Linguaggi di Programmazione
Corso C

Parte n.7
Automi a Pila e Grammatiche Libere

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

Dipartimento di Informatica
Università degli Studi di Bari

Automi a Pila

Per il teorema di Kleene la classe $\mathcal{L}_{FSL}$ è limitata
per *riconoscere* linguaggi di tipi superiori 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.

Automa a Pila non deterministico

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

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

dove:

- Q insieme finito e non vuoto degli *stati*
- X insieme finito e non vuoto dei simboli detto *alfabeto di ingresso*
- Γ insieme finito e non vuoto di simboli 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

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

dove:

- $q \in Q$ è lo *stato corrente* dell'unità di controllo
- $w = x_1 x_2 \cdots x_n \in X^*$ è la porzione di stringa di simboli di ingresso sul nastro da esaminare (con la 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

Le ID servono a descrivere le *transizioni* 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 il simbolo su cui insiste la testina e A il simbolo in cima alla pila

1. se δ è descritta dalla n-pla $(q, x, A, q', A_1 \cdots A_k)$ allora l'automa *può* operare la transizione:

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

   se $(q', A_1 \cdots A_k) \in \delta(q, x, A)$

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

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

   se $(q', A_1 \cdots A_k) \in \delta(q, \lambda, A)$

$\overset{*}{\Longrightarrow}$ la chiusura riflessiva e transitiva dell'operatore di transizione $\Longrightarrow$
$ID_1 \overset{*}{\Longrightarrow} ID_2$ si transita dalla descrizione ID_1 in ID_2 in un numero finito (anche nullo) di passi

Linguaggi Accettati da PDA

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

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

risulta così definito il *linguaggio accettato* da M *in condizione di pila vuota*:

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

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

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

risulta così definito il *linguaggio accettato* da M *in condizione di stato finale*:

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

Teorema di equivalenza

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

Esempio 1

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

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

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

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

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

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 2

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

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

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

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

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

simbolo pila	stato	a	b	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	-	-	-

Programma per PDA:

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)

ossia:

- 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: $(q_0, c, Z, q_1, Z)\ \forall Z \in \Gamma$

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

Esempio 3

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

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

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

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

- $Q = \{q_0, q_1\}$ $\quad q_0 =$ 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

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

Studiamo la necessità di forme normali per grammatiche libere:

Esempio

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

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

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 $Z_0 = S$
- $F = \emptyset$

simbolo 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			$\{(q_0, B), (q_0, \lambda)\}$

Ciò è giustificato dal seguente risultato

Teorema

Sia $G = (X, V, S, P)$ una grammatica libera con produzioni del tipo

$$A \longrightarrow x\alpha \qquad x \in X, \alpha \in V^*$$

Allora $L(G) = T(M)$ ove:

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

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

senza dimostrazione

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 del tipo:

1. $S \longrightarrow \lambda$
2. $A \longrightarrow BC$
   dove $A \in V$ 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. $A \longrightarrow a$
   dove $A \in V, a \in X$

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 $A \in V, \quad x \in X, \quad \alpha \in V^*$

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 prima di ricorsioni sinistre

Dimostrazione:
Costruttivamente

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

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

$X = \{0, 1, 2\} \quad V = \{S, A, B\}$
$P = \{S \longrightarrow 00A|B|1,\ A \longrightarrow 1AA|2,\ B \longrightarrow 0\}$

Algoritmo 1

input: $G = (X, V, S, P)$ grammatica libera
output: $G_1 = (X, V, S, P)$ in CNF

1. Conversione dei terminali che compaiono nelle parti destre di produzioni in non terminali e aggiunta delle produzioni appropriate

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

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

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

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

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

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

$X = \{a, b, c, d\} \quad V = \{S, A, B, C, D\}$
$P = \{S \longrightarrow AaB,\ A \longrightarrow \lambda,\ B \longrightarrow CD|c,\ C \longrightarrow BC|d,\ D \longrightarrow ab|a\}$

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 $V_1 = \{A \in V \mid A \stackrel{*}{\Longrightarrow} \lambda\}$ e $V_2 = V \setminus V_2$
Algoritmo di calcolo di $e(A) =$ vero sse $A \stackrel{*}{\Longrightarrow} \lambda$:

1. $\forall A \in V:\ e(A) :=$ falso
2. $\forall A \longrightarrow \lambda \in P:\ e(A) :=$ 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 $e(A) :=$ true
   e si marcano tute le occorrenze di A in parti destre di produzioni
4. se nel passo precedente qualche $e(A)$ è mutato allora torna al passo (c) altrimenti termina

Nell'esempio precedente costruiamo le nuove produzioni P_2:

$e()$	(a)	(b)	(c)	(d)
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 $S \stackrel{*}{\Longrightarrow} \lambda (\lambda \in L(G))$
2. Se $A \longrightarrow X_1, X_2 \ldots X_r,\ r \geq 1$
   allora apparterranno a P_2 le produzioni

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

ove: $Y_i = \begin{cases} X_i & \text{se } X_i \in X \cup V_2; \\ X_i | \lambda & X_i \in V_1 \end{cases}$

$P_2 =$
$\{S \longrightarrow \mathbf{aB} | \mathbf{AaB},\ A \longrightarrow \lambda,\ B \longrightarrow CD|c,\ C \longrightarrow BC|d,\ D \longrightarrow ab|a\}$

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 $s(A)$

$A \in V$ $\quad t(A) = \text{true} \quad \text{sse} \quad A \overset{*}{\Longrightarrow} w,\ w \in X^*$

$\quad s(A) = \text{true} \quad \text{sse} \quad S \overset{*}{\Longrightarrow} \alpha A \beta, \quad \alpha, \beta \in (X \cup V)^*$

1. $\forall A \in V : t(A) :=$ falso
2. se $A \longrightarrow x \in P, \quad x \in X^*$ allora $t(A) :=$ vero
   e si marcano con un apice t tutte le occorrenze di A in parti destre di produzioni di P
3. $s(S) :=$ vero
   e si marcano con un apice s tutte le occorrenze di S in parti sinistre di produzioni di P
4. (a) $\forall A \longrightarrow \alpha \in P$ se i non terminali di α sono marcati t allora
   si marcano con t tutte le A in parti destre di P

   (b) se A è marcato s allora $s(B) :=$ 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

$S^s \longrightarrow aB^t|\underline{\mathbf{AaB^t}}$
$B^s \longrightarrow C^tD^t|c$
$C^s \longrightarrow B^tC^t|d$
$D^s \longrightarrow ab|a$

essendo A il non terminale inutile, quindi $S \longrightarrow AaB$

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

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

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

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

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. $P_1 = P_1 \cup \{B \in c(A) \wedge B \longrightarrow \alpha \in P_1\}$

Non vi sono tali produzioni

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

Algoritmo

Se $B \longrightarrow \alpha_1|\alpha_2|\cdots|\alpha_n \in P$
allora sostituisco $A \longrightarrow B\beta$ con $A \longrightarrow \alpha_1\beta|\alpha_2\beta|\cdots|\alpha_n\beta$

Se $\forall i \in [1,n]$ α_i inizia con un non terminale allora termina per $A \longrightarrow B\beta$
altrimenti se $\exists \alpha_k = C\beta$ ripetere per $A \longrightarrow \alpha_k\beta$

L'algoritmo termina sse (ipotesi):

- non terminali inutili eliminati
- grammatica prima 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 $\alpha \in (X \cup V)^*$

A questo punto si trasforma ogni terminale in α in un non terminale aggiungendo un'opportuna produzione

La grammatica ottenuta sarà in GNF

$S \longrightarrow aB$
$B \longrightarrow \underline{\mathbf{CD}}|c$
$C \longrightarrow \underline{\mathbf{BC}}|d$
$D \longrightarrow ab|a$

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)
$D \longrightarrow ab|a$	tipo (b)

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

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

ottenendo la grammatica equivalente:

$S \longrightarrow aB$
$B \longrightarrow CD|c$
$C \longrightarrow \underline{\mathbf{CDC}}|\underline{\mathbf{cC}}|d$
$D \longrightarrow ab|a$

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

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

si trasformano le produzioni in: $A \longrightarrow w|wb$ e $B \longrightarrow vB|v|\lambda$

$S \longrightarrow aB$
$B \longrightarrow CD|c$
$C \longrightarrow \underline{\mathbf{CDC}}|cC|d$
$D \longrightarrow ab|a$

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

La grammatica diventa quindi:

$S \longrightarrow aB$
$B \longrightarrow CD|c$
$C \longrightarrow cC|d|cCE|dE$
$E \longrightarrow DCE|DC$
$D \longrightarrow ab|a$

L'introduzione di E cambia l'ordinamento in: $S \prec S \prec B \prec C \prec E \prec D$ e tutte le produzioni sono del tipo (a) o (b)

$B \longrightarrow CD$ diventa $B \longrightarrow cCD|dD|cCED|dED$ e
$E \longrightarrow DCE|CD$ diventa $E \longrightarrow abCE|aCE|abC|aC$

quindi:

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

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 $A \longrightarrow a$

$S \longrightarrow aB$
$B \longrightarrow cCD|dD|cCED|dED|c$
$C \longrightarrow cC|d|cCE|dE$
$E \longrightarrow abCE|aCE|abC|aC$
$D \longrightarrow ab|a$

diventa:

$S \longrightarrow aB$
$B \longrightarrow cCD|dD|cCED|dED|c$
$C \longrightarrow cC|d|cCE|dE$
$D \longrightarrow aF|a$
$E \longrightarrow aFCE|aCE|aFC|aC$
$F \longrightarrow b$

che è in GNF

Teorema

Un linguaggio libero è generabile da una grammatica in GNF ottenuta tramite questo algoritmo.

Teorema

Ogni linguaggio libero è riconosciuto da un automa a pila

Teorema

Ogni linguaggio accettato da un automa a pila è libero