Linguaggi Liberi dal Contesto

Grammatiche libere da contesto

una grammatica G libera da contesto è una quadrupla (V, Σ, P, S) tale che:

- V è un insieme di simboli (alfabeto non terminale)
- Σ è un insieme di simboli (alfabeto terminale)
- P è un insieme di regole o produzioni sintattiche
- $S \in V$ è l'assioma

le regole sono coppie

$$A \to \alpha$$

dove $A \in V$ (A è un simbolo non terminale) e α è una stringa di simboli terminali e non terminali ($\alpha \in (V \cup \Sigma)^\star$)

$(\{S, T\}, \{a, b\}, \{S \to aTb, T \to bSa, T \to ab\}, S)$

$A \to \alpha_1 | \ldots | \alpha_n$ abbrevia $A \to \alpha_1, \ldots, A \to \alpha_n$

Derivazioni

la stringa β diviene la stringa γ o la stringa γ produce la stringa β per una grammatica G

$$\beta \rightarrow \gamma$$

se $\beta, \gamma \in (V \cup \Sigma)^\star$ e $\beta = \delta A \eta$ e $\gamma = \delta \alpha \eta$ e $A \rightarrow \alpha \in P$

$\beta \rightarrow^n \gamma$ (β diviene γ in n passi) se $\beta = x_0 \rightarrow x_1 \ldots x_{n-1} \rightarrow x_n = \gamma$

$\beta \rightarrow^\star \gamma$ (β diviene riflessivamente e transitivamente γ) se $\beta \rightarrow^n \gamma$ per qualche $n \geq 0$

$\beta \rightarrow^+ \gamma$ (β diviene transitivamente γ) se $\beta \rightarrow^n \gamma$ per qualche $n \geq 1$

$(\{S, T\}, \{a, b\}, \{S \rightarrow aTb, T \rightarrow bSa | ab\}, S)$

- $aTb \rightarrow abSab$
- $S \rightarrow aTb \rightarrow abSab$
- $S \rightarrow^2 abSab$

Linguaggio generato da una grammatica

il linguaggio generato da una grammatica partendo dal non terminale A (notazione $L_A(G)$) è l'insieme delle stringhe di terminali prodotte da A

$$L_A(G) = \{x \in \Sigma^\star \mid A \to^+ x\}$$

il linguaggio generato da una grammatica (notazione $L(G)$) è l'insieme delle stringhe di terminali prodotte dall'assioma

$$L(G) = \{x \in \Sigma^\star \mid S \to^+ x\}$$

$G = (\{S, T\}, \{a, b\}, \{S \to aTb, T \to bSa|ab\}, S)$

- $T \to bSa \to baTba \to baabba$
- $T \to bSa \to baTba \to babSaba \to babaTbaba \to babaabbaba$
- $S \to aTb \to aabb$
- $S \to aTb \to abSab \to abaTbab \to abaabbab$

$L_T(G) = \{baabba, babaabbaba, \ldots\}$

$L(G) = \{aabb, abaabbab, \ldots\}$

Linguaggi liberi da contesto

Un linguaggio è libero dal contesto se esiste una grammatica libera che lo genera

due grammatiche $\mathbf{G}$ e $\mathbf{G'}$ sono debolmente equivalenti se generano lo stesso linguaggio, cioè $\mathbf{L(G) = L(G')}$

$\mathbf{G = (\{S\}, \{a, b\}, \{S \to aSb|ab\}, S)}$

$\mathbf{G' = (\{S, A, B\}, \{a, b\}, \{S \to ASB|AB, A \to a, B \to b\}, S)}$

$\mathbf{L(G) = L(G') = \{ab, aabb, aaabbb, \ldots\} = \{a^n b^n \mid n \geq 1\}}$

Grammatiche pulite I

una grammatica è pulita se:

1. ogni non terminale A è raggiungibile dall'assioma, cioè esiste una derivazione $S \rightarrow^{+} \alpha A \beta$
2. ogni non terminale A è definito, cioè genera un linguaggio non vuoto, $L_A(G) \neq \emptyset$
3. non ci sono derivazioni circolari che producono un non terminale a partire da se stesso, cioè della forma $A \rightarrow^{+} A$

ogni grammatica non pulita ha un'equivalente grammatica pulita che genera lo stesso linguaggio

primo passo: calcolo dei terminali definiti

$$\mathbf{DEF} := \{A \mid A \rightarrow u \in P \ \&\ u \in \Sigma^{\star}\}$$

$$\mathbf{DEF} := \mathbf{DEF} \cup \{B \mid B \rightarrow \alpha_1 \ldots \alpha_n \in P \\ \&\ \forall i (1 \leq i \leq n)\ \alpha_i \in \Sigma^{\star} \ \vee\ \alpha_i \in \mathbf{DEF}\}$$

Grammatiche pulite II

secondo passo: calcolo dei terminali raggiungibili

$$\mathbf{RAG} := \{\mathbf{S}\}$$

$$\mathbf{RAG} := \mathbf{RAG} \cup \{\mathbf{A} \mid \mathbf{B} \to \alpha\mathbf{A}\beta \in \mathbf{P} \;\&\; \mathbf{B} \in \mathbf{RAG}\}$$

terzo passo: calcolo dei terminali con circolarità

$$\mathbf{NULL} := \{\mathbf{A} \mid \mathbf{A} \to \epsilon \in \mathbf{P}\}$$

$$\mathbf{NULL} := \mathbf{NULL} \cup \{\mathbf{A} \mid \mathbf{A} \to \mathbf{B}_1 \dots \mathbf{B}_n \in \mathbf{P} \;\&\; \forall i(1 \leq i \leq n)\; \mathbf{B}_i \in \mathbf{NULL})\}$$

$$\mathbf{CIRC}(\mathbf{A}) := \{\mathbf{B} \mid \mathbf{A} \to \mathbf{C}_1 \dots \mathbf{C}_n\mathbf{B}\mathbf{C}_{n+1} \dots \mathbf{C}_m \in \mathbf{P} \\ \;\&\; \forall i(1 \leq i \leq m)\; \mathbf{C}_i \in \mathbf{NULL}\}$$

$$\mathbf{CIRC}(\mathbf{A}) := \mathbf{CIRC}(\mathbf{A}) \cup \{\mathbf{B} \mid \mathbf{D} \to \mathbf{C}_1 \dots \mathbf{C}_n\mathbf{B}\mathbf{C}_{n+1} \dots \mathbf{C}_m \in \mathbf{P} \\ \;\&\; \forall i(1 \leq i \leq m)\; \mathbf{C}_i \in \mathbf{NULL} \;\&\; \mathbf{D} \in \mathbf{CIRC}(\mathbf{A})\}$$

Grammatiche pulite III

quarto passo: eliminazione delle regole che contengono terminali indefiniti o irraggiungibili o che generano circolarità

$\mathbf{G} = (\{S, A, B, C, D, E, F, G\}, \{a, b, c\}, \{S \to ABC, A \to EAF|a, B \to Gb|b, C \to Sc|c, D \to d, E \to \epsilon, F \to \epsilon\}, S)$

$\mathbf{G}' = (\{S, A, B, C\}, \{a, b, c\}, \{S \to ABC, A \to a, B \to b, C \to Sc|c\}, S)$

Ricorsione e linguaggi finiti/infiniti

Una derivazione è ricorsiva se partendo da un non terminale produce in almeno un passo una stringa che contiene lo stesso non terminale, cioè se è della forma $A \to^+ \alpha A \beta$. E' ricorsiva sinistra se $\alpha = \epsilon$; ricorsiva destra se $\beta = \epsilon$.

Il non terminale A è detto ricorsivo.

$G = (\{S\}, \{a, b\}, \{S \to aSb | ab\}, S)$

$S \to aSb$

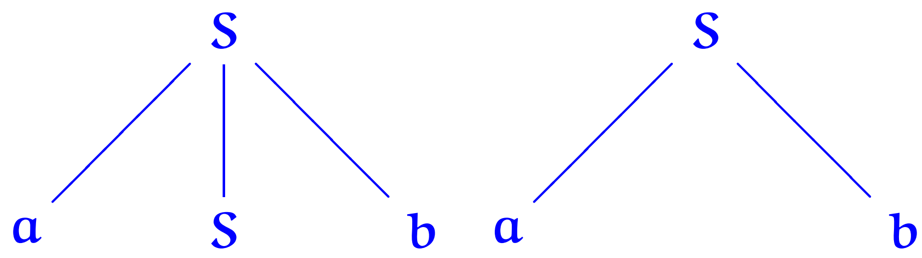
Sia G una grammatica pulita e priva di derivazioni circolari. Condizione necessaria e sufficiente perchè il linguaggio $L(G)$ sia infinito è che G permetta derivazioni ricorsive.

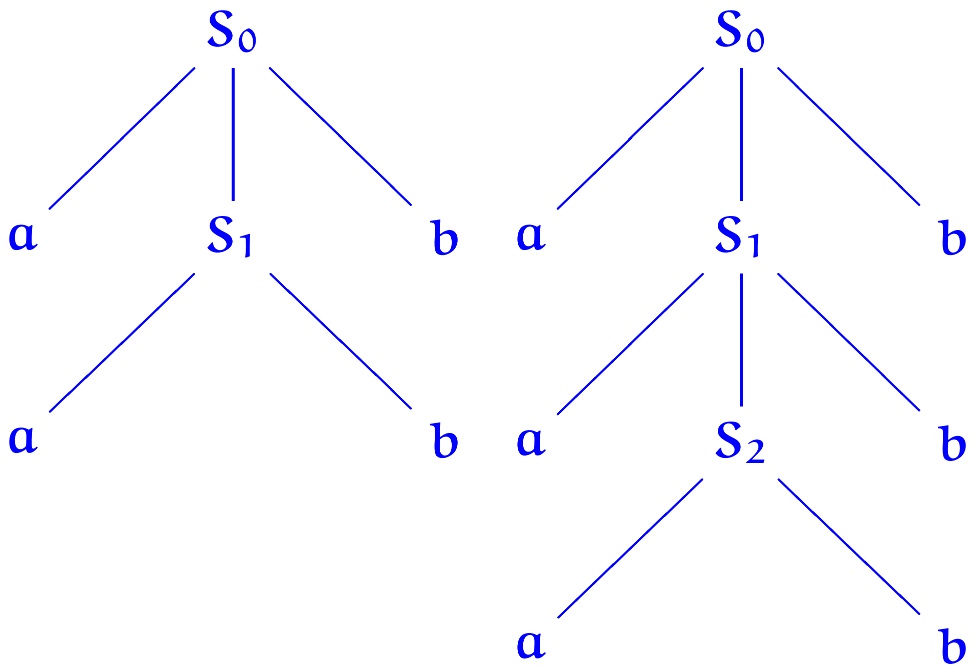
Per decidere se una grammatica ha delle ricorsioni basta esaminare la chiusura transitiva della relazione binaria fra non terminali definita da:
A produce B se la grammatica contiene una produzione della forma $A \to \alpha B \beta$.
La grammatica ha delle ricorsioni solo se la chiusura transitiva di questa relazione ha dei cicli.

Alberi sintattici

Possiamo rappresentare produzioni e derivazioni mediante alberi:

$\mathbf{G} = (\{S\}, \{a, b\}, \{S \to aSb|ab\}, S)$





Derivazioni canoniche

Una derivazione è canonica sinistra se ad ogni passo sostituisco il non terminale più a sinistra.

Una derivazione è canonica destra se ad ogni passo sostituisco il non terminale più a destra.

$\mathbf{G} = (\{S, A, B\}, \{a, b\}, \{S \to ASB|e, A \to ab, B \to cd\}, S)$

derivazione sinistra: $S \to ASB \to abSB \to abASBB \to ababSBB \to ababeBB \to ababecdB \to ababecdcd$

derivazione destra: $S \to ASB \to AScd \to AASBcd \to AAScdcd \to AAecdcd \to Aabecdcd \to ababecdcd$

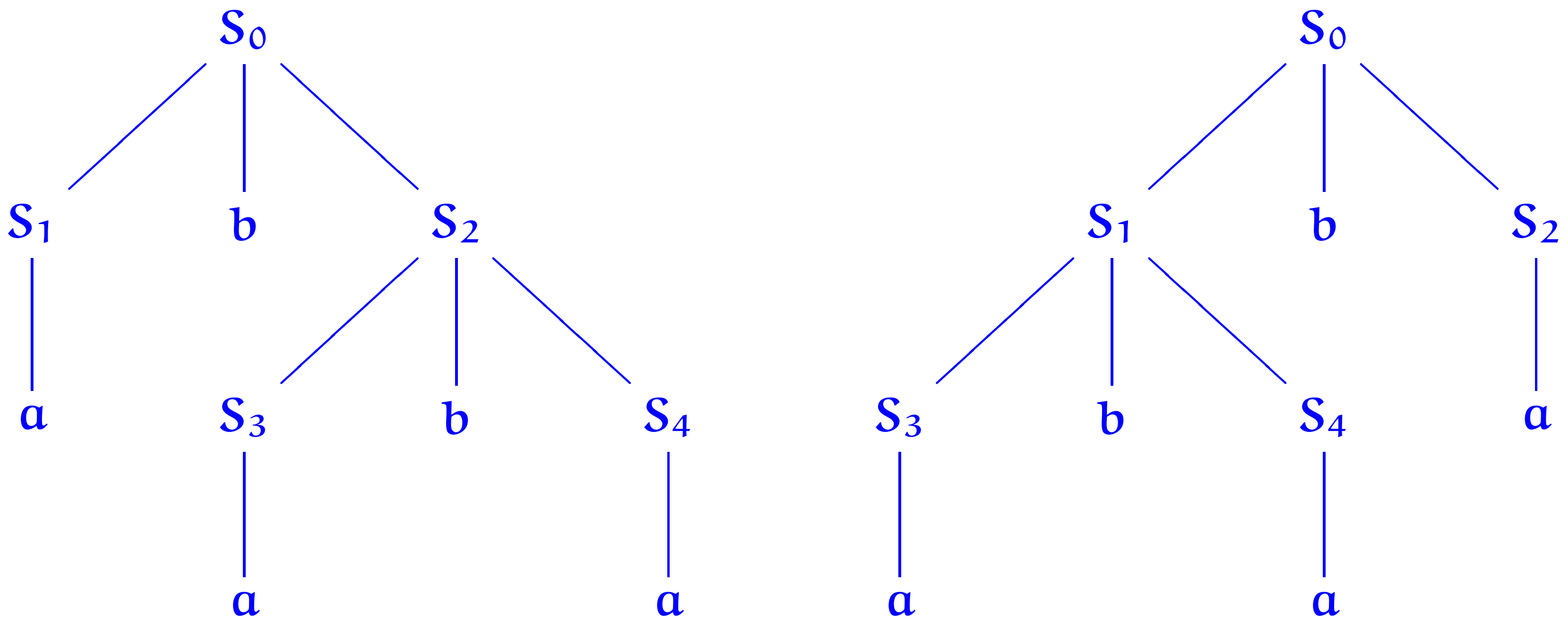
derivazione: $S \to ASB \to AScd \to abScd \to abASBcd \to abAScdcd \to ababScdcd \to ababecdcd$

ogni frase può essere generata con una derivazione sinistra o destra

Ambiguità I

Una frase è ambigua se ha due alberi sintattici distinti

$\mathbf{G} = (\{S\}, \{a, b\}, \{S \to SbS | a\}, S)$



Il grado di ambiguità di una frase è il numero dei suoi alberi sintattici distinti.

Ambiguità II

Non è decidibile se una grammatica è ambigua.

Un linguaggio è inerentemente ambiguo se tutte le grammatiche che lo generano sono ambigue.

$\{a^i b^i c^\star \mid i \geq 0\} \cup \{a^* b^k c^k \mid k \geq 0\} = \{a^i b^j c^k \mid i, j, k \geq 0 \;\&\; i = j \vee j = k\}$

$G = (\{S\}, \{a, b, c\}, \{S \to BC|AD, B \to aBb|\epsilon, C \to cC|\epsilon, A \to aA|\epsilon, D \to bBc|\epsilon\}, S)$

Grammatiche equivalenti

Ricordiamo: Due grammatiche sono debolmente equivalenti se generano lo stesso linguaggio.

Due grammatiche sono fortemente equivalenti se generano lo stesso linguaggio e assegnano ad ogni frase alberi sintattici isomorfi.

L'equivalenza debole è indecidibile, mentre quella forte è decidibile.

$\mathbf{G} = (\{S\}, \{a, b\}, \{S \to aSb|ab\}, S)$

$\mathbf{G'} = (\{S, A, B\}, \{a, b\}, \{S \to ASB|AB, A \to a, B \to b\}, S)$

$\mathbf{G''} = (\{T\}, \{a, b\}, \{T \to aTb|ab\}, T)$

$\mathbf{G}$ è debolmente equivalente a $\mathbf{G'}$ e fortemente equivalente a $\mathbf{G''}$.

Nel seguito trasformeremo grammatiche in grammatiche debolmente equivalenti: queste trasformazioni in generale possono produrre grammatiche non pulite, in questo caso useremo l'algoritmo di pulizia per pulirle.

Forme normali delle grammatiche I

Una grammatica può sempre essere trasformata in una grammatica equivalente che non contiene l'assioma nelle parti destre delle produzioni. Basta aggiungere un nuovo assioma S_0 e la produzione $S_0 \to S$.

Una regola è detta di copiatura se è della forma $A \to B$ con $B \in V$. Le regole di copiatura possono essere eliminate da ogni grammatica. Si calcola prima l'insieme dei non terminali in cui si può copiare un non terminale:

$$\mathbf{COPIA}(A) := \{A\}$$

$$\mathbf{COPIA}(A) := \mathbf{COPIA}(A) \cup \{C \mid B \to C \in P \;\&\; B \in \mathbf{COPIA}(A)\}$$

Si costruisce poi l'insieme delle nuove produzioni

$P' = \{A \to \alpha \mid \alpha \notin V \;\&\; (\exists B \in \mathrm{Copia}(A).B \to \alpha \in P)\}$

$G = (\{S, A\}, \{a, b\}, \{S \to SbS|A, A \to a\}, S)$

$G' = (\{S\}, \{a, b\}, \{S \to SbS, S \to a\}, S)$

Forma normale non annullabile I

Un non terminale è annullabile se produce la stringa vuota, cioè A è annullabile se $A \to^+ \epsilon$.

Si può calcolare l'insieme $\mathbf{NULL}$ dei non terminali annullabili:

$$\mathbf{NULL} := \{A \mid A \to \epsilon \in P\}$$

$$\mathbf{NULL} := \mathbf{NULL} \cup \{A \mid A \to B_1 \ldots B_n \in P \;\&\; \forall i(1 \leq i \leq n)\; B_i \in \mathbf{NULL})\}$$

Una grammatica è in forma normale non annullabile se soddisfa le condizioni:

1. nessun non terminale diverso dall'assioma è annullabile
2. l'assioma è annullabile solo se la stringa vuota appartiene al linguaggio.

Forma normale non annullabile II

Per trasformare una grammatica in un'altra che sia in forma normale non annullabile per ogni regola $A \to x_1 A_1 x_2 \ldots x_n A_n x_{n+1} \in P$ si aggiungono come regole alternative quelle ottenute cancellando in tutti i modi possibili dalla parte destra i non terminali A_i che sono annullabili. Si eliminano poi le regole $A \to \epsilon$ tranne che per $A = S$.

$\mathbf{G} = (\{S, A\}, \{a, b\}, \{S \to AbA, A \to a|\epsilon\}, S)$

$\mathbf{G}' = (\{S, A\}, \{a, b\}, \{S \to AbA|Ab|bA|b, A \to a\}, S)$

Forma normale di Chomsky I

Una grammatica è in forma normale di Chomsky se le sue regole sono solo dei seguenti due tipi:

1. regole omogenee binarie: la stringa è formata da due non terminali, cioè regole della forma $A \rightarrow BC$ con $B, C \in V$
2. regole terminali unitarie: la stringa è formata da un terminale, cioè regole della forma $A \rightarrow a$ con $a \in \Sigma$
3. la regola $S \rightarrow \epsilon$ solo se il linguaggio contiene ϵ

Forma normale di Chomsky II

Per trasfomare una grammatica in forma normale non annullabile in una grammatica in forma normale di Chomsky si possono iterare le seguenti trasformazioni:

1. per ogni regola $A \to B\alpha$ dove $B \in V$ e $\alpha \notin V$ si aggiunge il non terminale $<\alpha>$ e la regola è sostituita dalle regole $A \to B <\alpha>$ e $<\alpha> \to \alpha$
2. per ogni regola $A \to a\alpha$ dove $a \in \Sigma$ e $\alpha \neq \epsilon$ si aggiungono i non terminali $<a>, <\alpha>$ e la regola è sostituita dalle regole $A \to <a><\alpha>$, $<a> \to a$ e $<\alpha> \to \alpha$

$G = (\{S, A\}, \{a, b\}, \{S \to AbA|Ab|bA, A \to a\}, S)$

$G_1 = (\{S, A, <bA>, <b>\}, \{a, b\}, \{S \to A <bA> |A <b> | <b> A, A \to a,$
$<bA> \to bA, <b> \to b\}, S)$

$G_2 = (\{S, A, <bA>, <b>\}, \{a, b\}, \{S \to A <bA> |A <b> | <b> A, A \to a,$
$<bA> \to <b> A, <b> \to b\}, S)$

Trasformazioni delle ricorsioni sinistre in ricorsioni destre I

Espansione

se $B \to \beta_1 | \ldots | \beta_h$ sono tutte le alternative di B, la regola $A \to \alpha B \gamma$ può essere sostituita da $A \to \alpha\beta_1\gamma | \ldots | \alpha\beta_h\gamma$

Eliminazione delle ricorsioni immediate

se le alternative di A ricorsive a sinistra sono
$A \to A\beta_1 | \ldots | A\beta_h$ ($\beta_i \neq \epsilon$, $1 \leq i \leq h$), e le rimanenti alternative sono
$A \to \gamma_1 | \ldots | \gamma_k$, si crea un nuovo non terminale A' e si sostituiscono le regole precedenti con
$A \to \gamma_1 A' | \ldots | \gamma_k A'$
$A' \to \epsilon | \beta_1 A' | \ldots | \beta_h A'$

Eliminazione delle ricorsioni non-immediate

assumiamo che la grammatica sia in forma normale non annullabile e priva di cicli; sia $V = \{A_1, \ldots, A_m\}$ ed A_1 l'assioma.

Trasformazioni delle ricorsioni sinistre in ricorsioni destre II

for $i := 1$ to m

do for $j := 1$ to $i - 1$

do

espandi ogni regola della forma

$A_i \rightarrow A_j\,\alpha$

nelle regole

$A_i \rightarrow \gamma_1\,\alpha | \ldots \gamma_k\,\alpha$

dove

$A_j \rightarrow \gamma_1 | \ldots \gamma_k$

sono le alternative di A_j presenti

end do

elimina le ricorsioni sinistre immediate nelle alternative di A_i

end do

Trasformazioni delle ricorsioni sinistre in ricorsioni destre III

$A_1 \rightarrow A_2\, a$

$A_2 \rightarrow A_3\, b$

$A_3 \rightarrow A_1\, c | p$

$A_1 \rightarrow A_2\, a$

$A_2 \rightarrow A_3\, b$

$A_3 \rightarrow A_2\, ac | p$ (espansione)

$A_1 \rightarrow A_2\, a$

$A_2 \rightarrow A_3\, b$

$A_3 \rightarrow A_3\, b\, ac | p$ (espansione)

$A_1 \rightarrow A_2\, a$

$A_2 \rightarrow A_3\, b$

$A_3 \rightarrow pA_3'$

$A_3' \rightarrow \epsilon | b\, acA_3'$ (eliminazione ricorsione immediata)

Forma normale di Greibach I

Una grammatica (che genera un linguaggio che non contiene ϵ) è in forma normale di Greibach se le produzioni hanno la forma $A \to a\alpha$ dove $a \in \Sigma$ e $\alpha \in V^\star$.

Ogni grammatica che genera un linguaggio non contenente la stringa vuota si può trasformare nella forma normale di Greibach eliminando le ricorsioni sinistre, trasformandola nella forma normale non annullabile, eseguendo delle espansioni ed introducendo dei nuovi non terminali per avere i terminali solo all'inizio delle stringhe parti destre delle produzioni

Forma normale di Greibach II

$A_1 \to A_2\,a$

$A_2 \to A_3\,b$

$A_3 \to pA_3'$

$A_3' \to \epsilon|bacA_3'$

$A_1 \to A_2\,a$

$A_2 \to A_3\,b$

$A_3 \to pA_3'|p$

$A_3' \to bacA_3'|bac$ (forma normale non annullabile)

$A_1 \to A_2\,a$

$A_2 \to pA_3'b|pb$

$A_3 \to pA_3'|p$

$A_3' \to bacA_3'|bac$ (espansione)

Forma normale di Greibach III

$A_1 \rightarrow pA_3'ba|pba$

$A_2 \rightarrow pA_3'b|pb$

$A_3 \rightarrow pA_3'|p$

$A_3' \rightarrow bacA_3'|bac$ (espansione)

$A_1 \rightarrow pA_3'ba|pba$

$A_3' \rightarrow bacA_3'|bac$ (eliminazione dei non-terminali non raggiungibili)

$A_1 \rightarrow pA_3'BA|pBA$

$A_3' \rightarrow bACA_3'|bAC$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$ (omogenizzazione)