Pumping Lemma

- Dimostra che il linguaggio $L = \{0^n 1^m | n < m\}$ non è regolare.
- Dimostra che il linguaggio $L = \{0^{n^2} \mid n \geq 0\}$ non è regolare.
- Dimostra che il linguaggio $L = \{0^n 1^n 2^n \mid n \geq 0\}$ non è regolare.
- Dimostra che il linguaggio $L = \{w \in \{a,b\}* \mid w$ non contiene la sottostringa $abb\}$ non è regolare.
- Dimostra che il linguaggio $L = \{0^i 1^j 0^k \mid i,j,k \geq 0 \; e \; i = j \; o \; j = k\}$ non è regolare.
- Dimostra che il linguaggio $L = \{w \in \{a,b\}^* | \#a(w) = 2\#b(w)$ non è regolare

Linguaggi regolari

- Dimostra che il linguaggio $L = \{w \in \{a,b\}* \mid w$ contiene almeno tre "a" consecutivi$\}$ è regolare.
- Dimostra che il linguaggio $L = \{a^n b^m \mid n,m \geq 0 \; e \; n + m \; è \; pari\}$ è regolare.
- Dimostra che il linguaggio $L = \{w \in \{a,b\}* \mid w$ ha lo stesso numero di "a" e "b"$\}$ è regolare.
- Dimostrare che il linguaggio $L = \{w \in \{a,b\}* \mid w$ non contiene la sottostringa "aba"$\}$ è regolare.
- Dimostra che il linguaggio $L = \{w \in \{a,b\}* \mid w$ è una stringa palindroma che inizia e finisce con "a"$\}$ è regolare.
- Dimostra che il linguaggio $L = \{w \in \{a,b\}* \mid w$ ha un numero primo di "a"$\}$ è regolare.

Pumping Lemma per linguaggi context-free

- Dimostra che il linguaggio $L = \{a^n b^m c^k \mid n > m > k\}$ non è context-free.
- Dimostra che il linguaggio $L = \{ww^R w \mid w \in \{a,b\}*\}$ non soddisfa il pumping lemma per i linguaggi context-free.
- Dimostra che il linguaggio $L = \{a^n b^m c^k \mid n \neq m \neq k\}$ non è context-free.
- Dimostrare che il linguaggio $L = \{0^{n^2} \mid n \geq 0\}$ non è context-free.

Pumping Lemma

Di seguito alcuni esercizi svolti:

- Dimostra che il linguaggio $L = \{0^n 1^m | n < m\}$ non è regolare.

Soluzione:

Supponiamo per assurdo che L sia un linguaggio regolare. Quindi, per il pumping lemma, esiste una costante p (la lunghezza di pompare) tale che per ogni stringa w ∈ L di lunghezza almeno p, w può essere scritta come xyz, dove:

- $|y| > 0$
- $|xy| \leq p$
- per ogni $i \geq 0$, la stringa $xy^i z$ appartiene a L.

Prendiamo la stringa $w = 0^{\frac{p}{2}} 1^{\frac{3p}{2}} \in L$. Poiché $|w| = p \geq p$, per il pumping lemma possiamo scrivere w come xyz.

Poiché |xy| ≤ p, la stringa xy può contenere solo zeri, ovvero $y = 0^k$ per qualche $k > 0$. Quindi, la stringa xz deve contenere lo stesso numero di zeri e uni, ovvero $xz = 0^{\frac{p}{2}-} 1^{\frac{3p}{2}}$.

Ora consideriamo xy^2z. Poiché y è composto solo da zeri, xy^2z avrà più zeri che uni, ovvero $xy^2z = 0^{\frac{p}{2}+k} 1^{\frac{3p}{2}}$. Tuttavia, questa stringa non appartiene a L perché $n < m$, ovvero il numero di zeri è minore del numero di uni.

Quindi, abbiamo trovato una stringa w ∈ L che non può essere pompata, il che contraddice il pumping lemma. Concludiamo che L non può essere regolare.

- Dimostra che il linguaggio $L = \{0^{n^2} \mid n \geq 0\}$ non è regolare.

Soluzione:

Supponiamo per assurdo che L sia un linguaggio regolare. Quindi, per il pumping lemma, esiste una costante p (la lunghezza di pompare) tale che per ogni stringa w ∈ L di lunghezza almeno p, w può essere scritta come xyz, dove:

- $|y| > 0$
- $|xy| \leq p$
- per ogni $i \geq 0$, la stringa $xy^i z$ appartiene a L.

Prendiamo una stringa w ∈ L tale che |w| ≥ p. Sappiamo che w ha la forma 0^{n^2} per qualche n ≥ 0. Possiamo scrivere w come xyz, dove:

$x = \varepsilon$

$y = 0^k$ per qualche $k > 0$

$z = 0^{n^2 - k}$

Poiché |xy| ≤ p, allora y deve essere composto interamente da 0, ovvero y = 0^k per qualche k ≤ p. Inoltre, per il punto 3 del pumping lemma, xy^iz deve ancora essere una stringa di L per ogni i ≥ 0.

Consideriamo ora i casi in cui i = 2 e i = 3. In entrambi i casi, otteniamo la seguente stringa:

$$xy^2z = \varepsilon\, 0^k\, \varepsilon\, 0^k\, 0^{n^2 - k} = 0^{n^2 + k}$$

oppure

$$xy^3z = \varepsilon\, 0^k\, \varepsilon\, 0^k\, \varepsilon\, 0^k\, 0\text{^}(n^2 - k) = 0^{n^2 + 2k}$$

Tuttavia, nessuna di queste stringhe appartiene a L, perché la lunghezza della stringa deve essere un quadrato perfetto, e n^2 + k o n^2 + 2k non sono quadrati perfetti per k > 0.

Quindi, abbiamo trovato una stringa w ∈ L che non può essere pompata, il che contraddice il pumping lemma. Concludiamo che L non può essere regolare.

- Dimostra che il linguaggio $L = \{w \in \{a, b\}^* | \#a(w) = 2\#b(w)$ non è regolare
    - Tradotto "il numero di occorrenze di a è 2 volte il numero di occorrenze di b"

Soluzione:

Per dimostrare che il linguaggio L non è regolare, useremo il pumping lemma. Supponiamo per assurdo che L sia regolare e sia data una costante p dal pumping lemma.

Scegliamo la stringa $w = a^p\, b^{2p} \in L$. Poiché $|w| = 3p \geq p$, per il pumping lemma possiamo scrivere w come xyz, dove:

- $|y| > 0$
- $|xy| \leq p$
- per ogni $i \geq 0$, la stringa xy^iz appartiene a L.

Poiché $|xy| \leq p$, la sottostringa xy deve essere contenuta solo nella prima parte di w, ovvero nella sequenza a^p . Inoltre, poiché $|y| > 0, y$ deve contenere almeno un carattere a. Consideriamo ora il caso in cui $i = 2$. Allora:

$$xy^2z = a^{p+|y|}b^{2p}$$

Poiché y contiene almeno un carattere a, la stringa xy^2z contiene almeno un carattere a in più rispetto a b, il che significa che la stringa non appartiene a L. Questo contraddice il pumping lemma, dimostrando che L non è regolare.

Quindi, il linguaggio $L = \{w \in \{a, b\} * | \#a(w) = 2\#b(w)\}$ non è regolare.

Linguaggi regolari

- Dimostra che il linguaggio $L = \{w \in \{a,b\}* \mid w$ contiene almeno tre "a" consecutivi$\}$ è regolare.

Soluzione:

Per dimostrare che il linguaggio L è regolare, costruiamo un automa finito deterministico (DFA) che accetta tutte le stringhe in L.

L'automa per L ha tre stati: $q0, q1\ e\ q2$. Lo stato $q0$ è lo stato iniziale e $q2$ è lo stato finale.

La transizione da $q0$ a $q1$ avviene quando l'input è 'a', altrimenti rimaniamo in q0. La transizione da q1 a q2 avviene quando l'input è 'a', altrimenti torniamo a $q0$. In $q2$, qualsiasi input porta a rimanere in q2.

Formalmente, l'automa può essere descritto come segue:

- $Q = \{q0, q1, q2\}$;
- $\Sigma = \{a, b\}$;
- $\delta(q0, a) = q1, \delta(q0, b) = q0, \delta(q1, a) = q2, \delta(q1, b) = q0, \delta(q2, a) = q2, \delta(q2, b) = q2$;
- $q0$ è lo stato iniziale e $\{q2\}$ è l'insieme degli stati finali.

Questo DFA accetta ogni stringa che contiene almeno tre "a" consecutivi, in quanto in tal caso l'automa si trova nello stato finale q2. Inoltre, essendo un DFA, accetta solo le stringhe che appartengono a L.

Pertanto, abbiamo dimostrato che il linguaggio L = {w ∈ {a, b}* | w contiene almeno tre "a" consecutivi} è regolare.

- Dimostra che il linguaggio $L = \{a^n b^m \mid n, m \geq 0\ e\ n + m\ è\ pari\}$ è regolare.

Per dimostrare che il linguaggio L è regolare, costruiamo un automa finito deterministico (DFA) che accetta tutte le stringhe in L.

L'automa per L ha quattro stati: q0, q1, q2 e q3. Lo stato q0 è lo stato iniziale e q0, q1, q2 e q3 sono gli stati finali.

La transizione da q0 a q1 avviene quando l'input è 'a', altrimenti rimaniamo in q0. La transizione da q0 a q2 avviene quando l'input è 'b', altrimenti rimaniamo in q0. In q1, qualsiasi input porta a rimanere in q3, mentre in q3 qualsiasi input porta a rimanere in q1. In q2, qualsiasi input porta a rimanere in q2.

Formalmente, l'automa può essere descritto come segue:

- $Q = \{q0, q1, q2, q3\}$;
- $\Sigma = \{a, b\}$;
- $\delta(q0, a) = q1, \delta(q0, b) = q2, \delta(q1, a) = q3, \delta(q1, b) = q0, \delta(q2, a) = q0, \delta(q2, b) = q3, \delta(q3, a) = q1, \delta(q3, b) = q2$;
- q0 è lo stato iniziale e {q0, q1, q2, q3} è l'insieme degli stati finali.

Questo DFA accetta ogni stringa della forma $anbm$, dove n e m sono numeri non negativi e n + m è pari. Inoltre, essendo un DFA, accetta solo le stringhe che appartengono a L.

Pertanto, abbiamo dimostrato che il linguaggio L = $\{a^n b^m$ | n, m ≥ 0 e n + m è pari$\}$ è regolare.

- Dimostra che il linguaggio $L = \{w \in \{a,b\}* \mid w$ ha lo stesso numero di "a" e "b"$\}$ è regolare.

Per dimostrare che il linguaggio L è regolare, costruiamo un automa finito deterministico (DFA) che accetta tutte le stringhe in L.

L'automa per L ha tre stati: q0, q1 e q2. Lo stato q0 è lo stato iniziale e q2 è lo stato finale.

La transizione da q0 a q1 avviene quando l'input è 'a', altrimenti avviene la transizione da q0 a q2. La transizione da q1 a q0 avviene quando l'input è 'b', altrimenti avviene la transizione da q1 a q2. La transizione da q2 a q2 avviene con qualsiasi input.

Formalmente, l'automa può essere descritto come segue:

- $Q = \{q0, q1, q2\}$;
- $\Sigma = \{a, b\}$;
- $\delta(q0, a) = q1, \delta(q0, b) = q2, \delta(q1, a) = q2, \delta(q1, b) = q0, \delta(q2, a) = q2, \delta(q2, b) = q2$;
- q0 è lo stato iniziale e {q2} è l'insieme degli stati finali.

Questo DFA accetta solo le stringhe che hanno lo stesso numero di "a" e "b", in quanto l'automa si trova nello stato finale q2 solo se il numero di "a" e "b" è lo stesso. Inoltre, essendo un DFA, accetta solo le stringhe che appartengono a L.

Pertanto, abbiamo dimostrato che il linguaggio L = {w ∈ {a, b}* | w ha lo stesso numero di "a" e "b"} è regolare.

- Dimostrare che il linguaggio $L = \{w \in \{a,b\}* \mid w$ non contiene la sottostringa "aba"$\}$ è regolare.

Soluzione:

L'automa per L ha quattro stati: q0, q1, q2 e q3. Lo stato q0 è lo stato iniziale e q3 è lo stato finale.

La transizione da q0 a q1 avviene quando l'input è 'a', altrimenti rimaniamo in q0. La transizione da q0 a q3 avviene quando l'input è 'b', altrimenti rimaniamo in q0. La transizione da q1 a q2 avviene quando l'input è 'b', altrimenti torniamo in q0. La transizione da q2 a q3 avviene quando l'input è 'a', altrimenti torniamo in q0. In q3, qualsiasi input porta a rimanere in q3.

Formalmente, l'automa può essere descritto come segue:

- $Q = \{q0, q1, q2, q3\}$;
- $\Sigma = \{a, b\}$;
- $\delta(q0, a) = q1, \delta(q0, b) = q3, \delta(q1, a) = q0, \delta(q1, b) = q2, \delta(q2, a) = q3, \delta(q2, b) = q0, \delta(q3, a) = q3, \delta(q3, b) = q3$;
- q0 è lo stato iniziale e {q3} è l'insieme degli stati finali.

Questo DFA accetta solo le stringhe che non contengono la sottostringa "aba", in quanto l'automa ritorna allo stato q0 ogni volta che incontra la sottostringa "ab" e allora potrebbe iniziare a cercare una "a" successiva, ma se incontra una "b" passa allo stato finale q3. Inoltre, essendo un DFA, accetta solo le stringhe che appartengono a L.

Pertanto, abbiamo dimostrato che il linguaggio L = {w ∈ {a, b}* | w non contiene la sottostringa "aba"} è regolare attraverso la costruzione di un automa finito deterministico.

Pumping Lemma per linguaggi context-free

- Dimostra che il linguaggio $L = \{ww^R w \mid w \in \{a,b\}*\}$ non soddisfa il pumping lemma per i linguaggi context-free.

Supponiamo che L sia un linguaggio context-free. Allora, per il pumping lemma, esiste un intero positivo p tale che per ogni stringa w di L con |w| ≥ p, esiste una suddivisione $w = uvxyz$ tale che:

- |vxy| ≤ p;
- |vy| > 0;
- per ogni k ≥ 0, la stringa $uv^k xy^k z$ appartiene a L.

Consideriamo la stringa $w = a^p b^p a^p$, dove p è un intero positivo. Poiché |w| = 3p ≥ p, esiste una suddivisione w = uvxyz tale che |vxy| ≤ p e |vy| > 0. Sia v = a^i, y = a^j, con i + j > 0 e i + j ≤ p. Allora, $uv^2xy^2z = a^{p+i+j}b^p a^p$ non appartiene a L, in quanto non è una stringa della forma ww^Rw. Quindi, la terza condizione del pumping lemma non è soddisfatta, il che contraddice l'ipotesi che L sia context-free.

Pertanto, L non è context-free.

- Dimostra che il linguaggio $L = \{a^n b^m c^k \mid n \neq m \neq k\}$ non è context-free.

Supponiamo, per assurdo, che L sia un linguaggio context-free. Allora, esiste un intero positivo p tale che per ogni stringa w di L con |w| ≥ p, esiste una suddivisione w = uvxyz tale che:

- $|vxy| \leq p$;
- $|vy| > 0$;
- per ogni k ≥ 0, la stringa $uv^k xy^k z$ appartiene a L.

Consideriamo la stringa $w = a^p b^p c^k p$, dove p è un intero positivo tale che $p > 2p + 1$. Quindi, abbiamo che |w| = 3p > p.

Per il punto 2 del pumping lemma, la suddivisione w = uvxyz deve avere una sottostringa vy che contiene solo "a". Poiché |vy| > 0 e |vxy| ≤ p, allora v e y possono contenere al massimo p "a". Quindi, consideriamo la suddivisione $w = u'v'x'y'z'$, dove u' è la sottostringa di u che precede v, x' è la sottostringa di x che segue v e precede y, e z' è la sottostringa di z che segue y.

Inoltre, poiché |vxy| ≤ p, allora la sottostringa vxy deve essere contenuta completamente nella sequenza di "a" all'inizio della stringa w, ovvero nella sottostringa a^p. Quindi, possiamo scrivere v = a^i, x = a^j e y = a^k, dove i + j + k > 0 e i + k ≤ p.

Per il punto 3 del pumping lemma, la stringa uv^kxy^kz deve appartenere a L per ogni k ≥ 0. Scegliamo k = 2. Quindi, $uv^2xy^2z = a^{p+i+j+k}b^{p+j}c^{k-1}$ non appartiene a L, poiché il numero di "a" è diverso dal numero di "b" e dal numero di "c". Questo contraddice l'ipotesi che L sia un linguaggio context-free.

Pertanto, L non è context-free.

- Dimostrare che il linguaggio $L = \{0^{n^2} \mid n \geq 0\}$ non è context-free.

Supponiamo, per assurdo, che L sia un linguaggio context-free. Allora, esiste un intero positivo p tale che per ogni stringa w di L con |w| ≥ p, esiste una suddivisione w = uvxyz tale che:

- $|vxy| \leq p$;
- $|vy| > 0$;
- per ogni k ≥ 0, la stringa $uv^k xy^k z$ appartiene a L.

Consideriamo la stringa w = 0^(p^2), che appartiene a L. Per il pumping lemma esiste una scomposizione w = uvxyz tale che |vxy| <= p e |vy| > 0. Ci sono tre casi da considerare:

- La sottostringa vy contiene solo 0: in tal caso, la stringa uv^2xy^2z non può appartenere a L, poiché il numero di zeri nella stringa non è un quadrato perfetto.
- La sottostringa vy contiene solo zeri e almeno un 1: in tal caso, la stringa uv^2xy^2z non può appartenere a L, poiché il numero di zeri nella stringa non è un quadrato perfetto.
- La sottostringa vy contiene sia 0 che almeno un 1: in tal caso, la stringa uv^2xy^2z non può appartenere a L, poiché il numero di zeri nella stringa non è un quadrato perfetto.

In tutti e tre i casi, abbiamo ottenuto una contraddizione con la proprietà 3 del pumping lemma. Pertanto, L non è un linguaggio context-free.

Concludiamo che L non può essere generato da una grammatica context-free e quindi non è context-free.

Pumping Lemma

- Dimostra che il linguaggio $L = \{0^n 1^n 2^n \mid n \geq 0\}$ non è regolare.

Soluzione:

Supponiamo per assurdo che L sia regolare e sia data una costante p dal pumping lemma. Scegliamo la stringa $w = 0^p 1^p 2^p \in L$. Poiché $|w| = 3p \geq p$, per il pumping lemma possiamo scrivere w come xyz, dove:

- $|y| > 0$
- $|xy| \leq p$
- per ogni $i \geq 0$, la stringa $xy^i z$ appartiene a L.

Poiché $|xy| \leq p$, la sottostringa xy può contenere solo caratteri 0 o 1. Inoltre, poiché |y| > 0, y deve contenere almeno un carattere 0 o 1. Consideriamo ora il caso in cui i = 2. Allora:

$$xy^2z = 0^{p+|y|}1^p 2^p$$

Poiché y contiene almeno un carattere 0 o 1, la stringa xy^2z contiene più 0 o 1 rispetto a 2. Quindi, la stringa xy^2z non appartiene a L, il che contraddice il pumping lemma. Concludiamo che L non è regolare.

- Dimostra che il linguaggio $L = \{w \in \{a,b\}* \mid w$ non contiene la sottostringa $abb\}$ non è regolare.

Soluzione:

Supponiamo per assurdo che L sia regolare e sia data una costante p dal pumping lemma. Scegliamo la stringa $w = a^p b^p (ab)^p \in L$. Poiché |w| = $p + 2p^2 \geq p$, per il pumping lemma possiamo scrivere w come xyz, dove:

- $|y| > 0$
- $|xy| \leq p$
- per ogni $i \geq 0$, la stringa $xy^i z$ appartiene a L.

Poiché |xy| ≤ p, la sottostringa xy deve essere contenuta solo nella prima parte di w, ovvero nella sequenza a^p. Inoltre, poiché |y| > 0, y deve contenere almeno un carattere a. Consideriamo ora il caso in cui i = 2. Allora:

$$xy^2z = a^{p+|y|}b^p(ab)^p$$

Poiché y contiene almeno un carattere a, la stringa xy^2z contiene la sottostringa abb, il che contraddice la definizione di L. Quindi, L non è regolare.

- Dimostra che il linguaggio $L = \{0^i 1^j 0^k \mid i,j,k \geq 0 \; e \; i = j \; o \; j = k\}$ non è regolare.

Soluzione:

Supponiamo per assurdo che L sia regolare e sia data una costante p dal pumping lemma. Scegliamo la stringa $w = 0^p 1^p 0^p \in L$. Poiché $|w| = 3p \geq p$, per il pumping lemma possiamo scrivere w come xyz, dove:

- $|y| > 0$
- $|xy| \leq p$
- per ogni $i \geq 0$, la stringa $xy^i z$ appartiene a L.

Poiché $|xy| \leq p$, la sottostringa xy deve essere contenuta solo nella prima parte di w, ovvero nella sequenza $0^p 1^p$. Inoltre, poiché $|y| > 0, y$ deve contenere almeno un carattere 0 o 1. Consideriamo ora il caso in cui i = 2. Allora:

$$xy^2z = 0^{p+|y|}1^{p+|y|}0^p$$

Poiché y contiene almeno un carattere 0 o 1, la stringa xy^2z contiene una sequenza di 0 o 1 che interrompe la sequenza $0^p 1^p$ o la sequenza $1^p 0^p$, il che contraddice la definizione di L. Quindi, L non è regolare.

- Dimostra che il linguaggio $L = \{w \in \{a,b\}^* | \#a(w) = 2\#b(w)$ non è regolare
    - Tradotto "il numero di occorrenze di a è 2 volte il numero di occorrenze di b"

Soluzione:

Per dimostrare che il linguaggio L non è regolare, useremo il pumping lemma. Supponiamo per assurdo che L sia regolare e sia data una costante p dal pumping lemma.

Scegliamo la stringa $w = a^p b^{2p} \in L$. Poiché $|w| = 3p \geq p$, per il pumping lemma possiamo scrivere w come xyz, dove:

- $|y| > 0$
- $|xy| \leq p$
- per ogni $i \geq 0$, la stringa xy^iz appartiene a L.

Poiché $|xy| \leq p$, la sottostringa xy deve essere contenuta solo nella prima parte di w, ovvero nella sequenza a^p . Inoltre, poiché $|y| > 0, y$ deve contenere almeno un carattere a. Consideriamo ora il caso in cui $i = 2$. Allora:

$$xy^2z = a^{p+|y|}b^{2p}$$

Poiché y contiene almeno un carattere a, la stringa xy^2z contiene almeno un carattere a in più rispetto a b, il che significa che la stringa non appartiene a L. Questo contraddice il pumping lemma, dimostrando che L non è regolare.

Quindi, il linguaggio $L = \{w \in \{a,b\}* \mid \#a(w) = 2\#b(w)\}$ non è regolare.

1. (8 punti) Considera il linguaggio

$$L = \{0^m 1^n \mid m = n^3\}.$$

Dimostra che L non è regolare.

Per dimostrare che L non è regolare, possiamo utilizzare il pumping lemma per i linguaggi regolari. Supponiamo per assurdo che L sia regolare e sia generato da un automa a stati finiti M con n stati.

Scegliamo una stringa $w = 0^{n^3} 1^n \in L$, che ha lunghezza almeno n. Poiché w appartiene a L, esiste una suddivisione w = xyz, dove:

1. $|xy| \leq n$

2. $|y| > 0$

3. $xy^iz \in L \; per \; ogni \; i \geq 0$

Consideriamo la suddivisione di w in modo che la sottostringa y contenga solo simboli 0, ovvero $y = 0^k$ per qualche k > 0. Allora, la parte xz di w contiene esattamente $n^3 + 1 - k$ simboli 0 e n simboli 1. Poiché |xy| ≤ n, la sottostringa xy deve contenere solo simboli 0, ovvero $xy = 0^j$ per qualche j ≤ k.

In particolare, se scegliamo i = 2, allora xy^2z contiene k + j simboli 0 e n simboli 1. Dato che k > 0, abbiamo $k + j < n^3 + 1$, e quindi $k + j < n^3$. Quindi, se L fosse regolare, xy^2z dovrebbe ancora appartenere a L, il che implica che k + j deve essere uguale a n^3. Ma questo è impossibile, poiché abbiamo dimostrato che k + j < n^3, e quindi abbiamo ottenuto una contraddizione.

Concludiamo quindi che L non può essere generato da un automa a stati finiti, e quindi non è un linguaggio regolare.

1. (8 punti) *Considera il linguaggio*

$$L = \{0^m 1^n \mid m/n \text{ è un numero intero}\}.$$

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

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

  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.

2. Considera il linguaggio

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

Dimostra che L_2 non è regolare.

Per dimostrare che L_2 non è regolare, possiamo utilizzare il pumping lemma per i linguaggi regolari. Supponiamo per assurdo che L_2 sia regolare e sia generato da un automa a stati finiti M con n stati.

Scegliamo la stringa $w = 0^n 1^n \in L_2$, dove n è il parametro del lemma. Poiché w appartiene a L_2, esiste una suddivisione w = xyz, dove:

1. $|xy| \leq n$
2. $|y| > 0$
3. $xy^iz \in L_2 per\ ogni\ i \geq 0$

Consideriamo la suddivisione di w in modo che la sottostringa y contenga solo simboli 0, ovvero $y = 0^k$ per qualche k > 0. Allora, la parte xz di w contiene esattamente $n - k$ simboli 0 e n simboli 1. Poiché |xy| ≤ n, la sottostringa xy deve contenere solo simboli 0, ovvero $xy = 0^j$ per qualche j ≤ k.

In particolare, se scegliamo i = 2, allora xy^2z contiene k + j simboli 0 e n simboli 1. Dato che k > 0, abbiamo k + j < n, e quindi la lunghezza di xy^2z è inferiore a 2n. Ma abbiamo visto che la lunghezza di una stringa in L_2 è sempre uguale a n, quindi xy^2z non può essere in L_2, e questo contraddice l'ipotesi che L_2 sia generato da un automa a stati finiti.

Concludiamo quindi che L_2 non può essere generato da un automa a stati finiti e quindi non è un linguaggio regolare.

2. Considerate il linguaggio $L = \{www \mid w \in \{a,b\}^*\}$. Questo linguaggio è regolare? Dimostrare formalmente la risposta.

Il linguaggio L non è regolare e possiamo dimostrarlo utilizzando il pumping lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia regolare e sia generato da un automa a stati finiti M con n stati. Scegliamo la stringa w = a^n b^n a^n, dove n è il parametro del lemma. Poiché w appartiene a L, esiste una suddivisione w = xyz, dove:

1. |xy| ≤ n

2. |y| > 0

3. xy^iz ∈ L per ogni i ≥ 0

Consideriamo la suddivisione di w in modo che la sottostringa y contenga solo simboli a, ovvero y = a^k per qualche k > 0. Allora, la parte xz di w contiene esattamente n - k simboli a, n simboli b e n simboli a. Poiché |xy| ≤ n, la sottostringa xy deve contenere solo simboli a, ovvero xy = a^j per qualche j ≤ k.

In particolare, se scegliamo i = 2, allora xy^2z contiene k + j simboli a, n simboli b e n simboli a. Dato che k > 0, abbiamo k + j < 2n, e quindi la lunghezza di xy^2z è inferiore a 2n + n = 3n. Ma abbiamo visto che la lunghezza di una stringa in L è sempre uguale a 3n, quindi xy^2z non può essere in L, e questo contraddice l'ipotesi che L sia generato da un automa a stati finiti.

Concludiamo quindi che L non può essere generato da un automa a stati finiti e quindi non è un linguaggio regolare.

<u>Linguaggi regolari</u>

- Dimostra che il linguaggio $L = \{w \in \{a,b\} * | w$ contiene almeno tre "a" consecutivi} è regolare.

Soluzione:

Per dimostrare che il linguaggio L è regolare, costruiamo un automa finito deterministico (DFA) che accetta tutte le stringhe in L.

L'automa per L ha tre stati: $q0, q1\ e\ q2$. Lo stato $q0$ è lo stato iniziale e $q2$ è lo stato finale.

La transizione da $q0$ a $q1$ avviene quando l'input è 'a', altrimenti rimaniamo in q0. La transizione da q1 a q2 avviene quando l'input è 'a', altrimenti torniamo a $q0$. In $q2$, qualsiasi input porta a rimanere in q2.

Formalmente, l'automa può essere descritto come segue:

- $Q = \{q0, q1, q2\}$;
- $\Sigma = \{a, b\}$;
- $\delta(q0, a) = q1, \delta(q0, b) = q0, \delta(q1, a) = q2, \delta(q1, b) = q0, \delta(q2, a) = q2, \delta(q2, b) = q2$;
- $q0$ è lo stato iniziale e $\{q2\}$ è l'insieme degli stati finali.

Questo DFA accetta ogni stringa che contiene almeno tre "a" consecutivi, in quanto in tal caso l'automa si trova nello stato finale q2. Inoltre, essendo un DFA, accetta solo le stringhe che appartengono a L.

Pertanto, abbiamo dimostrato che il linguaggio L = {w ∈ {a, b}* | w contiene almeno tre "a" consecutivi} è regolare.

- Dimostra che il linguaggio $L = \{a^n b^m \mid n, m \geq 0 \; e \; n + m \; è \; pari\}$ è regolare.

Per dimostrare che il linguaggio L è regolare, costruiamo un automa finito deterministico (DFA) che accetta tutte le stringhe in L.

L'automa per L ha quattro stati: q0, q1, q2 e q3. Lo stato q0 è lo stato iniziale e q0, q1, q2 e q3 sono gli stati finali.

La transizione da q0 a q1 avviene quando l'input è 'a', altrimenti rimaniamo in q0. La transizione da q0 a q2 avviene quando l'input è 'b', altrimenti rimaniamo in q0. In q1, qualsiasi input porta a rimanere in q3, mentre in q3 qualsiasi input porta a rimanere in q1. In q2, qualsiasi input porta a rimanere in q2.

Formalmente, l'automa può essere descritto come segue:

- $Q = \{q0, q1, q2, q3\}$;
- $\Sigma = \{a, b\}$;
- $\delta(q0, a) = q1, \delta(q0, b) = q2, \delta(q1, a) = q3, \delta(q1, b) = q0, \delta(q2, a) = q0, \delta(q2, b) = q3, \delta(q3, a) = q1, \delta(q3, b) = q2$;
- q0 è lo stato iniziale e {q0, q1, q2, q3} è l'insieme degli stati finali.

Questo DFA accetta ogni stringa della forma $anbm$, dove n e m sono numeri non negativi e n + m è pari. Inoltre, essendo un DFA, accetta solo le stringhe che appartengono a L.

Pertanto, abbiamo dimostrato che il linguaggio L = {anbm | n, m ≥ 0 e n + m è pari} è regolare.

- Dimostra che il linguaggio $L = \{w \in \{a, b\} * | w$ è una stringa palindroma che inizia e finisce con "a"} è regolare.

Soluzione:

Per dimostrare che il linguaggio L è regolare, costruiamo un automa finito deterministico (DFA) che accetta tutte le stringhe in L.

L'automa per L ha tre stati: q0, q1 e q2. Lo stato q0 è lo stato iniziale e q2 è lo stato finale.

La transizione da q0 a q1 avviene quando l'input è 'a', altrimenti rimaniamo in q0. La transizione da q1 a q0 avviene quando l'input è 'a', altrimenti avviene la transizione da q1 a q2. La transizione da q2 a q2 avviene con qualsiasi input.

Formalmente, l'automa può essere descritto come segue:

- Q = {q0, q1, q2};
- Σ = {a, b};
- δ(q0, a) = q1, δ(q0, b) = q0, δ(q1, a) = q0, δ(q1, b) = q1, δ(q2, a) = q2, δ(q2, b) = q2;
- q0 è lo stato iniziale e {q2} è l'insieme degli stati finali.

Questo DFA accetta solo le stringhe che sono palindrome e iniziano e finiscono con "a". Inoltre, essendo un DFA, accetta solo le stringhe che appartengono a L.

Pertanto, abbiamo dimostrato che il linguaggio L = {w ∈ {a, b}* | w è una stringa palindroma che inizia e finisce con "a"} è regolare attraverso la costruzione di un automa finito deterministico.

- Dimostra che il linguaggio $L = \{w \in \{a, b\} * | w$ ha un numero primo di "a"} è regolare.

Soluzione:

Si può dimostrare che il linguaggio è regolare attraverso la costruzione di un automa finito deterministico (DFA).

L'automa per L ha due stati: q0 e q1. Lo stato q0 è lo stato iniziale e q1 è lo stato finale.

La transizione da q0 a q0 avviene quando l'input è 'b', altrimenti avviene la transizione da q0 a q1. La transizione da q1 a q1 avviene quando l'input è 'a', altrimenti avviene la transizione da q1 a q0.

Formalmente, l'automa può essere descritto come segue:

- Q = {q0, q1};
- Σ = {a, b};
- δ(q0, a) = q1, δ(q0, b) = q0, δ(q1, a) = q1, δ(q1, b) = q0;
- q0 è lo stato iniziale e {q1} è l'insieme degli stati finali.

Questo DFA accetta solo le stringhe che hanno un numero primo di "a", in quanto l'automa si trova nello stato finale q1 solo se incontra un numero primo di "a". Inoltre, essendo un DFA, accetta solo le stringhe che appartengono a L.

Pertanto, abbiamo dimostrato che il linguaggio L = {w ∈ {a, b}* | w ha un numero primo di "a"} è regolare attraverso la costruzione di un automa finito deterministico.

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:

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

Per dimostrare che stutter(L) è regolare se L è regolare, possiamo costruire un automa a stati finiti che accetta il linguaggio stutter(L).

L'idea di base è di costruire un automa che riconosce una stringa w in L, e per ogni transizione che legge un simbolo a, passa a uno stato in cui i simboli successivi sono ripetuti due volte. In questo modo, l'automa accetta tutte le stringhe in stutter(L).

Formalmente, l'automa a stati finiti per stutter(L) può essere costruito come segue:

1. Lo stato iniziale è lo stesso dello stato iniziale dell'automa per L.

2. Ogni stato finale dell'automa per L è anche uno stato finale per l'automa per stutter(L).

3. Per ogni transizione (q, a, r) dell'automa per L, dove q e r sono stati e a è un simbolo in {0, 1}, aggiungiamo le seguenti transizioni all'automa per stutter(L):

- (q, a, s) dove s è uno stato che legge il simbolo a e passa allo stato successivo in cui ogni simbolo viene ripetuto due volte.

- (s, a, s) per ogni simbolo a in {0, 1}.

- (s, b, r) per ogni simbolo b diverso da a in {0, 1}.

In questo modo, l'automa per stutter(L) legge ogni simbolo della stringa in L esattamente come l'automa per L, ma ogni volta che legge un simbolo a, passa allo stato s in cui ogni simbolo successivo viene ripetuto due volte. Quando l'automa legge un simbolo diverso da a, passa allo stato successivo dell'automa per L.

Poiché L è un linguaggio regolare e l'automa per L può essere costruito utilizzando un numero finito di stati e transizioni, l'automa per stutter(L) può essere costruito utilizzando un numero finito di stati e transizioni. Quindi, stutter(L) è un linguaggio regolare.

In conclusione, abbiamo dimostrato che se L è un linguaggio regolare sull'alfabeto {0, 1}, allora stutter(L) è un linguaggio regolare.

3. Sia L un linguaggio regolare su un alfabeto Σ. Supponete che il simbolo $\#$ appartenga all'alfabeto Σ e dimostrate che il seguente linguaggio è regolare:

$$dehash(L) = \{dehash(w) : w \in L\}$$

dove $dehash(w)$ è la stringa che si ottiene eliminando tutti i simboli $\#$ da w.

Per dimostrare che dehash(L) è regolare se L è regolare, possiamo costruire un automa a stati finiti che accetta il linguaggio dehash(L).

L'idea di base è di costruire un automa che riconosce una stringa w in L, e che in ogni stato riconosce il simbolo # e passa allo stato successivo senza leggerlo. In questo modo, l'automa accetta tutte le stringhe in dehash(L).

Formalmente, l'automa a stati finiti per dehash(L) può essere costruito come segue:

1. Lo stato iniziale è lo stesso dello stato iniziale dell'automa per L.

2. Ogni stato finale dell'automa per L è anche uno stato finale per l'automa per dehash(L).

3. Per ogni transizione (q, a, r) dell'automa per L, dove q e r sono stati e a è un simbolo in \Sigma, aggiungiamo le seguenti transizioni all'automa per dehash(L):

- (q, #, q) per ogni simbolo # in \Sigma.

- (q, b, r) per ogni simbolo b diverso da # in \Sigma.

In questo modo, l'automa per dehash(L) legge ogni simbolo della stringa in L esattamente come l'automa per L, ma ogni volta che legge un simbolo #, passa allo stato successivo senza leggerlo. Quando l'automa legge un simbolo diverso da #, passa allo stato successivo dell'automa per L.

Poiché L è un linguaggio regolare e l'automa per L può essere costruito utilizzando un numero finito di stati e transizioni, l'automa per dehash(L) può essere costruito utilizzando un numero finito di stati e transizioni. Quindi, dehash(L) è un linguaggio regolare.

In conclusione, abbiamo dimostrato che se L è un linguaggio regolare sull'alfabeto \Sigma, allora dehash(L) è un linguaggio regolare.

4. Sia L un linguaggio regolare su un alfabeto Σ. Dimostrare che anche il seguente linguaggio è regolare:

$$init(L) = \{w \in \Sigma^* : \text{ esiste } x \in \Sigma^* \text{ tale che } wx \in L\}$$

Per dimostrare che init(L) è regolare se L è regolare, possiamo costruire un automa a stati finiti che accetta il linguaggio init(L).

L'idea di base è di costruire un automa che riconosce una stringa w in init(L) se esiste una stringa x tale che wx appartiene a L. In altre parole, l'automa deve essere in grado di leggere ogni possibile prefisso di una stringa in L e accettare solo i prefissi che possono essere estesi in una stringa in L.

Formalmente, l'automa a stati finiti per init(L) può essere costruito come segue:

1. Lo stato iniziale è uno stato speciale q0 che non legge nessun simbolo e che ha una transizione per ogni stato finale dell'automa per L.

2. Per ogni stato finale r dell'automa per L, aggiungiamo una transizione (q0, \epsilon, r) all'automa per init(L).

3. Per ogni transizione (q, a, r) dell'automa per L, dove q e r sono stati e a è un simbolo in \Sigma, aggiungiamo le seguenti transizioni all'automa per init(L):

- (q, a, r) per ogni simbolo a in \Sigma diverso da #.
- (q, a, q) per ogni simbolo a in \Sigma diverso da #.
- (q, #, q0) per ogni simbolo # in \Sigma.

In questo modo, l'automa per init(L) legge ogni simbolo della stringa w in ordine e passa allo stato successivo dell'automa per L solo se la stringa letta fino a quel punto è un prefisso di una stringa in L. Se l'automa incontra un simbolo #, torna allo stato iniziale q0, che accetta solo stringhe vuote o prefissi di stringhe in L.

Poiché L è un linguaggio regolare e l'automa per L può essere costruito utilizzando un numero finito di stati e transizioni, l'automa per init(L) può essere costruito utilizzando un numero finito di stati e transizioni. Quindi, init(L) è un linguaggio regolare.

In conclusione, abbiamo dimostrato che se L è un linguaggio regolare sull'alfabeto \Sigma, allora init(L) è un linguaggio regolare.

<u>Pumping Lemma per linguaggi context-free</u>

- Dimostra che il linguaggio $L = \{a^n b^m c^k \,|n > m > k\}$ non è context-free.

Per dimostrare che L non è context-free, possiamo utilizzare il teorema di pumping lemma per i linguaggi context-free. Supponiamo, per assurdo, che L sia context-free. Allora, esiste un intero positivo p tale che per ogni stringa w di L con |w| ≥ p, esiste una suddivisione w = uvxyz tale che:

- |vxy| ≤ p;
- |vy| > 0;
- per ogni k ≥ 0, la stringa $uv^k xy^k z$ appartiene a L.

Consideriamo la stringa $w = a^p b^p c^p$, dove p è un intero positivo. Poiché $|w| = 3p$, esiste una suddivisione $w = uvxyz$ tale che $|vxy| \leq p$ e $|vy| > 0$. Ci sono tre casi possibili per la stringa vxy:

- vxy contiene solo "a";
- vxy contiene solo "b";
- vxy contiene solo "c".

In ogni caso, possiamo scegliere k = 2 e ottenere la stringa uv^2xy^2z, che non appartiene a L, in quanto contiene un numero di "a", "b" e "c" non uguale. Quindi, la terza condizione del teorema di pumping lemma non è soddisfatta, il che contraddice l'ipotesi che L sia context-free.

Pertanto, il linguaggio $L = \{a^n b^n c^n \mid n \geq 0\}$ non è context-free.

- Dimostra che il linguaggio $L = \{w \mid w$ ha un numero uguale di simboli $a, b \ e \ c\}$ non è context-free.

Soluzione:

Supponiamo per assurdo che L sia un linguaggio context-free. Applichiamo il pumping lemma per linguaggi context-free a L come segue:

1. Sia p la costante del pumping lemma per L.

2. Scegliamo la stringa w = $a^p \ b^p \ c^p$ in L.

3. Scriviamo w come $u \ v \ x \ y \ z$, dove |vxy| <= p, |vy| >= 1 e u, v, x, y, z sono stringhe qualsiasi tali che $w = u \ v \ x \ y \ z$.

4. Consideriamo ogni possibile scelta di v e y:

- Se v o y contiene solo un tipo di simbolo, allora la stringa u v^2 x y^2 z non appartiene a L.

- Se v e y contengono due tipi di simboli, allora vxy non può essere una sottostringa di a^p b^p c^p perché ha lunghezza maggiore di p. Quindi, la stringa u v^2 x y^2 z non appartiene a L.

- Se v e y contengono tutti e tre i simboli, allora vxy non può essere una sottostringa di a^p b^p c^p perché contiene almeno un simbolo ripetuto più di p/3 volte. Quindi, la stringa u v^2 x y^2 z non appartiene a L.

In ogni caso, abbiamo trovato una stringa u v^2 x y^2 z che non appartiene a L, in contraddizione con l'ipotesi che L sia un linguaggio context-free. Quindi, L non è context-free. Esercizio:

- Dimostra che il linguaggio $L = \{a^n \ b^m \ c^{n+m} \mid n > m\}$ non è context-free.

Soluzione:

Supponiamo per assurdo che L sia un linguaggio context-free. Applichiamo il pumping lemma per linguaggi context-free a L come segue:

Sia p la costante del pumping lemma per L.

Scegliamo la stringa $w = a^p \ b^{p-1} \ c^{2p-1}$ in L.

Scriviamo w come u v x y z, dove |vxy| <= p, |vy| >= 1 e u,v,x,y,z sono stringhe qualsiasi tali che w = u v x y z.

Consideriamo ogni possibile scelta di v e y:

- Se v contiene solo a, allora x contiene almeno un b e y contiene almeno un c. Quindi, la stringa u v^2 x y^2 z non appartiene a L perché il numero di b è maggiore del numero di c.
- Se v contiene solo b, allora x contiene solo a e y contiene solo c. Quindi, la stringa u v^2 x y^2 z non appartiene a L perché il numero di a è maggiore del numero di b.
- Se v contiene solo c, allora x contiene almeno un b e y contiene almeno un a. Quindi, la stringa u v^2 x y^2 z non appartiene a L perché il numero di c è maggiore del numero di a.
- Se v contiene almeno due tipi di simboli, allora vxy non può essere una sottostringa di a^p b^(p-1) c^(2p-1) perché ha lunghezza maggiore di p. Quindi, la stringa u v^2 x y^2 z non appartiene a L.

In ogni caso, abbiamo trovato una stringa u v^2 x y^2 z che non appartiene a L, in contraddizione con l'ipotesi che L sia un linguaggio context-free. Quindi, L non è context-free.

- Dimostra che il linguaggio $L = \{a^n\ b^n\ c^m\ d^m \mid n, m >= 1\}$ non è context-free.

Soluzione:

Supponiamo per assurdo che L sia un linguaggio context-free. Applichiamo il pumping lemma per linguaggi context-free a L come segue:

Sia p la costante del pumping lemma per L.

Scegliamo la stringa w = $a^p\ b^p\ c^p\ d^p$ in L.

Scriviamo w come u v x y z, dove |vxy| <= p, |vy| >= 1 e u,v,x,y,z sono stringhe qualsiasi tali che w = u v x y z.

Consideriamo ogni possibile scelta di v e y:

- Se v o y contiene solo un tipo di simbolo, allora la stringa u v^2 x y^2 z non appartiene a L.
- Se v e y contengono due tipi di simboli, allora vxy non può essere una sottostringa di a^p b^p c^p d^p perché contiene almeno un simbolo ripetuto più di p volte. Quindi, la stringa u v^2 x y^2 z non appartiene a L.
- Se v e y contengono tre tipi di simboli, allora vxy non può essere una sottostringa di a^p b^p c^p d^p perché contiene almeno due simboli ripetuti più di p volte. Quindi, la stringa u v^2 x y^2 z non appartiene a L.

In ogni caso, abbiamo trovato una stringa u v^2 x y^2 z che non appartiene a L, in contraddizione con l'ipotesi che L sia un linguaggio context-free. Quindi, L non è context-free.

Dimostrare che L è un linguaggio context free

2. (8 punti) Per ogni linguaggio L sull'alfabeto Σ, sia $\text{superstring}(L) = \{xyz \mid y \in L \text{ e } x, z \in \Sigma^*\}$. Dimostra che se L è un linguaggio context-free, allora anche $\text{superstring}(L)$ è un linguaggio context-free.

Per dimostrare che superstring(L) è un linguaggio context-free se L è context-free, possiamo costruire una grammatica context-free che genera superstring(L).

L'idea di base è di costruire una grammatica che genera tutte le possibili concatenazioni di stringhe x, y e z, dove y appartiene a L e x e z sono stringhe qualsiasi sull'alfabeto \Sigma.

Formalmente, la grammatica context-free G per superstring(L) può essere definita come segue:

1. Aggiungiamo un nuovo simbolo di start S alla grammatica G.

2. Per ogni regola A -> alpha della grammatica per L, aggiungiamo le seguenti regole alla grammatica G:

   - S -> aSb per ogni simbolo a,b in \Sigma.

   - S -> A per ogni regola A -> alpha della grammatica per L.

   - S -> aS per ogni simbolo a in \Sigma.

   - S -> Sb per ogni simbolo b in \Sigma.

   - S -> \epsilon.

In questo modo, la grammatica G genera tutte le possibili concatenazioni di stringhe x, y e z, dove y appartiene a L e x e z sono stringhe qualsiasi sull'alfabeto \Sigma. In particolare, le regole S -> aSb generano le possibili intersezioni tra le stringhe x, y e z, mentre le regole S -> A generano tutte le possibili stringhe y in L.

Poiché L è un linguaggio context-free e la grammatica per superstring(L) può essere costruita utilizzando un numero finito di regole, la grammatica per superstring(L) è anche context-free. Quindi, superstring(L) è un linguaggio context-free.

In conclusione, abbiamo dimostrato che se L è un linguaggio context-free sull'alfabeto \Sigma, allora superstring(L) è un linguaggio context-free.

2. (8 punti) Per ogni linguaggio L, sia $\text{substring}(L) = \{v \mid uvw \in L \text{ per qualche coppia di stringhe } u, w\}$. Dimostra che se L è un linguaggio context-free, allora anche $\text{substring}(L)$ è un linguaggio context-free.

Per dimostrare che substring(L) è un linguaggio context-free se L è un linguaggio context-free, possiamo utilizzare la costruzione di pompa per i linguaggi context-free.

Supponiamo che L sia generato da una grammatica context-free G = (V, Σ, R, S), dove V è l'insieme delle variabili, Σ è l'alfabeto di simboli terminali, R è l'insieme delle regole di produzione e S è il simbolo iniziale. Se prendiamo una stringa w ∈ substring(L), allora per definizione esistono due stringhe u e v tali che w = uvv', dove v' è una stringa qualsiasi. Dobbiamo dimostrare che esiste una grammatica context-free G' = (V', Σ, R', S'), dove V' è l'insieme delle variabili di G e di nuove variabili introdotte per la costruzione, R' è l'insieme delle regole di produzione di G e di nuove regole introdotte per la costruzione, e S' è il simbolo iniziale, tale che w ∈ L(G').

Per costruire G', introduciamo una nuova variabile A per ogni coppia di variabili A, B ∈ V di G. Ogni variabile A' rappresenta la presenza di una sottostringa compresa tra due occorrenze di A in L. In altre parole, A'

rappresenta tutte le stringhe che possono essere ottenute sostituendo A con una qualsiasi combinazione di stringhe di A e B. Le regole di produzione di G' sono le seguenti:

1. S' → S

2. A → ε, per ogni A ∈ V

3. A' → a, per ogni a ∈ Σ

4. A' → AB', per ogni A, B ∈ V

5. A' → A'A', per ogni A ∈ V

Le regole di produzione 1 e 2 servono per mantenere la stessa grammatica del linguaggio di partenza. La regola di produzione 3 introduce le stringhe di lunghezza 1 che sono presenti in substring(L). La regola di produzione 4 introduce tutte le combinazioni di stringhe di A e B, che rappresentano le sottostringhe di L contenenti A. Infine, la regola di produzione 5 ripete le variabili A', che rappresentano le sottostringhe contenenti A.

La grammatica G' può essere costruita in tempo polinomiale rispetto alla dimensione di G e di substring(L), e quindi substring(L) è un linguaggio context-free.

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.

Per dimostrare che suffix(L) è un linguaggio context-free se L è un linguaggio context-free, possiamo costruire una grammatica context-free che genera suffix(L).

L'idea di base è di costruire una grammatica che genera tutte le possibili stringhe v che sono suffissi di stringhe in L. In particolare, una stringa v è un suffisso di una stringa in L se e solo se esiste una stringa u tale che la concatenazione di u e v appartiene a L.

Formalmente, la grammatica context-free G per suffix(L) può essere definita come segue:

1. Aggiungiamo un nuovo simbolo di start S alla grammatica G.

2. Per ogni regola A -> alpha della grammatica per L, aggiungiamo le seguenti regole alla grammatica G:

   - S -> v per ogni stringa v in \Sigma^*.

   - S -> Au per ogni regola A -> alpha della grammatica per L e ogni stringa u in \Sigma^*.

In questo modo, la grammatica G genera tutte le possibili stringhe v che sono suffissi di stringhe in L. In particolare, le regole S -> v generano tutti i possibili suffissi di stringhe sull'alfabeto \Sigma, mentre le regole S -> Au generano tutti i possibili suffissi di stringhe in L.

Poiché L è un linguaggio context-free e la grammatica per suffix(L) può essere costruita utilizzando un numero finito di regole, la grammatica per suffix(L) è anche context-free. Quindi, suffix(L) è un linguaggio context-free.

In conclusione, abbiamo dimostrato che se L è un linguaggio context-free sull'alfabeto \Sigma, allora suffix(L) è un linguaggio context-free.

13. Se A e B sono linguaggi, definiamo $A \circ B = \{xy \mid x \in A, y \in B \text{ e } |x| = |y|\}$. Mostrare che se A e B sono linguaggi regolari, allora $A \circ B$ è un linguaggio context-free.

Per dimostrare che A.B è un linguaggio context-free se A e B sono linguaggi regolari, possiamo costruire una grammatica context-free che genera A.B.

L'idea di base è di costruire una grammatica che genera tutte le possibili coppie di stringhe x e y di lunghezza uguale, dove x appartiene ad A e y appartiene a B. In particolare, una stringa xy appartiene ad A.B se e solo se |x| = |y|.

Formalmente, la grammatica context-free G per A.B può essere definita come segue:

1. Aggiungiamo un nuovo simbolo di start S alla grammatica G.

2. Aggiungiamo un nuovo simbolo non terminale U alla grammatica G.

3. Aggiungiamo regole alla grammatica G per generare tutte le possibili stringhe di lunghezza uguale da A e B:

- S -> UA | \epsilon

- U -> aUc | bUd | \epsilon, dove a,b,c,d sono simboli in \Sigma e A e B sono gli automi a stati finiti per A e B, rispettivamente.

In questo modo, la grammatica G genera tutte le possibili coppie di stringhe x e y di lunghezza uguale, dove x appartiene ad A e y appartiene a B. In particolare, le regole U -> aUc e U -> bUd generano tutte le possibili coppie di stringhe x e y di lunghezza uguale, dove x appartiene ad A e y appartiene a B.

Poiché A e B sono linguaggi regolari, gli automi a stati finiti per A e B possono essere costruiti utilizzando un numero finito di stati e transizioni. Inoltre, la grammatica per A.B può essere costruita utilizzando un numero finito di regole. Quindi, la grammatica per A.B è una grammatica context-free.

In conclusione, abbiamo dimostrato che se A e B sono linguaggi regolari, allora A.B è un linguaggio context-free.

- Per ogni linguaggio L, sia $divide(L) = \{v \mid \frac{u}{w} \in L$ per qualche coppia di stringhe $u, w\}$. Dimostra che se L è un linguaggio context-free, allora anche $divide(L)$ è context-free.

Per dimostrare che divide(L) è un linguaggio context-free se L è un linguaggio context-free, possiamo costruire una grammatica context-free che genera divide (L).

Sia G = (V, Σ, P, S) una grammatica context-free per L. Costruiamo una nuova grammatica G' = (V', Σ, P', S') per divide(L) come segue:

1. V' = V ∪ {X, Y, Z}

2. P' = P ∪ { S' → S, S' → X, X → YZ, Y → ε, Y → aYb, Z → cZd, Z → ε }

3. S' = S

La regola S' → S genera tutte le stringhe in L, mentre la regola S' → X genera tutte le stringhe in divide(L) che non appartengono a L. La regola X → YZ genera tutte le stringhe in divide(L) che hanno la forma uvw, dove u e w appartengono a Σ^* e v appartiene a L. La regola Y → aYb genera sequenze di a e b di lunghezza crescente e la regola Z → cZd genera sequenze di c e d di lunghezza crescente. La regola Y → ε gestisce il caso in cui v è una stringa vuota, mentre la regola Z → ε gestisce il caso in cui u e w sono stringhe vuote.

Poiché abbiamo costruito una grammatica context-free per divide(L), segue che divide(L) è un linguaggio context-free.

Pertanto, se L è un linguaggio context-free, anche divide(L) è un linguaggio context-free.

Pumping Lemma

- Dimostra che il linguaggio $L = \{0^n 1^m 2^{n+m} \mid n, m \geq 0\}$ non è regolare

Supponiamo per assurdo che L sia un linguaggio regolare. Allora, per il Pumping Lemma, esiste un intero positivo p (il "pumping length") tale che per ogni stringa w appartenente a L di lunghezza almeno p, possiamo scrivere w come xyz, dove:

- |xy| <= p
- |y| > 0
- per ogni intero positivo i, la stringa xy^iz appartiene a L.

Scegliamo la stringa $w = 0^p\ 1^p\ 2^{2p}$ appartenente a L. Poiché $|w| = 3p\ >=\ p$, per il Pumping Lemma possiamo scrivere w come xyz, dove le proprietà 1 e 2 sono soddisfatte.

Consideriamo ora la stringa $xy^2z = 0^{p+k}\ 1^p\ 2^{2p+k}$, dove k=|y|. Poiché la proprietà 3 del Pumping Lemma deve ancora essere soddisfatta, dobbiamo dimostrare che xy^2z appartiene ancora a L.

In primo luogo, poiché $|xy|\ <=\ p$, la sottostringa xy può contenere solo zeri e/o uni. Inoltre, poiché |y| > 0, la sottostringa y deve contenere almeno un uno.

Consideriamo ora la lunghezza della stringa xy^2z. Poiché stiamo raddoppiando la sottostringa y, l'incremento di lunghezza sarà di almeno k unità. Pertanto, la lunghezza della stringa xy^2z sarà maggiore di p, ma non supererà la lunghezza massima di 2p per il terzo componente di L.

Tuttavia, la sottostringa xy^2 può contenere al massimo p zeri e p+1 uni. Pertanto, la stringa xy^2z non può appartenere a L, poiché il numero di zeri non è uguale al numero di due elevato alla somma di n e m.

Abbiamo così ottenuto una contraddizione, poiché abbiamo supposto che L sia un linguaggio regolare, ma abbiamo dimostrato che esiste una stringa in L che non può essere pompati. Di conseguenza, il linguaggio L non può essere regolare.

- Dimostra che il linguaggio $L = \{a^l b^m a^n \big|\ l + m \equiv n\ \ mod\ 3\}$ non è regolare

Per dimostrare che il linguaggio L={a^l b^m a^n | l+m≡n mod 3} non è regolare, useremo il Pumping Lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia un linguaggio regolare. Allora, per il Pumping Lemma, esiste un intero positivo p (il "pumping length") tale che per ogni stringa w appartenente a L di lunghezza almeno p, possiamo scrivere w come xyz, dove:

- |xy| <= p
- |y| > 0
- per ogni intero positivo i, la stringa xy^iz appartiene a L.

Scegliamo la stringa $w\ =\ a^{p+2}\ b^{2p}\ a^{2p+1}$ appartenente a L. Poiché $|w|\ =\ 5p + 3\ >=\ p$, per il Pumping Lemma possiamo scrivere w come xyz, dove le proprietà 1 e 2 sono soddisfatte.

Consideriamo ora la stringa xy^2z. Poiché la proprietà 3 del Pumping Lemma deve ancora essere soddisfatta, dobbiamo dimostrare che xy^2z appartiene ancora a L.

Sia k = |y|. Poiché |xy| <= p, la sottostringa xy può contenere solo a. Inoltre, poiché |y| > 0, la sottostringa y deve contenere almeno un a. Quindi, possiamo scrivere xy come a^r e y come a^s, dove r+s <= p e s > 0.

Consideriamo ora la lunghezza della stringa xy^2z. Poiché stiamo raddoppiando la sottostringa y, l'incremento di lunghezza sarà di almeno s unità. Pertanto, la lunghezza della stringa xy^2z sarà maggiore di p, ma non supererà la lunghezza massima di 5p+3 per il terzo componente di L.

Inoltre, la stringa xy^2z deve ancora soddisfare l'equazione l+m≡n mod 3. Poiché l = r+s, m = 2p e n = 2p+1, abbiamo l+m = r+s+2p e n = 2p+1. Quindi, l+m-n = r+s-1.

Poiché r+s <= p, abbiamo -p < r+s <= p e -p-1 < r+s-1 <= p-1. Pertanto, l+m-n = r+s-1 è compreso tra -2p-1 e 2p-1.

Poiché l+m-n ≡ 2p-1 ≡ -p+2 mod 3, la proprietà l+m≡n mod 3 non può essere soddisfatta per la stringa xy^2z. Quindi, xy^2z non appartiene a L, il che contraddice l'ipotesi che L sia un linguaggio regolare.

Pertanto, il linguaggio L non può essere regolare.

- Dimostra che il linguaggio $L = \{a^i b^j c^k \mid i, j, k \geq 0, i = j \; oppure \; j = k\}$ non è regolare

Per dimostrare che il linguaggio L={a^i b^j c^k | i,j,k≥0, i=j oppure j=k} non è regolare, useremo il Pumping Lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia un linguaggio regolare. Allora, per il Pumping Lemma, esiste un intero positivo p (il "pumping length") tale che per ogni stringa w appartenente a L di lunghezza almeno p, possiamo scrivere w come xyz, dove:

- |xy| <= p
- |y| > 0
- per ogni intero positivo i, la stringa xy^iz appartiene a L.

Scegliamo la stringa $w = a^p \, b^p \, c^p$ appartenente a L. Poiché $|w| = 3p >= p$, per il Pumping Lemma possiamo scrivere w come xyz, dove le proprietà 1 e 2 sono soddisfatte.

Consideriamo ora la stringa xy^2z. Poiché la proprietà 3 del Pumping Lemma deve ancora essere soddisfatta, dobbiamo dimostrare che xy^2z appartiene ancora a L.

Sia k = |y|. Poiché |xy| <= p, la sottostringa xy può contenere solo a e/o b. Inoltre, poiché |y| > 0, la sottostringa y deve contenere almeno una a o una b.

Ci sono due casi da considerare:

- Se y contiene solo a, allora la stringa xy^2z conterrà più a che b e c, il che significa che i non sarà uguale a né j né k. Pertanto, xy^2z non appartiene a L.
- Se y contiene solo b, allora la stringa xy^2z conterrà più b che a e c, il che significa che j non sarà uguale a né i né k. Pertanto, xy^2z non appartiene a L.

In entrambi i casi, xy^2z non appartiene a L, il che contraddice l'ipotesi che L sia un linguaggio regolare.

Pertanto, il linguaggio L non può essere regolare.

- Dimostra che il linguaggio $L = \{0^m 1^n 0^{(m*n)} : n, m \geq 0\}$ non è regolare

Per dimostrare che il linguaggio L={0^m 1^n 0^(m*n) : n,m≥0} non è regolare, useremo il Pumping Lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia un linguaggio regolare. Allora, per il Pumping Lemma, esiste un intero positivo p (il "pumping length") tale che per ogni stringa w appartenente a L di lunghezza almeno p, possiamo scrivere w come xyz, dove:

- |xy| <= p
- |y| > 0
- per ogni intero positivo i, la stringa xy^iz appartiene a L.

Scegliamo la stringa $w = 0^p 1^p 0^{p^2}$ appartenente a L. Poiché $|w| = p + p + p^2 >= p$, per il Pumping Lemma possiamo scrivere w come xyz, dove le proprietà 1 e 2 sono soddisfatte.

Consideriamo ora la stringa xy^2z. Poiché la proprietà 3 del Pumping Lemma deve ancora essere soddisfatta, dobbiamo dimostrare che xy^2z appartiene ancora a L.

Sia k = |y|. Poiché |xy| <= p, la sottostringa xy può contenere solo zeri. Inoltre, poiché |y| > 0, la sottostringa y deve contenere almeno un 0.

La lunghezza della stringa xy^2z sarà maggiore di p, ma non supererà la lunghezza massima di p + p + (p^2) per il terzo componente di L.

Consideriamo ora il numero di 0 e 1 nella stringa xy^2z. Poiché la sottostringa y contiene solo 0, la sottostringa xy^2 conterrà esattamente k zeri in più rispetto a xy. Pertanto, il numero di zeri nella stringa xy^2z sarà pari a p + k.

Poiché il terzo componente di L richiede che ci siano p^2 zeri nella stringa, dobbiamo avere p^2 = p + k, ovvero k = p^2 - p.

Consideriamo ora il numero di uni nella stringa xy^2z. Poiché la sottostringa y contiene solo zeri, il numero di uni nella stringa xy^2z sarà pari a p.

Tuttavia, poiché il terzo componente di L richiede che ci siano mn zeri nella stringa, e mn = p^2, il numero di uni nella stringa dovrebbe essere pari a p^2, il che non è il caso.

Quindi, xy^2z non appartiene a L, il che contraddice l'ipotesi che L sia un linguaggio regolare.

Pertanto, il linguaggio L non può essere regolare.

- Dimostra che il linguaggio $L = \{w_1 \# w_2 \# \dots \# w_n : w_i \neq w_j \; per \; i \neq j\}$ non è regolare

Per dimostrare che il linguaggio L={w_1#w_2#…#w_n :w_i≠w_j per i≠j} non è regolare, useremo il Pumping Lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia un linguaggio regolare. Allora, per il Pumping Lemma, esiste un intero positivo p (il "pumping length") tale che per ogni stringa w appartenente a L di lunghezza almeno p, possiamo scrivere w come xyz, dove:

- |xy| <= p
- |y| > 0
- per ogni intero positivo i, la stringa xy^iz appartiene a L.

Scegliamo la stringa $w = x\#y\#z$ appartenente a L, dove x, y e z sono stringhe distinte di lunghezza p. Poiché $|w| = 3p + 2 >= p$, per il Pumping Lemma possiamo scrivere w come xyz, dove le proprietà 1 e 2 sono soddisfatte.

Consideriamo ora la stringa xy^2z. Poiché la proprietà 3 del Pumping Lemma deve ancora essere soddisfatta, dobbiamo dimostrare che xy^2z appartiene ancora a L.

Sia k = |y|. Poiché |xy| <= p, la sottostringa xy può contenere solo una parte di x o di y o di entrambe. Inoltre, poiché |y| > 0, la sottostringa y deve contenere almeno un carattere.

La lunghezza della stringa xy^2z sarà maggiore di p, ma non supererà la lunghezza massima di 3p+2 per il primo componente di L.

Poiché xy^2z è ottenuta raddoppiando la sottostringa y, ci sono solo due casi da considerare:

Se y è una sottostringa di x, allora nella stringa xy^2z ci saranno due copie di x consecutive. Quindi, xy^2z non può appartenere a L.

Se y è una sottostringa di y, allora nella stringa xy^2z ci saranno due copie di y consecutive. Quindi, xy^2z non può appartenere a L.

In entrambi i casi, xy^2z non appartiene a L, il che contraddice l'ipotesi che L sia un linguaggio regolare.

Pertanto, il linguaggio L non può essere regolare.

- Dimostra che il linguaggio $L = \{xy: x = y,\ x \neq y\}$ non è regolare
    - Ad esempio, L contiene la stringa "abab", ma non contiene la stringa "aaaa".

Il linguaggio L={xy:x=y,x≠y} rappresenta l'insieme di tutte le coppie di stringhe uguali e diverse. Ad esempio, L contiene la stringa "abab", ma non contiene la stringa "aaaa".

Per dimostrare che L non è regolare, useremo il pumping lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia un linguaggio regolare. Allora, per il pumping lemma, esiste un intero positivo p (il "pumping length") tale che per ogni stringa w appartenente a L di lunghezza almeno p, possiamo scrivere w come xyz, dove:

- |xy| <= p
- |y| > 0
- per ogni intero positivo i, la stringa xy^iz appartiene a L.

Scegliamo la stringa $w = a^p\, a^p$. Poiché $|w| = 2p >= p$, per il pumping lemma possiamo scrivere w come xyz, dove le proprietà 1 e 2 sono soddisfatte.

Consideriamo ora la stringa xy^2z. Poiché la proprietà 3 del pumping lemma deve ancora essere soddisfatta, dobbiamo dimostrare che xy^2z appartiene ancora a L.

Sia k = |y|. Poiché |xy| <= p, la sottostringa xy può contenere solo a. Inoltre, poiché |y| > 0, la sottostringa y deve contenere almeno un carattere.

Per dimostrare che xy^2z non appartiene a L, dobbiamo dimostrare che xy^2z deve contenere due sottostringhe uguali e diverse.

Tuttavia, ogni sottostringa di xy^2z sarà una concatenazione di una sottostringa di x e una sottostringa di y. Poiché x e y sono uguali per definizione, ogni sottostringa di xy^2z sarà uguale a un'altra sottostringa di xy^2z.

Quindi, xy^2z non può contenere due sottostringhe uguali e diverse, il che significa che xy^2z appartiene a L.

Questo contraddice l'ipotesi che L non sia regolare, quindi L non può essere regolare.

Linguaggi regolari

- Sia L un linguaggio e sia a una stringa.
  Definiamo $\left(\frac{L}{a}\right)^R$ l'insieme delle stringhe w tali che wa^R appartiene ad L. Dimostrare che $\left(\frac{L}{a}\right)^R$ è regolare.

Per dimostrare che (L/a)^R è regolare, possiamo costruire un automa a stati finiti che riconosce questo linguaggio.

Sia M = (Q, Σ, δ, q0, F) un automa a stati finiti che riconosce il linguaggio L. L'idea è di costruire un nuovo automa a partire da M, in modo che riconosca (L/a)^R.

L'automa per (L/a)^R avrà gli stessi stati di M, ma avrà una nuova funzione di transizione δ' e un nuovo insieme di stati finali F'.

La funzione di transizione δ' è definita come segue:

δ'(q, b) = {δ(p, b) | p è un qualsiasi stato tale che δ(p, a) = q}

In altre parole, δ'(q, b) è l'insieme degli stati che possiamo raggiungere da q leggendo il simbolo b, partendo da qualsiasi stato p tale che leggendo a si raggiunge lo stato q in M.

L'insieme degli stati finali F' è definito come segue:

F' = {q | q0 ∈ δ(q, a), q ∈ Q}

In altre parole, un certo stato q è finale in F' se e solo se, partendo dallo stato iniziale q0 di M e leggendo la stringa a, si arriva nello stato q in M.

L'automa M' = (Q, Σ, δ', q0, F') così definito riconosce il linguaggio (L/a)^R.

Per dimostrare che M' è corretto, consideriamo una stringa w appartenente a (L/a)^R. Ciò significa che esiste una stringa v tale che wa^Rv appartiene a L. Quindi, se seguiamo la stringa w in M' partendo dallo stato iniziale q0, arriviamo a uno stato q tale che δ(q, a) è lo stato in cui si arriva in M leggendo a partire da q0. In altre parole, w porta M' allo stato q. Ma poiché wa^Rv appartiene a L, leggendo a^Rv in M si arriva ad uno stato finale f. Quindi, in M', leggendo v partendo dallo stato q, si arriva in uno stato finale f' tale che f' ∈ δ'(q, a). Ciò significa che M' accetta w.

Inoltre, se w non appartiene a (L/a)^R, allora non esiste alcuna stringa v tale che wa^Rv appartiene a L. Quindi, in M', leggendo a^R da uno stato q non si arriva mai ad uno stato finale di M, e quindi M' non accetta w.

Pertanto, abbiamo dimostrato che M' riconosce il linguaggio (L/a)^R, e quindi (L/a)^R è regolare.

- Sia L un linguaggio e sia a una stringa.
  Definiamo L_a^* come l'insieme delle stringhe w tali che ogni occorrenza di a è seguita da un numero pari di occorrenze di a. Dimostrare che L_a^* è un linguaggio regolare.

Per dimostrare che L_a^* è un linguaggio regolare, possiamo costruire un automa a stati finiti che riconosce questo linguaggio.

L'idea è di costruire un automa che tiene traccia del numero di occorrenze di a che sono state lette finora. Se il numero di occorrenze di a finora è dispari e stiamo leggendo un carattere diverso da a, l'automa deve accettare; altrimenti, se il numero di occorrenze di a finora è pari e stiamo leggendo un carattere diverso da a, l'automa deve continuare a leggere i caratteri successivi. Se stiamo leggendo un carattere a, l'automa deve passare ad uno stato in cui il numero di occorrenze di a finora diventa dispari.

Formalmente, l'automa M = (Q, Σ, δ, q0, F) è definito come segue:

Q: insieme degli stati, dove q0 è lo stato iniziale

Σ: alfabeto, dove Σ = {a} ∪ Σ', con Σ' che rappresenta l'alfabeto dei simboli diversi da a

δ: funzione di transizione, dove δ(q, b) = q per ogni stato q e simbolo b ∈ Σ'

δ(q, a) = q1 se il numero di occorrenze di a finora è pari, e δ(q, a) = q2 se il numero di occorrenze di a finora è dispari

δ(q1, a) = q2 e δ(q2, a) = q1

F: insieme degli stati finali, dove F = {q0, q1}

In altre parole, l'automa ha due stati per tenere traccia del fatto che il numero di occorrenze di a finora sia pari o dispari. Se l'automa si trova in uno stato q1 e legge un carattere a, passa allo stato q2 e viceversa. Se l'automa si trova in uno stato q1 e legge un carattere diverso da a, rimane in q1; se invece si trova in q2 e legge un carattere diverso da a, accetta.

Per dimostrare che M riconosce il linguaggio L_a^*, consideriamo una stringa w appartenente a L_a^*. Ciò significa che ogni occorrenza di a in w è seguita da un numero pari di occorrenze di a.

Supponiamo di leggere w in M partendo dallo stato iniziale q0. Inizialmente, il numero di occorrenze di a è pari. Se leggiamo un carattere diverso da a, l'automa rimane in q0, e il numero di occorrenze di a rimane pari. Se invece leggiamo un carattere a, l'automa passa allo stato q2, e il numero di occorrenze di a diventa dispari. Ogni successiva occorrenza di a sarà seguita da un numero pari di occorrenze di a, quindi l'automa passerà da q2 a q1 ogni volta che leggiamo un carattere a. Alla fine della lettura di w, il numero di occorrenze di a sarà pari, e quindi l'automa si troverà nello stato q1, che è uno stato finale.

Quindi, M accetta w. Inoltre, se una stringa non appartiene a L_a^*, allora l'automa non accetta la stringa, poiché in qualche punto si troverà in uno stato q2 con il numero di occorrenze di a dispari.

Pertanto, abbiamo dimostrato che M riconosce il linguaggio L_a^*, e quindi L_a^* è un linguaggio regolare.

- Sia L un linguaggio regolare e siano a e b due stringhe. Definiamo $L^R_{\{a,b\}}$ come l'insieme delle stringhe w tali che ogni occorrenza di a è immagazzinata in una pila e ogni occorrenza di b corrispondente elimina l'ultima occorrenza di a dalla pila. Dimostrare che $L^R_{\{a,b\}}$ è un linguaggio regolare.

Per dimostrare che L_{a,b}^R è un linguaggio regolare, possiamo costruire un automa a pila che riconosce questo linguaggio.

L'idea è di utilizzare una pila per tenere traccia delle occorrenze di a man mano che le leggiamo. Quando incontriamo una occorrenza di b, controlliamo se l'ultimo simbolo inserito nella pila è a; se sì, eliminiamo l'ultimo simbolo dalla pila e continuiamo la lettura. Se invece l'ultimo simbolo nella pila non è a, l'automa rifiuta la stringa.

Formalmente, l'automa a pila M = (Q, Σ, Γ, δ, q0, Z0, F) è definito come segue:

Q: insieme degli stati, dove q0 è lo stato iniziale

Σ: alfabeto di input

Γ: alfabeto dello stack

δ: funzione di transizione, dove δ(q, a, Z) = {(p, XZ') | (p, X) ∈ δ(q, a, Z)} per ogni stato q, simbolo di input a ∈ Σ, e simbolo dello stack Z ∈ Γ

δ(q, a, a) = {(q, ε)}

δ(q, b, a) = {(q, ε)}

δ(q, b, b) = {(q, b)}

δ(q, ε, Z) = {(qf, Z)} per ogni Z ∈ Γ

q0: stato iniziale

Z0: simbolo iniziale dello stack

F: insieme degli stati finali, dove qf è lo stato finale

In altre parole, l'automa ha uno stato q0 iniziale e uno stato finale qf. L'automa legge i simboli dell'input in modo ordinario, e ogni volta che legge una occorrenza di a, inserisce il simbolo a nello stack. Quando legge una occorrenza di b, controlla se l'ultimo simbolo inserito nello stack è a; se sì, elimina il simbolo a dallo stack. Se l'ultimo simbolo inserito nello stack non è a, l'automa rifiuta la stringa. Alla fine della lettura dell'input, l'automa accetta se e solo se lo stack è vuoto.

Per dimostrare che M riconosce il linguaggio L_{a,b}^R, consideriamo una stringa w appartenente a L_{a,b}^R. Ciò significa che ogni occorrenza di a è immagazzinata in una pila e ogni occorrenza di b corrispondente elimina l'ultima occorrenza di a dalla pila.

Supponiamo di leggere w in M partendo dallo stato iniziale q0 con lo stack iniziale contenente il simbolo Z0. Ogni volta che leggiamo una occorrenza di a, inseriamo il simbolo a nello stack. Quando leggiamo una occorrenza di b, controlliamo se l'ultimo simbolo inserito nella pila è a; se sì, eliminiamo l'ultimo simbolo dalla pila e continuiamo la lettura. Alla fine della lettura di w, il simbolo nello stack sarà Z0 se e solo se tutte le occorrenze di a sono state eliminate dalle occorrenze di b corrispondenti.

Quindi, M accetta w. Inoltre, se una stringa non appartiene a L_{a,b}^R, allora l'automa rifiuta la stringa, poiché in qualche punto si troverà in uno stato in cui l'ultimo simbolo inserito nello stack non è a, ma si sta leggendo una occorrenza di b.

Pertanto, abbiamo dimostrato che M riconosce il linguaggio L_{a,b}^R, e quindi L_{a,b}^R è un linguaggio regolare.

- Sia L un linguaggio regolare e sia a una stringa. Definiamo L_a^c come l'insieme delle stringhe w tali che ogni occorrenza di a è seguita da un numero di occorrenze di a uguale a c, dove c è una costante definita a priori. Dimostrare che L_a^c è un linguaggio regolare.

Per dimostrare che L_a^c è un linguaggio regolare, possiamo utilizzare il fatto che l'intersezione tra un linguaggio regolare e un linguaggio regolare è ancora un linguaggio regolare.

In particolare, possiamo definire un linguaggio regolare L_c come l'insieme di tutte le stringhe che contengono esattamente c occorrenze di a. Questo linguaggio può essere facilmente riconosciuto da un automa a stati finiti che tiene traccia del numero di occorrenze di a lette finora.

Quindi, possiamo definire L_a^c come l'intersezione tra il linguaggio L e il linguaggio L_c. In altre parole, L_a^c è l'insieme delle stringhe che appartengono sia a L che a L_c.

Poiché L e L_c sono entrambi linguaggi regolari, l'intersezione L ∩ L_c è ancora un linguaggio regolare. Quindi, L_a^c è un linguaggio regolare, in quanto l'intersezione di due linguaggi regolari è ancora un linguaggio regolare.

Pertanto, abbiamo dimostrato che L_a^c è un linguaggio regolare.

- Sia L un linguaggio regolare e sia a una stringa. Definiamo L_a^{**} come l'insieme delle stringhe w tali che ogni occorrenza di a è contenuta all'interno di un blocco di almeno due occorrenze di a. Dimostrare che L_a^{**} è un linguaggio regolare.

Per dimostrare che L_a^(**) è un linguaggio regolare, possiamo utilizzare l'operazione di concatenazione su linguaggi regolari e l'operazione di chiusura di Kleene.

In particolare, possiamo definire il linguaggio regolare R come l'insieme di tutte le stringhe che contengono almeno due occorrenze di a, ovvero R = (aΣa)(aΣa), dove Σ è l'alfabeto di L. Questo linguaggio può essere riconosciuto da un automa a stati finiti che legge le stringhe in modo ordinario e tiene traccia del fatto che due occorrenze di a sono state lette.

Quindi, possiamo definire L_a^() come il linguaggio ottenuto concatenando R con l'operazione di chiusura di Kleene, ovvero L_a^() = R*.

Poiché R è un linguaggio regolare, l'operazione di chiusura di Kleene su R produce ancora un linguaggio regolare. Quindi, L_a^(**) è un linguaggio regolare, in quanto può essere ottenuto concatenando un linguaggio regolare con l'operazione di chiusura di Kleene.

Pertanto, abbiamo dimostrato che L_a^(**) è un linguaggio regolare.

<u>Pumping Lemma per linguaggi context-free</u>

- Dimostra che il linguaggio $L = \{a^n b^j : n \leq j^2\}$ non è context-free

Per dimostrare che il linguaggio L={a^n b^j :n≤j^2} non è context-free, possiamo utilizzare il pumping lemma per i linguaggi context-free.

Supponiamo per assurdo che L sia un linguaggio context-free. Allora, per il pumping lemma per i linguaggi context-free, esiste un intero p (dipendente dal linguaggio L) tale che ogni stringa w di L di lunghezza almeno p può essere scritta come w = uvxyz, dove:

- |vxy| ≤ p
- |vy| ≥ 1
- Per ogni i ≥ 0, uv^ixy^iz appartiene a L.

Scegliamo la stringa $w = a^p b^{p^2} \in L$. Poiché |w| = p + p^2 > p, per il pumping lemma esiste una scomposizione w = uvxyz tale che soddisfa le proprietà 1, 2 e 3.

Poiché |vxy| ≤ p, allora v e y devono essere contenuti in a^p. Inoltre, poiché |vy| ≥ 1, almeno uno dei due deve contenere almeno un simbolo a. Quindi, sia v che y possono essere scritti come a^k per qualche k ≤ p.

Consideriamo ora la stringa uv^2xy^2z. Poiché v e y contengono solo il simbolo a e non il simbolo b, allora il numero di a nella stringa aumenterà di k, ovvero diventerà a^(p+k), mentre il numero di b non cambierà, rimanendo uguale a p^2. Quindi, la nuova stringa avrà un numero di a maggiore di p^2, che viola la definizione del linguaggio L, poiché non esiste alcun numero j tale che j^2 sia maggiore di p^2.

Pertanto, la scomposizione w = uvxyz non soddisfa la proprietà 3 del pumping lemma, il che implica che il linguaggio L non è context-free.

Quindi, abbiamo dimostrato che il linguaggio L={a^n b^j :n≤j^2} non è context-free.

- Dimostra che il linguaggio $L = \{a^n b^j c^k : k = jn\}$ non è context-free

Per dimostrare che il linguaggio L={a^n b^j c^k: k=jn} non è context-free, possiamo utilizzare il pumping lemma per i linguaggi context-free.

Supponiamo per assurdo che L sia un linguaggio context-free. Allora, per il pumping lemma per i linguaggi context-free, esiste un intero p (dipendente dal linguaggio L) tale che ogni stringa w di L di lunghezza almeno p può essere scritta come w = uvxyz, dove:

- |vxy| ≤ p
- |vy| ≥ 1
- Per ogni i ≥ 0, uv^ixy^iz appartiene a L.

Scegliamo la stringa $w = a^p\, b^p\, c^{np} \in L$. Poiché $|w| = 2p + np > p$, per il pumping lemma esiste una scomposizione w = uvxyz tale che soddisfa le proprietà 1, 2 e 3.

Poiché |vxy| ≤ p, allora v e y devono essere contenuti in a^p o b^p o c^(np). Inoltre, poiché |vy| ≥ 1, almeno uno dei due deve contenere almeno un simbolo.

Consideriamo ora i possibili casi per v e y:

- v e y sono entrambi contenuti in a^p. In questo caso, uv^2xy^2z avrà un numero di a maggiore di p, violando la definizione del linguaggio L.
- v e y sono entrambi contenuti in b^p. In questo caso, uv^2xy^2z avrà un numero di b maggiore di p, il che implica che j + k non sarà più uguale a jn. Quindi, la nuova stringa non appartiene più a L.
- v e y sono entrambi contenuti in c^(np). In questo caso, uv^2xy^2z avrà un numero di c maggiore di np, violando la definizione del linguaggio L.
- v e y sono contenuti in a^p e c^(np), rispettivamente. In questo caso, uv^2xy^2z avrà un numero di a maggiore di p e un numero di c maggiore di np, violando la definizione del linguaggio L.
- v e y sono contenuti in b^p e c^(np), rispettivamente. In questo caso, uv^2xy^2z avrà un numero di b maggiore di p, il che implica che j + k non sarà più uguale a jn, e un numero di c maggiore di np, violando la definizione del linguaggio L.

Pertanto, in tutti i casi, la scomposizione w = uvxyz non soddisfa la proprietà 3 del pumping lemma, il che implica che il linguaggio L non è context-free.

Quindi, abbiamo dimostrato che il linguaggio L={a^n b^j c^k:k=jn} non è context-free.

- Dimostra che il linguaggio $L = \{a^n b^n c^j : n \leq j\}$ non è context-free

Per dimostrare che il linguaggio L={a^n b^n c^j: n≤j} non è context-free, possiamo utilizzare il pumping lemma per i linguaggi context-free.

Supponiamo per assurdo che L sia un linguaggio context-free. Allora, per il pumping lemma per i linguaggi context-free, esiste un intero p (dipendente dal linguaggio L) tale che ogni stringa w di L di lunghezza almeno p può essere scritta come w = uvxyz, dove:

- |vxy| ≤ p
- |vy| ≥ 1
- Per ogni i ≥ 0, uv^ixy^iz appartiene a L.

Scegliamo la stringa $w = a^p\, b^p\, c^{2p} \in L$. Poiché |w| = 4p > p, per il pumping lemma esiste una scomposizione w = uvxyz tale che soddisfa le proprietà 1, 2 e 3.

Poiché |vxy| ≤ p, allora v e y devono essere contenuti in a^p o b^p o c^(2p). Inoltre, poiché |vy| ≥ 1, almeno uno dei due deve contenere almeno un simbolo.

Consideriamo ora i possibili casi per v e y:

- v e y sono entrambi contenuti in a^p o b^p. In questo caso, uv^2xy^2z avrà un numero di a e un numero di b non uguali, quindi la nuova stringa non appartiene più a L.
- v è contenuto in a^p e y è contenuto in b^p. In questo caso, uv^2xy^2z avrà un numero di a e un numero di b non uguali, quindi la nuova stringa non appartiene più a L.
- v e y sono entrambi contenuti in c^(2p). In questo caso, uv^2xy^2z avrà un numero di c maggiore di 2p, violando la definizione del linguaggio L.
- v è contenuto in a^p e y è contenuto in c^(2p). In questo caso, uv^2xy^2z avrà un numero di a maggiore di p e un numero di c maggiore di 2p, violando la definizione del linguaggio L.
- v è contenuto in b^p e y è contenuto in c^(2p). In questo caso, uv^2xy^2z avrà un numero di b maggiore di p e un numero di c maggiore di 2p, violando la definizione del linguaggio L.

Pertanto, in tutti i casi, la scomposizione w = uvxyz non soddisfa la proprietà 3 del pumping lemma, il che implica che il linguaggio L non è context-free.

Quindi, abbiamo dimostrato che il linguaggio L={a^n b^n c^j: n≤j} non è context-free.

- Dimostra che il linguaggio $L = \{a^i b^j c^k d^l \mid i < j < k < l\}$ non è context-free

Per dimostrare che il linguaggio L={a^i b^j c^k d^l ┤ | i<j<k<l} non è context-free, possiamo utilizzare il pumping lemma per i linguaggi context-free.

Supponiamo per assurdo che L sia un linguaggio context-free. Allora, per il pumping lemma per i linguaggi context-free, esiste un intero p (dipendente dal linguaggio L) tale che ogni stringa w di L di lunghezza almeno p può essere scritta come w = uvxyz, dove:

- |vxy| ≤ p
- |vy| ≥ 1
- Per ogni i ≥ 0, uv^ixy^iz appartiene a L.

Scegliamo la stringa w = a^p b^(2p) c^(3p) d^(4p) ∈ L. Poiché |w| = 10p > p, per il pumping lemma esiste una scomposizione w = uvxyz tale che soddisfa le proprietà 1, 2 e 3.

Poiché |vxy| ≤ p, allora v e y devono essere contenuti in a^p o b^(2p) o c^(3p) o d^(4p). Inoltre, poiché |vy| ≥ 1, almeno uno dei due deve contenere almeno un simbolo.

Consideriamo ora i possibili casi per v e y:

- v e y sono entrambi contenuti in a^p. In questo caso, uv^2xy^2z avrà un numero di a maggiore di p e quindi i < j < k < l non sarà più soddisfatto, violando la definizione del linguaggio L.
- v e y sono entrambi contenuti in b^(2p). In questo caso, uv^2xy^2z avrà un numero di b maggiore di 2p e quindi i < j < k < l non sarà più soddisfatto, violando la definizione del linguaggio L.
- v e y sono entrambi contenuti in c^(3p). In questo caso, uv^2xy^2z avrà un numero di c maggiore di 3p e quindi i < j < k < l non sarà più soddisfatto, violando la definizione del linguaggio L.
- v e y sono entrambi contenuti in d^(4p). In questo caso, uv^2xy^2z avrà un numero di d maggiore di 4p e quindi i < j < k < l non sarà più soddisfatto, violando la definizione del linguaggio L.
- v è contenuto in a^p e y è contenuto in b^(2p). In questo caso, uv^2xy^2z avrà un numero di a maggiore di p e un numero di b maggiore di 2p, violando la definizione del linguaggio L.
- v è contenuto in b^(2p) e y è contenuto in c^(3p). In questo caso, uv^2xy^2z avrà un numero di b maggiore di 2p e un numero di c maggiore di 3p, violando la definizione del linguaggio L.

- v è contenuto in c^(3p) e y è contenuto in d^(4p). In questo caso, uv^2xy^2z avrà un numero di c maggiore di 3p e un numero di d maggiore di 4p, violando la definizione del linguaggio L.
- v è contenuto in a^p e y è contenuto in c^(3p). In questo caso, uv^2xy^2z avrà un numero di a maggiore di p e un numero di c maggiore di 3p, violando la definizione del linguaggio L.
- v è contenuto in b^(2p) e y è contenuto in d^(4p). In questo caso, uv^2xy^2z avrà un numero di b maggiore di 2p e un numero di d maggiore di 4p, violando la definizione del linguaggio L.

Pertanto, in tutti i casi, la scomposizione w = uvxyz non soddisfa la proprietà 3 del pumping lemma, il che implica che il linguaggio L non è context-free.

Quindi, abbiamo dimostrato che il linguaggio L={a^i b^j c^k d^l ┤ | i<j<k<l} nonè context-free.

Forme normali

- Si definisca il linguaggio $infix(G)$ come l'insieme di tutte le sottostringhe di tutte le stringhe generate da G che non sono né prefissi né suffissi delle stringhe generate. Dimostrare che infix(G) è anch'esso un linguaggio context-free.

Per dimostrare che infix(G) è un linguaggio context-free, possiamo costruire una grammatica context-free che genera tutte le sottostringhe di tutte le stringhe generate da G che non sono né prefissi né suffissi.

Supponiamo di avere una grammatica context-free G che genera tutte le stringhe di un linguaggio L. Possiamo costruire una nuova grammatica context-free G' che genera infix(G) come segue:

Aggiungiamo un nuovo simbolo di partenza S' e una nuova regola S' → S, dove S è il simbolo di partenza di G.

Aggiungiamo una nuova regola S' → ε, che genera la stringa vuota.

Per ogni regola A → αBβ in G, dove α e β sono stringhe di simboli di G e B è un simbolo di G diverso dal simbolo di partenza S, aggiungiamo le seguenti regole a G':

A → αBβ

A → αXβ

X → YB

Y → ZC

Z → ε

In altre parole, per ogni regola di G che sostituisce un simbolo B con una stringa di simboli, introduciamo una nuova variabile X che sostituisce la stessa stringa di simboli ma con B al centro, e poi introduciamo ulteriori variabili Y e Z che sostituiscono il primo e l'ultimo simbolo della stringa di simboli, rispettivamente. La regola Z → ε è necessaria per tenere traccia delle sottostringhe che iniziano o finiscono con il simbolo B.

Aggiungiamo una nuova regola X → B, che genera la sottostringa B.

In questo modo, la grammatica G' genera tutte le sottostringhe di tutte le stringhe generate da G che non sono né prefissi né suffissi. Infatti, ogni stringa generata da G' deve contenere almeno un simbolo X, che rappresenta una stringa di simboli di G con B al centro. Le variabili Y e Z servono a tenere traccia delle sottostringhe che iniziano o finiscono con B, rispettivamente. La regola X → B genera la sottostringa B stessa.

Quindi, abbiamo dimostrato che infix(G) è un linguaggio context-free, poiché può essere generato da una grammatica context-free.

- Si definisca il linguaggio $common - substring(G)$ come l'insieme di tutte le sottostringhe che sono presenti in almeno due stringhe generate da G. Dimostrare che common-substring(G) è anch'esso un linguaggio context-free.

Per dimostrare che common-substring(G) è un linguaggio context-free, possiamo utilizzare la tecnica della coppia di grammatiche.

Sia G una grammatica context-free che genera un linguaggio L. Costruiamo una coppia di grammatiche context-free G1 e G2 che generano rispettivamente tutte le stringhe in L che contengono una sottostringa comune e tutte le sottostringhe comuni a almeno due stringhe in L.

La grammatica G1 è simile a G, ma con alcune regole aggiuntive per generare le stringhe che contengono una sottostringa comune. In particolare, per ogni coppia di regole A → αBβ e A' → α'B'β' in G tali che B e B' sono simboli diversi e le stringhe αBβ e α'B'β' hanno una sottostringa comune, aggiungiamo una nuova regola A'' → α''B''β'' in G1, dove α'' e β'' sono stringhe di simboli di G e B'' è un nuovo simbolo di G1. Questa nuova regola sostituisce la sottostringa comune con il simbolo B''.

La grammatica G2 è costruita come segue:

Aggiungiamo un nuovo simbolo di partenza S' e una nuova regola S' → S, dove S è il simbolo di partenza di G.

Per ogni simbolo non terminale A di G, aggiungiamo una nuova variabile A' a G2.

Aggiungiamo una nuova regola S' → ε.

Per ogni regola A → αBβ di G, dove B è un simbolo di G, aggiungiamo le seguenti regole a G2:

A' → αB'β

B' → B

B' → ε

In altre parole, per ogni regola di G che sostituisce il simbolo B con una stringa di simboli, introduciamo una nuova variabile B' che sostituisce la stessa stringa di simboli, e poi introduciamo una nuova regola che permette a B' di generare anche la stringa vuota.

Aggiungiamo una nuova regola S' → A', dove A è il simbolo di partenza di G.

In questo modo, la grammatica G2 genera tutte le sottostringhe comuni a almeno due stringhe in L, inclusi i casi in cui la sottostringa comune è vuota.

Infine, il linguaggio common-substring(G) è l'insieme di tutte le stringhe generate da G2, che è una grammatica context-free. Quindi, abbiamo dimostrato che common-substring(G) è un linguaggio context-free.

- Si definisca il linguaggio $shortest - unique - prefix(G)$ come l'insieme di tutte le sottostringhe che sono il più breve prefisso univoco di una stringa generata da G.
    - In altre parole, una sottostringa x appartiene a $shortest - unique - prefix(G)$ se e solo se x è prefisso di esattamente una stringa generata da G e non è prefisso di nessun'altra stringa generata da G.
    - Dimostrare che $shortest - unique - prefix(G)$ è anch'esso un linguaggio context-free.

Per dimostrare che shortest-unique-prefix(G) è un linguaggio context-free, possiamo utilizzare la tecnica della grammatica a stati finiti (FSA).

Costruiamo un FSA che accetta tutte le stringhe di shortest-unique-prefix(G). L'idea generale è quella di costruire un automa che riconosce i prefissi di ogni stringa generata da G, e quindi selezionare solo quelli che sono unici.

L'FSA ha uno stato iniziale e uno stato finale per ogni stringa generata da G. Ogni stato rappresenta un prefisso della relativa stringa, e ogni transizione rappresenta l'aggiunta di un nuovo carattere al prefisso. In particolare, per ogni regola A → α in G, dove α è una stringa di simboli di G, aggiungiamo un arco etichettato con α dallo stato iniziale allo stato finale corrispondente alla stringa α. Inoltre, aggiungiamo un arco etichettato con ogni simbolo terminale di G da ogni stato finale a uno stato di errore.

Per selezionare solo i prefissi unici, aggiungiamo una condizione di accettazione per gli stati finali. In particolare, uno stato finale rappresenta un prefisso unico se e solo se non esiste un altro stato finale che ha lo stesso prefisso.

Infine, possiamo convertire l'FSA in una grammatica context-free. In particolare, possiamo utilizzare la tecnica della grammatica a FSA, utilizzando l'FSA costruito come guida per generare le regole della grammatica.

In ogni caso, abbiamo dimostrato che shortest-unique-prefix(G) è un linguaggio context-free, poiché può essere generato da un FSA e quindi da una grammatica context-free.

- Si definisca il linguaggio $shortest - superstring(G)$ come l'insieme di tutte le stringhe che sono la concatenazione più breve di tutte le stringhe generate da G.
    - In altre parole, una stringa w appartiene a $shortest - superstring(G)$ se e solo se w è la concatenazione di tutte le stringhe generate da G in un ordine qualsiasi e non esiste una concatenazione più breve di tutte le stringhe generate da G.
    - Dimostrare che $shortest - superstring(G)$ è anch'esso un linguaggio context-free.

Il linguaggio shortest-superstring(G) non è necessariamente un linguaggio context-free. Infatti, il problema di trovare la concatenazione più breve di un insieme di stringhe è noto come il problema di shortest superstring e è NP-hard. Ciò significa che non esiste un algoritmo efficiente per risolvere il problema in generale, e quindi non esiste una grammatica context-free che possa generare tutte le stringhe di shortest-superstring(G).

Tuttavia, se si impone una restrizione sulle lunghezze delle stringhe generate da G, è possibile costruire una grammatica context-free che genera tutte le stringhe di shortest-superstring(G) sotto questa restrizione.

In particolare, supponiamo che tutte le stringhe generate da G abbiano lunghezza massima L. Possiamo costruire una grammatica context-free G' come segue:

Aggiungiamo un nuovo simbolo di partenza S'.

Per ogni simbolo A di G, aggiungiamo una nuova variabile A' a G'.

Aggiungiamo una nuova regola S' → ε.

Per ogni regola A → α in G, dove α è una stringa di simboli di G, aggiungiamo le seguenti regole a G':

A' → α

A' → αB

B → Cα

C → Dα

D → Eα

...

L-2. (L-3) → (L-2)α

L-1. (L-2) → ε

In altre parole, per ogni regola di G che sostituisce il simbolo A con la stringa α, introduciamo una nuova variabile A' che può generare la stringa α da sola o seguita da una sequenza di simboli B, C, D, ..., (L-3) che servono da "buffer" per le stringhe successive. L'ultimo simbolo (L-2) genera la stringa α senza alcun buffer, e il simbolo (L-1) genera la stringa vuota.

In questo modo, la grammatica G' genera tutte le possibili concatenazioni di tutte le stringhe generate da G, dove ogni stringa è separata da al massimo L-2 simboli di buffer. Infatti, ogni stringa generata da G' deve essere derivata da una variabile A' di G', dove A è un simbolo di G. La regola A' → α genera la stringa α da sola, mentre le regole successive generano le stringhe successive con un numero crescente di buffer.

Poiché tutte le stringhe generate da G hanno lunghezza massima L, ogni concatenazione di queste stringhe con al massimo L-2 simboli di buffer ha lunghezza massima 2L-4. Quindi, tutte le stringhe di shortest-superstring(G) con lunghezza massima 2L-4 possono essere generate da G'. Tuttavia, non possiamo garantire che tutte le stringhe di shortest-superstring(G) con lunghezza superiore a 2L-4 possano essere generate da G', poiché il problema di shortest superstring è NP-hard.

In conclusione, se tutte le stringhe generate da G hanno lunghezza massima L, possiamo costruire una grammatica context-free che genera tutte le stringhe di shortest-superstring(G) con lunghezza massima 2L-4. Tuttavia, in generale, shortest-superstring(G) non è un linguaggio context-free.

- Si definisca il linguaggio $smallest - context - free - grammar(G)$ come l'insieme di tutte le grammatiche context-free in forma normale di Chomsky che generano lo stesso linguaggio di G e che hanno il numero minimo di regole.
    - In altre parole, si cerca la forma normale di Chomsky più compatta che genera lo stesso linguaggio di G.
    - Dimostrare $che\ smallest - context - free - grammar(G)$ è anch'esso un linguaggio context-free.

Il linguaggio smallest-context-free-grammar(G) è un linguaggio di grammatiche context-free, poiché ogni grammatica in smallest-context-free-grammar(G) è una grammatica context-free.

Per dimostrare che smallest-context-free-grammar(G) è un linguaggio context-free, possiamo utilizzare la tecnica della grammatica a FSA (finite state automaton). In particolare, costruiamo un FSA che accetta tutte le stringhe di smallest-context-free-grammar(G) in modo che ogni stato rappresenti una grammatica context-free in forma normale di Chomsky con un certo numero di regole, e ogni transizione rappresenti l'aggiunta di una regola alla grammatica.

La costruzione dell'FSA può essere effettuata in modo simile alla costruzione della grammatica a FSA. Iniziamo con un singolo stato iniziale che rappresenta la grammatica iniziale di G in forma normale di Chomsky. Per ogni regola A → α in questa grammatica, aggiungiamo un arco etichettato con A → α dallo stato iniziale a uno stato finale che rappresenta la grammatica con una regola in più. In questo modo, generiamo tutte le grammatiche context-free in forma normale di Chomsky che generano lo stesso linguaggio di G.

Inoltre, possiamo mantenere un insieme di tutte le grammatiche context-free in forma normale di Chomsky che sono state raggiunte fino a quel momento, in modo da evitare di generare grammatiche duplicate. In questo modo, l'FSA accetta solo le grammatiche context-free in forma normale di Chomsky che hanno il numero minimo di regole.

Infine, possiamo convertire l'FSA in una grammatica context-free utilizzando la tecnica della grammatica a FSA.

In conclusione, abbiamo mostrato che smallest-context-free-grammar(G) è un linguaggio di grammatiche context-free, poiché può essere generato da un FSA e quindi da una grammatica context-free.

<u>Varie</u>

- Il linguaggio $L = \{a^n w w^R a^n : n \geq 0, w \in \{a,b\}^*$ è context-free o non lo è? Motivare la risposta.

Il linguaggio L={a^n ww^R a^n:n≥0,w∈{a,b}^*} è un linguaggio context-free.

Una possibile grammatica context-free che genera il linguaggio L è la seguente:

S → ε | aSa | bSb | A

A → aBaa | bBbb

B → ε | aB | bB

La produzione S → ε consente di generare la stringa vuota, che appartiene a L per n=0 e w=ε. Le produzioni aSa e bSb generano le stringhe a^n w w^R a^n e b^n w w^R b^n, rispettivamente, dove la sottostringa w può essere generata ricorsivamente dalla produzione A. La produzione A → aBaa | bBbb genera tutte le possibili stringhe w in {a,b}^*, mentre la produzione B → ε | aB | bB genera tutte le possibili lunghezze per la sottostringa w.

In questo modo, la grammatica genera tutte le stringhe del linguaggio L. In particolare, la produzione S → aSa e S → bSb consentono di generare stringhe con un numero arbitrario di simboli a e b all'inizio e alla fine della sottostringa w, mentre la produzione A → aBaa e A → bBbb consentono di generare la sottostringa w in modo che sia uguale alla sua inversione w^R.

In conclusione, il linguaggio L={a^n ww^R a^n:n≥0,w∈{a,b}^*} è un linguaggio context-free, poiché può essere generato da una grammatica context-free.

- Il linguaggio $L = \{a^n b^j a^n b^j : n, j \geq 0\}$ è context-free o non lo è? Motivare la risposta.

Il linguaggio L={a^n b^j a^n b^j :n,j≥0} non è un linguaggio context-free.

Per dimostrare che L non è context-free, possiamo utilizzare il pumping lemma per i linguaggi context-free.

Supponiamo per assurdo che L sia un linguaggio context-free. Allora, per il pumping lemma, esiste un intero p (dipendente dal linguaggio L) tale che ogni stringa z = a^n b^j a^n b^j con |z| ≥ p può essere scritta come z = uvwxy, dove:

- |vwx| ≤ p

- |vx| ≥ 1
- per ogni i ≥ 0, uv^iwx^iy ∈ L

Consideriamo la stringa $z = a^p\ b^p\ a^p\ b^p$. Poiché |z| = 4p ≥ p, per il pumping lemma esiste una scomposizione z = uvwxy che soddisfa le proprietà sopra elencate.

Osserviamo che la sottostringa vx deve essere composta solo da a o solo da b, perché altrimenti, se vx contiene sia a che b, la scomposizione può essere fatta in modo che v e x si sovrappongano tra la prima metà di z e la seconda metà di z, generando una stringa non appartenente a L. Quindi, supponiamo senza perdita di generalità che vx sia composta solo da a.

Poiché |vwx| ≤ p, la sottostringa vx è composta da a^k per qualche k tale che 1 ≤ k ≤ p. Inoltre, poiché |vx| ≥ 1, abbiamo k ≥ 1.

Consideriamo ora la stringa $uv^2wx^2y = a^{p+(k-1)}\ b^p\ a^{p+(k-1)}\ b^p$. Poiché k ≥ 1, abbiamo p+(k-1) > p, quindi la lunghezza della prima metà della stringa a^(p+(k-1)) b^p è maggiore della lunghezza della seconda metà. Pertanto, la stringa non può appartenere a L, in quanto il numero di a nella prima metà della stringa è maggiore del numero di a nella seconda metà.

Poiché abbiamo ottenuto una contraddizione, possiamo concludere che l'assunzione iniziale che L sia un linguaggio context-free è falsa. Pertanto, il linguaggio L={a^n b^j a^n b^j :n,j≥0} non è un linguaggio context-free.

- Il linguaggio $L = \{0^p \mid p$ è un numero primo e in base 2 il numero $p - 1$ è divisibile per 3$\}$ è regolare? Motivare la risposta.

Il linguaggio L={0^p | p è un numero primo e in base 2 il numero p-1 è divisibile per 3} non è un linguaggio regolare.

Per dimostrare che L non è regolare, possiamo utilizzare il pumping lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia un linguaggio regolare. Allora, per il pumping lemma, esiste un intero p (dipendente dal linguaggio L) tale che ogni stringa z in L con |z| ≥ p può essere scritta come z = xyz, dove:

- |xy| ≤ p
- |y| ≥ 1
- per ogni i ≥ 0, xy^iz ∈ L

Consideriamo la stringa z = 0^(p), dove p è un numero primo che in base 2 il numero p-1 è divisibile per 3 (ad esempio, p=7, perché 7-1=6=2*3). Poiché |z| = p ≥ p, per il pumping lemma esiste una scomposizione z = xyz che soddisfa le proprietà sopra elencate.

Osserviamo che la sottostringa y deve consistere solo di zeri, perché altrimenti, se y contiene un uno, la scomposizione può essere fatta in modo che y si sovrapponga tra la fine della prima metà di z e l'inizio della seconda metà di z, generando una stringa non appartenente a L.

Poiché y consiste solo di zeri e |xy| ≤ p, la parte rimanente di z (cioè la stringa xz) consiste di 0^(p-k) seguita da una sequenza finita di cifre binarie che non contengono due cifre consecutive uguali. In particolare, la parte rimanente di z può essere vista come la rappresentazione binaria di un numero primo q = p-k.

Ora, consideriamo la stringa xy^2z. Poiché y consiste solo di zeri, xy^2z consiste di 0^(p+jy) seguito dalla rappresentazione binaria di un numero primo q', dove j è il numero di zeri aggiunti alla sottostringa y e q' = q+j. Tuttavia, poiché q è un numero primo, non è divisibile per 3 in base 2. Quindi, se scegliamo j in modo che q+j sia divisibile per 3 in base 2, la stringa xy^2z non appartiene a L.

Ma questo è possibile, poiché esistono numeri primi q che soddisfano questa proprietà. Ad esempio, se q=7, abbiamo q-1=6=110_2 e q è divisibile per 3 in base 2, quindi scegliendo j=2, abbiamo q+j=9=1001_2, che è divisibile per 3 in base 2.

In conclusione, abbiamo ottenuto una contraddizione, poiché abbiamo trovato una stringa xy^2z che non appartiene a L, mentre il pumping lemma afferma che ogni stringa xy^iz con i ≥ 0 deve appartenere a L. Quindi, l'assunzione iniziale che L sia un linguaggio regolare è falsa. Pertanto, il linguaggio L={0^p | p è un numero primo e in base 2 il numero p-1 è divisibile per 3} non è un linguaggio regolare.

- Il linguaggio $L = \{w \in \{a, b^* \mid$ il numero di sottostringhe "aa" seguite da un numero pari di "b" in w è uguale al numero di sottostringhe "bb" seguite da un numero dispari di "a"} è regolare? Motivare la risposta.

Il linguaggio L = {w ∈ {a,b}^* | il numero di sottostringhe "aa" seguite da un numero pari di "b" in w è uguale al numero di sottostringhe "bb" seguite da un numero dispari di "a"} non è un linguaggio regolare.

Per dimostrare che L non è regolare, possiamo utilizzare il pumping lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia un linguaggio regolare. Allora, per il pumping lemma, esiste un intero p (dipendente dal linguaggio L) tale che ogni stringa z in L con |z| ≥ p può essere scritta come z = xyz, dove:

- |xy| ≤ p
- |y| ≥ 1
- per ogni i ≥ 0, xy^iz ∈ L

Consideriamo la stringa $z = (ab)^p \, a^{p+1} \, b^{2p}$. Poiché $|z| = 5p + 1 \geq p$, per il pumping lemma esiste una scomposizione z = xyz che soddisfa le proprietà sopra elencate.

Osserviamo che la sottostringa y deve consistere solo di "a" o solo di "b", perché altrimenti, se y contiene sia "a" che "b", la scomposizione può essere fatta in modo che y si sovrapponga tra una sottostringa "aa" seguita da un numero pari di "b" e una sottostringa "bb" seguita da un numero dispari di "a", generando una stringa non appartenente a L.

Supponiamo senza perdita di generalità che y consista solo di "a". Poiché |xy| ≤ p, la parte rimanente di z (cioè la stringa xz) consiste di un numero finito di "a" e una sequenza di "b" di lunghezza 2p. In particolare, la parte rimanente di z contiene esattamente p-1 sottostringhe "aa" seguite da un numero pari di "b".

Consideriamo ora la stringa xy^2z. Poiché y consiste solo di "a", xy^2z contiene una sottostringa "aa" seguita da un numero pari di "b" in più rispetto a z, ovvero contiene p sottostringhe "aa" seguite da un numero pari di "b". Inoltre, xy^2z contiene ancora esattamente p-1 sottostringhe "bb" seguite da un numero dispari di "a". Pertanto, xy^2z non appartiene a L, in quanto il numero di sottostringhe "aa" seguite da un numero pari di "b" è diverso dal numero di sottostringhe "bb" seguite da un numero dispari di "a".

Poiché abbiamo ottenuto una contraddizione, possiamo concludere che l'assunzione iniziale che L sia un linguaggio regolare è falsa. Pertanto, il linguaggio L = {w ∈ {a,b}^* | il numero di sottostringhe "aa" seguite da un numero pari di "b" in w è uguale al numero di sottostringhe "bb" seguite da un numero dispari di "a"} non è un linguaggio regolare.

- Il linguaggio $L = \{w \in \{a, b\}^* \mid w$ ha lo stesso numero di occorrenze di “ab” e “ba”} è regolare? Motivare la risposta

Il linguaggio L = {w ∈ {a,b}^* | w ha lo stesso numero di occorrenze di "ab" e "ba"} non è un linguaggio regolare.

Per dimostrare che L non è regolare, possiamo utilizzare il pumping lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia un linguaggio regolare. Allora, per il pumping lemma, esiste un intero p (dipendente dal linguaggio L) tale che ogni stringa z in L con |z| ≥ p può essere scritta come z = xyz, dove:

1. |xy| ≤ p

2. |y| ≥ 1

3. per ogni i ≥ 0, xy^iz ∈ L

Consideriamo la stringa $z = (ab)^p (ba)^p$. Poiché |z| = 4p ≥ p, per il pumping lemma esiste una scomposizione z = xyz che soddisfa le proprietà sopra elencate.

Osserviamo che la sottostringa y deve consistere solo di "a" o solo di "b", perché altrimenti, se y contiene sia "a" che "b", la scomposizione può essere fatta in modo che y si sovrapponga tra una sottostringa "ab" e una sottostringa "ba", generando una stringa non appartenente a L.

Supponiamo senza perdita di generalità che y consista solo di "a". Poiché |xy| ≤ p, la parte rimanente di z (cioè la stringa xz) consiste di un numero finito di "a" e una sequenza di "b" di lunghezza p seguita da una sequenza di "a" di lunghezza p. In particolare, la parte rimanente di z contiene esattamente p sottostringhe "ab" e p sottostringhe "ba".

Consideriamo ora la stringa xy^2z. Poiché y consiste solo di "a", xy^2z contiene una sottostringa "ab" in più rispetto a z e non contiene una sottostringa "ba" in più rispetto a z. Pertanto, xy^2z non appartiene a L, in quanto il numero di occorrenze di "ab" e "ba" in xy^2z non è uguale.

Poiché abbiamo ottenuto una contraddizione, possiamo concludere che l'assunzione iniziale che L sia un linguaggio regolare è falsa. Pertanto, il linguaggio L = {w ∈ {a,b}^* | w ha lo stesso numero di occorrenze di "ab" e "ba"} non è un linguaggio regolare.

Pumping Lemma

- Dimostra che il linguaggio $L = \{w \in \{0,1\}* \mid$ w è una stringa palindroma di lunghezza dispari e il numero di 0 in w è uguale al numero di 1} non è regolare.

Assumiamo per assurdo che L sia un linguaggio regolare. Allora, per il pumping lemma, esiste un intero positivo p tale che ogni stringa w ∈ L di lunghezza almeno p può essere scritta come w = xyz, dove y non è vuota e |xy| ≤ p, tale che xy^iz ∈ L per ogni i ≥ 0.

Scegliamo una stringa $w = 0^{p+1}\,1^p 0^{p+1} \in L$ di lunghezza 2p+2 > p. Per il pumping lemma, possiamo scrivere w come w = xyz, dove y non è vuota e |xy| ≤ p, tale che xy^iz ∈ L per ogni i ≥ 0. In particolare, y contiene solo 0 o solo 1. Poiché w è una stringa palindroma di lunghezza dispari, il primo e l'ultimo carattere di w devono essere diversi, cioè w[1] ≠ w[2p+2]. Ma poiché xy^iz ∈ L per ogni i ≥ 0, deve essere che xy contiene solo 0 o solo 1. Pertanto, se y contiene solo 0, allora la stringa xy^2z ha un numero dispari di 0 e un numero pari di 1, e quindi non appartiene a L. Allo stesso modo, se y contiene solo 1, allora la stringa xy^2z ha un numero pari di 0 e un numero dispari di 1, e quindi non appartiene a L. In entrambi i casi, abbiamo ottenuto una contraddizione. Pertanto, L non può essere un linguaggio regolare.

- Dimostra che il linguaggio $L = \{a^n b^m \mid n, m > 0$ e n è divisibile per m} non è regolare.

Per dimostrare che il linguaggio L = {a^n b^m | n,m > 0 e n è divisibile per m} non è regolare, useremo il pumping lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia regolare. Allora, per il pumping lemma, esiste un intero positivo p tale che ogni stringa w ∈ L con |w| ≥ p può essere scritta come w = xyz, dove:

|xy| ≤ p;

|y| > 0;

xy^iz ∈ L per ogni intero non negativo i.

Consideriamo la stringa $w = a^p\, b^{p-1} \in L$. Per il pumping lemma, esistono stringhe x, y e z tali che w = xyz, |xy| ≤ p e |y| > 0, e xy^iz ∈ L per ogni i non negativo.

Poiché |xy| ≤ p, la stringa y consiste solo di a. Pertanto, xy^2z ha la forma $a^{p+k}\, b^{p-1}$ con k > 0, poiché |y| > 0. Tuttavia, k non può essere divisibile per p-1, poiché in caso contrario, avremmo che n = p+k è divisibile per m = p-1, il che contraddice la definizione di L.

Pertanto, abbiamo trovato una stringa w ∈ L tale che xy^2z ∉ L, il che contraddice il pumping lemma. Concludiamo quindi che L non è un linguaggio regolare, come volevamo dimostrare.

- Dimostra che il linguaggio $L = \{1^k y \,|y \in \{0,1\}$ e y contiene al più k uni, per $k \geq 1$ non è regolare.

Per dimostrare che il linguaggio L = {1^k y | y ∈ {0,1}* e y contiene al più k uni, per k ≥ 1} non è regolare, useremo il pumping lemma per i linguaggi regolari.

Supponiamo per assurdo che L sia regolare. Allora, per il pumping lemma, esiste un intero positivo p tale che ogni stringa w ∈ L con |w| ≥ p può essere scritta come w = xyz, dove:

|xy| ≤ p;

|y| > 0;

xy^iz ∈ L per ogni intero non negativo i.

Consideriamo la stringa w = 1^p 0^(p-1) ∈ L. Per il pumping lemma, esistono stringhe x, y e z tali che w = xyz, |xy| ≤ p e |y| > 0, e xy^iz ∈ L per ogni i non negativo.

Poiché |xy| ≤ p, la stringa y consiste solo di 1. Pertanto, xy^2z ha la forma 1^(p+k) 0^(p-1) con k > 0, poiché |y| > 0. Tuttavia, se scegliamo y = 1^q con q > k, allora xy^2z contiene più di k uni, il che viola la definizione di L.

Pertanto, abbiamo trovato una stringa w ∈ L tale che xy^2z ∉ L, il che contraddice il pumping lemma. Concludiamo quindi che L non è un linguaggio regolare, come volevamo dimostrare.

- Sia $\Sigma = \{0,1,+,=\}$ e $ADD = \{x = y + z \mid x, y, z$ sono interi binari e x è la somma di y e di $z\}$.

  Dimostra che ADD non è regolare.

Per dimostrare che ADD non è un linguaggio regolare, useremo il pumping lemma per i linguaggi regolari.

Supponiamo per assurdo che ADD sia un linguaggio regolare. Allora, per il pumping lemma, esiste un intero positivo p tale che ogni stringa w ∈ ADD con |w| ≥ p può essere scritta come w = xyz, dove:

- |xy| ≤ p;

- |y| > 0;

- xy^iz ∈ ADD per ogni intero non negativo i.

Consideriamo la stringa $w = 1^p = 0^p + 1^p \in ADD$, dove il simbolo + rappresenta la somma binaria. Poiché |w| = 3p > p, esistono stringhe x, y e z tali che w = xyz, |xy| ≤ p e |y| > 0, e xy^iz ∈ ADD per ogni i non negativo.

Poiché |xy| ≤ p, la stringa y consiste solo di 1 o solo di 0, o contiene solo simboli + o solo simboli =, o contiene una combinazione di questi simboli. Esaminiamo ciascun caso:

- Se y contiene solo 1, allora xy^2z non è una somma binaria. Infatti, la stringa y contiene solo 1, quindi xy^2z ha più di p+1 cifre 1, il che implica che non può essere una somma binaria di due numeri binari di lunghezza p o meno.

- Se y contiene solo 0, allora xy^2z non è una somma binaria. Infatti, la stringa xy contiene solo 1^k per qualche k ≤ p, quindi xy^2z ha almeno p+1 cifre 1, il che implica che non può essere una somma binaria di due numeri binari di lunghezza p o meno.

- Se y contiene solo + o solo =, allora xy^2z non è una stringa ben formata. Infatti, la stringa y contiene solo + o solo =, quindi xy^2z contiene più di un simbolo + o più di un simbolo =, il che viola la sintassi delle somme binarie.

In tutti questi casi, abbiamo trovato una stringa xy^2z che non appartiene ad ADD, il che contraddice il pumping lemma. Pertanto, ADD non è un linguaggio regolare, come volevamo dimostrare.

- Se A è un insieme di numeri naturali e k è un numero naturale maggiore di 1, sia Bk(A) = {w| w è la rappresentazione in base k di qualche numero in A}.

  In questo caso, non sono ammessi gli 0 iniziali nella rappresentazione di un numero. Ad esempio, B2({3, 5}) = {11, 101} e B3({3, 5}) = {10, 12}. Fornite un esempio di un insieme A per il quale B2(A) è regolare, ma B3(A) non è regolare. Dimostrate che il vostro esempio funziona.

Consideriamo l'insieme A = {2, 6}. La rappresentazione binaria di 2 è 10 e la rappresentazione binaria di 6 è 110. Pertanto, B2(A) = {10, 110}, che è un insieme regolare poiché consiste solo di stringhe di lunghezza fissa.

D'altra parte, la rappresentazione ternaria di 2 è 2 e la rappresentazione ternaria di 6 è 20. Pertanto, B3(A) = {2, 20}, che non è regolare poiché entrambe le stringhe in B3(A) hanno lunghezza diversa.

Per dimostrare che B3(A) non è regolare, supponiamo per assurdo che sia regolare. Allora, per il pumping lemma per i linguaggi regolari, esiste un intero positivo p tale che ogni stringa w ∈ B3(A) con |w| ≥ p può essere scritta come w = xyz, dove:

- |xy| ≤ p;

- |y| > 0;

- xy^iz ∈ B3(A) per ogni intero non negativo i.

Consideriamo la stringa w = 20 ∈ B3(A). Poiché |w| = 2 ≥ p, esistono stringhe x, y e z tali che w = xyz, |xy| ≤ p e |y| > 0, e xy^iz ∈ B3(A) per ogni i non negativo.

Poiché B3(A) non contiene stringhe di lunghezza 1, la stringa y ha lunghezza maggiore di 1. Inoltre, poiché B3(A) non contiene stringhe con 0 iniziali, la stringa x non può contenere 0. Pertanto, x e y devono consistere solo di cifre 1 e 2. Inoltre, se y contiene almeno una cifra 1, allora xy^2z contiene almeno due cifre 1 e quindi non può essere in B3(A). Pertanto, y deve contenere solo cifre 2.

Ma allora, la stringa xy^2z ha lunghezza 2p, che è diversa dalla lunghezza di w, il che implica che xy^2z non può essere una stringa in B3(A). Pertanto, abbiamo trovato una stringa w ∈ B3(A) tale che xy^2z ∉ B3(A), il che contraddice il pumping lemma. Concludiamo quindi che B3(A) non è un linguaggio regolare, come volevamo dimostrare.

<u>Linguaggi regolari</u>

- Un all-NFA M è una $5-tupla\ (Q,\Sigma,\delta,q0,F)$ che accetta $x\ \in\ \Sigma*$ se ogni possibilestato in cui M potrebbe trovarsi dopo aver letto l'input x è uno stato di F. Si noti, al contrario, che un NFA ordinario accetta una stringa se qualche stato tra questi possibili stati è uno stato distato di accettazione. Dimostrare che tutte le NFA riconoscono la classe dei linguaggi regolari.

Per dimostrare che tutte le NFA riconoscono la classe dei linguaggi regolari, dobbiamo dimostrare che ogni linguaggio regolare può essere accettato da un all-NFA.

Sia L un linguaggio regolare. Esiste quindi un DFA D = (Q, Σ, δ, q0, F) che accetta L. Consideriamo l'all-NFA M = (Q, Σ, δ', q0, F), dove:

δ'(q, a) = {δ(q, a)} per ogni q ∈ Q e a ∈ Σ. In altre parole, il nuovo stato raggiunto da q leggendo a è l'insieme dei singoli stati raggiunti da q leggendo a secondo la funzione di transizione di D.

M dimostra di accettare L. Infatti, per ogni stringa x ∈ Σ*, se D accetta x, allora M accetta x in quanto ogni possibile stato di M dopo aver letto x appartiene a F. Al contrario, se D non accetta x, allora non esiste un percorso da q0 a uno stato di accettazione in D, il che implica che non esiste un percorso da q0 a uno stato di F in M dopo aver letto x.

Pertanto, abbiamo dimostrato che ogni linguaggio regolare può essere accettato da un all-NFA, il che dimostra che tutte le NFA riconoscono la classe dei linguaggi regolari.

- Siano B e C linguaggi in $\Sigma=\{0,1\}$. Si definisca

  $B\ \leftarrow^1\ C$ = {$w\ \in B$| per qualche $y \in C$, le stringhe w e y contengono un numero uguale di 1}.

  Dimostrare che la classe dei linguaggi regolari è chiusa sotto l'operazione 1←.

Per dimostrare che la classe dei linguaggi regolari è chiusa sotto l'operazione 1 ←, dobbiamo dimostrare che, dato un linguaggio regolare B e un linguaggio regolare C, il linguaggio B ←^1 C è anch'esso regolare.

Sia D = (Q,Σ,δ,q0,F) un DFA che accetta il linguaggio B, e sia E = (P,Σ,λ,p0,G) un DFA che accetta il linguaggio C. Consideriamo il DFA F = (Q × P, Σ, δ', (q0, p0), F × G), dove:

δ'((q, p), a) = (δ(q, a), λ(p, a)) per ogni q ∈ Q, p ∈ P e a ∈ Σ. In altre parole, il nuovo stato raggiunto da (q, p) leggendo a è la coppia dei nuovi stati raggiunti da q e p leggendo a secondo le funzioni di transizione di D ed E.

F dimostra di accettare B ←^1 C. Infatti, per ogni stringa w ∈ B ←^1 C, esiste una stringa y ∈ C tale che w e y contengono lo stesso numero di 1. Sia n il numero di 1 in w e y. Allora, possiamo costruire una sequenza di n caratteri a1, a2, ..., an tali che:

se ai = 1, allora w e y contengono entrambe almeno un 1 in posizione i;

se ai = 0, allora w e y contengono entrambe almeno uno 0 in posizione i.

Sia (q, p) lo stato di F raggiunto leggendo la stringa a1a2...an. Poiché D accetta B e E accetta C, esiste una stringa x tale che δ(q0, x) ∈ F e λ(p0, x) ∈ G. Inoltre, δ(q, ai) ∈ F e λ(p, ai) ∈ G per ogni i = 1, ..., n. Pertanto, δ'(q, p) = (δ(q, a1), λ(p, a1)) ∈ F × G, δ'(δ'(q, p), a2) = (δ(δ(q, a1), a2), λ(λ(p, a1), a2)) ∈ F × G, e così via fino a δ'(δ'(δ'(...(q, p), an), a(n+1)) = (δ(δ(δ'(...(q, an), a(n-1)), ...), a1), λ(λ(λ'(...(p, an), a(n-1)), ...), a1)) ∈ F × G.

Pertanto, abbiamo dimostrato che il linguaggio B ←^1 C è accettato dal DFA F, il che dimostra che la classe dei linguaggi regolari è chiusa sotto l'operazione 1 ←.

- Se A è un linguaggio qualsiasi, sia $A_{\frac{1}{2}\_}$ l'insieme di tutte le mezze stringhe di A tali che $A_{\frac{1}{2}\_} = \{x \mid \text{per qualche } y, |x| = |y| \text{ e } xy \in A\}$.
  Dimostra che se A è regolare, anche $A_{\frac{1}{2}\_}$ è regolare

Per dimostrare che se A è un linguaggio regolare, anche A_(1/2-) è regolare, costruiamo un automa finito non deterministico (NFA) per A_(1/2-) a partire dall'automa finito deterministico (DFA) per A.

L'automa per A ha un insieme di stati Q, un alfabeto Σ, una funzione di transizione δ : Q × Σ → Q e uno stato iniziale q0 ∈ Q e un insieme di stati finali F ⊆ Q.

Costruiamo l'automa per A_(1/2-) come segue. Gli stati dell'automa per A_(1/2-) sono coppie di stati dell'automa per A, ovvero Q_1/2- = Q × Q. Lo stato iniziale dell'automa per A_(1/2-) è (q0, q0), ovvero la coppia di stati iniziali dell'automa per A. Uno stato (q, p) ∈ Q_1/2- è finale se e solo se p è uno stato finale dell'automa per A.

La funzione di transizione dell'automa per A_(1/2-) è definita come segue. Per ogni coppia di stati (q, p) ∈ Q_1/2- e ogni simbolo a ∈ Σ, definiamo δ_1/2-((q, p), a) come la coppia di stati (δ(q, a), δ(p, a)).

Infine, definiamo la funzione di transizione estesa δ_1/2- : Q_1/2- × Σ* → Q_1/2- come segue. Per ogni coppia di stati (q, p) ∈ Q_1/2- e ogni stringa w ∈ Σ*, definiamo δ_1/2-((q, p), w) come segue:

- Se w = ε, allora δ_1/2-((q, p), w) = (q, p).

- Se w = xa per qualche simbolo a ∈ Σ e stringa x ∈ Σ*, allora δ_1/2-((q, p), w) = δ_1/2-((q, p), x)a.

L'automa per A_(1/2-) accetta una stringa w se e solo se esiste una stringa x tale che |x| = |w|/2 e tale che xw ∈ A. In altre parole, l'automa per A_(1/2-) accetta una coppia di stringhe (x, y) se e solo se xy ∈ A e |x| = |y| = |xy|/2.

Pertanto, abbiamo costruito un automa finito non deterministico per A_(1/2-) a partire dall'automa finito deterministico per A, il che dimostra che se A è un linguaggio regolare, anche A_(1/2-) è regolare.

- Definiamo l'operazione $evita$ per i linguaggi A e B come segue

  $A\ evita\ B\ = \{w|\ w\ \in\ A$ e w non contiene alcuna stringa in B come sottostringa$\}$.

  Dimostrare che la classe dei linguaggi regolari è chiusa sotto l'operazione $evita$.

Per dimostrare che la classe dei linguaggi regolari è chiusa sotto l'operazione evita, supponiamo che A e B siano due linguaggi regolari. Dobbiamo dimostrare che A evita B è anche un linguaggio regolare.

Per dimostrare ciò, costruiamo un automa finito deterministico (DFA) per A evita B. L'automa avrà gli stessi stati dell'automa per A, ma con alcune modifiche alla funzione di transizione.

In particolare, dobbiamo assicurarci che l'automa rifiuti tutte le stringhe che contengono una sottostringa in B. Per fare ciò, creiamo un nuovo stato di rifiuto q_r, che diventa lo stato finale dell'automa. Aggiungiamo transizioni da tutti gli stati di A a q_r per ogni simbolo in B. In altre parole, se uno stato di A legge un simbolo in B, l'automa passa immediatamente allo stato di rifiuto.

Inoltre, dobbiamo assicurarci che l'automa accetti solo le stringhe che non contengono alcuna sottostringa in B. Per fare ciò, manteniamo gli stati finali di A e assicuriamoci che l'automa non si muova mai nello stato di rifiuto. In particolare, se lo stato corrente dell'automa è uno stato finale di A e legge un simbolo in B, l'automa rimane nello stato finale di A invece di passare allo stato di rifiuto.

In questo modo, l'automa per A evita B accetta solo le stringhe in A che non contengono alcuna sottostringa in B, come richiesto. Pertanto, abbiamo dimostrato che la classe dei linguaggi regolari è chiusa sotto l'operazione evita.

- Sia L un linguaggio su un alfabeto Σ. Si definisce il linguaggio $CHAIN$ di L con sé stesso n volte, indicato con L^n, come l'insieme di tutte le stringhe che possono essere ottenute concatenando n copie di una stringa in L.
  - Dimostrare che se L è un linguaggio regolare, allora per ogni intero positivo n, il linguaggio L^n è anch'esso un linguaggio regolare.

Per dimostrare che L^n è un linguaggio regolare, possiamo utilizzare il teorema di Kleene che afferma che un linguaggio L è regolare se e solo se può essere riconosciuto da un automa a stati finiti (AFD).

Poiché L è un linguaggio regolare, esiste un AFD M che lo riconosce. L'idea è di costruire un nuovo AFD N che riconosce L^n.

La costruzione di N avviene per induzione su n.

Per il caso base, n = 1, l'AFD N è semplicemente una copia dell'AFD M.

Per il passo induttivo, supponiamo di avere un AFD N che riconosce L^(n-1). Per costruire l'AFD N' che riconosce L^n, possiamo utilizzare l'AFD N come base e aggiungere una copia di M per riconoscere una copia aggiuntiva di L.

In particolare, l'AFD N' ha lo stesso numero di stati di N moltiplicato per il numero di stati di M. Gli stati di N' sono coppie (q, p), dove q è uno stato di N e p è uno stato di M.

L'idea è di utilizzare gli stati di N per tenere traccia delle prime (n-1) copie di L, e gli stati di M per tenere traccia dell'ultima copia di L. In questo modo, possiamo riconoscere tutte le stringhe in L^n.

Inizialmente, l'AFD N' si trova nello stato iniziale (q0, p0), dove q0 è lo stato iniziale di N e p0 è lo stato iniziale di M.

Per ogni transizione (q, a, q') in N, aggiungiamo un gruppo di transizioni da (q, p) a (q', p) per ogni stato p in M che accetta la lettera a.

Inoltre, per ogni stato finale di N, aggiungiamo un arco di transizione epsilon da (q, p) a (q0, p0) per ogni stato p in M.

Infine, gli stati finali di N' sono tutte le coppie (q, p) dove q è uno stato finale di N e p è uno stato finale di M.

In questo modo, l'AFD N' riconosce il linguaggio L^n, dimostrando che è anch'esso un linguaggio regolare.

- Sia L un linguaggio su un alfabeto Σ. Si definisce il linguaggio $PROJECTION$ su una posizione specifica k come l'insieme di tutte le stringhe in L che hanno una lettera k-esima specifica. Dimostrare che se L è un linguaggio regolare, allora per ogni intero positivo k, il linguaggio $PROJECTION$ su k di L è anch'esso un linguaggio regolare.

Per dimostrare che il linguaggio PROJECTION su k di L è regolare, possiamo utilizzare il teorema di Kleene che afferma che un linguaggio L è regolare se e solo se può essere riconosciuto da un automa a stati finiti (AFD).

Sia M un AFD che riconosce L. L'idea è di costruire un nuovo AFD N che riconosce il linguaggio PROJECTION su k di L.

L'AFD N ha lo stesso insieme di stati di M, ma con alcune modifiche alle transizioni e agli stati finali.

In particolare, per ogni stato q di M, creiamo uno stato q' in N che rappresenta lo stato di M dopo aver letto le prime k lettere di una stringa. Ogni transizione (q, a, q') in M viene mappata a una transizione (q, a, q') in N solo se q' rappresenta lo stato di M dopo aver letto le prime k-1 lettere e la k-esima lettera è a.

Inoltre, uno stato q in N è finale se e solo se lo stato corrispondente q'' in M è finale.

Infine, lo stato iniziale di N è lo stato corrispondente allo stato iniziale di M dopo aver letto le prime k-1 lettere.

In questo modo, l'AFD N riconosce il linguaggio PROJECTION su k di L, dimostrando che è anch'esso un linguaggio regolare.

- Sia L un linguaggio su un alfabeto Σ. Si definisce il linguaggio $CONJUGATE$ di L come l'insieme di tutte le stringhe che possono essere ottenute invertendo l'ordine delle lettere di una stringa in L e sostituendo ogni lettera con la sua complementare.
    - In altre parole, se x è una stringa in L, allora il suo coniugato x' è ottenuto sostituendo ogni lettera di x con la sua complementare e invertendo l'ordine delle lettere.
    - Dimostrare che se L è un linguaggio regolare, allora il linguaggio $CONJUGATE$ di L è anch'esso un linguaggio regolare.

Per dimostrare che il linguaggio CONJUGATE di L è regolare, possiamo utilizzare il teorema di Kleene che afferma che un linguaggio L è regolare se e solo se può essere riconosciuto da un automa a stati finiti (AFD).

Sia M un AFD che riconosce L. L'idea è di costruire un nuovo AFD N che riconosce il linguaggio CONJUGATE di L.

L'AFD N ha gli stessi stati di M, ma con alcune modifiche alle transizioni e agli stati finali.

In particolare, per ogni stato q di M, creiamo uno stato q' in N che rappresenta lo stato di M dopo aver letto la stringa in ordine inverso. Ogni transizione (q, a, q') in M viene mappata a una transizione (q', a', q) in N, dove a' è la complementare di a.

Inoltre, uno stato q in N è finale se e solo se lo stato corrispondente q'' in M è finale.

Infine, lo stato iniziale di N è lo stato corrispondente allo stato finale di M.

In questo modo, l'AFD N riconosce il linguaggio CONJUGATE di L, dimostrando che è anch'esso un linguaggio regolare.

Pumping Lemma per linguaggi context-free

- Dimostra che $L = \{w\#t$ | w è una sottostringa di t, con $w, t \in \{a, b\}^*$ non è context-free.

Per dimostrare che L = {w#t | w è una sottostringa di t, con w,t∈{a,b}*} non è context-free, utilizzeremo il pumping lemma per i linguaggi context-free.

Supponiamo per assurdo che L sia context-free. Allora, per il pumping lemma esiste un intero positivo p tale che ogni stringa s di L con |s| ≥ p può essere scritta come s = uvxyz, dove:

1. |vy| > 0

2. |vxy| ≤ p

3. per ogni i ≥ 0, uv^ixy^iz ∈ L.

Consideriamo la stringa s = a^p # a^2p. Poiché |s| = 3p + 1 ≥ p, esiste una scomposizione s = uvxyz che soddisfa le proprietà del pumping lemma.

Poiché |vxy| ≤ p, la sottostringa vxy di s deve essere contenuta interamente in una delle tre parti di s, ovvero in w, # o t. Inoltre, poiché |vy| > 0, la sottostringa vxy deve contenere almeno un simbolo.

Ci sono tre casi da considerare a seconda di dove si trova la sottostringa vxy in s:

1. vxy è contenuta in w. In questo caso, vxy deve essere una sottostringa di a^i per qualche i compreso tra 1 e p. Pertanto, uv^ixy^iz non può appartenere a L poiché la sottostringa w sarà più lunga di t e non sarà più una sottostringa di t.

2. vxy è contenuta in #. In questo caso, vxy deve essere uguale a # o contenere solo #. Pertanto, uv^ixy^iz non può appartenere a L poiché la sottostringa w sarà vuota o conterrà solo # e non sarà più una sottostringa di t.

3. vxy è contenuta in t. In questo caso, vxy deve essere una sottostringa di a^ip per qualche i compreso tra 1 e 2. Pertanto, uv^ixy^iz non può appartenere a L poiché la sottostringa w sarà più lunga di t e non sarà più una sottostringa di t.

In tutti e tre i casi, abbiamo ottenuto una contraddizione con la proprietà del pumping lemma per L. Pertanto, L non può essere context-free.

- Dimostra che $L = \{0^n\#0^{2n}\#0^{3n} \mid n \geq 0\}$ non è context-free.

Per dimostrare che L = {0^n#0^2n#0^3n | n ≥ 0} non è context-free, utilizzeremo il pumping lemma per i linguaggi context-free.

Supponiamo per assurdo che L sia context-free. Allora, per il pumping lemma esiste un intero positivo p tale che ogni stringa s di L con |s| ≥ p può essere scritta come s = uvxyz, dove:

1. |vy| > 0

2. |vxy| ≤ p

3. per ogni i ≥ 0, uv^ixy^iz ∈ L.

Consideriamo la stringa s = 0^p#0^2p#0^3p. Poiché |s| = 6p + 1 ≥ p, esiste una scomposizione s = uvxyz che soddisfa le proprietà del pumping lemma.

Poiché |vxy| ≤ p, la sottostringa vxy di s deve essere contenuta interamente in una delle tre parti di s, ovvero nei primi p zeri, tra il primo e il secondo hashtag o dopo il secondo hashtag. Inoltre, poiché |vy| > 0, la sottostringa vxy deve contenere almeno un simbolo.

Ci sono quattro casi da considerare a seconda di dove si trova la sottostringa vxy in s:

1. vxy è contenuta nei primi p zeri. In questo caso, vxy deve essere una sottostringa di 0^i per qualche i compreso tra 1 e p. Pertanto, uv^ixy^iz non può appartenere a L poiché il numero di zeri nella seconda e terza parte di s non sarà più uguale al doppio e al triplo del numero di zeri nella prima parte.

2. vxy è contenuta tra il primo e il secondo hashtag. In questo caso, vxy deve essere una sottostringa di 0^i per qualche i compreso tra p+1 e 2p. Pertanto, uv^ixy^iz non può appartenere a L poiché il numero di zeri nella seconda parte di s non sarà più doppio del numero di zeri nella prima parte.

3. vxy è contenuta tra il secondo hashtag e la fine di s. In questo caso, vxy deve essere una sottostringa di 0^i per qualche i compreso tra 2p+1 e 3p. Pertanto, uv^ixy^iz non può appartenere a L poiché il numero di zeri nella terza parte di s non sarà più triplo del numero di zeri nella prima parte.

4. vxy contiene zeri in due o tre parti di s. In questo caso, vxy deve essere una sottostringa di 0^i#0^j#0^k per qualche i, j e k tali che i+j ≤ 2p e j+k ≥ 2p+1. Pertanto, uv^ixy^iz non può appartenere a L poiché il numero di zeri nella seconda parte di s non sarà più doppio del numero di zeri nella prima parte e il numero di zeri nella terza parte di s non sarà più triplo del numero di zeri nella prima parte.

In tutti e quattro i casi, abbiamo ottenuto una contraddizione con la proprietà del pumping lemma per L. Pertanto, L non può essere context-free.

- Dimostra che $\{t_1\#t_2 \dots \#t_k \mid k \geq 2, \forall t_i \in \{a,b\}^*, t_i = t_j$ per qualche $i \neq j\}$ non è context-free

Per dimostrare che il linguaggio L = {t_1#t_2...#t_k | k ≥ 2, ∀t_i ∈ {a,b}* , t_i = t_j per qualche i ≠ j} non è context-free, utilizzeremo il pumping lemma per i linguaggi context-free.

Supponiamo per assurdo che L sia context-free. Allora, per il pumping lemma esiste un intero positivo p tale che ogni stringa s di L con |s| ≥ p può essere scritta come s = uvxyz, dove:

1. |vy| > 0

2. |vxy| ≤ p

3. per ogni i ≥ 0, uv^ixy^iz ∈ L.

Consideriamo la stringa $s = a^p\#a^p\#b^p$. Poiché |s| = 2p + p + 2 ≥ p, esiste una scomposizione s = uvxyz che soddisfa le proprietà del pumping lemma.

Poiché k ≥ 2, la sottostringa vxy di s deve essere contenuta interamente in una delle t_i, ovvero in a^p, #, a^p o b^p. Inoltre, poiché |vy| > 0, la sottostringa vxy deve contenere almeno un simbolo.

Ci sono quattro casi da considerare a seconda di dove si trova la sottostringa vxy in s:

1. vxy è contenuta in uno dei due separatori #. In questo caso, vxy deve essere uguale a # o contenere solo #. Pertanto, uv^ixy^iz non può appartenere a L poiché la stringa s rimane composta da due sottostringhe uguali seguite da una diversa.

2. vxy è contenuta in a^p. In questo caso, vxy deve essere una sottostringa di a^i per qualche i compreso tra 1 e p. Pertanto, uv^ixy^iz non può appartenere a L poiché la sottostringa vxy non sarà uguale a nessun'altra sottostringa t_j.

3. vxy è contenuta in b^p. In questo caso, vxy deve essere una sottostringa di b^i per qualche i compreso tra 1 e p. Pertanto, uv^ixy^iz non può appartenere a L poiché la sottostringa vxy non sarà uguale a nessun'altra sottostringa t_j.

4. vxy contiene sia a che b. In questo caso, vxy deve essere una sottostringa di a^ib^j per qualche i, j ≥ 1. Pertanto, uv^ixy^iz non può appartenere a L poiché la sottostringa vxy non sarà uguale a nessun'altra sottostringa t_j.

In tutti e quattro i casi, abbiamo ottenuto una contraddizione con la proprietà del pumping lemma per L. Pertanto, L non può essere context-free.

- Abbiamo definito la CUT del linguaggio A come $CUT(A) = \{yxz | xyz \in A\}$.

  Dimostrare che CUT non è chiusa rispetto alle CFG.

Per dimostrare che la classe delle CFL non è chiusa sotto CUT, costruiamo un controesempio. Consideriamo il linguaggio A = {a^n b^n c^n | n ≥ 1}. Questo linguaggio non è context-free, come dimostrato dal lemma di pumping per i linguaggi non context-free.

Consideriamo ora la sua CUT, ovvero il linguaggio CUT(A) = {yxz | xyz ∈ A}. In particolare, consideriamo la stringa s = a^n b^n c^n per un qualche n ≥ 1. Poiché s appartiene ad A, allora s = xyz, dove x = a^i, y = a^j b^k e z = b^l c^n con i+j+k = n e j > 0.

Consideriamo ora una qualsiasi partizione di y come y = y_1 y_2, con y_1 composto da almeno un simbolo. Allora, la stringa w = xy_1 z non appartiene a CUT(A), poiché w = a^i y_1 b^(k+l) c^n non può essere scritta come yxu per nessuna scelta di y, x e u, poiché l'unico modo per rappresentare w come yxu richiederebbe di scegliere y_1 come parte di x, ma in questo caso la lunghezza di y_1 supererebbe la lunghezza massima consentita per la parte y del lemma di pumping per i CFL.

Pertanto, abbiamo dimostrato che esiste almeno un linguaggio A non context-free tale che CUT(A) non è un linguaggio context-free. Di conseguenza, la classe delle CFL non è chiusa sotto CUT.

- Sia $\Sigma = \{1, 2, 3, 4\}$ e C = $\{w \in \Sigma *\}$ In w, il numero di 1 è uguale al numero di 2 e il numero di 4 è uguale al numero di 3 è uguale al numero di 4}. Dimostrare che C non è context- free

Per dimostrare che C non è un linguaggio context-free, utilizzeremo il pumping lemma per i linguaggi context-free.

Supponiamo per assurdo che C sia un linguaggio context-free. Allora, per il pumping lemma esiste un intero positivo p tale che ogni stringa s di C con |s| ≥ p può essere scritta come s = uvxyz, dove:

1. |vy| > 0

2. |vxy| ≤ p

3. per ogni i ≥ 0, uv^ixy^iz ∈ C.

Consideriamo la stringa $s = 1^p\ 2^p\ 3^p\ 4^p$. Poiché |s| = 4p ≥ p, esiste una scomposizione s = uvxyz che soddisfa le proprietà del pumping lemma.

Poiché il numero di 1 è uguale al numero di 2 e il numero di 3 è uguale al numero di 4 in s, la sottostringa vxy di s deve contenere solo 1, 2 o entrambi. Inoltre, poiché |vy| > 0, la sottostringa vxy deve contenere almeno un simbolo.

Ci sono tre casi da considerare a seconda di quale tipo di simboli contiene vxy:

1. vxy contiene solo 1. In questo caso, uv^ixy^iz non può appartenere a C poiché il numero di 1 nella stringa non sarebbe più uguale al numero di 2.

2. vxy contiene solo 2. In questo caso, uv^ixy^iz non può appartenere a C poiché il numero di 2 nella stringa non sarebbe più uguale al numero di 1.

3. vxy contiene sia 1 che 2. In questo caso, vxy deve contenere almeno un 1 e un 2. Sia y = 1^a 2^b con a > 0 e b > 0. Allora, scegliamo i = 0 e otteniamo la stringa uv^0xy^0z = uxyz, che contiene meno 1 e meno 2 rispetto a s. In particolare, il numero di 1 nella stringa sarebbe minore del numero di 2, il che significa che la stringa non appartiene a C.

In tutti e tre i casi, abbiamo ottenuto una contraddizione con la proprietà del pumping lemma per C. Pertanto, C non può essere un linguaggio context-free.

<u>Forme normali</u>

- Per le stringhe w e t, si scrive w $ t se i simboli di w sono una permutazione dei simboli di t. In altre parole, w $ t se t e w hanno gli stessi simboli nella stessa riga.

  In altre parole, w $ t se t e w hanno gli stessi simboli nelle stesse quantità, ma eventualmente in ordine diverso.

  Per qualsiasi stringa w, definire SCRAMBLE(w) = {t| t $ w}. Per qualsiasi linguaggio A, sia

  $$SCRAMBLE(A) = \{t|\ t \in SCRAMBLE(w)\ per\ qualche\ w \in A\}.$$

  Mostrare che se Σ = {0,1}, allora lo SCRAMBLE di un linguaggio regolare è context-free.

Per dimostrare che lo SCRAMBLE di un linguaggio regolare è context-free quando Σ = {0,1}, costruiamo una grammatica context-free per SCRAMBLE(A) a partire da un DFA per A.

Sia M = (Q, Σ, δ, q0, F) un DFA che accetta il linguaggio regolare A. La grammatica context-free per SCRAMBLE(A) ha gli stessi simboli terminali Σ = {0, 1} e gli stessi simboli non terminali di M, ovvero Q.

La grammatica ha le seguenti regole di produzione:

- S → ε

- S → 0S1

- S → 1S0

- S → aS, per ogni a ∈ Σ e ogni stato q ∈ Q tale che non sia uno stato finale di M

- S → aSb, per ogni a, b ∈ Σ e ogni coppia di stati q, p ∈ Q tale che δ(q, a) = p e δ(p, b) = q

La regola di produzione S → ε rappresenta il caso in cui la stringa vuota appartiene a SCRAMBLE(A). Le regole di produzione S → 0S1 e S → 1S0 generano stringhe in cui il simbolo 0 e il simbolo 1 sono permutati. Le regole di produzione S → aS generano stringhe in cui l'ordine dei simboli non terminali corrisponde

all'ordine degli stati visitati dal DFA quando legge il simbolo a. Infine, le regole di produzione S → aSb generano stringhe in cui gli stati visitati dal DFA quando legge il simbolo a sono gli stessi visitati quando legge il simbolo b, ma in ordine inverso.

Ogni stringa t generata dalla grammatica context-free per SCRAMBLE(A) può essere vista come una permutazione delle stringhe w che il DFA M accetta, dove i simboli di t corrispondono ai simboli di w distribuiti in modo diverso. Pertanto, SCRAMBLE(A) è context-free.

In conclusione, abbiamo dimostrato che se Σ = {0,1}, lo SCRAMBLE di un linguaggio regolare è context-free.

- Sia L un linguaggio context-free. Si definisca il linguaggio L' come l'insieme di tutte le stringhe w ∈ {a, b}* che possono essere ottenute sostituendo una qualsiasi sottostringa di una stringa in L con una qualsiasi stringa in {a, b}*.

  Ad esempio, se L = {a^n b^n | n ≥ 1}, allora L' contiene tutte le stringhe ottenute sostituendo una qualsiasi sottostringa di una stringa di L con una qualsiasi stringa in {a, b}*.

  Dimostrare che se L è context-free, allora L' è anch'esso un linguaggio context-free.

Per dimostrare che L' è un linguaggio context-free, costruiamo una grammatica context-free che genera L'.

Sia G = (V, Σ, R, S) una grammatica context-free che genera L. Aggiungiamo alla grammatica G un nuovo simbolo di start S', e definiamo R' come l'insieme delle regole di produzione:

S' → S

S → aSb | ε

La produzione S → aSb genera una stringa in L, mentre la produzione S' → S mantiene la struttura di L. Inoltre, ε è ammesso come stringa in L', in quanto ε può essere ottenuta sostituendo la stringa vuota di L con la stringa vuota di {a, b}*.

Ora, consideriamo una qualsiasi stringa w ∈ L'. Poiché w può essere ottenuta sostituendo una qualsiasi sottostringa di una stringa in L con una qualsiasi stringa in {a, b}, allora w può essere scritta come w = xuy, dove x e y sono stringhe in {a, b} e u è una stringa in L.

Poiché L è un linguaggio context-free e G è una grammatica context-free che genera L, allora esiste una derivazione S →* u in G. Quindi, la stringa w può essere generata dalla grammatica G' = (V, Σ, R', S'), poiché si può ottenere la derivazione S' → S →* u → xuy.

Pertanto, abbiamo costruito una grammatica context-free G' che genera L', dimostrando così che L' è un linguaggio context-free.

Concludiamo quindi che se L è un linguaggio context-free, allora L' è anch'esso un linguaggio context-free.

- Sia L un linguaggio context-free su un alfabeto Σ. Si definisce il linguaggio L' come segue: una stringa w appartiene a L' se e solo se w può essere scritta come w = uvxyz, dove u, v, x, y, z sono stringhe in Σ*, tali che:
  - uvyz contiene almeno una lettera
  - per ogni i ≥ 0, la stringa $u(v^i)x(y^i)z$ appartiene a L.
  - Dimostrare che L' è un linguaggio context-free.

Per dimostrare che L' è un linguaggio context-free, costruiamo una grammatica context-free che genera L'.

Sia G = (V, Σ, R, S) una grammatica context-free che genera L. Aggiungiamo alla grammatica G un nuovo simbolo di start S', e definiamo R' come l'insieme delle regole di produzione:

S' → S

S → AB

A → aAa | bAb | ε

B → cBd | ε

La produzione S' → S genera una stringa in L. La produzione S → AB divide la stringa in due parti, A e B, in modo che A contenga una parte delle lettere e B contenga l'altra parte delle lettere.

La produzione A → aAa, A → bAb e A → ε generano tutte le possibili combinazioni di lettere in A. La produzione B → cBd genera una sequenza di lettere c e d, che possono essere utilizzate per creare una stringa vuota o per aggiungere un numero arbitrario di c e d alla stringa generata da A.

Pertanto, la grammatica G' genera tutte le stringhe di L', poiché ogni stringa di L' può essere generata da una derivazione S' →* w dove w = uvxyz, u ∈ Σ*, v = a^k, x = b^k, y = c^k e z = d^k per qualche intero positivo k.

Poiché abbiamo costruito una grammatica context-free che genera L', possiamo concludere che L' è un linguaggio context-free.

- Sia L un linguaggio context-free su un alfabeto Σ. Si definisce il linguaggio complemento di L, indicato con L', come segue: una stringa w appartiene a L' se e solo se w non appartiene a L. Dimostrare che se L è un linguaggio context-free, allora L' è anch'esso un linguaggio context-free.

Sia G = (V, Σ, R, S) una grammatica context-free che genera L. Aggiungiamo alla grammatica G un nuovo simbolo di start S', e definiamo R' come l'insieme delle regole di produzione:

S' → A | ε

A → aA | bA | cA | dA | E

E → Aa | Ab | Ac | Ad | Ba | Bb | Bc | Bd | ... | Za | Zb | Zc | Zd

B → bB | cB | dB | aB | E

...

Z → zZ | aZ | bZ | cZ | dZ | E

La produzione S' → A genera tutte le stringhe che appartengono a L', poiché ogni stringa che può essere generata dalla grammatica G appartiene a L, e ogni stringa che non può essere generata dalla grammatica G appartiene a L'.

La produzione A → aA, A → bA, A → cA e A → dA generano tutte le possibili combinazioni di lettere in A. La produzione E → Aa, E → Ab, E → Ac e E → Ad generano tutte le possibili combinazioni di una lettera di A seguita da una lettera diversa.

Inoltre, le produzioni B → bB, B → cB, B → dB e B → aB generano tutte le possibili combinazioni di lettere in B, mentre le produzioni Z → zZ, Z → aZ, Z → bZ, Z → cZ e Z → dZ generano tutte le possibili combinazioni di lettere in Z.

Pertanto, la grammatica G' genera tutte le stringhe di L', poiché ogni stringa che può essere generata dalla grammatica G appartiene a L, e ogni stringa che non può essere generata dalla grammatica G appartiene a L'.

Poiché abbiamo costruito una grammatica context-free che genera L', possiamo concludere che L' è un linguaggio context-free.

Esercizi Automi 26-04

Prima Parte

Pumping Lemma

Esercizio 1 Considerate l'insieme delle stringhe il cui simbolo centrale è una a, cioè il linguaggio

$$L = \{x \in \{a,b\}^* \mid \text{il simbolo di } x \text{ in posizione } \lfloor \tfrac{|x|}{2} \rfloor \text{ è una } a\}.$$

Dimostrate che esiste un insieme infinito di stringhe distinguibili rispetto a L. Concludete quindi che L non è regolare. Dimostrate poi lo stesso risultato utilizzando il pumping lemma per i linguaggi regolari.

- Dimostrare che il linguaggio $L = \{b^n c^{2^k} : n \geq 1, k \geq 1\}$ non è regolare

Calcolare la lunghezza minima di pumping (pumping length) [nota: non sempre esiste]

- $\{0^n 1^n 2^m 3^m | n, m \geq 0\}$
- $\{x\#y | x, y \in \{0,1\}^* \; e \; |x| = 2|y|\}$
- $\{0^n 1^m 2^{n-m} | n \geq m \geq 0\}$
- $L = \{a^n b a^{2n} \mid n \geq 0\}$
- $L = \{a^i b^j \mid j = i \; oppure \; j = 2i\}$
- $L = \{a^i w \mid i \geq 0, w \in \{b,c\}^* \; e \; se \; i = 1$, allora w ha lo stesso numero di b e di $c\}$

For $\Sigma = \{a, b\}$, let us consider the regular language:

$$L = \{x \mid x = a^{2+3k} \text{ or } x = b^{10+12k}, k \geq 0\}$$

Which one of the following can be a pumping length (the constant guaranteed by the pumping lemma) for L?

(A) 3 (B) 5 (C) 9 (D) 24

Dimostrare che L è regolare

Esercizio 5 Dimostrate che se L è un linguaggio regolare, allora lo è anche il linguaggio:

$$\text{cyc}(L) = \{x_1 x_2 \in \Sigma^* \mid x_1, x_2 \in \Sigma^* \text{ e } x_2 x_1 \in L\}.$$

Fornite una costruzione per ottenere un automata che accetta il linguaggio $\text{cyc}(L)$ a partire da un automa che accetta L.

Esercizio 8 Dimostrate che se L è un linguaggio regolare, allora lo è anche il linguaggio:

$$\log(L) = \{w \in \Sigma^* \mid \exists y \in \Sigma^* \text{ con } |y| = 2^{|w|} \text{ e } wy \in L\}.$$

Esercizio 9 Dimostrate che se L è un linguaggio regolare, allora lo è anche il linguaggio:

$$\mathrm{ROOT}(L) = \{w \in \Sigma^* \mid w^{|w|} \in L\}.$$

- Per una stringa $w = a_1a_2a_3a_4a_5a_6$ …, si definisca $third(w) = a_3a_6a_9$ …
  Per un linguaggio L, si definisca $third(L) = \{third(w): w \in L\}$. Si mostri che se L è regolare, allora $third(L)$ è anche regolare.

- Si dimostri che $min(L) = \{w \mid w$ è in L, ma nessun prefisso proprio di w è in $L\}$ è regolare

- Si dimostri che $max(L) = \{w \mid w$ è in L e per nessuna x diversa da wx è in $L\}$ è regolare

- Si dimostri che $init(L) = \{w \mid$ per qualche x, wx è in $L\}$ è regolare

<u>Pumping Lemma per linguaggi context-free</u>

- Si dia un esempio di un linguaggio non context-free ma che si comporta come tale per il pumping lemma. Si provi che l'esempio funziona.
- Si mostri che il linguaggio $L = \{a^n b^n c^i : n \leq i \leq 2n\}$ non è context-free.
- Si mostri che il linguaggio $L = \{ww^R w \mid w$ è una stringa di 0 e di 1$\}$ non è context-free.

<u>Context-Free oppure No ?</u>

- Si decida se $L = \{x \in \{a,b\}^* \mid N_a(x) < N_b(x) < 2N_a(x)\}$ è context-free o meno.
  - Si intende che N_a sia il numero di occorrenze di a, etc. etc.
- Si decida se $L = \{w\#x \mid w^R$ è una sottostringa di x per $w, x \in \{0,1\}^*\}$ è context-free o meno.
- Si decida se $L = \{w\#x \mid w$ è una sottostringa di $x\}$ è context-free o meno
- Si decida che $L = \{a^j b^i c^{2i} \mid i, j \geq 0\}$ è context-free o meno.

<u>Forme normali</u>

- Per definizione, il linguaggio $half(L)$ è definito come segue:

  $$half(L) = \{w \mid \exists x \in L, |x| = 2|w| \; e \; w \text{ è } una \; sottostringa \; di \; x\}$$

  In altre parole, $half(L)$ è il linguaggio delle sottostringhe di lunghezza pari degli elementi del linguaggio L. Si dimostri che tale linguaggio è o meno context-free.

- Sia $G = (V, \Sigma, R, S)$ una grammatica context-free che genera il linguaggio L. L'alfabeto di input di $FOLLOW(L)$ è Σ. Per ogni simbolo non terminale A in V, $FOLLOW(L)(A)$ è definito come l'insieme di tutti i simboli dell'alfabeto Σ che possono seguire direttamente A in una derivazione di una stringa in L. Si mostri che se L è context-free, $FOLLOW(L)$ è context-free.

- Definiamo la *rotational closure* di A come $RC(A) = \{yx \mid xy \in A\}$. Si mostri che $RC(A)$ è chiuso rispetto alla classe dei linguaggi context-free.

- Si mostri che se L è regolare allora $RL = \{x | \exists y \; s.t. \; xyx^R \in L\}$ è regolare.

- Si mostri che per un linguaggio L definito su un alfabeto Σ, con

$$SW(L) = \{y \in \Sigma^* | \exists x \in \Sigma^* \; tale \; che \; xyz \in L\}$$

allora $SW(L)$è regolare.

- $NOEXTEND(L)$ è un linguaggio definito come:

$$NOEXTEND(L) = \{x \in \Sigma* \mid per \; ogni \; y \in \Sigma*, xy \notin L\}$$

In altre parole, NOEXTEND(L) è l'insieme di tutte le stringhe che non possono essere estese da nessuna delle stringhe di L. Si mostri che è regolare.