Eliminare la ricorsione sx immediata, con e senza le epsilon mosse

Per trasformare da una grammatica ad un'altra equivalente senza ricorsione sinistra immediata, metto in evidenza i non terminali con ricorsione sinistra immediata

$A \rightarrow A\beta_1 \mid \ldots \mid A\beta_n \mid \gamma_1 \mid \ldots \mid \gamma_m$

$\beta_1,\ldots,\beta_n \in (\Sigma \cup V)^*$
$\gamma_1,\ldots,\gamma_n \in (\Sigma \cup V)^*$ ma non iniziano con A

Con la ε produzione può essere trasformato così:
$A \rightarrow \gamma_1 A' \mid \ldots \mid \gamma_m A'$
$A' \rightarrow \beta_1 A' \mid \ldots \mid \beta_n A' \mid \varepsilon$

Senza la ε produzione può essere trasformato così:
$A \rightarrow \gamma_1 \mid \ldots \mid \gamma_m \mid \gamma_1 A' \mid \ldots \mid \gamma_m A'$
$A' \rightarrow \beta_1 \mid \ldots \mid \beta_n \mid \beta_1 A' \mid \ldots \mid \beta_n A'$

Oppure regole del terenz:

$A \rightarrow A\alpha_1 \mid \ldots \mid A\alpha_n \mid \beta_1 \mid \ldots \mid \beta_m$

$\alpha_1,\ldots,\alpha_n \in (\Sigma \cup V)^*$
$\beta_1,\ldots,\beta_n \in (\Sigma \cup V)^*$ ma non iniziano con A

Con la ε produzione può essere trasformato così:
$A \rightarrow \beta_1 A' \mid \ldots \mid \beta_m A'$
$A' \rightarrow \alpha_1 A' \mid \ldots \mid \alpha_n A' \mid \varepsilon$

Senza la ε produzione può essere trasformato così:
$A \rightarrow \beta_1 \mid \ldots \mid \beta_m \mid \beta_1 A' \mid \ldots \mid \beta_m A'$
$A' \rightarrow \alpha_1 \mid \ldots \mid \alpha_n \mid \alpha_1 A' \mid \ldots \mid \alpha_n A'$

ES:

$X \rightarrow XXa \mid Xb \mid AXa \mid a$
Con epsilon prod.
$X \rightarrow AXaX' \mid aX'$
$X' \rightarrow XaX' \mid bX' \mid \varepsilon$

$X \rightarrow AXa \mid a \mid AXaX' \mid aX' \mid$
$X' \rightarrow Xa \mid b \mid XaX' \mid bX' \mid$

Definire le grammatiche lineari destre.

Le grammatiche lineari destre sono grammatiche che sono incluse per definizione nelle grammatiche lineari quindi con un solo asse di crescita, quindi di conseguenza sono a sua volta incluse nell'insieme delle grammatiche context free.
E sono così definite:
$A \rightarrow \propto$
Dove $A \in V$
e $\propto \in \sum^{*} (X \cup \varepsilon)$
Con $X \in V$
Quindi non posso mettere niente dopo il non terminale X .
In sostanza non riusciamo a fare delle ricorsioni centrali quindi non riusciremo più a generare bilanciamenti.

Definire le grammatiche lineari destre e dimostrare che i linguaggi generati da tali grammatiche sono propriamente inclusi nei linguaggi generati dalle grammatiche lineari.

Grammatica lineare: $\sum^{*} (V \cup \varepsilon) \sum^{*}$
lineare dx sono così definite:
$A \rightarrow \propto$
Dove $A \in V$
e $\propto \in \sum^{*} (X \cup \varepsilon)$
Con $X \in V$
Dalle grammatiche lineari riusciamo a generare tutti i linguaggi generati da una gramm lin dx ma non viceversa.
Infatti per definizione gramm lin dx sono contenute nelle grammatiche lin dobbiamo definire che sono diverse.
Per farlo bisogna trovare un linguaggio definibile da una gramm lin e non da una lin dx
es $L\{a^n b^n \backslash n >= 0\}$
non riesco a definirlo con una lin dx mentre con una lineare si a causa dei bilanciamenti.

Dimostrare che la classe dei linguaggi generati dalle grammatiche lineari è strettamente contenuta nella classe dei linguaggi context-free

Definizione grammatica context free:
Sono della forma:
$A \rightarrow \propto$
Dove $A \in V$
e $\propto \in (\sum \cup X)^{*}$
Con $X \in V$
Libere dal contesto.
Dobbiamo dimostrare che L(gram context free) $\supset$ L(gram.lin).
Definizione grammatica lineare :
Per definizione è una grammatica context free in cui io uso solo le regole:
$A \rightarrow \propto$
Dove $A \in V$
e $\propto \in \sum^{*} (X \cup \varepsilon) \sum^{*}$
Con $X \in V$
Ora dobbiamo dimostrare che la inclusione è stretta.
Possiamo dire che la inclusione è stretta perché le grammatiche lineari hanno un solo asse di crescita mentre quelle context free ne possono avere più di uno.
$L=\{a1^n\ b1^n\ a2^m b2^m ... a_k^h a_k^h\}$

Qui ci vuole più di un asse di crescita perché vogliamo una lista di bilanciamenti e non riusciamo ad esprimerla con una grammatica lineare.

Un linguaggio denotato da una espressione regolare è chiuso rispetto all'unione?
La chiusura di un linguaggio rispetto ad un'operatore significa che applicando questo operatore a qualsiasi e.r che genera quel linguaggio riottengo un linguaggio appartenente alla classe dei linguaggi regolari.Nel caso appunto della classe dei linguaggi regolari questo vale per l'operazione di U, *,^{n};$^+$,complemento(non posso ancora dimostrarlo)

Si definisca formalmente il linguaggio L(G) generato da una grammatica context free G, a partire dalla nozione di derivazione
I linguaggi context free è l'insieme dei linguaggi generati da una grammatica context free.
Formalmente è una quadrupla:
<V,∑,P,S>
P sono le regole della grammatica.r
Derivazione:
Svolgimento
Partendo da 2 forme di frasi misto di terminali e non terminali **β e γ ∈ (V ∪ ∑)***
Dato β formato da un prefisso η, un infisso A, ed un suffisso δ:
β = η A δ

Dato γ formato da un prefisso η, un infisso ξ, ed un suffisso δ:
γ = η ξ δ
Con una regola di produzione appartenente all'insieme P delle regole del linguaggio
(A → ξ ∈ P) con A ∈ V
Si possono utilizzare ^{n} passi facendo tutte le derivazioni partendo dallo start symbol se so quanti ne devp fare per finire tutti gli oggetti terminali
Si possono utilizzare * passi facendo tutte le derivazioni partendo dallo start symbol, e termina solo quando finisco tutti gli oggetti terminali se non so quanti ne ho bisogno(potrebbe anche essere infinito come i linguaggi parlati)
Il linguaggio è definito da:
L(G) = {x ∈ ∑* \ S →* x}
Dove G è appunto una grammatica context free
dove S è lo start symbol del linguaggio, e →* indica che si raggiunge x in un certo numero di passi.
S ->*x sono tutte le possibili frasi solo di terminali derivabili dallo star symbol.

Le grammatiche context free , senza produzioni autoinclusive generano sempre linguaggi regolari ?
Se ci sono produzioni autoinclusive, il linguaggio generato è sempre non regolare?
Nel caso in cui non ci siano produzioni autoinclusive siamo sicuri che generano sempre linguaggi regolari perchè di sicuro non saranno presenti delle dipendenze
es:
S→ aS/S1
S1→ bS1/b
Non possiamo dire che sono presenti delle dipendenze quindi sicuramente genereremo un linguaggio regolare.

Nel caso in cui invece abbiamo delle produzioni autoinclusive, non è detto che sia generato un linguaggio non regolare.
Es
S→ ASB/epsilon
A→ a/epsilon
B→ b/epsilon

In questo caso non ci sono dipendenze quindi anche con una ricorsione autoinclusiva è possibile generare un linguaggio regolare.

Definire le funzioni di transizione δ e δ* per gli automi a stati finiti deterministici(+ dim)
δ in un automa a stati finiti deterministico è definito nel seguente modo:
$\delta: Q \times \sum \rightarrow Q$
Dato uno stato e un simbolo appartenente all'alfabeto in input la funzione di transizione mi dice in quale stato vado a finire,oppure potrebbe anche non andare in nessuno stato e fallire.
$Q \times \sum \rightarrow$
In questa situazione fallisce.
Mentre il δ* serve per definire i linguaggi accettati dall'automa.
L'automa riconosce tutte e sole le stringhe tale per cui se io parto da uno stato iniziale e analizzo la stringa arrivo a uno stato finale, quindi dobbiamo parlare di stringhe non più di singolo simbolo dell'alfabeto

Quindi $\delta^* = Q \times \sum^* \rightarrow Q$

oppure $\delta^* = Q \times \sum^* \rightarrow$

In questo caso fallisco
Se vuole dimostrazione:
caso base: δ*(q,a)= δ(q,a) dove a è un simbolo dell'alfabeto
Opero per induzione:
Prendo una stringa lunga n+1, cioè una stringa lunga n più un simbolo dell'alfabeto
Per induzione posso immaginare di sapere cosa succede se parto da "q" e analizzo la stringa y.

Se io so che per δ*(q,y) ottengo uno stato per ipotesi perché $\delta^* = Q \times \sum^* \rightarrow Q$

Quindi per ipotesi induttiva posso dire che:$\delta^*(q, y.a) = \delta(\delta^*(q,y), a)$ dove $y \in \sum^n$ mentre $a \in \sum$

Linguaggio riconosciuto dall'automa: $L(M)=\{x \in \sum^* \setminus \delta^*(q,x) \in F\}$
Definire formalmente i linguaggi riconosciuti dagli automi a stati finiti deterministici.
Linguaggio riconosciuto dall'automa: $L(M)=\{x \in \sum^* \setminus \delta^*(q,x) \in F\}$

Dare la definizione formale della funzione di transizione δ^ per automi a stati finiti non deterministici con ε-mosse (sulla base della funzione δ)
Il **δ** è definito nel seguente modo:
δ: $Q \times (\sum \cup \{\varepsilon\}) \rightarrow 2^Q$
In quanto anche epsilon ci può portare a una mossa, e essendo non deterministico posso andare in più stati.
δ^ è il delta che ci permette di riconoscere i linguaggi accettati dall'automa.
Non posso più dire che $\delta^*(q,\varepsilon)=q$
Nozione di chiusura transitiva(ε - chiusura):Tutti gli stati raggiungibili da q utilizzando solo le ε mosse.
Per associatività: $q \in$ ε-chiusura(q)
Possiamo dire che $q' \in$ ε-chiusura(q) se $q' \in \delta(r,\varepsilon) \wedge r \in$ ε-chiusura(q)
r è uno stato raggiungibile da q in ε mosse e visto che dallo stato r con ε mosse arrivo allo stato q' allora $q' \in$ ε-chiusura(q)
Possiamo dire che:

δ^(q,ε)=ε-chiusura(q)
δ^(q,y.a)=ε-chiusura(p)
p={p\ ∃r ∈ δ^(q,y) Λp∈δ(r,a)}
r è uno stato appartenente agli stati raggiungibili da q con input la stringa y considerando le epsilon mosse per ognuno di questi stati ci applico il δ con il simbolo a in input e una volta fatto questo devo applicarci la epsilon chiusura
Il linguaggio riconosciuto dall'automa è così definito:
L(N)={x\ x∈ Σ^* \ δ^ (qo,x) ∩ F≠Φ}
Definire formalmente i linguaggi riconosciuti dagli automi a stati finiti non deterministici con ε-mosse.
Il linguaggio riconosciuto dall'automa è così definito:
L(N)={x\ x∈ Σ^* \ δ^ (qo,x) ∩ F≠Φ}
Definizione automa a stati finiti non deterministici senza ε-mosse
N<Q, ∑, δ, q0, F>
δ=Qx∑→ 2^Q è un'insieme dell parti
Ovviamente può anche fallire e non andare da nessuna parte.
Tutto il resto rimane uguale a quello deterministico
Definiamo il linguaggio riconosciuto dall'automa
δ*=Qx∑*--> 2^Q
Ad ogni passo posso raggiungere più di uno stato.Definiamo δ* in modo induttivo.
δ*(q,ε)=q
δ*(q,a)=δ(q,a)
δ*(q,y.a)=δ*(δ*(q,y),a)=$U_{q' \in \delta^*(q,y)}$ δ*(q',a)
Linguaggio riconosciuto dall'automa
L(N)={x\ x∈ Σ^* \ δ* (qo,x) ∩ F≠Φ}

Definire formalmente i linguaggi riconosciuti dagli automi a stati finiti non deterministici senza ε-mosse.

Il linguaggio riconosciuto dall'automa è così definito:
L(N)={x\ x∈ Σ^* \ δ* (qo,x) ∩ F≠Φ}

Definire formalmente la nozione di linguaggio denotato da una espressione regolare (si inizi definendo la nozione di implicazione)

Per implicazione si intende il passaggio dal generale allo specifico.
Siano e1…..en espressioni regolari, ci sono 4 regole di implicazione:

1) *(e1 ∪ … ∪ e_n) ⇒ e_i 1<=i<=n prende un'espressione regolare "i" qualsiasi appartenente all'insieme generale.(un linguaggio può poi essere denotato da una e.r)*
2) *e^* ⇒ e…..e k volte, k>=0*
3) *e^+ ⇒ e…..e k volte, k>0*
4) *e^n⇒ e….e n volte*

Ogni regola quindi mi fa passare da un linguaggio a uno più piccolo, per esempio nella prima regola se ne sceglie solo una.
e1 → e2 *(e1 deriva e2 se scomponendo e1,e2 si mantengono 2 parti invariate e la parte restante la ottengo tramite regole di implicazione)*

Se il linguaggio di e2 è un sottoinsieme del linguaggio e1, cioè è incluso in e1. Con la derivazione immediata si applica l'implicazione a una parte dell'espressione.
Per derivare si intende :
Siamo e1, e2 espressioni regolari
e1 = αβϑ
e2 = αδϑ
Allora e1→ e2 (e1 deriva e2) se β ⇒ δ (β implica δ)
In questo passaggio si utilizza una regola di implicazione.
e → * e' significa che e' è derivato da e in n >=0 passi.
Oppure potrei scrivere:
e1-->^{n}e' se so quanti passi faccio.
La *definizione* formale è : L(e) :{ x ∈ Σ* | e → * x}
Un linguaggio denotato da un'e.r. è definito nel seguente modo: data una x appartenente all'universo(di quel alfabeto), esiste una espressione regolare e che arriva a x applicando un numero di passi non definito di derivazione.

Definire formalmente l'automa a stati finiti deterministico M = <Q', ∑', δ', q'0, F'> equivalente ad un automa non deterministico N = <Q, ∑, δ, q0, F> con ε-mosse.
Per passare da un automa a stati finiti non deterministico con epsilon mosse a un automa a stati finiti deterministico devo applicare i 2 passi dell'algoritmo.
1)Togliere le epsilon mosse: dobbiamo sostanzialmente passare a un automa N'' nel quale non sono presenti le epsilon mosse da un automa N con le epsilon mosse:
Questo automa N'' sarà uguale a quello di partenza con alcune differenze su δ e F che qua definiamo δ'' e F'' .Quindi Q'' = Q , ∑'' = ∑, q0''= q0.
Ricordiamo che in partenza l'automa con ε mosse aveva un delta definito così:δ:Qx(∑U{ε})-->2^{Q}.
Il δ'' deve essere definito così:δ'':Qx∑-->2^{Q}.
Se io ho un input devo andare esattamente dove mi porterebbe quel input prendendo le scorciatoie.
Quindi il δ'' se partiamo da un automa ε mosse deve essere definito così δ''(q, a) = δ^(q, a)=ε-chiusura(δ(ε-chiusura(q),a))
Ricordo che in questo automa non deterministico senza epsilon mosse per vedere qual'è il linguaggio riconosciuto devo definire δ* che opera sulle stringhe quindi

$\delta^*(q, y.a) = \delta^*(\delta^*(q,y), a)=U_{q' \in \delta^*(q,y)} \delta^*(q',a)$

$$F'' = \begin{cases} F \cup \{q0\} \text{ se } \varepsilon\text{-chiusura}(q0) \cap F \neq \Phi \\ F \text{ altrimenti} \end{cases}$$

2)Ora elimino il non determinismo per passare a un automa deterministico M
Per farlo si utilizza una complessità in spazio esponenziale in modo da avere un riconoscimento in tempo lineare alla lunghezza dell'input.
In partenza ho un automa N= <Q,∑, δ, q0, F> non deterministico senza epsilon mosse e devo definire un automa M= <Q', ∑, δ', q0', F'>
Dove Q' ⊆2^{Q} i suoi stati sono quindi tutti i sottoinsiemi dello stato di partenza(alcuni possono venire esclusi)
qo'=[qo] è lo stesso di quello originale
∑' = ∑
Come stati finali ho tutti quegli stati che contengono almeno uno stato finale dell'automa di partenza.
F'={[qi,...,qj]\ {qi,...,qj} ∩F≠Φ}
sono tutti gli insiemi di stati che contengono almeno uno stato finale

Probabilmente avrò più stati finali rispetto a quelli iniziali.
Per la funzione di transizione avremo:
$\forall$ [qi, ... qj] $\in$ Q' $\wedge$ $\forall$ a $\in \sum$
δ'([qi, ... qj], a) = [pi, ... pk]
se e solo se $U_{q\in\{qi,..,qj\}}$ δ(q, a) = {pi, ... pk}

In pratica come funzione di transizione avrò una funzione che parte da un insieme di stati [q0],[q1],[q2],[q0q1],[q0q2],[q0q1q2] e per ogni insieme di stati prendo tutti gli stati che ho all'interno e ci applico il delta del non deterministico senza epsilon mosse con il simbolo a ($\forall$ a $\in \sum$) (quindi in sostanza vedo dove arrivo con le epsilon chiusure prendendo le scorciatoie)
Quindi in sostanza applico δ'(qi,a) che corrisponde in termini di formula a δ'([qi, ... qk], a) = [pi, ... pk] per farlo facciamo cosi:
δ'([q0],a),δ'([q1],a),δ'([q2],a),δ'([q0q1],a),δ'([q1q2],a),δ'([q0q2],a),δ'([q0,q1,q2],a).
Dove però δ'([q0q1],a),δ'([q1q2],a),δ'([q0q2],a),δ'([q0,q1,q2],a) corrispondono alle unioni dei vari δ'([q0],a),δ'([q1],a),δ'([q2],a) che corrispondono nell automa non deterministico a δ(q0,a),δ(q1,a) e δ(q2,a) e da questo si spiega nella formula: se e solo se $U_{q\in\{qi,..,qj\}}$ δ(q, a) = {pi, ... pk}
Per ogni stato di quell'insieme quindi vedo in che stati vado a finire applicando il delta e ci faccio l'unione
Questa cosa la faccio per ogni insieme di stati di Q' ($\forall$ [qi, ... qk] $\in$ Q')
Ricordiamo che l'automa riconosce tutte e sole le stringhe tale per cui se io parto da uno stato iniziale e analizzo la stringa arrivo a uno stato finale.
Quindi $\delta^*=Qx\sum^*$-->Q
oppure $\delta^*=Qx\sum^*$-->In questo caso fallisco
Definisco δ*:
caso base: δ*(q,a)= δ(q,a) dove a è un simbolo dell'alfabeto
Per ipotesi induttiva posso dire:
$\delta^*(q,y.a)= \delta(\delta^*(q,y),a)$ dove y $\in \sum^n$ mentre a $\in \sum$
Linguaggio riconosciuto dall'automa: $L(M)=\{x \in \sum^* \setminus \delta^*(q,x) \in F\}$

Definire formalmente l'automa a stati finiti non deterministico N = <Q', ∑', δ', q'0, F'> senza ε-mosse equivalente ad un automa non deterministico N = <Q, ∑, δ, q0, F> con ε-mosse.
E' solo il primo passo dell'algoritmo che si trova sopra.

Definire formalmente l'automa a stati finiti deterministico M = <Q', ∑', δ', q'0, F'> equivalente ad un automa non deterministico N = <Q,∑, F, δ, q0, F> senza ε-mosse.
E' solo il secondo passo dell'algoritmo.
2)elimino il non determinismo per passare a un automa deterministico M
Per farlo si utilizza una complessità in spazio esponenziale in modo da avere un riconoscimento in tempo lineare alla lunghezza dell'input.
In partenza ho un automa N= <Q,∑ , δ, q0, F> non deterministico senza epsilon mosse e devo definire un automa M= <Q', ∑', δ', q0', F'>
Dove $Q' \subseteq 2^Q$ i suoi stati sono quindi tutti i sottoinsiemi dello stato di partenza(alcuni possono venire esclusi)
qo'=[qo] è lo stesso di quello originale
∑' = ∑

Come stati finali ho tutti quegli stati che contengono almeno uno stato finale dell'automa di partenza.
F'={[qi,...,qj]\ {qi,...,qj} ∩F≠Φ}
sono tutti gli insiemi di stati che contengono almeno uno stato finale
Probabilmente avrò più stati finali rispetto a quelli iniziali.
Per la funzione di transizione avremo:
∀ [qi, ... qj] ∈ Q' ^ ∀ a ∈ ∑
δ'([qi, ... qj], a) = [pi, ... pk]
se e solo se $U_{q\in\{qi,..,qj\}}$ δ(q, a) = {pi, ... pk}

Dire quando una frase di un linguaggio è ambigua e fare un esempio

Una frase di un linguaggio è ambigua quando possiede piú di un albero di derivazione.
Linguaggio inerentemente ambiguo:
Qualunque grammatica che genera quel linguaggio è ambigua ,e in particolare una grammatica è ambigua se esiste almeno una frase che ha più di un albero di derivazione per ottenerla.
ES(fatto in classe cercate di avere un po di inventiva):
E→ E+E/E*E/d
la frase d1+d2*d3 può avere più di un albero di derivazione, a seconda se svolgo prima il "+" o il "*".

I linguaggi finiti sono sempre regolari? Motivare la risposta

Sono regolari perché formati da un numero finito di frasi, un insieme finito di frasi può essere rappresentato dall'unione di più frasi.
ES: L= {pippo, pluto}
Pippo U pluto è un'espressione regolare in quanto l'unione di espressioni regolari genera un espressione regolare.
A loro volta le frasi sono espressioni regolari in quanto concatenazione di terminali.
pippo = p ·i·p·p·o

Definire formalmente (per casi) le espressioni regolari

1)Linguaggio vuoto :Ø è un e.r
Il linguaggio vuoto è l'annichilitore nell'algebra dele e.r relativamente all'op di concatenamento
2)epsilon è un espressione regolare
3)Per ogni carattere dell'alfabeto Σ quel carattere è un'e.r.
Se s,t sono espressioni regolari allora
4)s.c ,
5) sUt
6)s* sono espressioni regolari.

Dato un linguaggio generato da una grammatica lineare strettamente destro dire se è contenuto nella classe dei linguaggi regolari

Descrivo algoritmo costruttivo che passa da 1 linguaggio generato da una qualunque gramm street lin dx mi va a definire la e.r corrispondente.
Utilizziamo le grammatiche lineari dx e non strettamente lineari destre perché intanto sono equivalenti.
Dobbiamo dimostrare che L(gram lin dx)≡L(er)
Per farlo dobbiamo dimostrare sia ⊇ che ⊆.
Noi dimostreremo solo ⊆.
Quindi dobbiamo dim che L(gram lin dx) sia contenuto in L(e.r)

Per farlo dobbiamo creare un algoritmo che prende in input una grammatica lin dx e mi da un'er
Pensiamo ai non terminali con variabili e ai terminali come costanti.

Si deve trasformare in un sistema di equazione

4 passi:

1) trasformo in un sistema di eq
   $X \dashrightarrow \propto 1/../\propto 2$ con $\propto : \sum^{*} (V \cup \varepsilon)$

   $X = \propto 1 \cup .. \cup \propto 2$

2) Raccolgo le variabili
3) Elimino le ricorsioni dirette
4) Risoluzione mediante Gauss

Dopo il passo 4 se è ancora presente una ricorsione diretta si torna al passo 2 come se fosse un ciclo fino a trovare un'er
Ogni singolo passo preserva il linguaggio perché sostituisco sempre con valori equivalenti come in un'equazione per l'appunto.
Es:
X-->abX / bX / cA / dA /efB/g/hd
1)

X=abX U bX U cA U dA U efB U g U hd
2)
Raccolgo le var:
X=(ab U b)X U (cU d)A U efB U g U hd
rosso= u
blu =v
3)Elimino le ricorsioni dirette
X=u.X U v
Diventa
X=u*.(v)

X=(ab U b)*.((cU d)A U efB U g U hd)
La parentesi è importante!

4)Si applica Gauss sostituendo in questo caso non si sostituisce niente in quanto non avevamo in partenza i valori A e B
Nel caso alla fine avessi ancora delle ricorsioni si torna al punto 2.
Dimostrare che i linguaggi generati dalle grammatiche lineare destre sono regolari.
Probabilmente bastava fare l'algoritmo dei 4 passi fatto sopra.
Qui abbiamo la dimostrazione nella quale si dice che le gramm lineari destre sono propriamente incluse nelle lineari.
Definizione grammatica lineare :
Per definizione è una grammatica context free in cui io uso solo le regole:
$A \rightarrow \propto$
Dove $A \in V$
e $\propto \in \sum^{*} (X \cup \varepsilon) \sum^{*}$
Con $X \in V$

Definizione grammatica lineare dx
$A \rightarrow \propto$
Dove $A \in V$
e $\propto \in \sum^{*} (X \cup \varepsilon)$
Con $X \in V$
I linguaggi generati dalle grammatiche lineari destre sono regolari in quanto non è possibile fare dei bilanciamenti in quanto non abbiamo più la ricorsione centrale.
Per dimostrarlo dobbiamo quindi dimostrare che L(gram. lin)$\supset$L(gram.lin dx)
Quindi dobbiamo dimostrare che non sono uguali ma solo che le gramm. lin dx sono incluse in quelle lineari.
1)Per definizione noi sappiamo che i linguaggi generati da grammatiche lineari destre sono inclusi nei linguaggi generati dalle grammatiche lineari.
2) Dobbiamo trovare un esempio per cui $\supset$ questa inclusione sia stretta.
Es: $\sum=\{a,b\}$
$L_{\sum}=\{a^n b^n \backslash n>=0\}$
In questo caso non riusciamo a generare un linguaggio del genere a partire da una grammatica lineare destra in quanto non riusciamo a fare i bilanciamenti.

Quando un linguaggio è deterministico? I linguaggi finiti sono deterministici? E i linguaggi regolari? E i linguaggi context free? Motivare le risposte.
Un linguaggio è deterministico quando può essere definito da un automa in cui
$\forall$ stato $q \in Q$ e
$\forall$ terminale in input $a \in \sum$,
esiste al più una sola regola di transizione $\delta(q, a)$
I linguaggi finiti sono regolari, ed i linguaggi regolari sono deterministici.
I linguaggi context free non sono deterministici, un esempio di linguaggio context free è il linguaggio delle palindrome senza marca di centro, che può essere riconosciuto solo con un automa a pila non deterministico.
Quando un linguaggio è deterministico?
Un linguaggio è deterministico quando può essere definito da un automa in cui
$\forall$ stato $q \in Q$ e
$\forall$ terminale in input $a \in \sum$,
esiste al più una sola regola di transizione $\delta(q, a)$
$Q \times \sum \rightarrow Q$
I linguaggi regolari sono tutti deterministici?Motivare la risposta
Sì, perché sono finiti e riconosciuti da automi deterministici.

Il linguaggio delle palindromi (con e senza marca di centro) è regolare? E' deterministico? Commentare le risposte

Il linguaggio delle palindromi con e senza marca da centro non sono regolari perché ci sono delle dipendenze che con un linguaggio generato da una grammatica regolare non riesco a ottenere mentre per quanto riguarda il determinismo , quello con marca da centro è deterministico perchè si può prende una sola strada per esempio abcba dato che abbiamo la marca da centro riusciamo a dare un'unica strada cioè a destra della marca di centro il primo termine non ancora visitato è una b poi vado a sinistra e il primo termine non ancora visitato dopo la marca di centro deve essere una b e così via.Mentre se siamo in assenza di marca da centro abbiamo 2 possibilità ad ogni iterazione o il simbolo che viene considerato è una marca da centro oppure non lo è quindi dovremo tenere in memoria le 2 possibilità e fare una fork .