Gemini 2

🎓 Simulazione Secondo Parziale - PROVA 8 (New Patterns)

Obiettivo: Affrontare logiche diverse (somme negli esponenti, palindromi con centro, stati di rifiuto espliciti).

❓ Esercizio 1: Proprietà di Prefisso (3 Linguaggi)

Domanda:
Stabilire se i seguenti linguaggi godono della proprietà di prefisso.

1. $L_1 = \{a^n b^k \mid k > n \geq 1\}$ (Più `b` che `a`)
2. $L_2 = \{a^n b^n c^n \mid n \geq 1\}$ (Conteggio triplo uguale)
3. $L_3 = \{w \in \{0,1\}^* \mid w \text{ finisce con } 00\}$

💡 Svolgimento Guidato Passo-Passo:

1. Analisi L_1 ($k > n$):

- **Logica:** Le `b` devono essere strettamente maggiori delle `a`.
- **Test:**
  - Stringa corta: $n = 1, k = 2 \Rightarrow x = abb$. Valida ($2 > 1$).
  - Estensione: Aggiungo una `b` $\Rightarrow y = abbb$.
  - Verifica y: $n = 1, k = 3$. Valida ($3 > 1$).
- **Conclusione:** x è prefisso di y.
- **Risposta: NO**.

2. Analisi L_2 ($a^n b^n c^n$):

- **Logica:** Struttura rigida. Ogni volta che aumento `a`, devo aumentare `b` E `c` contemporaneamente.

- **Test:**
  - Stringa corta ($n = 1$): $x = abc$.
  - Estensione: Provo ad aggiungere caratteri in coda.
    - Aggiungo `c`: $abcc$ ($n_a = 1, n_b = 1, n_c = 2$). **No**.
    - Aggiungo `a`: Ordine sbagliato.
- **Conclusione:** Non posso allungare la stringa senza dover modificare anche l'inizio (le `a`).
- **Risposta: SÌ**.

3. Analisi L_3 (Suffix `00`):

- **Logica:** Basta che gli ultimi due simboli siano zeri.
- **Test:**
  - Stringa corta: $x = 100$. Valida.
  - Estensione: Aggiungo `00` alla fine. $\Rightarrow y = 10000$.
  - Verifica y: Finisce con `00`? Sì.
- **Conclusione:** x è prefisso di y.
- **Risposta: NO**.

ⓘ Esercizio 2: DPDA e CFG (Pattern Somma)

Domanda:
Dato il linguaggio $L = \{a^n b^m c^{n+m} \mid n, m \geq 1\}$.
(Logica: le `c` finali sono la somma delle `a` e delle `b` precedenti).

1. Scrivere la **Grammatica (CFG)** che genera L.
2. Costruire il **DPDA**.

💡 Svolgimento Guidato Passo-Passo:

1. Costruzione CFG:

- Dobbiamo generare le `a` che si bilanciano con le ultime `c`, e in mezzo le `b` che si bilanciano con le prime `c` (del blocco c).
- Struttura: $a^n(b^m c^m)c^n$.
- Regola esterna (S): Genera $a \dots c$ e poi passa dentro.

$$S \to aSc \mid aBc$$

(Nota: aBc è il passo base per le `a`: ne metto almeno una e passo alle `b`).
- Regola interna (B): Genera $b \dots c$.

$$B \to bBc \mid bc$$

- **Grammatica Finale:**

$$S \to aSc \mid aBc$$

$$B \to bBc \mid bc$$

2. Costruzione DPDA:

- **Strategia:**
  1. Push `A` per ogni `a`.
  2. Push `B` per ogni `b` (sopra le A).
  3. Con le `c`, prima faccio POP delle `B` (match b/c), poi faccio POP delle `A` (match a/c).
- **Transizioni δ:**
  - q_0 (Leggo `a`): $\delta(q_0, a, Z_0) = \{(q_0, AZ_0)\}$, $\delta(q_0, a, A) = \{(q_0, AA)\}$.
  - $q_0 \to q_1$ (Arriva `b`): $\delta(q_0, b, A) = \{(q_1, BA)\}$.
  - q_1 (Leggo `b`): $\delta(q_1, b, B) = \{(q_1, BB)\}$.
  - $q_1 \to q_2$ (Arriva `c` - fase 1): $\delta(q_1, c, B) = \{(q_2, \epsilon)\}$. *(Inizio a consumare B)*.
  - q_2 (Consumo `B`): $\delta(q_2, c, B) = \{(q_2, \epsilon)\}$.
  - $q_2 \to q_3$ (Finito B, inizio A): $\delta(q_2, c, A) = \{(q_3, \epsilon)\}$. *(Nota: uso lo stesso input `c` ma cambio simbolo stack)*.
  - q_3 (Consumo `A`): $\delta(q_3, c, A) = \{(q_3, \epsilon)\}$.
  - Accept: $\delta(q_3, \epsilon, Z_0) = \{(q_f, Z_0)\}$.

ⓘ Esercizio 3: CFG to NPDA (Palindromo)

Domanda:
Data la grammatica $S \to aSa \mid bSb \mid c$.
Mostrare la trace per la stringa `abcba`.

💡 Svolgimento Guidato Passo-Passo:

1. Logica:
Questa grammatica genera palindromi di lunghezza dispari con centro `c`. La stringa `abcba` è corretta.

2. Trace:
Obiettivo: Svuotare pila e input.

Passo	Stato	Input Residuo	Stack	Azione
1	q	`abcba`	S	Start
2	q	`abcba`	aSa	Espansione $S \to aSa$ (vedo `a` in input)
3	q	`bcba`	Sa	Match `a`
4	q	`bcba`	$bSba$	Espansione $S \to bSb$ (vedo `b` in input)
5	q	`cba`	Sba	Match `b`
6	q	`cba`	cba	Espansione $S \to c$ (centro)
7	q	`ba`	ba	Match `c`
8	q	`a`	a	Match `b`
9	q	ϵ	ϵ	Match `a` $\to$ **OK**

ⓘ Esercizio 4: Macchina di Turing (Accept/Reject Espliciti)

Domanda:
Progettare (descrivere a parole e trace) una MT che accetta il linguaggio $L = \{a^n b^n \mid n \geq 1\}$ usando stati espliciti q_{acc} e q_{rej}.
Trace richiesta per: `aabb`.

💡 **Svolgimento Guidato Passo-Passo:**

1. Logica (Zig-Zag):

- Inizio (q_0): Se vedo `a`, la marco (es. divento `X`), vado a destra in fondo per cercare una `b`.
- Cerca `b` (q_1): Scorro tutto a destra. Se trovo `b`, la marco (`Y`) e torno indietro. Se non trovo `b` (solo a o blank troppo presto) $\to q_{rej}$.
- Ritorno (q_2): Scorro a sinistra fino alla prima `X`, poi passo avanti di uno (q_0).
- Fine: Se in q_0 trovo solo `Y` (tutto marcato), vado in q_{acc}.

2. Trace `aabb`:

- $(q_0, \underline{a}abb)$
- $(q_1, X\underline{a}bb) \to$ scorre a dx fino alla prima b.
- $(q_2, Xa\underline{Y}b) \to$ trovata b, marcata Y, torna indietro a sinistra.
- $(q_0, X\underline{a}Yb) \to$ trova X, va subito a destra sulla a.
- $(q_1, XX\underline{Y}b) \to$ marca seconda a, va a dx cercando b (salta le Y).
- $(q_2, XXY\underline{Y}) \to$ trovata seconda b, marcata, torna indietro.
- $(q_0, XX\underline{Y}Y) \to$ trova X, va a destra. Trova Y.
- Tutto marcato correttamente $\to q_{acc}$.

ⓘ Esercizio 5: Teoria (Ricorsione e Funzioni)

Domanda:

1. Definire il concetto di **Insieme Ricorsivo** collegandolo alla **Funzione Caratteristica**.
2. Quando una funzione $f : \mathbb{N} \to \mathbb{N}$ si dice **Calcolabile**?

💡 **Svolgimento Guidato Passo-Passo:**

1. Insieme Ricorsivo e Funzione Caratteristica:
Un insieme $A \subseteq \mathbb{N}$ (o un linguaggio) è detto **Ricorsivo** se la sua funzione caratteristica $\chi_A(x)$ è calcolabile.
La funzione caratteristica è definita come:

$$\chi_A(x) = \begin{cases} 1 & \text{se } x \in A \\ 0 & \text{se } x \notin A \end{cases}$$

In pratica: esiste un algoritmo (MT) che, dato x, mi restituisce sempre 1 o 0 in tempo finito.

2. Funzione Calcolabile:
Una funzione f è calcolabile (secondo la tesi di Church-Turing) se esiste una Macchina di Turing che, preso in input x sul nastro, termina la computazione lasciando sul nastro il valore $f(x)$. Se la funzione è parziale, la MT potrebbe non terminare per input dove $f(x)$ non è definita.