1. ELIMINAZIONE PRODUZIONI EPSILON

Si ha che se L è un CFL, allora  ha una grammatica priva

di produzioni epsilon.

La variabile A è annullabile se:



Sia A annullabile. Rimpiazzeremo una regola del tipo:

 con 

e cancelleremo tutte le regole con corpo epsilon.

ALGORITMO PER CALCOLARE SIMBOLI ANNULLABILI:

Data una grammatica G = (V; T; P; S),

L’insieme dei simboli annullabili è n(G).

**Base:**

Se A produce epsilon, allora A è annullabile.



**Induzione:**

Se c’è una produzione A che produce tutti simboli annullabili, allora A è annullabile:



Allora 

ESEMPIO:

Sia G la grammatica:



Per prima cosa, troviamo tutti i simboli annullabili n(G).

Sicuramente A e B fanno parte di n(G) perché producono epsilon direttamente, e quindi anche S, perché produce tutti i simboli annullabili AB.

n(G) = {A,B,S}

Per prima cosa cambiamo S, con la regola vista prima.

Per ogni simbolo annullabile aumentano le combinazioni possibili di presenza/assenza di ogni simbolo annullabile.

In questo caso però non possono essere assenti tutti e 2 i simboli perché sennò rimarrebbe solo epsilon.

Quindi S diventa:



Per A possono anche essere assenti entrambe le A, perché c’è comunque il terminale a che non fa produrre epsilon.

Ci sono quindi 4 combinazioni possibili per A:



La seconda e la terza produzione sono le stesse.

La stessa cosa vale anche per B:



La nuova grammatica G1 quindi diventa:



1. ELIMINAZIONE PRODUZIONI UNITA’

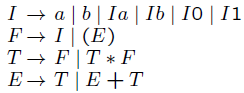
Date le variabili A e B,



È una produzione unità.

Le produzioni unità possono essere eliminate.

Esempio: data la grammatica:



Ha le produzioni unità: 

Esempio di espansione per eliminare la produzione unità :

Si espande la variabile T e si aggiunge alle produzioni di E tutte le produzioni di T:



La stessa cosa si Fa per eliminare la produzione unità F:



E infine si fa anche per la produzione unità I:



**Nota:**

Questo metodo non funziona se ci sono dei cicli nelle produzioni, esempio:



Per avere un metodo funzionante anche per questo tipo di grammatiche bisogna usare un metodo basato sulle coppie unità.

METODO DELLE COPPIE UNITA’ PER ELIMINARE LE PRODUZIONI UNITA’

(A;B) e' una coppia unità se  usando solamente produzioni unità.

**Nota:**

Se si ha la grammatica:



Allora  ma non usando solamente produzioni unità.

Data una grammatica G = (V; T; P; S),

L’insieme di tutte le coppie unità è u(G).

**Base:**

Ogni variabile forma una coppia unità con se stessa:



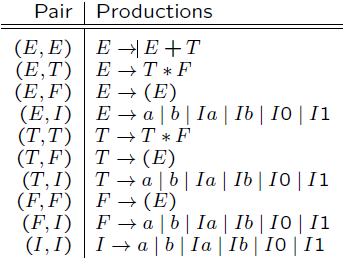
Cioè:  

**Induzione:**



Allora aggiungi (A,C) a u(G).

Esempio, data la grammatica di prima:



Questa nuova grammatica accetta lo stesso linguaggio senza produzioni unità.

1. ELIMINAZIONE SIMBOLI INUTILI:

Un simbolo X è **utile** per una grammatica G = (V; T; P; S), se

esiste una derivazione



per una stringa di terminali w.

Simboli che non sono utili sono detti inutili.

Un simbolo X è **generante** se X



per qualche w € T

Un simbolo X e' **raggiungibile** se



Se eliminiamo prima i simboli non generanti, e poi quelli non raggiungibili, rimarranno solamente simboli utili.

ESEMPIO:

(Prima si eliminano i simboli non generanti, poi i simboli non raggiungibili, altrimenti si può avere una grammatica con ancora simboli inutili)

Consideriamo la grammatica G:



S e A sono generanti, B non lo e'. Se eliminiamo B dobbiamo

eliminare anche:



Altrimenti si ha S che produce solo A, e non è più la forma normale di Chomsky.

Quindi rimane la grammatica:



Ora solo S è raggiungibile, A non è raggiungibile (ma è generante).

Si elimina quindi A e rimane la grammatica semplificata:



**Nota:**

Se avessimo eliminato prima i simboli non raggiungibili e poi i simboli non generanti, avremmo rimasto la stessa grammatica, in quanto tutti i simboli sono raggiungibili (vedi algoritmo sotto).

ALGORITMO PER CALCOLARE SIMBOLI GENERANTI:

Data una grammatica G = (V; T; P; S),

L’insieme dei simboli generanti si indica con g(G).

**Base:**

Ogni terminale genera se stesso:

g(G) = T

**Induzione:**

Se A produce simboli che si sanno già essere generanti, allora A è generante.



Nell’esempio di prima, i simboli generanti g(G) all’inizio sono tutti i terminali, cioè {a,b}.

Poi S produce a, quindi si aggiunge S in g(G), poi A produce b, quindi A si aggiunge a g(G).

g(G) = {a,b,S,A}

ALGORITMO PER CALCOLARE SIMBOLI RAGGIUNGIBILI:

Data una grammatica G = (V; T; P; S),

L’insieme dei simboli raggiungibili si indica con r(G).

**Base:**

Il simbolo iniziale S è sicuramente raggiungibile.



**Induzione:**

Se A è raggiungibile, allora tutti i simboli delle produzioni che hanno A in testa sono raggiungibili.

Se:



Allora aggiungi tutti i simboli in  in r(G).

Esempio di prima:

Inizialmente r(G) = {S}

Poi si aggiunge ad r(G) A,B, ed a, prodotti da S.

Infine si aggiunge b, prodotto da A.

r(G) = {S,A,B,a,b}

FORMA NORMALE DI CHOMSKY:

Una volta che abbiamo pulito la grammatica, possiamo trasformare la grammatica nella CNF (Chomsky Normal Form):



A,B,C sono variabili in V, a è un terminale in T.

Per ottenere una grammatica in CNF bisogna:

* Pulire la grammatica
* Modificare le produzioni con 2 o più simboli in modo tale che siano tutte variabili
* Ridurre il corpo delle regole di lunghezza superiore a 2 in cascate di produzioni con corpi da 2 variabili.

Per il passo 2, per ogni terminale a che compare in un corpo

di lunghezza >= 2, creare una nuova variabile, ad esempio A, e

sostituire a con A in tutti i corpi, e aggiungere la nuova regola

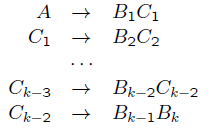


Per il passo 3, per ogni regola nella forma:



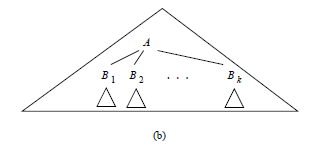
k >= 3, introdurre le nuove variabili C1,C2,…Ck-2, e sostituire la

regola con:

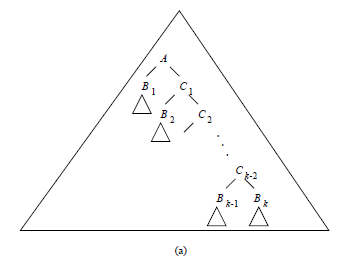


Effetti del passo 3:

Prima:

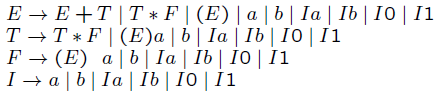


Dopo:



ESEMPIO CONVERSIONE GRAMMATICA IN CNF:

Partiamo dalla grammatica pulita precedentemente:

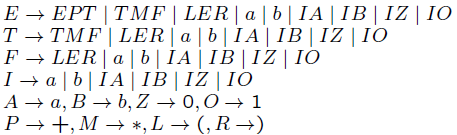


Per il passo 2 dobbiamo trovare tutte le regole di produzione che hanno almeno 2 simboli, e sostituire ogni singolo terminale con una nuova variabile.

Es:

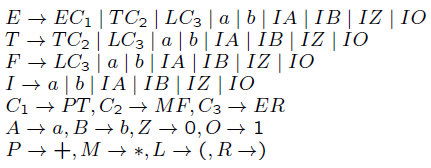
E + T diventa EPT, dove P produce +

Se si fa il passo 2 per ogni terminale, si ottiene la grammatica:



Per il passo 3, rimpiazziamo ad esempio EPT con EC1, dove C1 produce PT.

Quindi la grammatica in CNF finale è:



**ESERCIZI**

**7.1.1: elimina simboli inutili dalla grammatica:**

S → AB | CA

A → a

B → BC | AB

C → aB | b

Prima si trova l'insieme dei simboli generanti g(G):

g(G) = {a,b,A,C,S}

a,b per la base dell'algoritmo.

A, C perché generano simboli terminali

S perché genera CA, che sono tutti generanti.

Si elimina quindi dalla grammatica il simbolo B, non generante:

S → CA

A → a

C → b

Adesso si calcola l'insieme dei simboli raggiungibili:

r(G) = {S,A,C}

S per la base dell'algoritmo

A e C perché sono nella produzione di S.

Quindi tutti i simboli sono raggiungibili, e la grammatica pulita e quella trovata prima.

**7.1.2 pulisci la grammatica e trova la CNF della seguente grammatica:**

S → ASB | epsilon

A → aAS | a

B → SbS | A | bb

1) Primo passo della pulizia: si eliminano le produzioni epsilon:

Si calcola l'insieme delle variabili nullable:

**Base**: una variabile è nullable se produce direttamente epsilon

**Induzione**: una variabile è nullable se produce tutte variabili nullable

n(G) = {S}

Per ogni variabile nullable, si riscrive la grammatica con tutte le combinazioni di presenza / assenza della variabile (senza generare nuovamente epsilon):

S → ASB | AB

A → aAS | aA | a

B → SbS | Sb | bS | b | A | bb

2) Secondo passo della pulizia: si eliminano le produzioni unità:

(B,B) → SbS | Sb | bS | b | bb

(B,A) → aAS | aA | a

La grammatica quindi diventa:

S → ASB | AB

A → aAS | aA | a

B → SbS | Sb | bS | b | bb | aAS | aA | a

3) Terzo passo della pulizia: si eliminano i simboli inutili: (prima i non generanti, poi i non raggiungibili)

trovo l'insieme dei simboli generanti:

g(G) = {a,b,A,B,S}, sono tutti generanti

trovo l'insieme dei simboli raggiungibili:

r(G) = {S,A,B} sono tutti raggiungibili

Quindi non ci sono simboli inutili nella grammatica di prima.

4) Dalla grammatica pulita, si costruisce la CNF (Chomsky Normal Form)

- tutte le produzioni con almeno 2 simboli, diventano con tutte variabili

- per ogni produzione con almeno 3 simboli, del tipo A → B1 B2 … Bk, k >= 3, si aggiungono (k-2) variabili C1 C2 … C(k-2), e le produzioni diventano del tipo:

A → B1 C1, C1 → B2 C2, … C(k-3) → B(k-3) C(k-2), C(k-2) → B(k-1) Bk

quindi: trasformo tutte le produzioni di lunghezza >= 2 in tutte varabili:

S → ASB | AB

A → XAS | XA | a

B → SYS | SY | YS | b | YY | XAS | XA | a

X → a

Y → b

poi spezzo tutte le produzioni di lunghezza >= 3:

S → AD | AB

D → SB

A → XE | XA | a

E → AS

B → SF | SY | YS | b | YY | XE | XA | a

F → YS

X → a

Y → b

Questa è la grammatica finale in CNF.