CONTEXT-FREE GRAMMAR (CFG)

Una CFG è una tupla $\mathcal{G} = (V, \Sigma, R, S)$, dove:

- V: insieme delle VARIABILI della grammatica;
- Σ: insieme di TERMINALI;
- R: REGOLE DI DERIVAZIONE o PRODUZIONE;
- S: variabile di partenza.

Una CFG produrrà stringhe formate solo da terminali.
Avendo scelta nelle derivazioni, una CFG è sempre NONDETERMINISTICA!

- Esempio:
$\mathcal{G} = (V, \Sigma, R, S)$; $V = \{A, B\}$, $\Sigma = \{0, 1\}$, $S = A$, $R = \{A \to 0A1, A \to B, B \to \varepsilon\}$

Per esempio: $A \to 0A1 \to 00A11 \to 00B11 \to 00\varepsilon 11 \to \underline{0011}$
Genera qualunque stringa del tipo $0^k 1^k$, cioè genera $B = \{0^m 1^m \mid m \geq 0\}$, che è NON regolare.

* Una CFG $\mathcal{G}$ è in grado di generare linguaggi NON regolari, come B!

- **CFL: Context-Free Language:**

Data una CFG $\mathcal{G}$, l'insieme di tutte le stringhe generabili da $\mathcal{G}$ è il linguaggio generato dalla grammatica, ed è detto CONTEXT-FREE LANGUAGE:

$$\mathcal{L}(\mathcal{G}) := \{w \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} w\}$$

TEOREMA: $\boxed{\text{Ogni linguaggio regolare è un CFL.}}$

Se $\mathcal{L}$ è regolare $\Rightarrow \exists$ DFA $M = (Q, \Sigma, \delta, q_0, F)$ tale che $\mathcal{L}(M) = \mathcal{L}$.

* Costruiamo una CFG $\mathcal{G} = (V, \Sigma, R, S)$, tale che:

- $V = \{R_i \mid q_i \in Q\}$: per ogni stato in Q, c'è una variabile in V;
- R è fatto così: ogni volta che ho una transizione $\delta(q_i, a) = q_j$ avrò $R_i \to aR_j$ più le condizioni di accettazione:
$$R = \{R_i \to aR_j \mid \delta(q_i, a) = q_j\} \cup \{R_i \to \varepsilon \mid q_i \in F\}$$
- $S = R_0$.

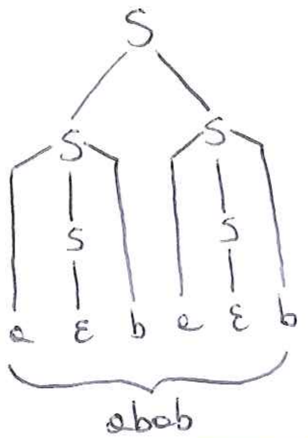
- Esempio: PARSE TREE

$G_2 = (\{S\}, \{a,b\}, R, S)$; $R: S \to aSb \mid SS \mid \varepsilon$

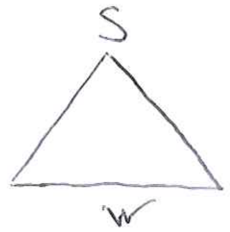
Qual è $L(G_2)$?

$L(G_2) = \{\varepsilon, ab, abab, aabb, \dots\}$

Posso rappresentare le derivazioni con un ALBERO SINTATTICO (PARSE TREE):



$S \Rightarrow^* w \in \Sigma^*$



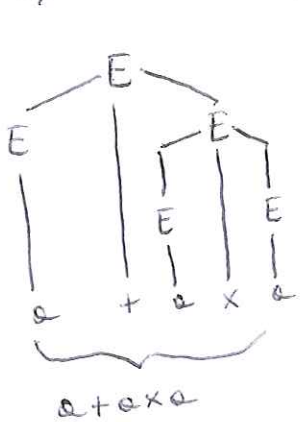
TEOREMA: $\boxed{\text{Se } A \text{ e } B \text{ sono CFL} \Rightarrow A \cup B \text{ è un CFL}}$

Unisco le 2 CFG e aggiungo una variabile iniziale S'' con le regole $S'' \to S \mid S'$.

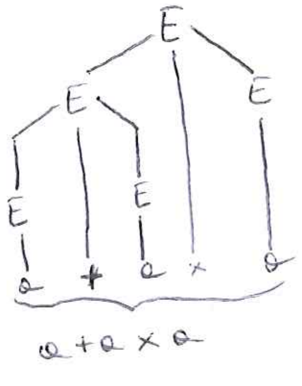
- Esempio:

$G_4 = (\{E\}, \{a, +, \times, (,)\}, R, S \equiv E)$; $R: E \to E \times E \mid E + E \mid (E) \mid a$

$E \Rightarrow^* a + a \times a$, ma questa stringa ha PIÙ PARSE TREES!



oppure



I 2 modi corrispondono al dare precedenza prima all'operatore "+" rispetto al "×" e viceversa.

→ LA CFG è AMBIGUA!

- Derivazione a sinistra:

Una derivazione $u \Rightarrow^* v$ è A SINISTRA se, ad ogni passo, è la variabile più a sinistra nella stringa ad essere sostituita.

Per esempio:

$$xAyBC \Rightarrow xzyBC \Rightarrow xzywC \Rightarrow xzywt$$

- stringa derivata in maniera ambigua:

Una stringa $w \in L(G)$ è DERIVATA AMBIGUAMENTE se possiede 2 o più derivazioni a sinistra, cioè se possiede 2 o più ALBERI SINTATTICI.

- GRAMMATICA AMBIGUA:

Una CFG G è AMBIGUA se $\exists\, w \in L(G)$ tale che w è derivata ambiguamente.
Basta che esiste 1 SOLA STRINGA w!

FORMA NORMALE DI CHOMSKY (CNF)

Una CFG $G = (V, \Sigma, R, S)$ si dice essere in FORMA NORMALE DI CHOMSKY se OGNI REGOLA di derivazione ha una delle seguenti forme:

- $A \rightarrow BC$, con $A \in V$, $B, C \in V \setminus \{S\}$
- $A \rightarrow a$, con $a \in \Sigma$
- $S \rightarrow \varepsilon$

TEOREMA: Ogni CFL è generabile da una CFG in FORMA NORMALE DI CHOMSKY

- COSTRUZIONE: Sia L un CFL $\Rightarrow \exists$ CFG $G = (V, \Sigma, R, S)$ che genera L.

Trasformiamo G in CNF.

(1) Aggiungiamo una nuova variabile di partenza S_0 e la regola $S_0 \rightarrow S$

(2) Risolviamo le regole del tipo $A \rightarrow \varepsilon$, ma con $A \neq S_0$.
Supponiamo che esiste $B \rightarrow uAwAz$ ed $A \rightarrow \varepsilon$
Sostituiamole con le seguenti regole: $B \rightarrow uwAz \mid uAwz \mid uwz$

- CASO PARTICOLARE:
se $B \rightarrow A$ ed $A \rightarrow \varepsilon$, mettiamo come nuova regola $B \rightarrow \varepsilon$ SOLO SE $B \rightarrow \varepsilon$ non è già stata rimossa.

(3) Sostituiamo le regole $A \to B$.

Si cercano le regole $B \to u$ e, per ogni regola, si sostituisce $A \to u$, <u>SOLO SE</u> $A \to u$ non è già stata rimossa in precedenza (u può essere anche una variabile, $u \in V$).

(4) Regole troppo lunghe: $A \to u_1 \dots u_k$, $k > 2$.

Si introducono: $A \to u_1' A_1$, $A_1 \to u_2' A_2$, ..., $A_{k-2} \to u_{k-1}' u_k'$

dove $u_i' = \begin{cases} u_i & \text{se } u_i \in V \\ U_i & \text{se } u_i \in \Sigma \text{ (è un terminale)} \end{cases}$

(5) Risolviamo gli U_i: $U_i \to u_i$, $\forall u_i \in \Sigma$.

Esempio:

G: $V = \{S, A, B\}$, $\Sigma = \{a, b\}$, $S = S$, R:
$S \to ASA \mid aB$
$A \to B \mid S$
$B \to b \mid \varepsilon$

1) Aggiungiamo S_0 e $S_0 \to S$;

2) Rimuoviamo le regole unarie: partiamo da $\boxed{B \to \varepsilon}$

$S_0 \to S$
$S \to ASA \mid aB \mid a$
$A \to B \mid S \mid \varepsilon$
$B \to b$

$\boxed{A \to \varepsilon}$ $\longrightarrow$
$S_0 \to S$
$S \to ASA \mid aB \mid a \mid SA \mid AS \mid S$
$A \to B \mid S$
$B \to b$

$S \to S$ è inutile, si rimuove

(3) Rimuoviamo le altre regole unarie:

$\boxed{S_0 \to S}$: $S_0 \to ASA \mid aB \mid a \mid SA \mid AS$

$\boxed{A \to B}$: $A \to b \mid S$

$\boxed{A \to S}$: $A \to b \mid ASA \mid aB \mid a \mid SA \mid AS$

(4) Regole con più di 2 termini:

$\boxed{S_0 \to ASA}$
$\boxed{S \to ASA}$
$\boxed{A \to ASA}$

Introduciamo $A_1 \to SA$; quindi:

$S_0 \to AA_1$, $S \to AA_1$, $A \to AA_1$

Dunque abbiamo:

$$S_0 \to AA_1 \mid aB \mid a \mid SA \mid AS$$
$$S \to AA_1 \mid aB \mid a \mid SA \mid AS$$
$$A \to b \mid AA_1 \mid aB \mid a \mid SA \mid AS$$
$$B \to b$$
$$A_1 \to SA$$

(5) Sono rimaste da sistemare: $\boxed{S_0 \to aB}$, $\boxed{S \to aB}$, $\boxed{A \to aB}$

Introduciamo $U_1 \to a$ ed otteniamo:

R: $$\boxed{\begin{array}{l} S_0 \to AA_1 \mid U_1B \mid a \mid SA \mid AS \\ S \to AA_1 \mid U_1B \mid a \mid SA \mid AS \\ A \to b \mid AA_1 \mid U_1B \mid a \mid SA \mid AS \\ B \to b \\ A_1 \to AS \\ U_1 \to a \end{array}}$$

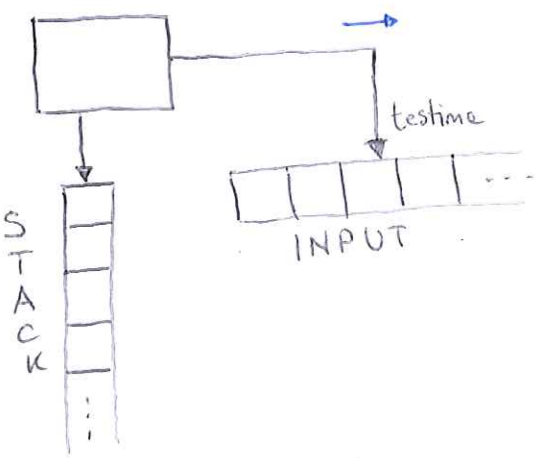
Se $V = \{S_0, S, A, B, A_1, U\} \Rightarrow \mathcal{G} = (V, \Sigma, R, S_0)$ è in FORMA NORMALE DI CHOMSKY

• PDA : Push-Down Automaton

Un PDA è una tupla $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, dove:

- Q := insieme degli STATI INTERNI
- Σ := alfabeto di input (del nastro)
- Γ := alfabeto dello stack
- $\delta : (Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon) \longmapsto 2^{Q \times \Gamma_\varepsilon}$, con $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$, $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$

  è NON DETERMINISTICO, poiché può portare a SOTTOINSIEMI di configurazioni (stato, stack),
- $q_0 \in Q$:= START STATE
- $F \subseteq Q$:= stati di accettazione.



* Al contrario dei DFA, ha capacità di MEMORIA INFINITA, ma può leggere solo l'elemento affiorante sullo stack.

* La testina può muoversi solo verso destra!

- Computazione Accettante (PDA):

PDA M; stringa $w \in \Sigma^*$; computazione $M(w)$ è ACCETTANTE se:

1) $w = w_1 \dots w_m$, dove $w_i \in \Sigma_\varepsilon$
2) $\exists$ sequenza $r_0, r_1, \dots, r_m$ di stati, con $r_i \in Q$
3) $\exists$ sequenza $s_0, s_1, \dots, s_m$ di simboli di stack, con $s_i \in \Gamma^*$

   s_i è CONFIGURAZIONE attuale dello STACK.

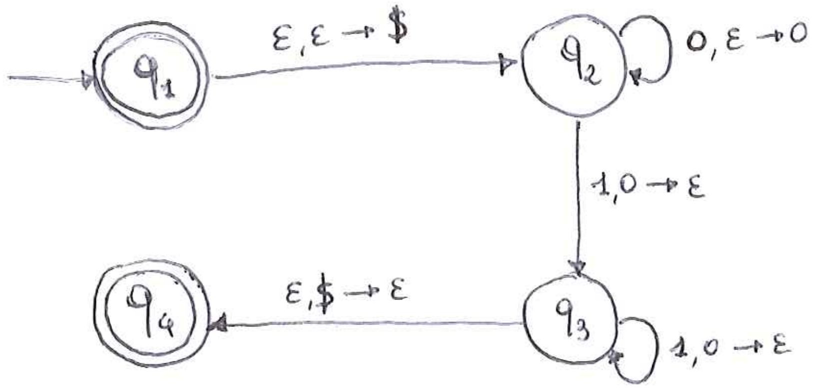
tali che:

- $r_0 = q_0$
- $s_0 = \varepsilon$
- $(r_{i+1}, b) = \delta(r_i, w_{i+1}, a)$, dove $s_i = at$, $s_{i+1} = bt$ e $a, b \in \Gamma_\varepsilon$, $t \in \Gamma^*$
- $r_m \in F$

Esempio:

Sia $B = \{0^n 1^n \mid n \geq 0\}$, che è un CFL.

Il PDA $M = (Q, \Sigma, \Gamma, \delta, q_1, F)$ che lo riconosce è:



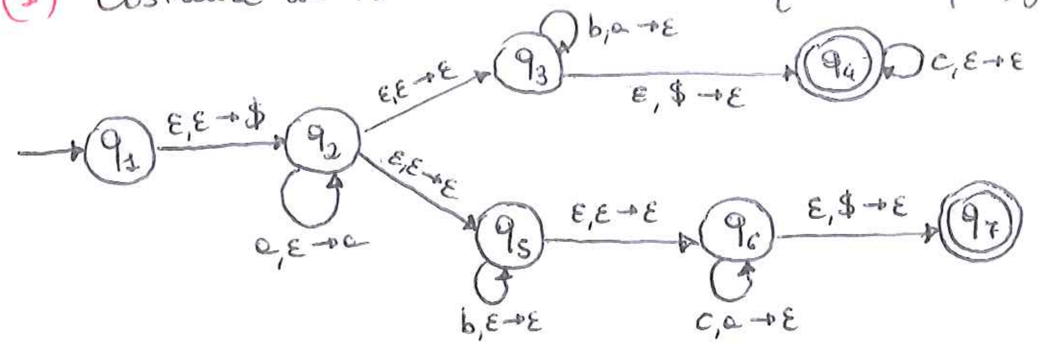
Come fa la macchina a capire che lo stack è vuoto?

Si fa all'inizio una PUSH di un simbolo speciale $ di fine stack.

Poi, alla fine, usa il NONDETERMINISMO nell'indovinare quando l'input sia terminato!

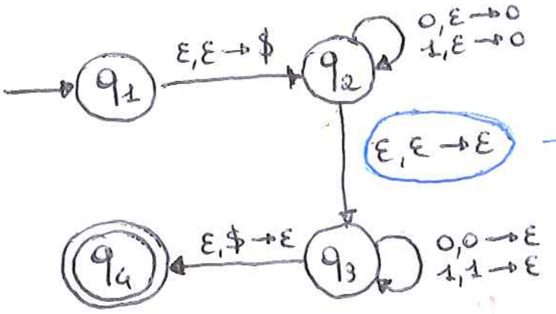
Esercizi:

(1) Costruire un PDA che riconosce $A = \{ a^i b^j c^k \mid i,j,k \geq 0 \wedge (i=j \vee i=k) \}$



* **NOTA**: sfruttiamo il NONDETERMINISMO. Inoltre, A è un CFL!

(2) Il linguaggio $B = \{ ww^R \mid w \in \{0,1\}^* \}$ è un CFL, in quanto una CFG che lo genera è $G = (\{S\}, \{0,1\}, R, S)$, con $R: S \to 0S0 \mid 1S1 \mid \varepsilon$.
Costruire un PDA che lo riconosce.



NONDETERMINISMO per indovinare quando finisce w ed inizia w^R

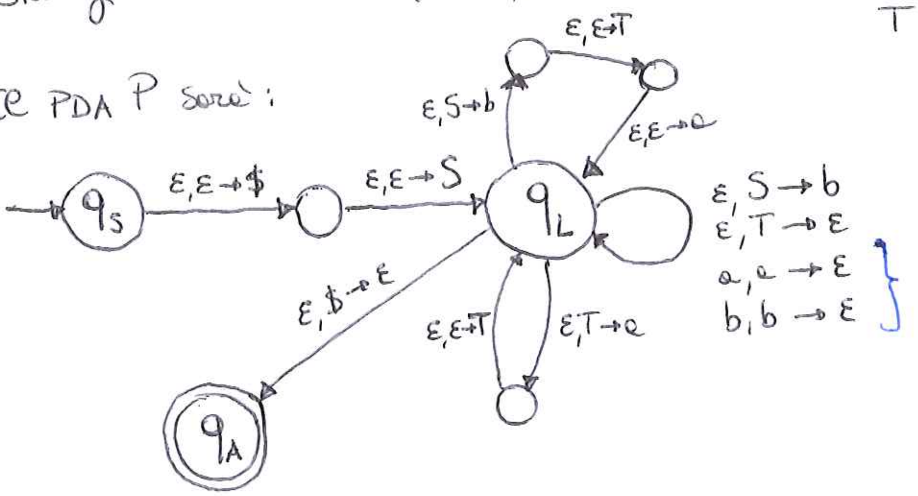
TEOREMA

$$\boxed{\mathcal{L} \text{ è un CFL} \iff \exists \text{ un PDA } P \text{ tale che } \mathcal{L}(P) = \mathcal{L}}$$

- Da CFG a PDA: $(\Rightarrow)$

Sia G tale che: $V = \{S, T\}$, $\Sigma = \{a, b\}$, R: $S \to aTb \mid b$
$T \to Ta \mid \varepsilon$

Il PDA P sarà:



POP dei terminali simili, per poter operare sulla variabile sullo stack, di modo che siano affiorati.

Abbiamo dimostrato che: $CFL \equiv CFG \equiv PDA$

- **Corollario**: $\boxed{\text{Se } A \text{ è un linguaggio regolare} \Rightarrow A \text{ è un CFL.}}$

Proof: A è regolare $\Rightarrow \exists$ NFA N tale che $\mathcal{L}(N) = A$.

$*$ Ma, un NFA è un PDA che non usa lo stack!

$\Rightarrow \exists$ PDA N tale che $\mathcal{L}(N) = A \Rightarrow A$ è un CFL! c.v.d.!!!

Riepilogo:

Regular Languages $\subseteq$ (sottoclasse) CFL

Regular Languages	$\subseteq$ sottoclasse	CFL
$\updownarrow$		$\updownarrow$
DFA		CFG
$\updownarrow$		$\updownarrow$
NFA		PDA
$\updownarrow$		
GNFA		
$\updownarrow$		
REX		

Pumping Lemma (CFL)

Se A è un CFL, allora esiste una "pumping lenght" $p > 0$ tale che:

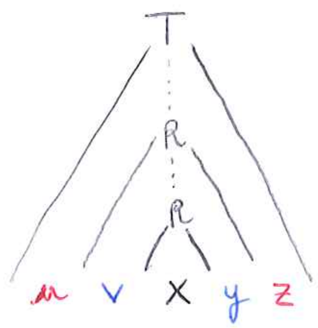
se $s \in A$, $|s| \geq p \Rightarrow s = uvxyz$, tale che valgono:

(1) $\forall i \geq 0$: $uv^ixy^iz \in A$;

(2) $|vy| > 0$;

(3) $|vxy| \leq p$.

L'idea è che, se A è CFL, c'è una CFG G che lo riconosce; tuttavia, se $|s| \geq p$, il numero di variabili $|V|$ è finito.
Per il principio della piccionaia, ci sarà una certa variabile R che si ripete, e, al massimo, avrà un'espansione del tipo $R \to aRb$, con a e b parti di stringhe.



$\to$ Posso suddividere S in 5 sottostringhe!

$*$ La condizione $|vxy| \leq p$, serve per non far ripetere R nell'ultimo albero sintattico.

Esercizi:

(1) Dimostrare che $B = \{a^n b^n c^n \mid n \geq 0\}$ NON è CFL.

Per assurdo, sia CFL $\Rightarrow$ vale il pumping lemma ($\exists\, p > 0$).

Consideriamo $s = a^p b^p c^p \Rightarrow |s| \geq p$

Dunque $s = uvxyz$, tale che:
- $\forall i \geq 0 : uv^ixy^iz \in B$
- $|vy| > 0$
- $|vxy| \leq p$

Sfruttando $|vxy| \leq p$, si vede che non esiste nessuna suddivisione che permette di pompare allo stesso modo sia a che b che c $\Rightarrow$ ASSURDO $\Rightarrow$ B non è CFL!

(2) $C = \{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}$. Dimostrare che C NON è CFL.

Lo sia, per assurdo. $\Rightarrow \exists\, p > 0$ tale che vale il PL.

Sia $s = a^p b^p c^p$, $|s| \geq p$.

$s = uvxyz$. Per la condizione $|vxy| \leq p$, si ha che $vxy \notin a^+b^+c^+$, quindi si può sempre pompare verso l'alto o verso il basso per non far più appartenere s' a C $\Rightarrow$ ASSURDO $\Rightarrow$ C non è CFL!

(3) Homework:

Abbiamo visto che $\{ww^R \mid w \in \{0,1\}^*\}$ è un CFL.

Dimostrare che $D = \{ww \mid w \in \{0,1\}^*\}$ NON è un CFL.

Lo sia per assurdo $\Rightarrow \exists\, p > 0$ per PL $\Rightarrow$ Sia $s = 0^p 1^p 0^p 1^p$, $|s| \geq p$ $(w = 0^p 1^p)$

$\Rightarrow s = uvxyz$, Poiché $|vxy| \leq p$, abbiamo 3 casi:
- (1) $vxy \in 1^+$: ma, per $i = 2 \rightarrow uvvxyyz \notin D$, poiché gli 1 nelle parti DX/SX differiscono
- (2) $vxy \in 0^+$: analogo ad (1)
- (3) $vxy \in 1^+0^+ \vee 0^+1^+$: non si otterrà mai una stringa pompabile a piacimento verso l'alto o verso il basso che $\in D$.

$\Rightarrow$ ASSURDO $\Rightarrow$ D non è CFL!

* **NOTA**: stavolta $s = 0^p 1 0^p 1$ non sarebbe bastata, perché con $vxy = 0^k 1 0^k$ sarebbe stata pompabile.

- UNIONE, CONCATENAZIONE, KLEENE'S STAR

$\boxed{\text{I CFL sono chiusi rispetto all'unione } \cup \text{, alla concatenazione } \circ \text{ e alla Kleene's Star } * .}$

Basta combinare insieme le CFG G_A e G_B, aggiungendo una nuova variabile di partenza S con le rispettive regole:

$\cup :=$ $S \to S_A \mid S_B$

$\circ :=$ $S \to S_A S_B$

$* :=$ $S \to S S_A \mid \varepsilon$

- REVERSAL e OMOMORFISMO:

$\boxed{\text{I CFL sono chiusi rispetto al reversal } ^R \text{ e agli omomorfismi}}$

* INTERSEZIONE:

$\boxed{\text{I CFL \underline{non} sono chiusi rispetto all' INTERSEZIONE } \cap}$

Proof: Mostriamo un esempio.

$B = \{a^m b^m c^m \mid m \geq 0\}$ abbiamo visto essere NON CFL.

Possiamo esprimerlo come:

$$B = \{a^i b^m c^m \mid m \geq 0 \wedge i \geq 0\} \cap \{a^m b^m c^i \mid m \geq 0 \wedge i \geq 0\}$$

Questi 2 linguaggi sono CFL, in quanto sono della forma $\{a^n b^n \mid n \geq 0\}$, che è CFL.

Tuttavia, B non lo è. $\Rightarrow$ I CFL NON sono chiusi rispetto all' INTERSEZIONE.

ok!!! ✓

RICAPITOLANDO

	UNIONE	CONCATENAZIONE	STAR	INTERS.	REVERSAL	HOMOMORFISM	COMPL.
Linguaggi Regolari	✓	✓	✓	✓	✓	✓	✓
CFL	✓	✓	✓	✗	✓	✓	✗

- **DPDA : Deterministic PDA**

Un DPDA è un PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ con una FUNZIONE DI TRANSIZIONE δ differente:

$$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longmapsto (Q \times \Gamma_\varepsilon) \cup \{\emptyset\}$$

(anziché di $2^{(Q \times \Gamma_\varepsilon)}$ → NONDETERMINISMO)

$$(q, a, x) \longrightarrow \begin{cases} \emptyset \\ (q', x') \end{cases}$$

Con la seguente CONDIZIONE DI DETERMINISMO più stringente:

- $\forall q \in Q,\ \forall a \in \Sigma,\ \forall x \in \Gamma$:

$$\left| \{\delta(q,a,x),\ \delta(q,a,\varepsilon),\ \delta(q,\varepsilon,x),\ \delta(q,\varepsilon,\varepsilon)\} \setminus \{\emptyset\} \right| = 1$$

In pratica, ESATTAMENTE 1 delle possibili transizioni da uno stato può essere lecita; non posso avere più opportunità con gli stessi simboli → DETERMINISMO

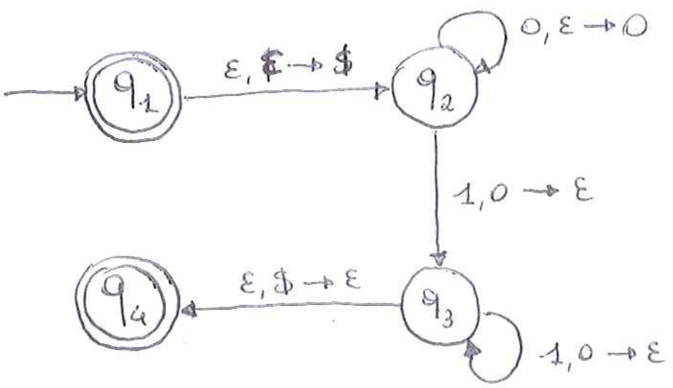
* Se lo stack non è vuoto, c'è 1 e 1 sola possibilità di procedere;

* Se lo stack è vuoto, o c'è una mossa legale che non fa una POP ($\delta(q,a,\varepsilon)$, $\delta(q,\varepsilon,\varepsilon)$), oppure NON ci sono mosse legali per procedere.

- I linguaggi riconosciuti dai DPDA sono detti DCFL: Deterministic Context-Free Language.

• Esempi :

(1) $B=\{0^n1^n \mid n \geq 0\}$ è un CFL . Il PDA che lo riconosce è :



In realtà è anche un DPDA! Poiché c'è sempre e soltanto 1 UNICA mossa legale da poter fare, quindi è deterministico.

⇒ B non solo è un CFL , ma è un DCFL !

(2) $\{a^ib^jc^k \mid i,j,k \geq 0 \wedge (i=j \vee i=k)\}$ non è un DCFL.

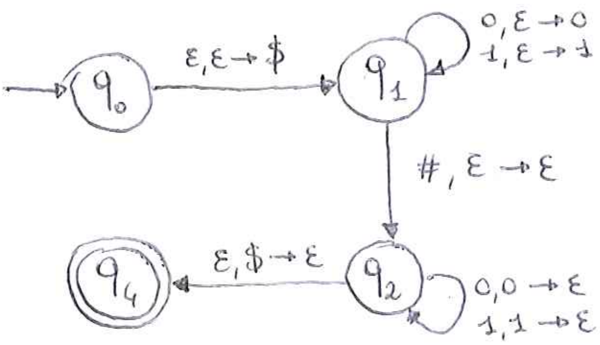
Infatti, ho per forza bisogno del NONDETERMINISMO per poter scegliere se confrontare le a con le b oppure le a con le c.

(3) $\{ww^R \mid w \in \{0,1\}^*\}$ è un CFL , ma non un DCFL ! Uso il nondeterminismo per indovinare quando termina w e inizia w^R.

(4) Homework :

Dimostrare che $\{w\#w^R \mid w \in \{0,1\}^*\}$ è un DCFL.

Un DPDA che lo riconosce è il seguente :



⇒ $\{w\#w^R \mid w \in \{0,1\}^*\}$ è un Deterministic Context-Free Language !

ok!!!

• **TEOREMA**: $\boxed{\text{Ogni linguaggio regolare è un DCFL.}}$

Se A è regolare $\Rightarrow \exists$ DFA M tale che $\mathcal{L}(M) = A$.

Ma, una DFA è un DPDA che non usa lo STACK!

$\Rightarrow \exists$ DPDA M tale che $\mathcal{L}(M) = A \Rightarrow A$ è un DCFL!

• **TEOREMA**: $\boxed{\text{I DCFL sono chiusi rispetto al COMPLEMENTO }^c.}$

Deriva dal seguente **LEMMA**:

[Ogni DPDA, ha un DPDA equivalente che legge SEMPRE l'INTERA STRINGA in input, anche quando rifiuta.]

• **Corollario**: $\boxed{\text{Se } A \text{ è un CFL, ma } A^c \text{ non è un CFL} \Rightarrow A \text{ non è un DCFL}}$

Se A fosse un DCFL $\Rightarrow$ anche A^c sarebbe un DCFL (chiusura sul complemento), ma non è nemmeno CFL $\Rightarrow A$ non è DCFL!

ok!!!

• **LEMMA**: $\boxed{\text{Se } A \text{ è un CFL e } B \text{ è REGOLARE} \Rightarrow A \cap B \text{ è un CFL.}}$

A è CFL $\Rightarrow \exists$ PDA $M_A = (Q_A, \Sigma, \Gamma, \delta_A, q_0^A, F_A)$ t.che $\mathcal{L}(M_A) = A$

B è REGOLARE $\Rightarrow \exists$ DFA $M_B = (Q_B, \Sigma, \delta_B, q_0^B, F_B)$ t.che $\mathcal{L}(M_B) = B$.

Costruiamo un nuovo PDA M:

$$M = (Q_A \times Q_B, \Sigma, \Gamma, \delta', (q_0^A, q_0^B), F_A \times F_B) \Rightarrow \mathcal{L}(M) = A \cap B$$

$\Rightarrow A \cap B$ è un CFL!

ok!!!

• Esempio:

$A = \{a^i b^j c^k \mid i,j,k \geq 0 \wedge (i \neq j \vee i \neq k)\}$ è un CFL, poiché si può usare lo stesso PDA che usavamo per vedere se $i = j \vee i = k$.

Però non è un ~~DCFL~~ DCFL! Infatti:

$A^c = \{a^n b^n c^n \mid n \geq 0\}$ non è CFL (via Pumping Lemma).

$\Rightarrow$ A è CFL, ma A^c non è CFL $\Rightarrow$ A non è DCFL! ok!

• TEOREMA: La classe dei DCFL non è chiusa rispetto a unione, intersezione, star, concatenazione, reversal e omomorfismo.

RIEPILOGO

	$\cup$	$\circ$	$*$	$\cap$	R	$h()$	c
Regular	✓	✓	✓	✓	✓	✓	✓
CFL	✓	✓	✓	✗	✓	✓	✗
DCFL	✗	✗	✗	✗	✗	✗	✓

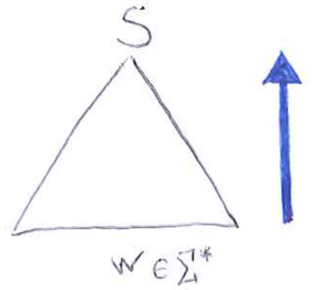
- **ENDMARKED LANGUAGE**:

Sia $A \subseteq \Sigma^*$ e $\dashv \notin \Sigma$. Un LINGUAGGIO MARCATO ("endmarked language") è:

$$A\dashv := \{ w\dashv \mid w \in A \}$$

- **TEOREMA**: $\boxed{\text{Un linguaggio } A \text{ è un DCFL} \iff A\dashv \text{ è un DCFL}}$

- **PROCEDIMENTO DI RIDUZIONE sulle CFG**:



- REDUCE STEP: $\underset{\text{reducing string}}{u} \rightarrowtail \underset{\text{reduced string}}{u'}$

- Se $u \rightarrowtail \dots \rightarrowtail v \Rightarrow u \rightarrowtail^* v$ Esiste una RIDUZIONE tale per cui u è RIDUCIBILE a v

- Se $u \rightarrowtail^* S \Rightarrow$ Esiste una RIDUZIONE da u.

- Esempio:

$S \to abSb \mid \varepsilon$

(1) DERIVAZIONE: $S \Rightarrow abSb \Rightarrow ababSbb \Rightarrow ababbb$

(2) RIDUZIONE: $ababbb \rightarrowtail ababSbb \rightarrowtail abSb \rightarrowtail S$

Anche altre riduzioni sono possibili, ma non tutte portano ad S.

- **RIDUZIONE A SINISTRA**:

È una DERIVAZIONE a DESTRA alla rovescia.

"LEFTMOST REDUCTION" := ogni stringa da ridurre, viene ridotta solo DOPO che tutte le altre reducing string alla sua SINISTRA.

- Sia $w \in \mathcal{L}(G)$ e sia $w = u_1 \rightarrowtail u_2 \rightarrowtail \dots \rightarrowtail u_i \rightarrowtail u_{i+1} \rightarrowtail \dots \rightarrowtail S$ una LEFTMOST REDUCTION.

Consideriamo il passo $u_i \rightarrowtail u_{i+1}$ e supponiamo che la regola di DERIVAZIONE sia $T \to h \Rightarrow u_i = xhy$, con $T \in V$, $h, x \in (\Sigma \cup V)^*$,

$u_{i+1} = xTy$ MA: $y \in \Sigma^*$ (Solo TERMINALI, perché sto facendo una RIDUZIONE A SINISTRA)

- **HANDLE** (maniglia):

Un HANDLE di u_i è la coppia $(h, T \to h)$. E' definito solo per STRINGHE VALIDE, cioè che appaiono in una qualsiasi derivazione di una stringa generata dalla grammatica.

- **TEOREMA**:

> Se una CFG $\mathcal{G}$ è NON AMBIGUA $\Rightarrow$ Ogni stringa valida ha 1 UNICO handle.

Proof: Se $\mathcal{G}$ non è ambigua $\Rightarrow \forall w \in \mathcal{L}(\mathcal{G}), \exists!$ parse tree $\underset{w}{\overset{S}{\Delta}} \Rightarrow$
$\Rightarrow \exists!$ derivazione destra $\Rightarrow \exists!$ RIDUZIONE A SINISTRA $\Rightarrow$
$\Rightarrow \exists!$ HANDLE $\forall u_i$, passo di riduzione di w. ok!! ■

- **COROLLARIO**:

> Se $\exists$ stringa $w \in \mathcal{L}(\mathcal{G})$ tale che w ha più di 1 handle, ALLORA $\mathcal{G}$ è AMBIGUA!

- Esempio:

$$\mathcal{G}_0: \left.\begin{array}{l} R \to S \mid T \\ S \to aSb \mid ab \\ T \to aTbb \mid aT \mid a \end{array}\right\} \quad \text{DERIVAZIONI:} \quad \begin{array}{l} R \Rightarrow S \Rightarrow aSb \Rightarrow aabb \\ R \Rightarrow T \Rightarrow aTbb \Rightarrow aabb \end{array}$$

- RIDUZIONI:

$$\left.\begin{array}{l} aabb \rightarrowtail aTbb \rightarrowtail T \rightarrowtail R \\ aabb \rightarrowtail aSb \rightarrowtail S \rightarrowtail R \end{array}\right\} \text{ Ho 2 RIDUZIONI A SINISTRA!}$$

$\Rightarrow$ Per $w = aabb$ ho 2 HANDLE DIVERSI: $(a, T \to a)$ e $(ab, S \to ab)$

$\Rightarrow \mathcal{G}_0$ è una CFG AMBIGUA!

- Esempio:

$$\mathcal{G}_1: \begin{array}{l} R \to S \mid T \\ S \to aSb \mid ab \\ T \to aTbb \mid abb \end{array} \quad \Rightarrow \mathcal{L}(\mathcal{G}_1) = B \cup C, \text{ con: } \begin{array}{l} B = \{a^m b^m \mid m \geq 1\} \\ C = \{a^m b^{2m} \mid m \geq 1\} \end{array}$$

Se ho: $\underset{B}{\underset{\Uparrow}{aaabbb}} \rightarrowtail aaSbb \rightarrowtail aSb \rightarrowtail S \rightarrowtail R$

$\underset{C}{\underset{\Uparrow}{aaabbbbbb}} \rightarrowtail aaTbbbb \rightarrowtail aTbb \rightarrowtail T \rightarrowtail R$

- Esempio:

G_2: $R \to 1S \mid 2T$
$S \to aSb \mid ab$
$T \to aTbb \mid abb$

$\Rightarrow L(G_2) = B \cup C$, con $B = \{1a^m b^m \mid m \geq 1\}$
$C = \{2a^m b^{2m} \mid m \geq 1\}$

$L(G_2)$ è un DCFL ! Leggendo il 1° carattere, si sa se $w \in B$ o $w \in C$, non deve indovinare in maniera NONDETERMINISTICA.

- Esempio (derivazione a sinistra)

G_3: $S \to T\dashv$
$T \to T(T) \mid \varepsilon$

$$()()\dashv \rightarrowtail T()()\dashv \rightarrowtail T(T)()\dashv \rightarrowtail T()\dashv \rightarrowtail T(T)\dashv \rightarrowtail T\dashv \rightarrowtail S.$$

- HANDLE FORZATO:

Un handle di una stringa valida $v = xhy$ è un HANDLE FORZATO se h è l'UNICO HANDLE per OGNI STRINGA valida xhw, con $w \in \Sigma^*$.

Cioè se l'handle è UNIVOCAMENTE DETERMINATO a prescindere dalla sottostringa w che segue l'handle stesso!

- DCFG: Deterministic CFG

Una CFG è una **DCFG** se ogni stringa valida per la grammatica ha un HANDLE FORZATO.