Gemini 3

🎓 Simulazione Secondo Parziale - PROVA 11

Obiettivo: Consolidare i pattern "nested" (annidati) e la teoria sulla calcolabilità.

ⓘ Esercizio 1: Proprietà di Prefisso (3 Linguaggi)

Domanda:
Stabilire se i seguenti linguaggi godono della proprietà di prefisso (Prefix-Free). Motivare la risposta con controesempi o ragionamenti strutturali.

1. $L_1 = \{a^n b^m \mid 0 < n < m\}$ (Più `b` che `a`).
2. $L_2 = \{a^n b^n c^k \mid n \geq 1, k \geq 0\}$ (Coda di `c` opzionale).
3. $L_3 = \{a^n b^m c^m d^n \mid n, m \geq 1\}$ (Struttura a specchio annidata).

💡 **Svolgimento Guidato:**

1. Analisi L_1 ($n < m$):

- **Logica:** Le `b` devono superare le `a`.
- **Test:**
  - Stringa valida minima ($n = 1, m = 2$): $x = abb$.
  - Posso aggiungere lettere? Se aggiungo una `b` $\Rightarrow y = abbb$.
  - Verifica y: $n = 1, m = 3$. La condizione $1 < 3$ è vera.
- **Conclusione:** x ("abb") è prefisso di y ("abbb").
- **Risposta: NO**.

2. Analisi L_2 ($a^n b^n c^k, k \geq 0$):

- **Logica:** Una base bilanciata $a^n b^n$ seguita da zero o più `c`.
- **Test:**
  - Caso $k = 0$ ($n = 1$): $x = ab$.
  - Estensione con $k = 1$: $y = abc$.
- **Conclusione:** x è prefisso di y. Il vincolo $k \geq 0$ permette di fermarsi subito dopo le `b` o di proseguire, rompendo la proprietà.
- **Risposta: NO**.

3. Analisi L_3 ($a^n b^m c^m d^n$):

- **Logica:** Struttura "a cipolla": $a[b^m c^m]d$. I blocchi esterni a/d racchiudono quelli interni b/c.
- **Test:**
  - Stringa minima ($n = 1, m = 1$): $x = abcd$.
  - Estensione?
    - Aggiungere `d`? No, perché le `d` devono combaciare con le `a` iniziali (che sono fisse nel prefisso).
    - Aggiungere `c` o `b`? No, romperei l'ordine $b \to c \to d$.
- **Conclusione:** Una volta chiusa la parentesi esterna (le `d`), la stringa è sigillata. Non si può estendere.
- **Risposta: SÌ**.

Esercizio 2: DPDA e CFG (Pattern Annidato)

Domanda:
Dato il linguaggio $L = \{a^n b^m c^m d^n \mid n, m \geq 1\}$.

1. Scrivere la **CFG** che genera L.
2. Costruire il **DPDA** che riconosce L.

💡 **Svolgimento Guidato:**

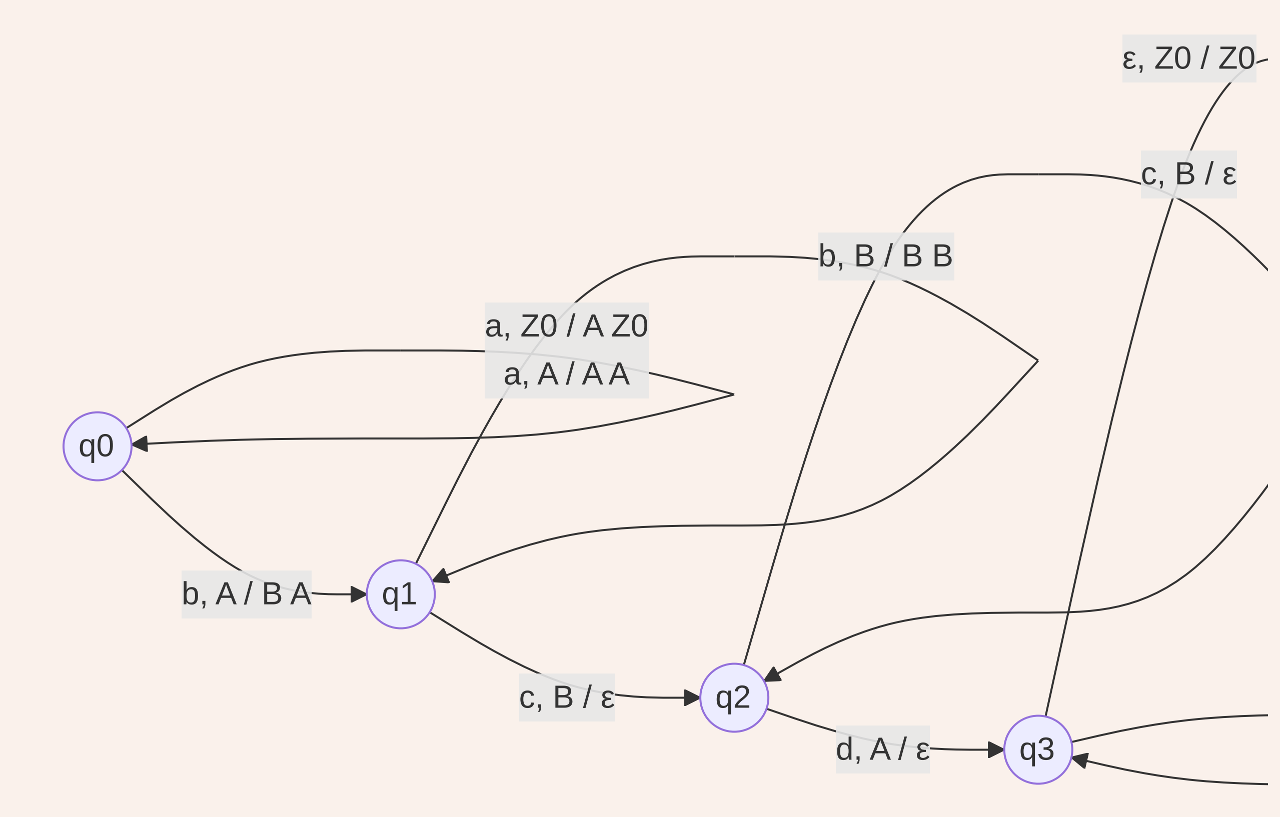
1. Costruzione CFG:

- Il linguaggio ha una dipendenza esterna (a/d) che racchiude una dipendenza interna (b/c).
- $S \to aSd \mid aAd$ (Gestisce $a^n \dots d^n$ e passa al centro).
- $A \to bAc \mid bc$ (Gestisce il centro $b^m c^m$).

2. Costruzione DPDA:

- **Strategia:**
  1. Push `A` per ogni `a`.
  2. Push `B` per ogni `b`.
  3. Pop `B` per ogni `c`.
  4. Pop `A` per ogni `d`.
- **Funzione di Transizione:**
  - q_0 (Leggo `a`): Push `A`.
  - $q_0 \to q_1$ (Leggo `b`): Push `B`.
  - q_1 (Leggo `b`): Push `B`.
  - $q_1 \to q_2$ (Leggo `c`): Pop `B`.
  - q_2 (Leggo `c`): Pop `B`.
  - $q_2 \to q_3$ (Leggo `d`): Pop `A`. *(Qui transito solo se la cima è A, cioè ho finito le B)*.
  - q_3 (Leggo `d`): Pop `A`.
  - Accettazione: $\delta(q_3, \epsilon, Z_0) = \{(q_f, Z_0)\}$.

Diagramma:



ⓘ Esercizio 3: CFG to NPDA (Trace)

Domanda:
Data la grammatica:

$$S \to aS \mid Sb \mid c$$

(Genera stringhe con una 'c' centrale circondata da 'a' a sinistra e 'b' a destra in qualsiasi numero).
Mostrare la trace per la stringa `aacb`.

💡 Svolgimento Guidato:

1. Logica:
Sostituiamo S fino a trovare il terminale `c`.

2. Trace (`aacb`):
| Passo | Stato | Input | Stack | Azione |

| :--- | :--- | :--- | :--- | :--- |
1	q	`aacb`	S	Start
2	q	`aacb`	aS	Espansione $S \to aS$
3	q	`acb`	S	Match `a`
4	q	`acb`	aS	Espansione $S \to aS$
5	q	`cb`	S	Match `a`
6	q	`cb`	Sb	Espansione $S \to Sb$
7	q	`cb`	cb	Espansione $S \to c$
8	q	`b`	b	Match `c`
9	q	ϵ	ϵ	Match `b` $\to$ **OK**

ⓘ Esercizio 4: Macchina di Turing (Esecuzione Trace)

Domanda:
Data la MT definita dalle seguenti transizioni (Stato iniziale q_0):

- $\delta(q_0, a) = (q_0, b, R)$
- $\delta(q_0, b) = (q_0, a, R)$
- $\delta(q_0, B) = (q_1, B, L)$ *(B = Blank)*
- $\delta(q_1, a) = (q_2, a, R)$
- $\delta(q_1, b) = (q_2, b, R)$

Scrivere la sequenza di configurazioni per l'input: **`abba`**.
Nota: Descrivere cosa fa la macchina.

💡 Svolgimento Guidato:

Trace:

1. $(q_0, \underline{a}bba)$
2. $\vdash (q_0, b\underline{b}ba)$ (Letto `a` $\to$ scrive `b`, va a R)

3. $\vdash (q_0, ba\underline{b}a)$ (Letto `b` $\to$ scrive `a`, va a R)
4. $\vdash (q_0, baa\underline{a})$ (Letto `b` $\to$ scrive `a`, va a R)
5. $\vdash (q_0, baab\underline{B})$ (Letto `a` $\to$ scrive `b`, va a R. Trova Blank)
6. $\vdash (q_1, baa\underline{b})$ (Letto `B` $\to$ resta `B`, va a L, stato q_1)
7. $\vdash (q_2, baab\underline{B})$ (Letto `b` $\to$ resta `b`, va a R, stato q_2)
   La macchina si arresta.

Funzione: La macchina inverte tutti i bit (`a` $\leftrightarrow$ `b`) e si ferma alla fine del nastro.

ⓘ Esercizio 5: Teoria (Linguaggio Universale)

Domanda:

1. Dare la definizione del **Linguaggio Universale** L_u.
2. A quale classe di linguaggi appartiene L_u? (Ricorsivo o RE?)
3. Dimostrare perché non appartiene all'altra classe (spiegazione logica basata sulle dispense).

💡 **Svolgimento Guidato (basato su LC-RA-1920_Zetaexe):**

1. Definizione:
L_u è l'insieme delle stringhe binarie che codificano coppie $\langle M, w \rangle$, dove M è una Macchina di Turing e w è un input tale che M **accetta** w.

2. Classificazione:
L_u è un linguaggio **Ricorsivamente Enumerabile (RE)** ma **NON Ricorsivo**.

- È RE perché esiste una MdT (la Macchina Universale U) che può simularlo e accettare se M accetta.

3. Dimostrazione (Non Ricorsivo):
Si dimostra per assurdo.

- Se L_u fosse Ricorsivo, allora anche il suo complemento $\overline{L_u}$ sarebbe Ricorsivo (proprietà dei linguaggi ricorsivi).
- Se $\overline{L_u}$ fosse ricorsivo, potremmo costruire una macchina che decide il **Linguaggio Diagonale (L_d)**.
- Ma sappiamo (dal teorema di diagonalizzazione di Cantor) che L_d **non è nemmeno RE**.
- Quindi l'ipotesi che L_u sia ricorsivo porta a una contraddizione.