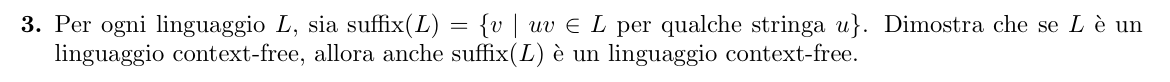
Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, bianco, schermata

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, documento, schermata

Descrizione generata automaticamente

Per dimostrare che suffix(L) è un linguaggio context-free se L è context-free:

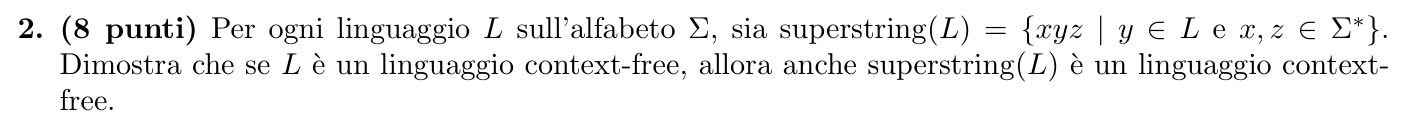
Sia G = (V, Σ, R, S) la grammatica context-free che genera L. Costruiamo una nuova grammatica G' = (V', Σ, R', S') che genera suffix(L):

V' = V ∪ {S'} dove S' è un nuovo simbolo non terminale R' contiene tutte le regole di R, più le seguenti nuove regole:

* S' → S
* S' → aS' per ogni a ∈ Σ

Questa grammatica G' genera suffix(L) perché:

* Può generare qualsiasi stringa in L (usando S' → S e poi le regole originali)
* Può aggiungere qualsiasi prefisso arbitrario a una stringa di L (usando ripetutamente S' → aS')

Poiché G' è una grammatica context-free, suffix(L) è un linguaggio context-free.

Sia G = (V, Σ, R, S) la grammatica context-free che genera L. Costruiamo una nuova grammatica G' = (V', Σ, R', S') che genera superstring(L):

V' = V ∪ {S', A, B} dove S', A, B sono nuovi simboli non terminali R' contiene tutte le regole di R, più le seguenti nuove regole:

* S' → ASB
* A → aA | ε per ogni a ∈ Σ
* B → bB | ε per ogni b ∈ Σ

Questa grammatica G' genera superstring(L) perché:

* Può generare qualsiasi prefisso x ∈ Σ\* usando le regole di A
* Può generare qualsiasi stringa y ∈ L usando le regole originali di G
* Può generare qualsiasi suffisso z ∈ Σ\* usando le regole di B

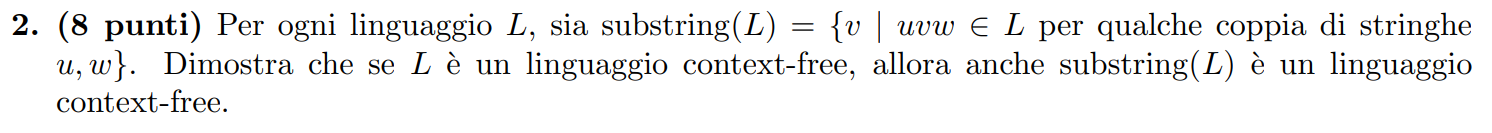
Poiché G' è una grammatica context-free, superstring(L) è un linguaggio context-free.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, linea, algebra

Descrizione generata automaticamentePer dimostrare che stutter(L) è context-free se L è context-free, costruiamo una grammatica context-free per stutter(L) a partire da una grammatica G per L.

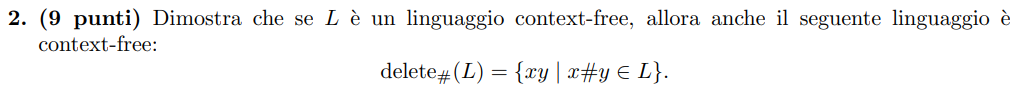
Sia G = (V, Σ, R, S) la grammatica per L. Costruiamo G' = (V', Σ, R', S') per stutter(L):

V' = V ∪ {S', A\_a | a ∈ Σ} S' è il nuovo simbolo iniziale R' contiene le seguenti regole:

* S' → S
* S' → A\_a S per ogni a ∈ Σ
* A\_a → aa per ogni a ∈ Σ
* Tutte le regole di R

G' genera stutter(L) perché:

* Può generare qualsiasi stringa di L (usando S' → S e le regole di R)
* Può inserire coppie di simboli identici in qualsiasi posizione (usando A\_a → aa)

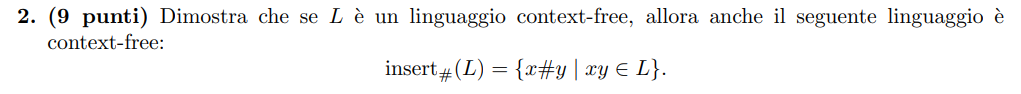
Quindi stutter(L) è context-free.

Per dimostrare che delete#(L) = {xy | x#y ∈ L} è context-free se L è context-free:

Sia G = (V, Σ ∪ {#}, R, S) la grammatica per L. Costruiamo G' = (V', Σ, R', S') per delete#(L):

V' = V ∪ {S'} S' è il nuovo simbolo iniziale R' contiene le seguenti regole:

* S' → S
* Per ogni regola A → αBβ in R, dove α e β non contengono #, aggiungiamo A → αβ in R'
* Per ogni regola A → α#β in R, aggiungiamo A → αβ in R'

G' genera delete#(L) perché simula G ma "salta" il simbolo # quando lo incontra.

Per dimostrare che insert#(L) = {x#y | xy ∈ L} è context-free se L è context-free:

Sia G = (V, Σ, R, S) la grammatica per L. Costruiamo G' = (V', Σ ∪ {#}, R', S') per insert#(L):

V' = V ∪ {S'} ∪ {A\_# | A ∈ V} S' è il nuovo simbolo iniziale R' contiene le seguenti regole:

* S' → S\_#
* Per ogni A → α in R, aggiungiamo A\_# → α\_#
* Per ogni simbolo terminale a ∈ Σ, aggiungiamo a\_# → a#a e a\_# → a
* Per ogni variabile A ∈ V, aggiungiamo A\_# → A#

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamenteG' genera insert#(L) perché simula G ma permette di inserire # in qualsiasi posizione.

Dimostrazione: Sia A = (Q, {0,1}, δ, q0, F) un DFA che riconosce B. Costruiamo una grammatica context-free G = (V, {0,1}, R, S) per SCRAMBLE(B) come segue:

V = {S} ∪ {[p,q,n0,n1] | p,q ∈ Q, n0,n1 ≥ 0}

Le regole R sono:

1. S → [q0,qf,0,0] per ogni qf ∈ F
2. [p,q,n0,n1] → 0[p,q,n0+1,n1] | 1[p,q,n0,n1+1] per ogni p,q ∈ Q
3. [p,q,n0+1,n1] → 0[δ(p,0),q,n0,n1] per ogni p,q ∈ Q
4. [p,q,n0,n1+1] → 1[δ(p,1),q,n0,n1] per ogni p,q ∈ Q
5. [p,p,0,0] → ε per ogni p ∈ Q

L'idea è che [p,q,n0,n1] rappresenta uno stato in cui:

* p è lo stato corrente in A
* q è lo stato finale che vogliamo raggiungere
* n0 e n1 sono i numeri di 0 e 1 ancora da leggere

Le regole 2-4 permettono di generare e consumare 0 e 1 in qualsiasi ordine, simulando una permutazione. La regola 5 permette di terminare quando abbiamo raggiunto lo stato desiderato e consumato tutti i simboli.

Per dimostrare la correttezza:

* Se w ∈ B, allora esiste una derivazione in G che genera ogni permutazione di w.
* Se t può essere derivato in G, allora esiste w ∈ B tale che t è una permutazione di w.

Quindi, SCRAMBLE(B) è generato da una grammatica context-free, e quindi è un linguaggio context-free.

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

Dimostrazione: Siano A e B linguaggi context-free. Esistono quindi grammatiche context-free G\_A = (V\_A, Σ\_A, R\_A, S\_A) e G\_B = (V\_B, Σ\_B, R\_B, S\_B) che generano A e B rispettivamente.

Costruiamo una grammatica context-free G = (V, Σ, R, S) per MIX(A,B):

V = V\_A ∪ V\_B ∪ {S, X, Y} dove S, X, Y sono nuovi simboli non in V\_A o V\_B Σ = Σ\_A ∪ Σ\_B S è il nuovo simbolo iniziale R contiene tutte le regole in R\_A e R\_B, più le seguenti nuove regole:

1. S → XY | ε
2. X → S\_AX | ε
3. Y → S\_BY | ε

Per dimostrare la correttezza di G:

1. Se w ∈ MIX(A,B), allora w = x\_1y\_1...x\_ny\_n con x\_i ∈ A, y\_i ∈ B. Possiamo derivare w in G usando le regole 1-3 per generare la struttura alternata, e poi le regole di G\_A e G\_B per generare le stringhe x\_i e y\_i rispettivamente.
2. Se w può essere derivato in G, allora per la struttura delle regole, w avrà la forma x\_1y\_1...x\_ny\_n dove ogni x\_i è derivabile in G\_A e ogni y\_i è derivabile in G\_B. Quindi w ∈ MIX(A,B).

Poiché G è una grammatica context-free che genera MIX(A,B), MIX(A,B) è un linguaggio context-free.

Immagine che contiene testo, Carattere, schermata, algebra

Descrizione generata automaticamenteQuindi, la classe dei linguaggi context-free è chiusa per l'operazione MIX.

Dimostrazione: Sia L un linguaggio context-free. Esiste quindi una grammatica context-free G = (V, Σ, R, S) che genera L.

Costruiamo una grammatica context-free G' = (V', Σ, R', S') per Rep(L):

V' = V ∪ {S', A} dove S' e A sono nuovi simboli non in V S' è il nuovo simbolo iniziale R' contiene tutte le regole in R, più le seguenti nuove regole:

1. S' → SA
2. A → SA | ε
3. Per ogni a ∈ Σ, aggiungiamo la regola a → aa

Per dimostrare la correttezza di G':

1. Se w ∈ Rep(L), allora esiste una stringa v ∈ L tale che w = Rep(v). Possiamo derivare w in G' usando le regole 1-2 per generare la struttura di v, e poi la regola 3 per raddoppiare ogni carattere.
2. Se w può essere derivato in G', allora per la struttura delle regole, w avrà la forma Rep(v) dove v è derivabile in G. Quindi w ∈ Rep(L).

Poiché G' è una grammatica context-free che genera Rep(L), Rep(L) è un linguaggio context-free.

Immagine che contiene testo, Carattere, schermata, algebra

Descrizione generata automaticamenteQuindi, la classe dei linguaggi context-free è chiusa per l'operazione Rep.

Dimostrazione: Poiché B è regolare, esiste un DFA A = (Q, Σ, δ, q0, F) che riconosce B. Costruiamo una grammatica context-free G = (V, Σ, R, S) per PALINDROMIZE(B):

V = {S} ∪ {[q, r] | q, r ∈ Q} S è il simbolo iniziale R contiene le seguenti regole:

1. S → [q0, q] per ogni q ∈ F
2. [q, r] → a[δ(q, a), r]a per ogni q, r ∈ Q e a ∈ Σ
3. [q, q] → ε per ogni q ∈ Q

L'idea è che [q, r] genera tutte le stringhe w tali che δ\*(q, w) = r, dove δ\* è l'estensione di δ alle stringhe.

Per dimostrare la correttezza di G:

1. Se ww^R ∈ PALINDROMIZE(B), allora w ∈ B. Quindi, esiste una sequenza di stati q0, q1, ..., qn = qf in A tale che δ(qi, wi) = qi+1 e qf ∈ F. Possiamo derivare ww^R in G usando le regole 1, 2 ripetutamente, e infine 3.
2. Se una stringa può essere derivata in G, per la struttura delle regole, deve essere della forma ww^R dove w porta A dallo stato iniziale a uno stato finale, quindi w ∈ B e ww^R ∈ PALINDROMIZE(B).

Poiché G è una grammatica context-free che genera PALINDROMIZE(B), PALINDROMIZE(B) è un linguaggio context-free.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Sia L un linguaggio context-free su Σ.

Esiste quindi una grammatica context-free G tale che L = L(G).

Costruiamo una nuova grammatica G' su Γ come segue:

- G' ha gli stessi non-terminali di G

- Per ogni produzione A → α in G, G' ha la produzione A → T(α),

dove T è applicata a ogni simbolo di α

- Gli stessi simboli iniziali

G' genera esattamente le stringhe T(w) per ogni w ∈ L(G).

Infatti, se S =>\* w in G, allora S =>\* T(w) in G',

e viceversa se S =>\* w' in G', w' = T(w) per qualche w tale che S =>\* w in G.

Quindi L(G') = T(L(G)) = T(L).

Poiché G' è context-free, anche T(L) lo è.