Grammatiche e Linguaggi Liberi da Contesto

• Abbiamo visto che molti linguaggi non sono regolari. Consideriamo allora classi più grandi di linguaggi.

• *Linguaggi Liberi da Contesto* (CFL) sono stati usati nello studio dei linguaggi naturali dal 1950, e nello studio dei compilatori dal 1960.

• Le *grammatiche libere da contesto* (CFG) sono la base della sintassi BNF (Backus-Naur-Form).

• Oggi i CFL sono importanti per XML.

Studieremo: CFG, i linguaggi che generano, gli alberi sintattici, e daremo cenni di automi a pila e delle proprietà di chiusura dei CFL.

Esempio informale di CFG

Consideriamo $L_{pal} = \{w \in \Sigma^* : w = w^R\}$

Per esempio: `otto` $\in L_{pal}$, `madamimadam` $\in L_{pal}$.

Sia $\Sigma = \{0, 1\}$ e supponiamo che L_{pal} sia regolare.

Usando il Pumping Lemma per i linguaggi regolari si arriva ad una contraddizione (suggerimento: considerare la stringa $0^n 1 0^n \in L_{pal}$).

Definiamo L_{pal} induttivamente:

Base: $\epsilon, 0$, e 1 sono palindromi.

Induzione: Se w è una palindroma, anche $0w0$ e $1w1$ lo sono.

Nessun altra stringa è una palindroma.

Le CFG costituiscono un modo formale per dare definizioni del tipo appena dato per L_{pal}.

1. $P \to \epsilon$
2. $P \to 0$
3. $P \to 1$
4. $P \to 0P0$
5. $P \to 1P1$

0 e 1 sono *terminali*

P è una *variabile* (o *nonterminale*, o *categoria sintattica*)

P è in questa grammatica anche il *simbolo iniziale*.

1–5 sono *produzioni* (o *regole*)

Definizione formale di CFG

Una *grammatica libera da contesto* è una quadrupla

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

dove

V è un insieme finito di *variabili*.

T è un insieme finito di *terminali*.

P è un insieme finito di *produzioni* della forma $A \to \alpha$, dove A è una variabile e $\alpha \in (V \cup T)^*$

S è una variabile distinta chiamata il *simbolo iniziale*.

Esempio: $G_{pal} = (\{P\}, \{0, 1\}, A, P)$, dove

$$A = \{P \to \epsilon, P \to 0, P \to 1, P \to 0P0, P \to 1P1\}.$$

A volte raggruppiamo le produzioni con la stessa testa:
$A = \{P \to \epsilon | 0 | 1 | 0P0 | 1P1\}$.

Esempio: Le espressioni regolari su $\{0, 1\}$ possono essere definite dalla grammatica

$$G_{regex} = (\{E\}, \{0, 1\}, A, E)$$

dove

$$A = \{E \to \mathbf{0}, E \to \mathbf{1}, E \to E.E, E \to E + E, E \to E^\star, E \to (E)\}$$

Esempio: espressioni (semplificate) in un tipico linguaggio di programmazione. Gli operatori sono + e *, e gli operandi sono identificatori, cioè stringhe in $L((\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{0}+\boldsymbol{1})^*)$

Le espressioni sono definite dalla grammatica

$$G = (\{E, I\}, T, P, E)$$

dove $T = \{+, *, (,), a, b, 0, 1\}$ e P è il seguente insieme di produzioni:

1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

Derivazioni usando le grammatiche

- *Derivazioni*, usando le produzioni dalla testa al corpo

Sia $G = (V, T, P, S)$ una CFG, $A \in V$,
$\{\alpha, \beta\} \subset (V \cup T)^*$, e $A \to \gamma \in P$.

Allora scriviamo

$$\alpha A \beta \underset{G}{\Rightarrow} \alpha \gamma \beta$$

o, se è ovvia la G,

$$\alpha A \beta \Rightarrow \alpha \gamma \beta$$

e diciamo che da $\alpha A \beta$ *si deriva* $\alpha \gamma \beta$.

Definiamo $\overset{*}{\Rightarrow}$ la chiusura riflessiva e transitiva di $\Rightarrow$, cioè:

Base: Sia $\alpha \in (V \cup T)^*$. Allora $\alpha \overset{*}{\Rightarrow} \alpha$.

Induzione: Se $\alpha \overset{*}{\Rightarrow} \beta$, e $\beta \Rightarrow \gamma$, allora $\alpha \overset{*}{\Rightarrow} \gamma$.

Esempio: Derivazione di $a * (a + b00)$ da E nella grammatica delle espressioni:

$$E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * (E) \Rightarrow$$

$$a * (E + E) \Rightarrow a * (I + E) \Rightarrow a * (a + E) \Rightarrow a * (a + I) \Rightarrow$$

$$a * (a + I0) \Rightarrow a * (a + I00) \Rightarrow a * (a + b00)$$

Nota: ad ogni passo potremmo avere varie regole tra cui scegliere, ad esempio
$I * E \Rightarrow a * E \Rightarrow a * (E)$, oppure
$I * E \Rightarrow I * (E) \Rightarrow a * (E)$.

Nota: non tutte le scelte portano a derivazioni di una particolare stringa, per esempio

$$E \Rightarrow E + E$$

non ci fa derivare $a * (a + b00)$.

Il linguaggio di una grammatica

Se $G(V, T, P, S)$ è una CFG, allora il *linguaggio di* G è

$$L(G) = \{w \in T^* : S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

cioè l'insieme delle stringhe su T^* derivabili dal simbolo iniziale.

Se G è una CFG, chiameremo $L(G)$ un
linguaggio libero da contesto.

Esempio: $L(G_{pal})$ è un linguaggio libero da contesto.

Alberi sintattici

• Se $w \in L(G)$, per una CFG, allora w ha un *albero sintattico*, che ci dice la struttura (sintattica) di w

• w potrebbe essere un programma, una query SQL, un documento XML, ...

• Gli alberi sintattici sono una rappresentazione alternativa alle derivazioni e alle inferenze ricorsive.

• Ci possono essere diversi alberi sintattici per la stessa stringa

• Idealmente ci dovrebbe essere solo un albero sintattico (la "vera" struttura), cioè il linguaggio dovrebbe essere non ambiguo.

• Sfortunatamente, non sempre possiamo rimuovere l'ambiguità.

Costruzione di un albero sintattico

Sia $G = (V, T, P, S)$ una CFG. Un albero è un *albero sintattico* per G se:

1. Ogni nodo interno è etichettato con una variabile in V.
2. Ogni foglia è etichettata con un simbolo in $V \cup T \cup \{\epsilon\}$. Ogni foglia etichettata con ϵ è l'unico figlio del suo genitore.
3. Se un nodo interno è etichettato A, e i suoi figli (da sinistra a destra) sono etichettati

   $$X_1, X_2, \ldots, X_k,$$

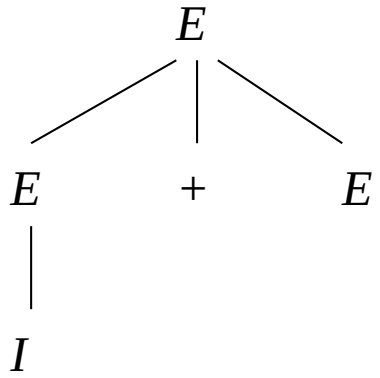
   allora $A \to X_1 X_2 \ldots X_k \in P$.

Esempio: nella grammatica

1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$

$\vdots$

il seguente è un albero sintattico:

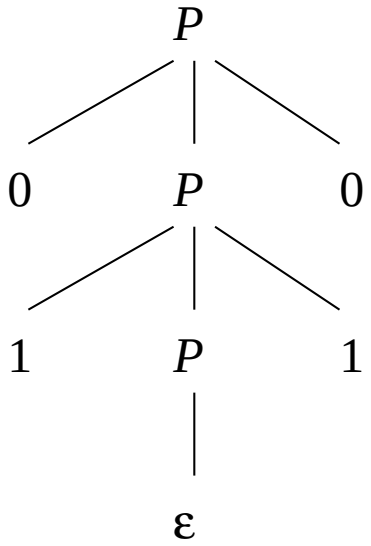


Questo albero sintattico mostra la derivazione $E \stackrel{*}{\Rightarrow} I + E$

Esempio: nella grammatica

1. $P \to \epsilon$
2. $P \to 0$
3. $P \to 1$
4. $P \to 0P0$
5. $P \to 1P1$

il seguente è un albero sintattico:



Mostra la derivazione $P \stackrel{*}{\Rightarrow} 0110$.

Il prodotto di un albero sintattico

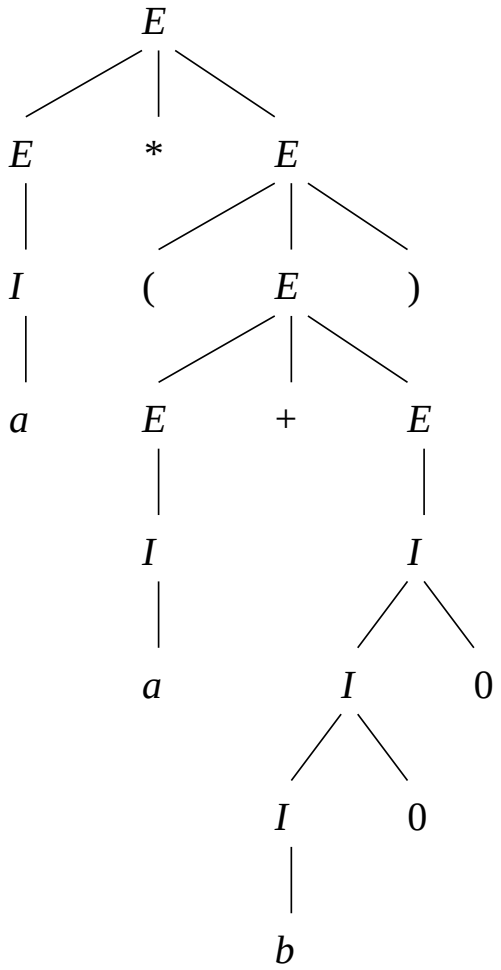
Il *prodotto* di un albero sintattico è la stringa di foglie da sinistra a destra.

Importanti sono quegli alberi sintattici dove:

1. Il prodotto è una stringa terminale.
2. La radice è etichettata dal simbolo iniziale.

L'insieme dei prodotti di questi alberi sintattici è il linguaggio della grammatica.

Esempio:



Il prodotto è $a * (a + b00)$.

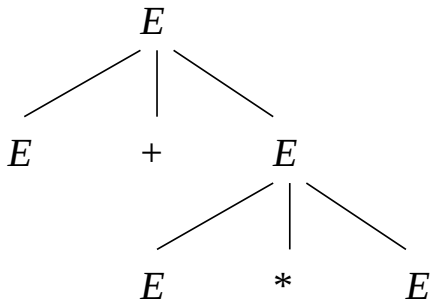
Ambiguità in Grammatiche e Linguaggi

Nella grammatica

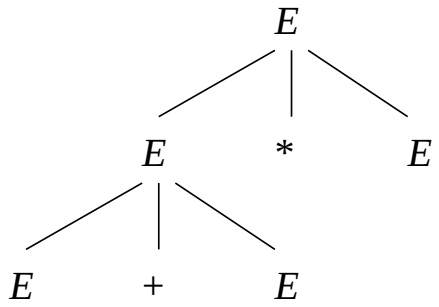
1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$

...

la forma sentenziale $E + E * E$ ha due derivazioni: $E \Rightarrow E + E \Rightarrow E + E * E$ e $E \Rightarrow E * E \Rightarrow E + E * E$. Questo ci dà due alberi sintattici:



(a)



(b)

L'esistenza di varie *derivazioni* di per se non è pericolosa, è l'esistenza di vari alberi sintattici che rovina la grammatica.

Esempio: Nella stessa grammatica

5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

la stringa $a + b$ ha varie derivazioni:

$$E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$$

e

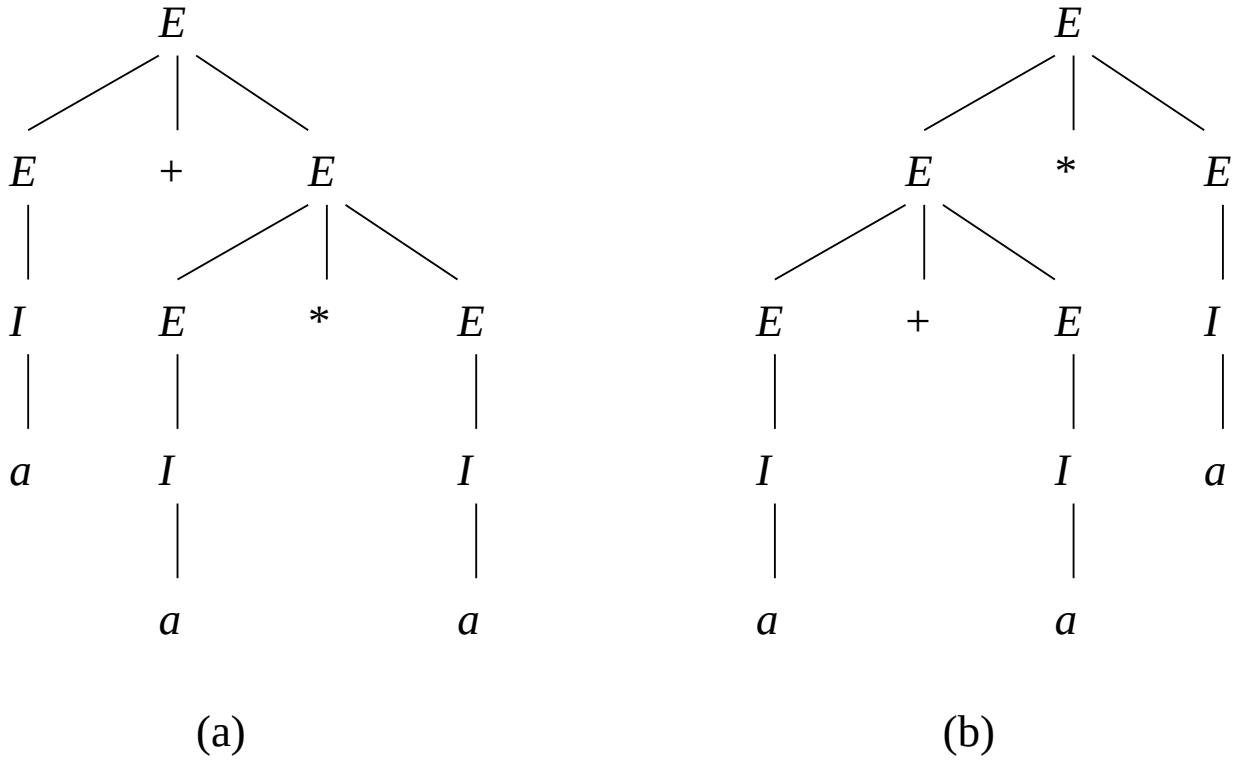
$$E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow I + b \Rightarrow a + b$$

Però il loro albero sintattico è lo stesso, e la struttura di $a + b$ è quindi non ambigua.

Definizione: Sia $G = (V, T, P, S)$ una CFG. Diciamo che G è *ambigua* se esiste una stringa in T^* che ha più di un albero sintattico.

Se ogni stringa in $L(G)$ ha al più un albero sintattico, G è detta *non-ambigua*.

Esempio: La stringa terminale $a + a * a$ ha due alberi sintattici:



Rimuovere l'ambiguità dalle grammatiche

Buone notizie: a volte possiamo rimuovere l'ambiguità

Cattive notizie: non c'è nessun algoritmo per farlo in modo sistematico

Ancora cattive notizie: alcuni CFL hanno solo CFG ambigue

Studiamo la grammatica

$$E \to I \mid E + E \mid E * E \mid (E)$$

$$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

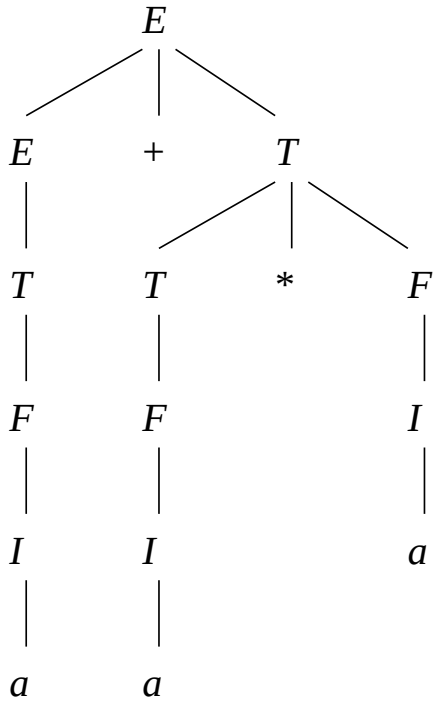
Non c'è precedenza tra * e +

Non c'è raggruppamento di sequenze di operatori: $E + E + E$ è inteso come $E + (E + E)$ o come $(E + E) + E$?

Soluzione:

1. $I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
2. $F \to I \mid (E)$
3. $T \to F \mid T * F$
4. $E \to T \mid E + T$

Ora l'unico albero sintattico per $a + a * a$ è:



Ambiguità inerente

Un CFL L è *inerentemente ambiguo* se *tutte* le grammatiche per L sono ambigue.

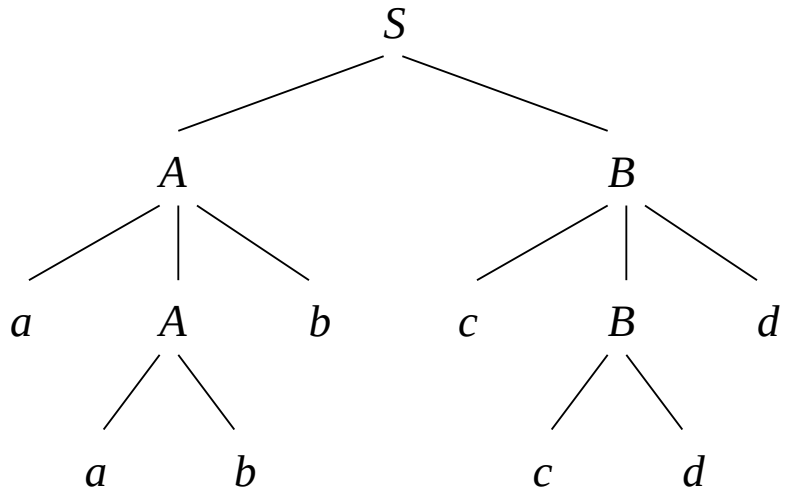
Esempio: Consideriamo $L =$

$$\{a^n b^n c^m d^m : n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n : n \geq 1, m \geq 1\}.$$

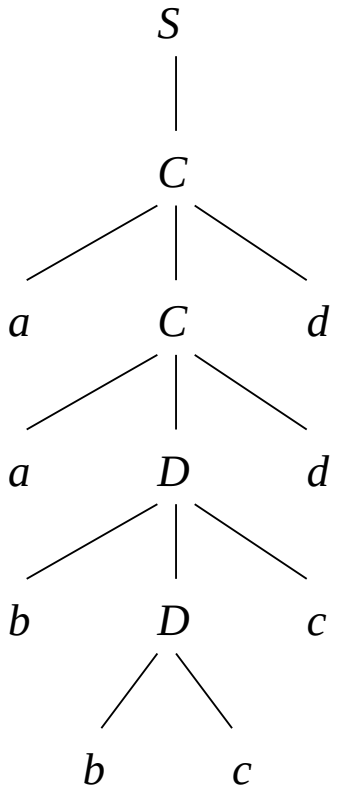
Una grammatica per L è

$$S \to AB \mid C$$
$$A \to aAb \mid ab$$
$$B \to cBd \mid cd$$
$$C \to aCd \mid aDd$$
$$D \to bDc \mid bc$$

Guardiamo la struttura sintattica della stringa *aabbccdd*.



(a)



(b)

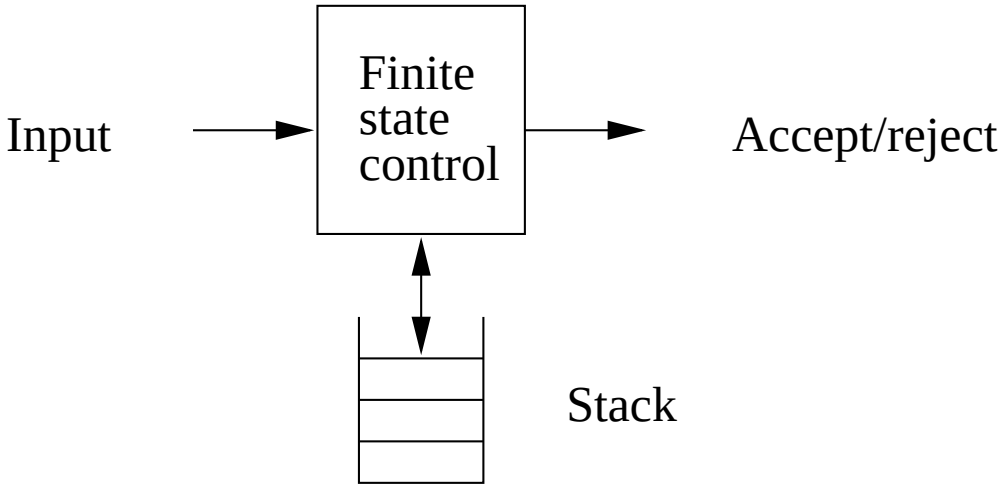
Può essere provato che *ogni* grammatica per L si comporta come questa. Il linguaggio L è quindi inerentemente ambiguo.

Automi a pila

Un automa a pila (PDA) è in pratica un ϵ-NFA con una pila.

In una transizione un PDA:

1. Consuma un simbolo di input.
2. Va in un nuovo stato (o rimane dove è).
3. Rimpiazza il top della pila con una stringa
   (non fa niente, o elimina il top della pila, o mette una stringa in cima alla pila)



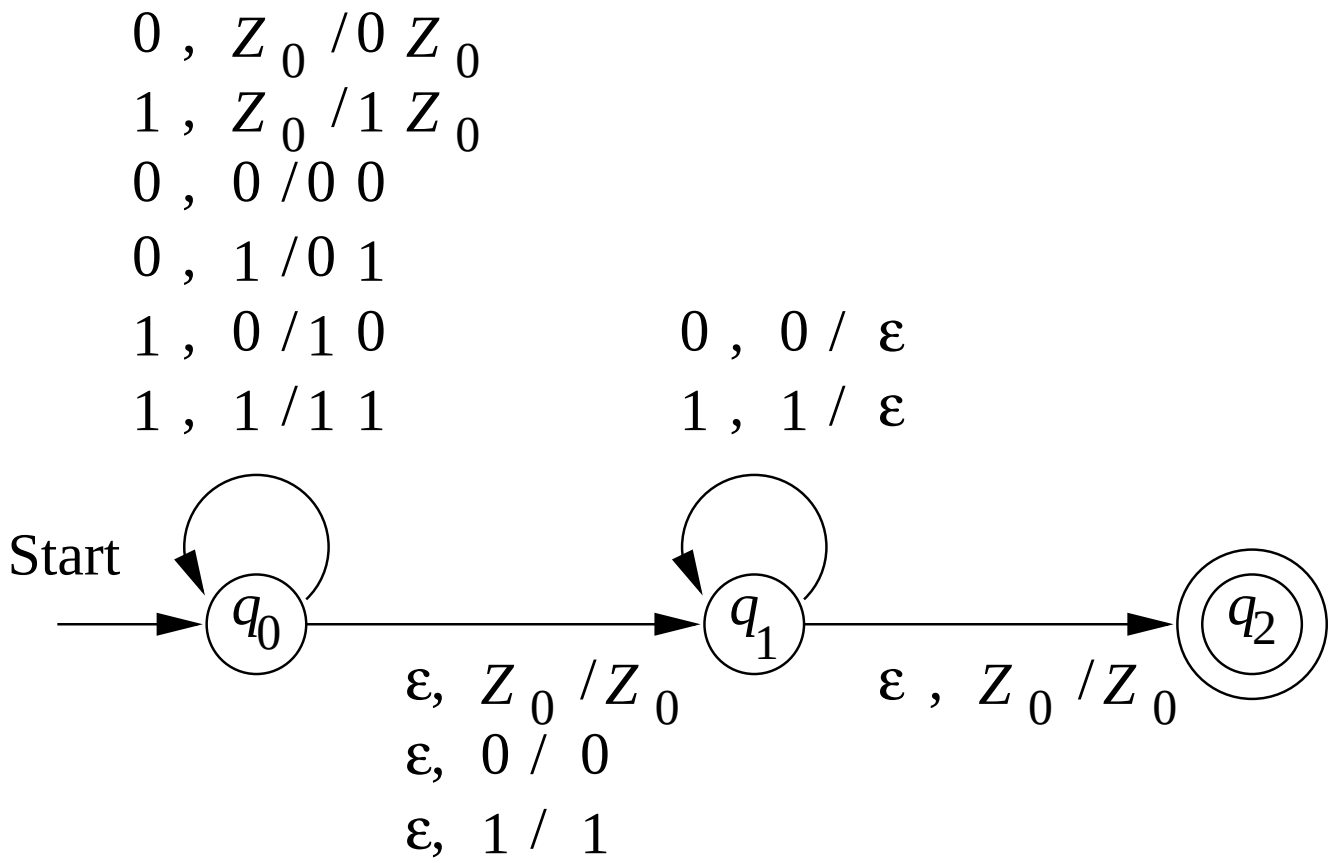
Esempio: Consideriamo

$$L_{wwr} = \{ww^R : w \in \{0,1\}^*\},$$

con "grammatica" $P \to 0P0,\ P \to 1P1,\ P \to \epsilon$. Un PDA per L_{wwr} ha tre stati, e funziona come segue:

1. Scommette che sta leggendo w. Rimane nello stato 0, e mette il simbolo di input sulla pila.
2. Scommette che sta nel mezzo di ww^R. Va spontaneamente nello stato 1.
3. Sta leggendo la testa di w^R. La paragona al top della pila. Se sono uguali, fa un pop della pila, e rimane nello stato 1. Se non sono uguali, si ferma.
4. Se la pila è vuota, va nello stato 2 e accetta la stringa.

Il PDA per L_{wwr} come diagramma di transizione:



definizione formale di PDA

Un PDA è una tupla di 7 elementi:

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F),$$

dove

- Q è un insieme finito di stati,
- Σ è un *alfabeto finito di input,*
- Γ è un *alfabeto finito di pila,*
- $\delta : Q \times \Sigma \cup \{\epsilon\} \times \Gamma \to 2^{Q \times \Gamma^*}$ è la *funzione di transizione,*
- q_0 è lo *stato iniziale,*
- $Z_0 \in \Gamma$ è il *simbolo iniziale* per la pila, e
- $F \subseteq Q$ è l'insieme di *stati di accettazione.*

Equivalenza di PDA e CFG

Un linguaggio è

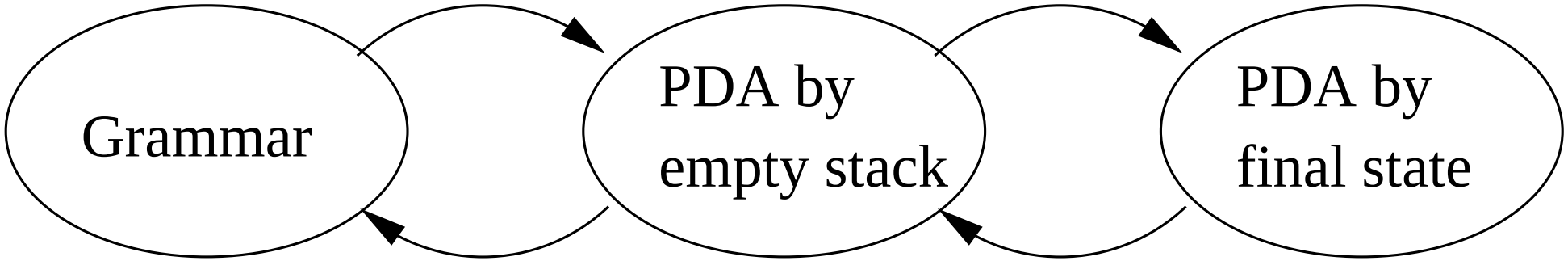
generato da una CFG

se e solo se è

accettato da un PDA per pila vuota

se e solo se è

accettato da un PDA per stato finale

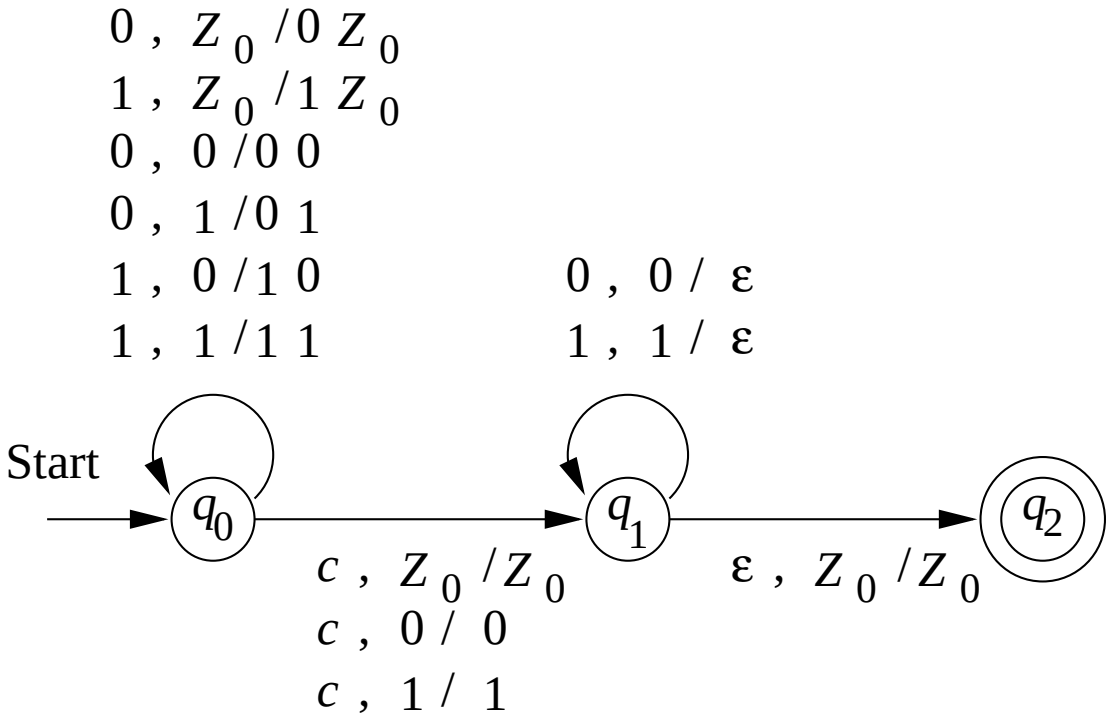


PDA deterministici (DPDA)

Un PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ è *deterministico* se e solo se

1. $\delta(q, a, X)$ è sempre o vuoto o con un solo elemento.
2. Se $\delta(q, a, X)$ non è vuoto, allora $\delta(q, \epsilon, X)$ deve essere vuoto.

Esempio: $L_{wcwr} = \{wcw^R : w \in \{0,1\}^*\}$.



Vale che Regolari$\subset L(\text{DPDA}) \subset$ CFL

Infatti

- **Teorema 6.17:** Se L è regolare, allora $L = L(P)$ per qualche DPDA P.

Quindi Regolari $\subseteq L(\text{DPDA})$.

- $L_{wcwr} \in L(\text{DPDA}) \setminus$ Regolari

Quindi Regolari $\subset L(\text{DPDA})$.

- Ci sono linguaggi in CFL$\setminus L(\text{DPDA})$.

  Esempio: L_{wwr}.

Quindi $L(\text{DPDA}) \subset$ CFL.

Proprietà di CFL

- *Semplificazione* di una CFG. Se un linguaggio è un CFL, ha una grammatica di una forma speciale.

- *Pumping Lemma per CFL.* Simile ai linguaggi regolari.

- *Proprietà di chiusura.* Alcune delle proprietà di chiusura dei linguaggi regolari valgono anche per i CFL.

Forma normale di Chomsky

Ogni CFL (senza ϵ) è generato da una CFG dove tutte le produzioni sono della forma

$$A \to BC, \text{ o } A \to a$$

dove A, B, e C sono variabili, e a è un simbolo terminale. Questa è detta forma normale di Chomsky (CNF), e per ottenerla dobbiamo

1. Eliminare i *simboli inutili*, quelli che non appaiono in nessuna derivazione $S \stackrel{*}{\Rightarrow} w$, per simbolo iniziale S e terminale w.
2. Eliminare le produzioni ϵ, della forma $A \to \epsilon$.
3. Eliminare le *produzioni unità*, cioè produzioni della forma $A \to B$, dove A e B sono variabili.

Esempio

Iniziamo dalla grammatica

$E \to E + T \mid T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$T \to T * F \mid (E)a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$F \to (E) \;\; a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Usando le regole
$A \to a, B \to b, Z \to 0, O \to 1$
$P \to +, M \to *, L \to (, R \to)$
e otteniamo la grammatica

$E \to EPT \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
$T \to TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
$F \to LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
$I \to a \mid b \mid IA \mid IB \mid IZ \mid IO$
$A \to a, B \to b, Z \to 0, O \to 1$
$P \to +, M \to *, L \to (, R \to)$

Per il passo 3, rimpiazziamo

$E \to EPT$ con $E \to EC_1, C_1 \to PT$

$E \to TMF, T \to TMF$ con
$E \to TC_2, T \to TC_2, C_2 \to MF$

$E \to LER, T \to LER, F \to LER$ by
$E \to LC_3, T \to LC_3, F \to LC_3, C_3 \to ER$

La grammatica in CFN finale è

$E \to EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
$T \to TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
$F \to LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
$I \to a \mid b \mid IA \mid IB \mid IZ \mid IO$
$C_1 \to PT, C_2 \to MF, C_3 \to ER$
$A \to a, B \to b, Z \to 0, O \to 1$
$P \to +, M \to *, L \to (, R \to)$

Pumping lemma per CFL

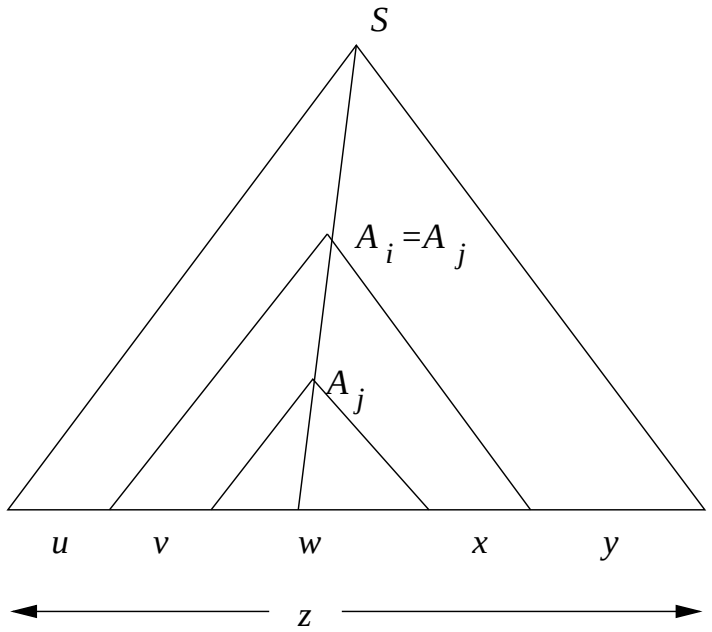
In ogni stringa sufficientemente lunga di un CFL si possono trovare due sottostringhe vicine che è possibile eliminare o ripetere (insieme), ottenendo sempre stringhe del linguaggio.

Pumping lemma per CFL . Sia L un CFL. Esiste una costante n tale che $\forall z \in L$ con $\mid z \mid \geq n$, possiamo scrivere $z = uvwxy$ con le seguenti condizioni:

1. $\mid vwx \mid \leq n$
2. $vx \neq \epsilon$
3. per ogni $i \geq 0$, $uv^i wx^i y \in L$.

Prova informale:

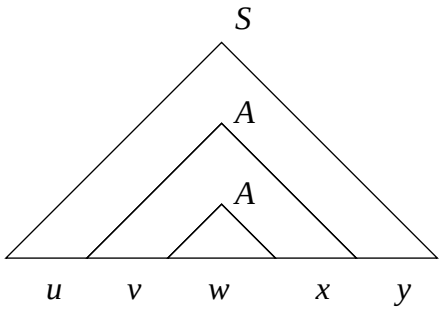
Se la stringa z è sufficientemente lunga, l'albero sintattico che produce $z = uvwxy$ ha un simbolo non terminale che si ripete in un cammino dalla radice ad una foglia. Supponiamo $A_i = A_j$.
Allora può essere individuato il sottoalbero con radice A_j e chiamato w il suo prodotto. Inoltre, dato il sottoalbero con radice A_i, chiamiamo vwx il suo prodotto.

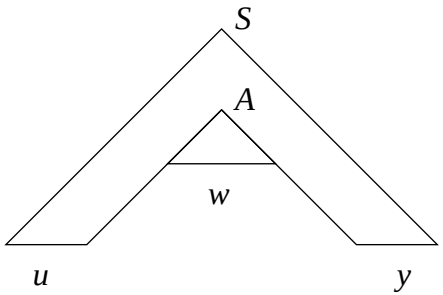


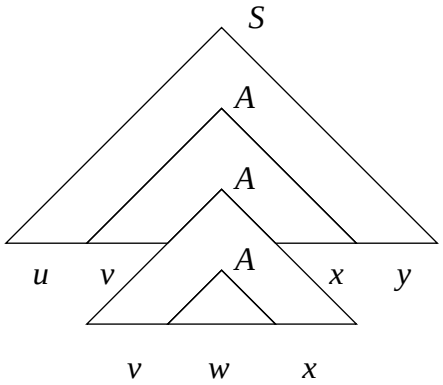
Dato che $A_i = A_j$, possiamo rimpiazzare il sottoalbero di A_i con quello di A_j, ottenendo quindi uwy, che deve ancora appartenere a L.

Oppure possiamo rimpiazzare il sottoalbero di A_j con quello di A_i, ottenendo $uvvwxxy$, ancora generata da L.

Pumping lemma per CFL







Esempio

Consideriamo $L = \{0^k 1^k 2^k \mid k \geq 1\}$.

Dato un n generico, scegliamo $z = 0^n 1^n 2^n$. Comunque noi spezziamo z in $uvwxy$, con $\mid vwx \mid \leq n$ e v e x non entrambi vuote, vwx non può contenere sia 0 che 2 perché l'ultimo 0 e il primo 2 sono lontani n+1 posti. Ci sono i seguenti casi:

- vwx non contiene 2. Allora vx ha solo 0 e 1. Quindi uwy, che dovrebbe essere in L, ha n 2, ma meno di n 0 o 1.
- vwx non contiene 0. Analogo.

Esempi

I CFL non sanno abbinare coppie con lo stesso numero di simboli, se le coppie sono intrecciate.

Esempio: $L = \{0^i 1^j 2^i 3^j \mid i, j \geq 1\}$.

Dato n, scegliamo $z = 0^n 1^n 2^n 3^n$. Quindi vwx contiene un solo simbolo o due simboli. In ogni caso, le stringhe generate non sono in L.

I CFL non sanno abbinare due stringhe di lunghezza arbitraria, se sono su un alfabeto di più di un simbolo.

Esempio: $L = \{ww \mid w \in \{0, 1\}^*\}$.

Dato n, scegliamo $z = 0^n 1^n 0^n 1^n$. Comunque la scomponiamo, non otteniamo stringhe di L.

Proprietà di chiusura dei CFL

Teorema 7.24: I CFL sono chiusi sotto unione, concatenazione, chiusura di Kleene e chiusura positiva +.

Teeorema: Se L è CF, allora lo è anche anche L^R.

Prova: Supponiamo che L sia generato da $G = (V, T, P, S)$. Costruiamo $G^R = (V, T, P^R, S)$, dove

$$P^R = \{A \to \alpha^R : A \to \alpha \in P\}$$

Si mostra per induzione sulla lunghezza delle derivazioni in G e in G^R che $(L(G))^R = L(G^R)$.

I CFL non sono chiusi sotto l'intersezione

Sia $L_1 = \{0^n 1^n 2^i : n \geq 1, i \geq 1\}$. Allora L_1 è CF con grammatica

$$S \to AB$$
$$A \to 0A1|01$$
$$B \to 2B|2$$

Inoltre, $L_2 = \{0^i 1^n 2^n : n \geq 1, i \geq 1\}$ è CF con grammatica

$$S \to AB$$
$$A \to 0A|0$$
$$B \to 1B2|12$$

Invece, $L_1 \cap L_2 = \{0^n 1^n 2^n : n \geq 1\}$ non è CF.

Teorema 7.27: Se L è CF, e R è regolare, allora $L \cap R$ è CF.