2. (9 punti) Dimostra che se L è un linguaggio context-free, allora anche il seguente linguaggio è context-free:

$$\text{insert}_{\#}(L) = \{x\#y \mid xy \in L\}.$$

Parti da G,, CFG del linguaggio L di partenza.

Creiamo una nuova grammatica G' in forma normale di Chomsky per il nostro linguaggio $insert_{\#}$

$$x = aaa, y = bbbbb, x, y \in \{a, b\}^* \rightarrow insert_{\#}(L) = aaa\#bbbbb$$

Struttura (più pratica):

- Regola iniziale $S' = S$
- Produzione
  - $S \rightarrow AB$
  - Abbiamo una nuova regola che per ogni A e B produzione ci permette di aggiungere un cancelletto
  - $A \rightarrow x\ canc\ y$
  - $canc \rightarrow \#$
- Una volta messo il cancelletto, otteniamo solo simboli terminali
  - $A \rightarrow a$ (regola terminale = finisce producendo un carattere senza mettere di nuovo il cancelletto)

Struttura più formale:

- Per ogni variabile V di G, conteniamo sia V che V' che vengono prodotte da G'
- Tutte le stesse regole di G' in G
- $\forall v \in G$, abbiamo delle regole intermedie $V \rightarrow V', V' \rightarrow \epsilon$
- Per ogni regola $V \rightarrow B$ di G, la regola $V' \rightarrow \#B$
- Se incontriamo un cancelletto nella produzione intermedia, la grammatica produrrà $V' \rightarrow \epsilon$ oppure $V \rightarrow v$

$G' = G$ per variabile iniziale e rispetta L.

2. (12 punti) Una variabile A in una grammatica context-free G è *persistente* se compare in ogni derivazione di ogni stringa w in $L(G)$. Data una grammatica context-free G e una variabile A, considera il problema di verificare se A è persistente.

(a) Formula questo problema come un linguaggio $PERSISTENT_{\text{CFG}}$.

(b) Dimostra che $PERSISTENT_{\text{CFG}}$ è decidibile.

a) $PERSISTENT_{CFG} = \{\langle G \rangle \mid G$ è una CFG, $A \in G$, A è persistente$\}$

b) Descrizione di una TM che fa quello che deve fare il problema (1)

Adesso facciamo “una descrizione a livello implementativo” / “ad alto livello”

Es. “per visualizzare il concetto logico di Macchina di Turing”

TM ##->[][][][][][][][]## (dove [] è una cella di memoria, -> è la testina, # è il confine (bound))

[a][w]awawawawawawaww (qua stiamo descrivendo a parole quello che vorremmo, in questo caso quando rifiuta; ad ogni regola, non ho sempre il fatto di "quando metti "a" allora metti "w")

Sul punto in rosso rifiuta.

Se avessimo invece un input:

[a][w]awawawawawawawa

Allora accetterebbe! (Se fosse un DFA, si direbbe "termina uno stato di accettazione/stato finale)

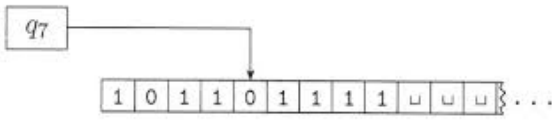


FIGURE **3.4**
A Turing machine with configuration $1011q_701111$

Descrizione regolare:

M è una TM, prende in input $\langle G, A\rangle$, G CFG e $A \in G$

- Verifica che ogni regola di $CFG\ G$ dia in output una produzione che contiene w
- Se M non vede w in una regola $\in L$ la stringa rifiuta
- Continua finché non esaurisce tutte le regole
- Se M accetta allora accetta, altrimenti rifiuta

Dimostriamo che M è un decisore → si comporta sempre in modo deciso

- Se input $\langle G, A\rangle$, allora A è persistente, in quanto la TM M verifica che ogni regola produca correttamente la variabile richiesta per ogni singolo passaggio. Se così non fosse, allora rifiuta altrimenti accetta.
- Se A è persistente, dato l'input $\langle G, A\rangle$, la TM rifiuta dato che ogni regola non produce la stringa richiesta, quindi A non può essere definita persistente.

Volendo per riduzione → vedi gli indecidibili.

$$A_{CFG} \leq_m PERSISTENT_{CFG}$$

Se $PERSISTENT_{CFG}$ è decidibile, allora A_{CFG}.

2. (12 punti) Considera il linguaggio

$$\text{FORTY-TWO} = \{\langle M, w\rangle \mid M \text{ termina la computazione su } w \text{ avendo solo 42 sul nastro}\}.$$

Dimostra che FORTY-TWO è indecidibile.

$$A_{TM} \leq_m Forty - Two$$

A_{TM} = {<M,w> | M is a TM and M accepts w}

Se A_{TM} è indecidibile, allora $Forty - Two$ è indecidibile.

- F è una funzione di riduzione con $\langle M, w\rangle$, M è una TM (attento: che fa A_{TM} seguendo la descrizione di $Forty - Two$) e w una stringa)

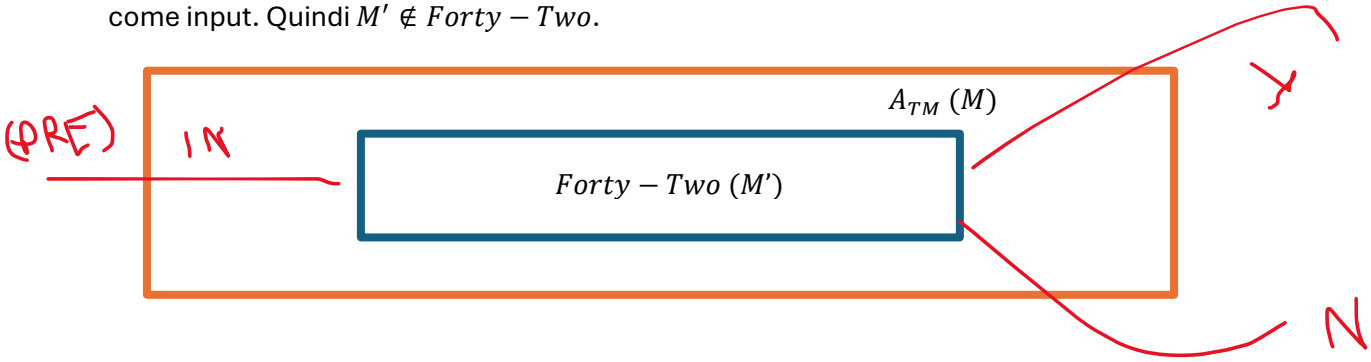
- M copia l'input sul nastro
- Se $w \neq 42$, va in loop o rifiuta
- Se $w = 42$, esegui M'
  - Se M' accetta, allora accetta
  - Altrimenti rifiuta

- Restituisci $\langle M' \rangle$

Mostriamo che F calcola una funzione di riduzione da A_{TM}:

$$\langle M, w \rangle \in A_{TM} \Leftrightarrow M' \in Forty - Two$$

- Se $\langle M, w \rangle \in A_{TM}$, la macchina trova 42 sul nastro e si ferma accettando
- Se $\langle M, w \rangle \notin A_{TM}$, la macchina rifiuta o va in loop in quanto sul nastro non era presente 42 come input. Quindi $M' \notin Forty - Two$.



Complemento di A_TM ≤m F se e solo se M' ∉ F.

In altre parole, il complemento di ATM, indicato come ATM, è l'insieme delle Turing machine M tali che (M, w) appartiene ad ATM se e solo se M' non appartiene a F.

Se sei nel caso negativo (rifiuta/va in loop) parte il complemento di A_TM e se realizzi le condizioni del problema (esempio – Forty-Two) allora rifiuti. In questo caso, vuol dire che per chiusura

Esempio completo:

Sia (M, w) un'istanza di $\overline{A_{TM}}$. Definiamo la TM M' che, su input x:

1. Esegue M su input w.
2. Se M accetta w, accetta.
3. Altrimenti (se M rifiuta o va in loop su w), calcola per 42 passi sul nastro (per vedere se appare 42) e poi si ferma rifiutando.

Mostriamo che (M, w) ∈ $\overline{A_{TM}}$ se e solo se (M', w) ∈ FORTY-TWO:

⇒) Se (M, w) ∈ $\overline{A_{TM}}$, allora M termina la computazione su w. Quindi per costruzione M' simulerà M su w e poi accetterà. Inoltre terminerà avendo usato solo 42 celle del nastro (o meno). Quindi (M', w) ∈ FORTY-TWO.

⇐) Se (M', w) ∈ FORTY-TWO, allora M' termina la computazione su w usando solo 42 celle del nastro. Per come abbiamo definito M', ciò può accadere solo se M termina su w, quindi (M, w) ∈ $\overline{A_{TM}}$.

3. (9 punti) Una *macchina di Turing ad albero binario* usa un albero binario infinito come nastro, dove ogni cella nel nastro ha un figlio sinistro e un figlio destro. Ad ogni transizione, la testina si sposta dalla cella corrente al padre, al figlio sinistro oppure al figlio destro della cella corrente. Pertanto, la funzione di transizione di una tale macchina ha la forma

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{P, L, R\},$$

dove P indica lo spostamento verso il padre, L verso il figlio sinistro e R verso il figlio destro. La stringa di input viene fornita lungo il ramo sinistro dell'albero.

Mostra che qualsiasi macchina di Turing ad albero binario può essere simulata da una macchina di Turing standard.

$TM_{Nastro\ singolo}$ che simula $TM_{Albero\ binario}$

S = TM a nastro singolo che su input w:

- Ricopio l'input sul nastro
- Se $\delta(r, a) = (s, b, P)$, controlliamo dove esiste la prima cella con valore P. Se non esiste nessuna cella con P, la macchina rifiuta (nel caso in cui abbiamo appena iniziato l'input).
    - Se la cella P non è stata marcata, rappresenta un nuovo padre
    - In questo caso, passo al primo puntatore L e scorro tutti i figli sinistri
    - Marco dove ho trovato l'ultimo L
    - Parto a scorrere tutti i figli destri dal primo R
    - Tutti questi saranno scorsi fino al prossimo P, che viene poi marcato
- Se $\delta(r, a) = (s, b, L)$, controlliamo di essere nella prima cella dopo un padre P marcato.
    - Se ciò avviene, comincio a scorrere tutti i simboli a sinistra dell'albero finché non ho finito i simboli che sono a sinistra prima di un prossimo P
    - Una volta finiti questi simboli, uso un marcatore (°) per indicare che ho finito i simboli di sinistra
    - Proseguo con i simboli di destra R
- Se $\delta(r, a) = (s, b, R)$, controlliamo di essere nella prima cella dopo un padre P marcato
    - Se ciò avviene, comincio a scorrere tutti i simboli a destra dell'albero finché non ho finito i simboli che sono a destra prima di un prossimo P
    - Una volta finiti questi simboli, uso un marcatore (*) per indicare che ho finito i simboli di destra
    - Avrò un puntatore ad un padre P (^) che mi dice "torna al nodo padre precedente"
- Trovato un padre precedente, vado al padre successivo e continuo così scorrendo tutti gli input finché esistono dei padri P
- Se M arriva in uno stato di accettazione, allora accetta altrimenti rifiuta
- Altrimenti, riprendi la simulazione di cui sopra (Padre/Sinistro/Destro)