7. Automi a Pila e Grammatiche Libere

Nicola Fanizzi (*fanizzi@di.uniba.it*)

Dipartimento di Informatica
Università degli Studi di Bari

20 aprile 2016

Automi con Memoria

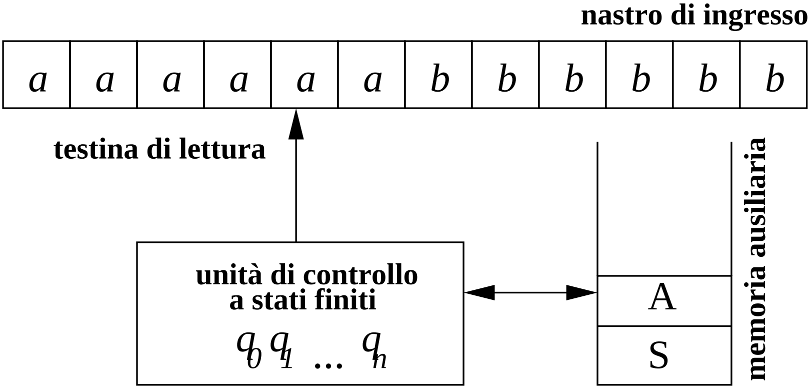
Per *riconoscere* linguaggi superiori rispetto a quelli di $\mathcal{L}_{FSL}$ occorrono altri strumenti più potenti:
Dato un alfabeto finito X di ingresso:

nastro di ingresso: contiene i simboli dell'alfabeto di ingresso X; su un simbolo insiste una *testina di lettura*

memoria ausiliaria: ha capacità virtualmente illimitata ed un proprio *alfabeto di memoria* (o *di lavoro*)
Se l'organizzazione della memoria è uno stack, l'automa risultante si definirà *automa a pila* (o *automa push-down*, *PDA*)

unità di controllo: controlla le transizioni dell'automa, in base al contenuto della memoria e del simbolo letto dalla testina sul nastro.

Modello di PDA



Un *automa a pila non deterministico* è una n-pla:

$$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

- Q insieme finito e non vuoto degli *stati*
- X insieme finito e non vuoto detto *alfabeto di ingresso*
- Γ insieme finito e non vuoto detto *alfabeto della pila*
- δ funzione di transizione:

  $$\delta : Q \times (X \cup \lambda) \times \Gamma \longrightarrow \wp(Q \times \Gamma^*)$$

  scritta anche come $(q', \sigma) \in \delta(q, x, Z)$ ovvero (q, x, Z, q', σ)
- $q_0 \in Q$ è lo *stato iniziale*
- $Z_0 \in \Gamma$ è il *simbolo iniziale* della pila
- $F \subseteq Q$ è l'insieme degli *stati finali*

Descrizioni Istantanee

Una *descrizione istantanea* (*ID*) per un PDA è una terna

$$(q, w, \sigma) \in Q \times X^* \times \Gamma^*$$

- $q \in Q$ è lo *stato corrente* dell'unità di controllo
- $w = x_1 x_2 \cdots x_n \in X^*$ è la sottostringa della stringa sul nastro ancora da esaminare (testina posizionata su x_1)
- $\sigma = Z_1 Z_2 \cdots Z_m$ è il contenuto della pila con Z_1 in cima e Z_m al fondo

Servono a descrivere lo *stato globale* di un PDA in ogni istante.

descrizione iniziale (q_0, w, Z_0)

descrizione finale (q, λ, σ) con $q \in F$, $\sigma \in \Gamma^*$

Sia M nello stato q, sia x letto dalla testina e A in cima alla pila:

1. se δ contiene $(q, x, A, q', A_1 \cdots A_k)$
   ossia se $(q', A_1 \cdots A_k) \in \delta(q, x, A)$
   M *può* operare la transizione:

$$(q, xw, A\sigma) \Longrightarrow (q', w, A_1 \cdots A_k\sigma)$$

2. se δ è descritta dalla n-pla $(q, \lambda, A, q', A_1 \cdots A_k)$
   ossia se $(q', A_1 \cdots A_k) \in \delta(q, \lambda, A)$
   allora la sua esecuzione può provocare la transizione:

$$(q, w, A\sigma) \Longrightarrow (q', w, A_1 \cdots A_k\sigma)$$

$\stackrel{*}{\Longrightarrow}$ è la chiusura riflessiva e transitiva dell'operatore $\Longrightarrow$
Con $ID_1 \stackrel{*}{\Longrightarrow} ID_2$ si indica la pssibilità di transitare dalla descrizione ID_1 alla ID_2 in un numero finito (anche nullo) di passi

Condizioni di Accettazione

$w \in X^*$ è *accettata* dal PDA M **in condizione di stato finale** sse:

$$(q_0, w, Z_0) \stackrel{*}{\Longrightarrow} (q, \lambda, \sigma) \qquad q \in F \text{ e } \sigma \in \Gamma^*$$

linguaggio accettato da M **in condizione di stato finale**:

$$T(M) = \{w \in X^* \mid (q_0, w, Z_0) \stackrel{*}{\Longrightarrow} (q, \lambda, \sigma) \text{ con } q \in F \text{ e } \sigma \in \Gamma^*\}$$

$w \in X^*$ è *accettata* dal PDA M **in condizione di pila vuota** sse:

$$(q_0, w, Z_0) \stackrel{*}{\Longrightarrow} (q, \lambda, \lambda) \qquad q \in Q$$

linguaggio accettato da M *in condizione di pila vuota*:

$$T(M) = \{w \in X^* \mid (q_0, w, Z_0) \stackrel{*}{\Longrightarrow} (q, \lambda, \lambda) \text{ con } q \in Q\}$$

Teorema.
La classe dei linguaggi accettati da PDA in condizione di pila vuota è equivalente alla classe dei linguaggi accettati in condizione di stato finale

Cenni sulla Dimostrazione.
Se si raggiunge uno stato finale e la pila non è vuota occorre ripulire la pila dai simboli restanti senza cambiare stato

Se la pila è vuota bisogna obbligare l'automa a transitare in uno stato finale senza modificare la situazione della pila

Esempio 1. $L = \{w \in \{a,b\}^* \mid w$ ha lo stesso numero di a e di $b\}$

$G = (X, V, S, P)$

- $X = \{a, b\}$
- $V = \{S\}$
- $P = \{S \longrightarrow ab \mid ba \mid SS \mid aSb \mid bSa\}$

$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$

- $Q = \{q_0\}$
- $\Gamma = \{Z_0, A, B\}$
- $F = \emptyset$

simbolo pila	stato	a	b
A	q_0	AA	λ
B	q_0	λ	BB
Z_0	q_0	AZ_0	BZ_0

Programma per PDA (è omesso lo stato q_0):

1. (a, Z_0, AZ_0)
2. (b, Z_0, BZ_0)
3. (a, A, AA)
4. (b, B, BB)
5. (a, B, λ)
6. (b, A, λ)
7. (λ, Z_0, λ) λ-regola per cancellare il fondo della pila

esempio di elaborazione

stringa *abba*

ID iniziale: $(q_0, abba, Z_0) \Longrightarrow (q_0, bba, AZ_0) \Longrightarrow (q_0, ba, Z_0) \Longrightarrow (q_0, a, BZ_0) \Longrightarrow (q_0, \lambda, Z_0) \Longrightarrow (q_0, \lambda, \lambda)$

la stringa è accettata.

---

stringa *aababa*

ID iniziale: $(q_0, aababa, Z_0) \Longrightarrow (q_0, ababa, AZ_0) \Longrightarrow (q_0, baba, AAZ_0) \Longrightarrow (q_0, aba, AZ_0) \Longrightarrow (q_0, ba, AAZ_0) \Longrightarrow (q_0, a, AZ_0) \Longrightarrow (q_0, \lambda, AAZ_0)$

la stringa non è accettata.

Esempio 2.

$L = \{wcw^R \mid w \in \{a, b\}^*\}$

$G = (X, V, S, P)$

- $X = \{a, b, c\}$
- $V = \{S\}$
- $P = \{S \longrightarrow c \mid aSa \mid bSb\}$

$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$

- $Q = \{q_0, q_1\}$ $\quad q_0 = \text{lettura} \quad q_1 = \text{match}$
- $\Gamma = \{Z_0, A, B\}$
- $F = \emptyset$

simbolo pila	stato	a	b	c
A	q_0	(q_0, AA)	(q_0, BA)	(q_1, A)
A	q_1	(q_1, λ)	-	-
B	q_0	(q_0, AB)	(q_0, BB)	(q_1, B)
B	q_1	-	(q_1, λ)	-
Z_0	q_0	(q_0, AZ_0)	(q_0, BZ_0)	(q_1, Z_0)
Z_0	q_1	-	-	-

Si osservi che c segnala il cambiamento di stato interno (centro della stringa palindroma) q_1 nel quale si comincia a svuotare la pila riempita nello stato q_0

1. (q_0, a, Z_0, q_0, AZ_0)
2. (q_0, a, A, q_0, AA)
3. (q_0, a, B, q_0, AB)
4. (q_0, b, Z_0, q_0, BZ_0)
5. (q_0, b, A, q_0, BA)
6. (q_0, b, B, q_0, BB)
7. (q_0, c, Z_0, q_1, Z_0)
8. (q_0, c, A, q_1, A)
9. (q_0, c, B, q_1, B)
10. $(q_1, a, A, q_1, \lambda)$
11. $(q_1, b, B, q_1, \lambda)$
12. $(q_1, \lambda, Z_0, q_1, \lambda)$ λ-regola

Programma per PDA:

- la 1. la 2. e la 3. si possono riassumere con: $(q_0, a, Z, q_0, AZ)\ \forall Z \in \Gamma$
- la 4. la 5. e la 6. si possono riassumere con: $(q_0, b, Z, q_0, BZ)\ \forall Z \in \Gamma$
- la 7. la 8. e la 9. si possono riassumere con: $(q_0, c, Z, q_1, Z)\ \forall Z \in \Gamma$

Esempio 3.

$L = \{ww^R \mid w \in \{a,b\}^*\}$

$G = (X, V, S, P)$

- $X = \{a, b\}$
- $V = \{S\}$
- $P = \{S \longrightarrow \lambda \mid aSa \mid bSb\}$

$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$

- $Q = \{q_0, q_1\}$ $\quad q_0 =$ lettura $\quad q_1 =$ match
- $\Gamma = \{Z_0, A, B\}$
- $F = \emptyset$

simbolo pila	stato	a	b
A	q_0	$\{(q_0, AA), (q_1, \lambda)\}$	(q_0, BA)
A	q_1	(q_1, λ)	-
B	q_0	(q_0, AB)	$\{(q_0, BB), (q_1, \lambda)\}$
B	q_1	-	(q_1, λ)
Z_0	q_0	(q_0, AZ_0)	(q_0, BZ_0)
Z_0	q_1	-	-

Programma per PDA:

1. $(q_0, a, Z, q_0, AZ)\ \forall Z \in \Gamma$
2. (q_0, b, Z, q_0, BZ)
3. $(q_0, \lambda, Z, q_1, Z)$ (invece della λ-regola precedente)
4. $(q_1, a, A, q_1, \lambda)$
5. $(q_1, b, B, q_1, \lambda)$
6. $(q_1, \lambda, Z_0, q_1, \lambda)$

automa non deterministico con due possibilità, trovandosi in q_0

- leggere il prossimo simbolo dal nastro
- passare in q_1

esempio di elaborazione

stringa *abba*

$(q_0, abba, Z_0) \Longrightarrow (q_0, bba, AZ_0) \Longrightarrow (q_0, ba, BAZ_0)$
$\overset{1}{\Longrightarrow} \underline{(q_0, a, BBAZ_0) \Longrightarrow (q_0, \lambda, ABBAZ_0)}.$
$\overset{5}{\Longrightarrow} (q_1, a, AZ_0) \Longrightarrow (q_1, \lambda, Z_0) \Longrightarrow (q_1, \lambda, \lambda)$

la stringa è accettata.

Osservazioni.

- $L_1 = \{wcw^R \mid w \in \{a, b\}^*\}$
  accettato da un PDA deterministico
- $L_2 = \{ww^R \mid w \in \{a, b\}^*\}$
  accettato da un PDA non deterministico
- si può dimostrare che

$$\nexists M \in PDA \text{ deterministico tale che } T(M) = L_2$$

  pertanto la classe dei linguaggi riconosciuta dai PDA deterministici è inclusa strettamente in quella dei linguaggi riconosciuti da PDA non determinisitici

Forme Normali

Esempio.
$G = (X, V, S, P)$

- $X = \{0, 1, 2\}$
- $V = \{S, A, B\}$
- $P = \{S \longrightarrow 0SAB \mid 1, \quad A \longrightarrow 1A \mid 1, \quad B \longrightarrow 2B \mid 2\}$

Data la forma delle produzioni, la lettura del primo simbolo (terminale) può essere usata in modo predittivo per decidere il resto della stringa che si dovrà derivare

Costruiamo l'automa a pila equivalente: $M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$

- $Q = \{q_0\}$
- $\Gamma = \{S, A, B\}$ con $Z_0 = S$
- $F = \emptyset$

top pila	stato	0	1	2
S	q_0	(q_0, SAB)	(q_0, λ)	
A	q_0		$\{(q_0, A), (q_0, \lambda)\}$	
B	q_0			$\{(q_0, B), (q_0, \lambda)\}$

Teorema. Sia $G = (X, V, S, P)$ una grammatica libera con produzioni del tipo $A \longrightarrow x\alpha$ con $x \in X, \alpha \in V^*$.
Allora esiste un PDA M, tale che $L(G) = T(M)$ con:

$$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

- $Q = \{q_0\}$
- $\Gamma = V$
- $Z_0 = S$
- $F = \emptyset$
- $\forall A \longrightarrow x\alpha \in P\colon\ (q_0, \alpha) \in \delta(q_0, x, A)$

Occorre riportare le produzioni di una grammatica libera in una forma particolare per poter effettuare il passaggio ad un automa riconoscitore mediante questo teorema

Forma Normale di Chomsky

Una grammatica libera $G = (X, V, S, P)$ è
in **forma normale di Chomsky**
se ogni produzione è di uno dei tipi seguenti:

1. $S \longrightarrow \lambda$
2. $A \longrightarrow BC$
   con $A \in V$ e $\begin{cases} B, C \in V \setminus \{S\} & \text{se } S \longrightarrow \lambda \in P \\ B, C \in V & \text{altrimenti} \end{cases}$
3. $A \longrightarrow a$
   con $A \in V, a \in X$

Forma Normale NLR

Una grammatica libera $G = (X, V, S, P)$ è in **forma normale priva di ricorsioni sinistre** (*NLR*, *No Left Recursion*)
se non ha produzioni del tipo:

$$A \longrightarrow Av$$

dove $A \in V, v \in (V \cup X)^*$

Forma Normale di Greibach

Una grammatica libera $G = (X, V, S, P)$
è in **forma normale di Greibach** (*GNF*, *Greibach Normal Form*)
se ogni produzione è del tipo:

1. $S \longrightarrow \lambda$
2. $A \longrightarrow x\alpha$
   dove $A \in V, \quad x \in X, \quad \alpha \in V^*$

Passaggio alle Forme Normali

Teorema. Sia G una grammatica libera. Allora esistono $G_i,\ i = 1, 2, 3$ equivalenti a G (cioè $L(G) = L(G_i)$) tali che

- G_1 è in forma normale di Chomsky
- G_2 è in forma normale di Greibach
- G_3 è in forma normale NLR priva di ricorsioni sinistre

Dim. (costruttivamente)

- da G a G_1 CNF con l'Algoritmo 1
- da G_1 a G_2 GNF con l' Algoritmo 2
- da G_2 a G_3 NLR come passo dell'Algoritmo 2

Esempio Conduttore Algoritmo 1.

Grammatica libera $G = (X, V, S, P)$ con

$X = \{0, 1, 2\}$

$V = \{S, A, B\}$

$P = \{S \longrightarrow 00A \mid B \mid 1,\ A \longrightarrow 1AA \mid 2,\ B \longrightarrow 0\}$

Algoritmo 1

input: $G = (X, V, S, P)$ grammatica libera
output: $G_1 = (X, V, S, P)$ in CNF
Passo 1.
Conversione dei terminali che compaiono nelle parti destre di produzioni in non terminali
Aggiunta delle produzioni appropriate per tali non terminali:

$$S \longrightarrow \mathbf{B}\mathbf{B}A \mid B \mid 1$$
$$A \longrightarrow \mathbf{C}AA \mid 2$$
$$\mathbf{C} \longrightarrow \mathbf{1}$$
$$B \longrightarrow 0$$

Passo 2.
Suddivisione delle produzioni in cui le parti destre hanno più di due simboli non terminali

$S \longrightarrow B\mathbf{D} \mid B \mid 1$
$\mathbf{D} \longrightarrow \mathbf{BA}$
$A \longrightarrow C\mathbf{E} \mid 2$
$\mathbf{E} \longrightarrow \mathbf{AA}$
$C \longrightarrow 1$
$B \longrightarrow 0$

Passo 3.
Sostituzione dei non terminali che costituiscono, da soli, parti destre di qualche produzione

$$
\begin{aligned}
S &\longrightarrow BD \mid \mathbf{0} \mid 1 \\
D &\longrightarrow BA \\
A &\longrightarrow CE \mid 2 \\
E &\longrightarrow AA \\
C &\longrightarrow 1 \\
B &\longrightarrow 0
\end{aligned}
$$

Esempio Conduttore Algoritmo 2.

Si consideri la grammatica $G = (X, V, S, P)$ con:

$X = \{a, b, c, d\}$

$V = \{S, A, B, C, D\}$

$P = \{$

$$
\begin{aligned}
S &\longrightarrow AaB, \\
A &\longrightarrow \lambda, \\
B &\longrightarrow CD \mid c, \\
C &\longrightarrow BC \mid d, \\
D &\longrightarrow ab \mid a
\end{aligned}
$$

$\}$

Algoritmo 2

input: $G = (X, V, S, P)$ in CNF
output: $G_1 = (X, V, S, P)$ in GNF
Passo 1. Eliminazione delle λ-regole
Dividiamo V in due parti $V_1 = \{A \in V \mid A \overset{*}{\Longrightarrow} \lambda\}$ e $V_2 = V \setminus V_1$
Algoritmo di calcolo di $e(A) =$ vero sse $A \overset{*}{\Longrightarrow} \lambda$:

1. $\forall A \in V: \;\; e(A) \leftarrow$ false
2. $\forall A \longrightarrow \lambda \in P: \;\; e(A) \leftarrow$ true
   e si marcano (con ') le occorrenze di A che appaiono nelle parti destre delle produzioni di G
3. $\forall A \longrightarrow \alpha \in P$ si cancellano da α i non terminali marcati;
   se la parte destra diventa vuota allora $e(A) \leftarrow$ true
   e si marcano tutte le A occorrenti in parti destre
4. se nel passo (3) qualche $e(A)$ è mutato allora torna al passo (3) altrimenti termina

Nell'esempio precedente costruiamo le nuove produzioni P_2:

$e()$	(1)	(2)	(3)	(4)
S	0	0	0	0
A	0	1	1	1
B	0	0	0	0
C	0	0	0	0
D	0	0	0	0

quindi $V_1 = \{A\}$

Per il lemma della stringa vuota:

1. $S \longrightarrow \lambda \in P_2$ sse $S \overset{*}{\Longrightarrow} \lambda$ (quando $\lambda \in L(G)$)
2. Se $A \longrightarrow X_1 X_2 \ldots X_r, \quad r \geq 1$
   allora apparterranno alla nuovo insieme P_2 le produzioni

$$A \longrightarrow Y_1 Y_2 \ldots Y_r$$

   con:

$$Y_i = \begin{cases} X_i & \text{se } X_i \in X \cup V_2; \\ X_i \mid \lambda & \text{se } X_i \in V_1 \quad \text{(2 prod. per ogni scelta di } X_i\text{)} \end{cases}$$

$$\begin{aligned} P_2 = \{\ & S \longrightarrow \mathbf{aB} \mid \mathbf{AaB}, \\ & A \longrightarrow \lambda, \qquad \text{(si può eliminare)} \\ & B \longrightarrow CD \mid c, \\ & C \longrightarrow BC \mid d, \\ & D \longrightarrow ab \mid a\ \} \end{aligned}$$

Passo 2. Eliminazione dei non terminali A inutili

- A non genera alcuna stringa terminale
- da S non deriva alcuna forma di frase contenente A

Algoritmo per il calcolo di $t(A)$ e $s(A)$

$$A \in V \quad \begin{array}{lll} t(A) = \text{true} & \text{sse} & A \stackrel{*}{\Longrightarrow} w, \; w \in X^* \\ s(A) = \text{true} & \text{sse} & S \stackrel{*}{\Longrightarrow} \alpha A \beta, \quad \alpha, \beta \in (X \cup V)^* \end{array}$$

1. $\forall A \in V : t(A) \leftarrow$ falso
2. se $A \longrightarrow x \in P, \quad x \in X^*$ allora $t(A) \leftarrow$ vero
   e si marcano con un apice t tutte le occorrenze di A in parti destre di produzioni di P
3. $s(S) \leftarrow$ vero
   e si marcano con un apice s tutte le occorrenze di S in parti sinistre di produzioni di P
4.
   - $\forall A \longrightarrow \alpha \in P$ se i non terminali di α sono marcati t allora si marcano con t tutte le A in parti destre di P
   - se A è marcato s allora $s(B) \leftarrow$ vero $\quad \forall A \longrightarrow \alpha B \beta$
     e si marcano con s tutte le occorrenze di B in parti sinistre di P
5. se nel passo 4. qualcosa è mutato allora torna al passo 4.
6. $\forall A$ tale che $t(A) =$ falso oppure $s(A) =$ falso: Cancella da P tutte le produzioni in cui compare A.

	t	s	t	s	t	s	t	s	t	s
S	0	0	0	1	1	1	1	1	1	1
A	0	0	0	0	0	1	0	1	0	1
B	0	0	1	0	1	1	1	1	1	1
C	0	0	1	0	1	0	1	1	1	1
D	0	0	1	0	1	0	1	1	1	1

$S^s \longrightarrow aB^t \mid \underline{\mathbf{AaB^t}}$
$B^s \longrightarrow C^t D^t \mid c$
$C^s \longrightarrow B^t C^t \mid d$
$D^s \longrightarrow ab \mid a$
essendo A il non terminale inutile,
quindi $S \longrightarrow AaB$ può essere eliminata

Passo 3. Eliminazione dei non terminali ciclici $A \stackrel{+}{\Longrightarrow} A$

Algoritmo per il calcolo di $c(A) = \{B \mid A \stackrel{+}{\Longrightarrow} B\}$

1. $\forall A \in V: \ c(A) \leftarrow \emptyset$
2. $\forall A \longrightarrow \alpha \in P$ con $\alpha = \beta B \gamma$ e $\beta \stackrel{*}{\Longrightarrow} \lambda$ e $\gamma \stackrel{*}{\Longrightarrow} \lambda$
   si pone $c(A) \leftarrow c(A) \cup \{B\}$
3. $\forall A \in V$ se $B \in c(A)$ allora $c(A) \leftarrow c(A) \cup c(B)$
4. se al passo 3. è mutata la composizione di un insieme $c(\cdot)$
   allora torna a 2.
   altrimenti STOP

Nel nostro caso:

$c(\cdot)$	1	2	3
S	$\emptyset$	$\emptyset$	$\emptyset$
B	$\emptyset$	$\emptyset$	$\emptyset$
C	$\emptyset$	$\emptyset$	$\emptyset$
D	$\emptyset$	$\emptyset$	$\emptyset$

quindi non vi sono non terminali ciclici

Passo 4. Eliminazione delle produzioni $A \longrightarrow B$

Algoritmo

1. Partizionare P in P_1 e P_2 ove
   $P_2 = \{A \longrightarrow B \in P \mid A, B \in V\}$
2. $P_1 \leftarrow P_1 \cup \{A \longrightarrow \alpha \mid B \in c(A) \wedge B \longrightarrow \alpha \in P_1\}$

NB: Si possono produrre NT ciclici.

Nell'esempio non vi sono tali produzioni

Passo 5. Eliminazione delle produzioni $A \longrightarrow B\beta$

Algoritmo

Per ognuna di queste produzioni:

1. Se $B \longrightarrow \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n \in P$
   allora sostituisco $A \longrightarrow B\beta$ con $A \longrightarrow \alpha_1\beta \mid \alpha_2\beta \mid \cdots \mid \alpha_n\beta$
2. Se $\forall i \in [1, n]$: α_i inizia con un terminale
   allora termina per $A \longrightarrow B\beta$
   altrimenti se $\exists \alpha_k = C\gamma$ ripetere per $A \longrightarrow C\gamma\beta$

NB: Si possono generare NT inutili

L'algoritmo termina sse (ipotesi):

- non terminali inutili eliminati in precedenza
- grammatica priva di ricorsioni sinistre tali che $A \stackrel{*}{\Longrightarrow} A\alpha$

Sotto queste ipotesi le produzioni della grammatica risultante sono del tipo $A \longrightarrow x\alpha$ con $x \in X$ e $\alpha \in (X \cup V)^*$

Applico i passi fino al prossimo 6. (ed eventualmente anche i precedenti).

A questo punto (passo 7.) si trasforma ogni terminale in α in un non terminale aggiungendo un'opportuna produzione

La grammatica ottenuta sarà in GNF

$$
\begin{aligned}
S &\longrightarrow aB \\
B &\longrightarrow \underline{\mathbf{CD}} \mid c \\
C &\longrightarrow \underline{\mathbf{BC}} \mid d \\
D &\longrightarrow ab \mid a
\end{aligned}
$$

Ordine scelto $S \prec B \prec C \prec D$

$S \longrightarrow aB$	tipo $(b) : A_i \longrightarrow xv, \quad x \in X, v \in (X \cup V)^*$
$B \longrightarrow CD$	tipo $(a) : A_i \longrightarrow A_j v, \quad i < j$
$B \longrightarrow c$	tipo (b)
$C \longrightarrow BC$	nè (a) nè (b) essendo $B \prec C$
$C \longrightarrow d$	tipo (b)
$D \longrightarrow ab \mid a$	tipo (b)

Trasformazione di $C \longrightarrow BC$ usando $B \longrightarrow CD \mid c$

$$C \longrightarrow \overbrace{CD}^{B}\, C \mid \overbrace{c}^{B}\, C$$

ottenendo la grammatica equivalente:

$$\begin{aligned}
S &\longrightarrow aB \\
B &\longrightarrow CD \mid c \\
C &\longrightarrow \underline{\mathbf{CDC}} \mid \underline{\mathbf{cC}} \mid d \\
D &\longrightarrow ab \mid a
\end{aligned}$$

Passo 6. Eliminazione delle ricorsioni sinistre $A \longrightarrow Av$

Per ogni NT tale che $A \longrightarrow Av \mid w$
con $v, w \in (X \cup V)^*, w \neq A\gamma$ e $\gamma \in (X \cup V)^*$

- si sostituiscono le produzioni $A \longrightarrow Av \mid w$ con:

$$\begin{array}{ll} 1. & A \longrightarrow w \mid wB \\ 2. & B \longrightarrow vB \mid v \mid \lambda \end{array}$$

$$
\begin{aligned}
S &\longrightarrow aB \\
B &\longrightarrow CD \mid c \\
C &\longrightarrow \underline{\mathbf{CDC}} \mid cC \mid d \\
D &\longrightarrow ab \mid a
\end{aligned}
$$

Unica ricorsione sinistra: $C \longrightarrow CDC$
trasformata in: $C \longrightarrow cC \mid d \mid cCE \mid dE$
con $E \longrightarrow DCE \mid DC$

La grammatica diventa quindi:

$$
\begin{aligned}
S &\longrightarrow aB \\
B &\longrightarrow CD \mid c \\
C &\longrightarrow cC \mid d \mid cCE \mid dE \\
E &\longrightarrow DCE \mid DC \\
D &\longrightarrow ab \mid a
\end{aligned}
$$

L'introduzione di E cambia l'ordinamento in:

$$S \prec B \prec C \prec E \prec D$$

e tutte le produzioni sono del tipo (a) o (b)
$E \longrightarrow DCE \mid DC$ diventa $E \longrightarrow abCE \mid aCE \mid abC \mid aC$
$B \longrightarrow CD$ diventa $B \longrightarrow cCD \mid dD \mid cCED \mid dED$
quindi:

$$
\begin{aligned}
S &\longrightarrow aB \\
B &\longrightarrow \underline{\mathbf{cCD}} \mid \underline{\mathbf{dD}} \mid \underline{\mathbf{cCED}} \mid \underline{\mathbf{dED}} \mid c \\
C &\longrightarrow cC \mid d \mid cCE \mid dE \\
E &\longrightarrow \underline{\mathbf{abCE}} \mid \underline{\mathbf{aCE}} \mid \underline{\mathbf{abC}} \mid \underline{\mathbf{aC}} \\
D &\longrightarrow ab \mid a
\end{aligned}
$$

Passo 7. Introduzione di nuovi non terminali
le produzioni della grammatica sono del tipo

$$A \longrightarrow x\alpha$$

con $x \in X$ e $\alpha \in (X \cup V)^*$
Per ogni $a \in X$ in α
si introduca un nuovo non terminale A
e si aggiunga una produzione $A \longrightarrow a$

Nell'esempio:

$$
\begin{array}{lcl}
S & \longrightarrow & aB \\
B & \longrightarrow & cCD \mid dD \mid cCED \mid dED \mid c \\
C & \longrightarrow & cC \mid d \mid cCE \mid dE \\
E & \longrightarrow & abCE \mid aCE \mid abC \mid aC \\
D & \longrightarrow & ab \mid a
\end{array}
$$

diventa:

$$
\begin{array}{lcl}
S & \longrightarrow & aB \\
B & \longrightarrow & cCD \mid dD \mid cCED \mid dED \mid c \\
C & \longrightarrow & cC \mid d \mid cCE \mid dE \\
D & \longrightarrow & aF \mid a \\
E & \longrightarrow & aFCE \mid aCE \mid aFC \mid aC \\
F & \longrightarrow & b
\end{array}
$$

che è in GNF

Teoremi di Equivalenza

Teorema. Un linguaggio libero è generabile da una grammatica in GNF ottenuta tramite l'algoritmo.

Teorema. Ogni linguaggio libero è riconosciuto da un automa a pila

Teorema. Ogni linguaggio accettato da un automa a pila è libero