da q?
- Se l'automa finisce la lettura in uno stato finale:
  - allora accetta, sbagliando, la parola $0^j 1^i$
- Se l'automa finisce la lettura in uno stato non finale:
  - allora rifiuta, sbagliando, la parola $0^i 1^i$
- In entrambi i casi abbiamo ingannato l'automa, quindi L_{01} non può essere regolare

Theorem (Pumping Lemma per Linguaggi Regolari)

Sia L un linguaggio regolare. Allora

- *esiste una lunghezza $k > 0$ tale che*
- *ogni parola $w \in L$ di lunghezza $|w| \geq k$*
- *può essere spezzata in $w = xyz$ tale che:*
  1. $y \neq \varepsilon$ *(il secondo pezzo è non vuoto)*
  2. $|xy| \leq k$ *(i primi due pezzi sono lunghi al max k)*
  3. $\forall i \geq 0$, $xy^i z \in L$ *(possiamo "pompare" y rimanendo in L)*

Dimostrazione:

- Supponiamo che L sia un linguaggio regolare
- Allora è riconosciuto da un DFA con, supponiamo, k stati
- Consideriamo una parola $w = a_1 a_2 \dots a_n \in L$ di lunghezza $n \geq k$
- Consideriamo gli stati nella computazione di A per w:

$$p_0 p_1 p_2 \dots p_k \dots p_n$$

- Siccome in $p_0, p_1, \dots, p_k$ ci sono $k+1$ stati, ne esiste uno che si ripete:

esistono $l < m$ tali che $p_l = p_m$ e $m \leq k$

- Possiamo spezzare w in tre parti $w = xyz$:
  1. $x = a_1 a_2 \dots a_l$
  2. $y = a_{l+1} a_{l+2} \dots a_m$
  3. $z = a_{m+1} a_{m+2} \dots a_n$
- che rispettano le condizioni del Lemma:
  - $y \neq \varepsilon$ perché $l < m$
  - $|xy| \leq k$ perché $m \leq k$

The Pumping Lemma as a 2-Person Game

1. You pick the language L to be proved nonregular.
2. Your adversary picks n, but does not reveal to you what n is. You must devise a move for all possible n's.
3. You pick w, which may depend on n. $|w| \geq n$.
4. Your adversary picks a factoring of $w = xyz$. Your adversary does not reveal what the factors are, only that they satisfy the constraints of the theorem: $|y| > 0$ and $|xy| \leq n$.
5. You "win" by picking k, which may be a function of n, x, y, and z, such that $xy^k z \notin L$.

- L'avversario sceglie la lunghezza k
- Noi scegliamo una parola w
- L'avversario spezza w in xyz
- Noi scegliamo i tale che $xy^i z \notin L$
- allora abbiamo vinto

- Ogni linguaggio regolare soddisfa il Pumping Lemma.
- Un linguaggio che falsifica il Pumping Lemma non può essere regolare:
  - per ogni lunghezza $k \geq 0$
  - esiste una parola $w \in L$ di lunghezza $|w| \geq k$ tale che
  - per ogni suddivisione $w = xyz$ tale che:
    1. $y \neq \varepsilon$ (il secondo pezzo è non vuoto)
    2. $|xy| \leq k$ (i primi due pezzi sono lunghi al max k)
  - esiste un $i \geq 0$ tale che $xy^i z \notin L$ (possiamo "pompare" y ed uscire da L)
- Attenzione: esistono linguaggi non regolari che rispettano il Pumping Lemma!

2. (a) Dimostrare che il linguaggio $L_1 = \{0^{2n}1^n : n \geq 0\}$ *non* è regolare.

Il linguaggio non è regolare. Supponiamo per assurdo che lo sia:

- sia h la lunghezza data dal Pumping Lemma;
- consideriamo la parola $w = 0^{2h}1^h$, che appartiene ad L_1 ed è di lunghezza maggiore di h;
- sia $w = xyz$ una suddivisione di w tale che $y \neq \varepsilon$ e $|xy| \leq h$;
- poiché $|xy| \leq h$, allora xy è completamente contenuta nel prefisso 0^{2h} di w, e quindi sia x che y sono composte solo da 0. Inoltre, siccome $y \neq \varepsilon$, possiamo dire che $y = 0^p$ per qualche valore $p > 0$. Allora la parola xy^2z è nella forma $0^{2h+p}1^h$, e quindi non appartiene al linguaggio perché il numero di 0 non è uguale al doppio del numero di 1 (dovrebbero essere $h + p/2$ mentre sono solo h).

Abbiamo trovato un assurdo quindi L_1 non può essere regolare.

2. Considerate il linguaggio $L = \{0^{2n}1^m0^n : n, m \geq 0\}$. Questo linguaggio è regolare? Dimostrare formalmente la risposta.

Il linguaggio non è regolare. Supponiamo per assurdo che lo sia:

- *sia h la lunghezza data dal Pumping Lemma;*
- *consideriamo la parola $w = 0^{2h}10^h$, che appartiene ad L ed è di lunghezza maggiore di h;*
- *sia $w = xyz$ una suddivisione di w tale che $y \neq \varepsilon$ e $|xy| \leq h$;*
- *poiché $|xy| \leq h$, allora xy è completamente contenuta nel prefisso 0^{2h} di w, e quindi sia x che y sono composte solo da 0. Inoltre, siccome $y \neq \varepsilon$, possiamo dire che $y = 0^p$ per qualche valore $p > 0$. Allora la parola xy^2z è nella forma $0^{2h+p}10^h$, e quindi non appartiene al linguaggio perché il numero di 0 nella prima parte della parola non è uguale al doppio del numero di 0 nella seconda parte della parola.*

Abbiamo trovato un assurdo quindi L_1 non può essere regolare.

2. *Considera il linguaggio*

$$L_2 = \{w \in \{0,1\}^* \mid w \text{ contiene lo stesso numero di } 00 \text{ e di } 11\}.$$

Dimostra che L_2 non è regolare.

Usiamo il Pumping Lemma per dimostrare che il linguaggio non è regolare.

Supponiamo per assurdo che L_2 sia regolare:

- sia k la lunghezza data dal Pumping Lemma;
- consideriamo la parola $w = 0^k1^k$, che appartiene ad L_2 ed è di lunghezza maggiore di k;
- sia $w = xyz$ una suddivisione di w tale che $y \neq \varepsilon$ e $|xy| \leq k$;
- poiché $|xy| \leq k$, allora x e y sono entrambe contenute nella sequenza di 0. Inoltre, siccome $y \neq \emptyset$, abbiamo che $x = 0^q$ e $y = 0^p$ per qualche $q \geq 0$ e $p > 0$. z contiene la parte rimanente della stringa: $z = 0^{k-q-p}1^k$. Consideriamo l'esponente $i = 0$: la parola xy^0z ha la forma

  $$xy^0z = xz = 0^q0^{k-q-p}1^k = 0^{k-p}1^k$$

  e contiene un numero di occorrenze di 00 minore delle occorrenze di 11. Di conseguenza, la parola non appartiene al linguaggio L_2, in contraddizione con l'enunciato del Pumping Lemma.

1. (8 punti) *Considera il linguaggio*

$$L = \{0^m1^n \mid m/n \text{ è un numero intero}\}.$$

Dimostra che L non è regolare.

Usiamo il Pumping Lemma per dimostrare che il linguaggio non è regolare.

Supponiamo per assurdo che L sia regolare:

- sia k la lunghezza data dal Pumping Lemma;
- consideriamo la parola $w = 0^{k+1}1^{k+1}$, che è di lunghezza maggiore di k ed appartiene ad L perché $(k+1)/(k+1) = 1$;
- sia $w = xyz$ una suddivisione di w tale che $y \neq \varepsilon$ e $|xy| \leq k$;
- poiché $|xy| \leq k$, allora x e y sono entrambe contenute nella sequenza di 0. Inoltre, siccome $y \neq \varepsilon$, abbiamo che $x = 0^q$ e $y = 0^p$ per qualche $q \geq 0$ e $p > 0$. z contiene la parte rimanente della stringa: $z = 0^{k+1-q-p}1^{k+1}$. Consideriamo l'esponente $i = 0$: la parola xy^0z ha la forma

  $$xy^0z = xz = 0^q0^{k+1-q-p}1^{k+1} = 0^{k+1-p}1^{k+1}.$$

  Si può notare che $(k+1-p)/(k+1)$ è un numero strettamente compreso tra 0 e 1, e quindi non può essere un numero intero. Di conseguenza, la parola non appartiene al linguaggio L, in contraddizione con l'enunciato del Pumping Lemma.

La chiusura per le operazioni regolari rende possibile anche capire se un certo linguaggio è regolare.

5. For languages A and B, let the *perfect shuffle* of A and B be the language

$$\{w \mid w = a_1 b_1 \cdots a_k b_k, \text{ where } a_1 \cdots a_k \in A \text{ and } b_1 \cdots b_k \in B, \text{ each } a_i, b_i \in \Sigma\}.$$

Show that the class of regular languages is closed under perfect shuffle.

Answer: Let $D_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $D_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ be two DFAs that recognize A and B, respectively. Here, we shall construct a DFA $D = (Q, \Sigma, \delta, q, F)$ that recognizes the perfect shuffle of A and B.

The key idea is to design D to alternately switch from running D_A and running D_B after each character is read. Therefore, at any time, D needs to keep track of (i) the current states of D_A and D_B and (ii) whether the next character of the input string should be matched in D_A or in D_B. Then, when a character is read, depending on which DFA should match the character, D makes a move in the corresponding DFA accordingly. After the whole string is processed, if both DFAs are in the accept states, the input string is accepted; otherwise, the input string is rejected.

Formally, the DFA D can be defined as follows:

(a) $Q = Q_A \times Q_B \times \{A, B\}$, which keeps track of all possible current states of D_A and D_B, and which DFA to match.

(b) $q = (q_A, q_B, A)$, which states that D starts with D_A in q_A, D_B in q_B, and the next character read should be in D_A.

(c) $F = F_A \times F_B \times \{A\}$, which states that D accepts the string if both D_A and D_B are in accept states, and the next character read should be in D_A (i.e., last character was read in D_B).

(d) δ is as follows:

i. $\delta((x, y, A), a) = (\delta_A(x, a), y, B)$, which states that if current state of D_A is x, the current state of D_B is y, and the next character read is in D_A, then when a is read as the next character, we should change the current state of A to $\delta_A(x, a)$, while the current state of B is not changed, and the next character read will be in D_B.

ii. Similarly, $\delta((x, y, B), b) = (x, \delta_B(y, b), A)$.

Riferimento da dire in aula: es. 2 preparazione esame sul Moodle)

Linguaggio regolare significa che esiste un DFA che lo riconosce. Tutte le parole di L sono di lunghezza > 1 (altrimenti non si potrebbe togliere una lettera). Togliendo una lettera ad una parola composta da una sola lettera otteniamo un set vuoto, che per definizione è un linguaggio regolare. Nel caso l'automa sia composto da due o più simboli dell'alfabeto dobbiamo semplicemente costruire un automa che sostituisce la transizione che accettava y e conuna epsilon transizione allo stato successivo

4. Sia $A/b = \{w \mid wb \in A\}$. Mostrare che se A è un linguaggio regolare e $b \in \Sigma$, allora A/b è regolare.

Per dimostrare che A/B è regolare, possiamo costruire un automa a stati finiti, in particolare un ε-NFA che, partendo dall'automa a stati finiti A=(Q U $\{q_0\}$', Σ, q_0, δ, F) che ha lo stesso insieme di stati, stesso stato iniziale e gli stessi stati finali. Essendo che si deve sempre essere la produzione di "w", lo stato iniziale conterrà, con l'aggiunta di una ε-transizione, un automa con tutti gli stati finali raggiungibili dal vecchio stato iniziale, con l'aggiunta della nuova stringa. Gli stati finali sono inoltre gli stessi.

Il linguaggio A/b riconosce tutte le stringhe che hanno la proprietà che se concateno b ad una qualsiasi stringa ottengo una stringa che appartiene al linguaggio regolare A.

Dato che i linguaggi regolari sono chiusi rispetto alla concatenazione, per ottenere una stringa appartenente ad A concatenando b a w significa che anche w deve appartenere ad un linguaggio regolare quindi A/b è un linguaggio regolare.

Si dimostra che i linguaggi regolari sono chiusi per concatenazione collegando lo stato finale di un DFA che riconosce un linguaggio regolare con un epsilon transizione allo stato iniziale di un DFA che riconosce un altro linguaggio regolare in modo da concatenare le due stringhe riconosciute da ciascun DFA.

1.55 The pumping lemma says that every regular language has a pumping length p, such that every string in the language can be pumped if it has length p or more. If p is a pumping length for language A, so is any length $p' \geq p$. The ***minimum pumping length*** for A is the smallest p that is a pumping length for A. For example, if $A = 01^*$, the minimum pumping length is 2. The reason is that the string $s = 0$ is in A and has length 1 yet s cannot be pumped, but any string in A of length 2 or more contains a 1 and hence can be pumped by dividing it so that $x = 0$, $y = 1$, and z is the rest. For each of the following languages, give the minimum pumping length and justify your answer.

[A]a. 0001^*
[A]b. 0^*1^*
c. $001 \cup 0^*1^*$
[A]d. $0^*1^+0^+1^* \cup 10^*1$
e. $(01)^*$
f. ε
g. $1^*01^*01^*$
h. $10(11^*0)^*0$
i. 1011
j. Σ^*

b.

String ε is in the language but it could not be pumped so the pumping length could not be 0. According to pumping lemma's three condition arises which are as follows:

- If user divide the string in xyz as x is ε
- y is first symbol $(0|1)$
- z being the everything after then it holds.

Hence the minimum pumping length is 1.

The string 001 is in the language but if it is generated by **001** then it cannot be pumped. If string s is larger than 3 and in the language, then it is generated by 0^*1^*. Dividing the string according to Pumping Lemma's condition into a string xyz where x is ε, y be the first symbol and z be the remaining, user can pump the string.

Hence the minimum pumping length is 4.

a.

The pumping length could not be 3 as 000 being in the language it cannot be pumped. Consider the string length be 4 or more and divide in xyz as x being 000, y being the first 1 and z being everything after then it satisfies every condition of pumping lemma.

Hence the minimum pumping length is 4.

d.

The string 11 is in the language but it cannot be pumped so the length is not 2. Let s be the string in the language of length at least 3. If s is generated by $0^*1^+0^+1^*$ it can be divided in xyz as x is ε, y is the first string and z is everything else so it can be pumped.

Again if s is generated by 10^*1 then also user can write it as xyz where x is 1, y is 0, and z is the remainder so it could be pumped.

Hence the minimum pumping length is 3.

e.

Let s be a string in the language.

Now s could be ε but it cannot be pumped so the length is not 0. Next s could be 01 which if divide in xyz as x is empty string ε, y is 01, and z is everything after then it satisfies the three conditions of pumping lemma.

Pumping length is not 1 because since there is no string of length 1 in the language.

Hence the minimum pumping length is 2.

f.

Let s be a string in the language then s is ε and according to pumping lemma it cannot be pumped. As per the pumping lemma, pumping length should greater then equal to 1.

Hence the minimum pumping length is 0.

Step 7 of 10

g.

The minimum pumping length of the language could not be 2 as 00 being in the language it could not be pumped. Let s be the string of length at least 3 in the language so minimally s could be 100 or 010 or 001.

Now dividing s in xyz in all the three cases user get, for 100 x is ε, y is 1, and z is the remainder, for 010 x is 0, y is 1, and z is the remainder and for 001 x is 00, y is 1, and z is the remainder. All satisfies Pumping Lemma's three conditions.

Hence the minimum pumping length is 3.

h.

The minimum length string in the given language is 100 but it cannot be pumped. Next minimum length string is 10100. If divide it according to the pumping lemma in xyz then x be 10, y be 10, and z be the rest of the string then y can be pumped.

Hence the minimum pumping length is 4.

j.

Say s be a string in the language Σ^*. According to pumping lemma if divide s in xyz then x be the empty string, y is $(\varepsilon|0|1)$ and z is empty string. Now ε could not be pumped.

Hence the minimum pumping length is 1.