2° PARTE → TM (VARIANTI) → TRANSIZIONE

INDECIDIBILITÀ
NP-HARD
} RIDUZIONE

TM SINGOLO

→ VARIANTE

(VICEVERSA)

F. DI TRANSIZIONE

1. Una *macchina di Turing con reset a sinistra* è una variante delle comuni macchine di Turing, dove la funzione di transizione ha la forma:

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{R, RESET\}.$$

Se $\delta(q, a) = (r, b, RESET)$, quando la macchina si trova nello stato q e legge a, la testina scrive b sul nastro, salta all'estremità sinistra del nastro ed entra nello stato r. Per sapere su quale cella saltare la macchina usa il simbolo speciale $\triangleright$ per identificare l'estremità di sinistra del nastro. Questo simbolo si può trovare solo in una cella del nastro, e non può essere sovrascritto o cancellato. La computazione di una macchina di Turing con reset a sinistra sulla parola w inizia con $\triangleright w$ sul nastro. Si noti che queste macchine non hanno la solità capacità di muovere la testina di una cella a sinistra.

Mostrare che le macchine di Turing con reset a sinistra riconoscono la classe dei linguaggi Turing-riconoscibili.

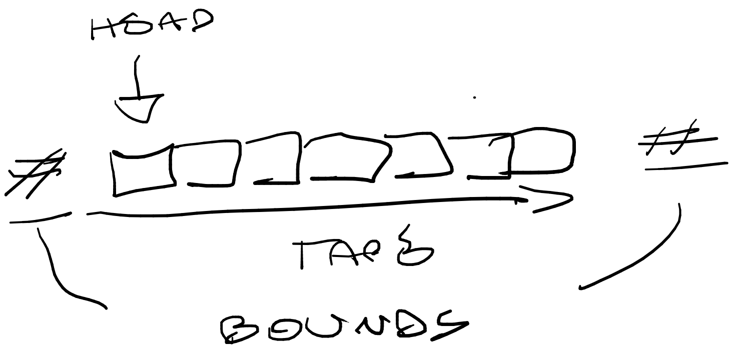
INPUT "a", OUTPUT "b" CON TRANSIZIONE "RESET"

SINGOLO: $\delta : Q \times \underbrace{\Gamma}_{\text{TAPE}} \mapsto Q \times \Gamma \times \{L, R\}$

$\subseteq$

RESET: $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, RESET\}$

IMPLEMENTAZIONE AD ALTO LIVELLO → A SX / A DX ...



TURING-RICONOSCIBILE → "ACCETTA" MA NON DECIDE SEMPRE

TM SINGOLO → TM RESET

SINGOLO: $\delta : Q \times \underset{\text{TAPE}}{\Gamma} \longmapsto Q \times \Gamma \times \{L, R\}$

$\subseteq$

RESET: $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, RESET\}$

$\Longleftrightarrow$

$(\Longrightarrow)$

SINGOLO $\xrightarrow{\text{SIMULA}}$ RESET SX.

TM STANDARD $\subseteq$

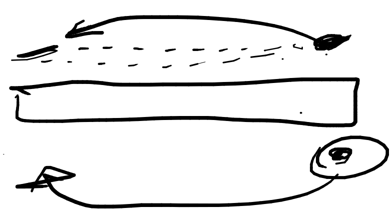
- $\delta(q, a) = (r, b, R)$ → S ESEGUE, MUOVE LA TESTINA A DX
  E SCRIVE b
  SUL NASTRO

- $\delta(q, a) = (r, b, RESET)$ → S ESEGUE,
  SCRIVE b
  SUL NASTRO
  ARRIVA A "RESET"

$(\Longleftarrow)$

RESET $\xrightarrow{\text{SIMULA}}$ SINGOLO

TM NON HA MEMORIA, DI SOLITO PUOI FARE COSÌ [METTI UN SIMBOLO (•) → PALLINO]



S SALTA FINCHÉ NON FINISCE INPUT

$S =$ "Su input w:
1. scrive il simbolo $\triangleright$ subito prima dell'input, in modo che il nastro contenga $\triangleright w$.
2. Se la mossa da simulare è $\delta(q, a) = (r, b, R)$, allora S la esegue direttamente: scrive b sul nastro, muove la testina a destra e va nello stato r. RIGHT
3. Se la mossa da simulare è $\delta(q, a) = (r, b, RESET)$, allora S esegue le seguenti operazioni: scrive b sul nastro, poi muove la testina a sinistra e va nello stato r_{RESET}. La macchina rimane nello stato r_{RESET} e continua a muovere la testina a sinistra finché non trova il simbolo $\triangleright$. A quel punto la macchina sposta la testina un'ultima volta a sinistra, poi di una cella a destra per tornare sopra al simbolo di fine nastro. La computazione riprende dallo stato r.
4. Se non sei nello stato di accettazione o di rifiuto, ripeti da 2."

(⟸) OGNI MACCHINA RICONOSCO RESET A SX

$\delta(q, a) = (r, b, R)$

→ SCRIVI, COPI L'INPUT, E SPOSTI A DX CON LO STATO R

a a a b b e

◁

RESET A SX

$\delta(q, a) = (r, b, L)$

→ SCRIVO "r" SUL NASTRO

▷ → CONTINUO A DX

RESET

$\sqcup$ = BLANK = NULL = INPUT SVUOTATO

→ CONTINUA FINCHÉ NON HO MOSSO A $\sqcup$ OGNI SIMBOLO DI RESET (◁)

▷ a a a a a

$\sqcup$ (abbe)

$\neq$ ▷ ◁ R $\sqcup$

SE Σ ACCETTA → ACCETTA

→ RIFIUTA

L'algoritmo usa un nuovo simbolo $\triangleleft$ per identificare la fine della porzione di nastro usata fino a quel momento, e può marcare le celle del nastro ponendo un punto al di sopra di un simbolo.

$M =$ "Su input w:

1. Scrive il simbolo $\triangleleft$ subito dopo l'input, per marcare la fine della porzione di nastro utilizzata. Il nastro contiene $\triangleright w \triangleleft$.
2. Simula il comportamento di S. Se la mossa da simulare è $\delta(q, a) = (r, b, R)$, allora M la esegue direttamente: scrive b sul nastro, muove la testina a destra e va nello stato r. Se muovendosi a destra la macchina si sposta sulla cella che contiene $\triangleleft$, allora questo significa che S ha spostato la testina sulla parte di nastro vuota non usata in precedenza. Quindi M scrive un simbolo blank marcato su questa cella, sposta $\triangleleft$ di una cella a destra, e fa un reset a sinistra. Dopo il reset si muove a destra fino al blank marcato, e prosegue con la simulazione mossa successiva.
3. Se la mossa da simulare è $\delta(q, a) = (r, b, L)$, allora S esegue le seguenti operazioni:
   3.1 scrive b sul nastro, marcandolo con un punto, poi fa un reset a sinistra
   3.2 Se il simbolo subito dopo $\triangleright$ è già marcato, allora vuol dire che S ha spostato la testina sulla parte vuota di sinistra del nastro. Quindi M scrive un blank e sposta il contenuto del nastro di una cella a destra finché non trova il simbolo di fine nastro $\triangleleft$. Fa un reset a sinistra e prosegue con la simulazione della prossima mossa dal nuovo blank posto subito dopo l'inizio del nastro. Se il simbolo subito dopo $\triangleright$ non è marcato, lo marca, resetta a sinistra e prosegue con i passi successivi.

   3.3 Si muove a destra fino al primo simbolo marcato, e poi a destra di nuovo.
   3.4 se la cella in cui si trova è marcata, allora è la cella da cui è partita la simulazione. Toglie la marcatura e resetta. Si muove a destra finché non trova una cella marcata. Questa cella è quella immediatamente precedente la cella di partenza, e la simulazione della mossa è terminata
   3.5 se la cella in cui si trova non è marcata, la marca, resetta, si muove a destra finché non trova una marcatura, cancella la marcatura e riprende da 3.3.
4. Se non sei nello stato di accettazione o di rifiuto, ripeti da 2."

TEMPLATE/
ES.
NOTEVOLE...

1. *Una* macchina di Turing bidimensionale *utilizza una griglia bidimensionale infinita di celle come nastro. Ad ogni transizione, la testina può spostarsi dalla cella corrente ad una qualsiasi delle quattro celle adiacenti. La funzione di transizione di tale macchina ha la forma*

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{\uparrow, \downarrow, \rightarrow, \leftarrow\},$$

dove le frecce indicano in quale direzione si muove la testina dopo aver scritto il simbolo sulla cella corrente.

Dimostra che ogni macchina di Turing bidimensionale può essere simulata da una macchina di Turing deterministica a nastro singolo.

$S =$ "su input w:

1. Sostituisce w con la configurazione iniziale $\#\#w\#\#$ e marca con $\hat{}$ il primo simbolo di w.
   Q0 QF → STATI!
2. Scorre il nastro finché non trova la cella marcata con $\hat{}$.
3. Aggiorna il nastro in accordo con la funzione di transizione di B:

DATO INPUT "Q", VAI A SCRIVERE "b" SUL NASTRO A SX →

   - Se $\delta(r, a) = (s, b, \rightarrow)$, scrive b non marcato sulla cella corrente, sposta $\hat{}$ sulla cella immediatamente a destra. Poi sposta di una cella a destra tutte le marcature con un pallino. Se in qualsiasi momento S sposta una marcatura sopra un $\#$, S scrive un blank marcato al posto del $\#$ e sposta il contenuto del nastro da questa cella fino al $\#\#$ finale, di una cella più a destra.
   - Se $\delta(r, a) = (s, b, \leftarrow)$, scrive b non marcato sulla cella corrente, sposta $\hat{}$ sulla cella immediatamente a sinistra. Poi sposta di una cella a sinistra tutte le marcature con un pallino. Se in qualsiasi momento S sposta una marcatura sopra un $\#$, S scrive un blank marcato al posto del $\#$ e sposta il contenuto del nastro da questa cella fino al $\#\#$ iniziale, di una cella più a sinistra.
   - Se $\delta(r, a) = (s, b, \uparrow)$, scrive b marcato con un pallino nella cella corrente, e sposta $\hat{}$ sulla prima cella marcata con un pallino posta a sinistra della cella corrente. Se questa cella marcata non esiste, aggiunge una nuova riga composta solo da blank all'inizio della configurazione.
   - Se $\delta(r, a) = (s, b, \downarrow)$, scrive b marcato con un pallino nella cella corrente, e sposta $\hat{}$ sulla prima cella marcata con un pallino posta a destra della cella corrente. Se questa cella non esiste, aggiunge una nuova riga composta solo da blank alla fine della configurazione.
4. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di B, allora *accetta*; se la simulazione raggiunge lo stato di rifiuto di B allora *rifiuta*; altrimenti prosegue con la simulazione dal punto 2."

DECIDIBILE ⟷ REGOLARE

[ALGORITMO DECIDE "abba"] - "abba"

} STOP IN T. FINITO

INDECIDIBILE ⟷ NON REGOLARE
=
PUMPING LEMMA

↓ TROVO UNA "FALLA" / NON LO RISOLVO SEMPRE ...

0110

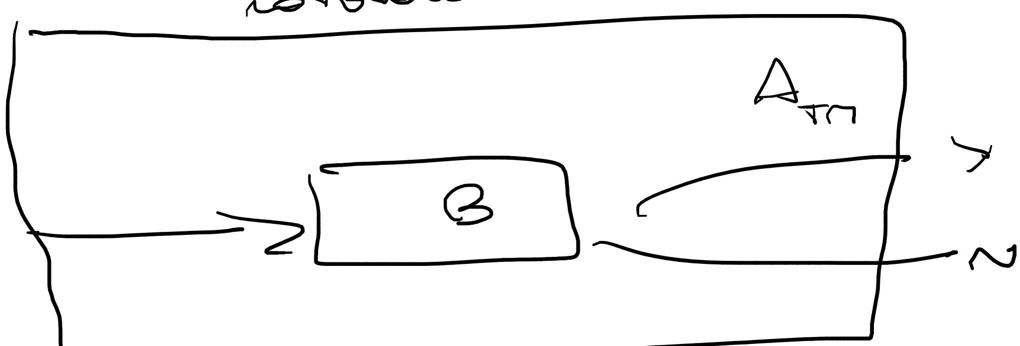
011100 → HO >#1

DECIDIBILITÀ → ∃ UN ALGORITMO CHE TERMINA (N. FINITO PASSI)

INDECIDIBILITÀ → DIFFICILE PROVARE CHE UN ALGORITMO SIA "IMPOSSIBILE"

RIDUZIONE $(A \leq_m B)$ $\leq_m$ = MAPPING

PROBL. NOTEVOLE   TUO PROBLEMA



$A_{TM} = \{ A$ è una TM che riconosce $w \}$ INPUT

$A \le_m B \rightarrow$ A è indecidibile

$\Downarrow$

B è indecidibile

$$A \begin{cases} A_{TM} \rightarrow \text{riconosce } w \\ (\overline{A_{TM}}) \rightarrow \text{riconosce } \neq w \\ HALT_{TM} \end{cases}$$

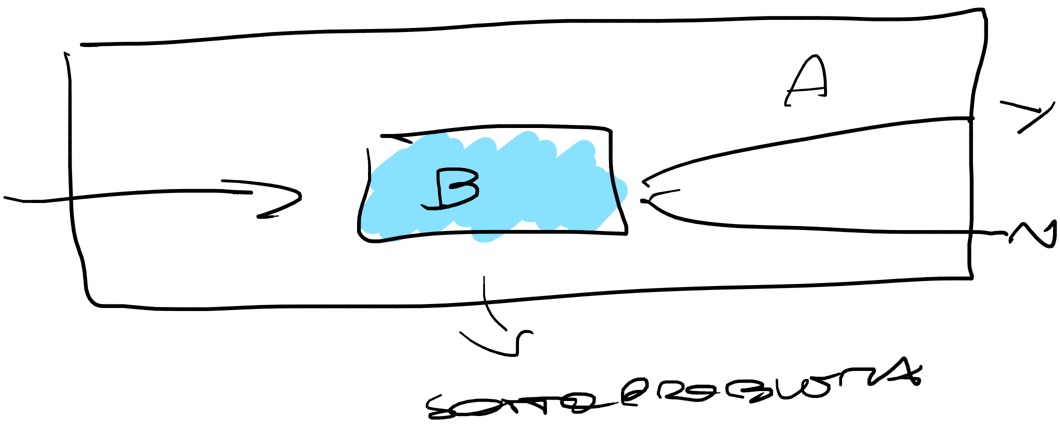
indecidibili

Halting problem

A è una TM si ferma su "w"

$$\begin{cases} A_{TM} \le_m \text{ problema} \\ HALT_{TM} \le_m \text{ problema} \end{cases}$$

NOTA: $A \le_m B \rightarrow$ se B è decidibile, A lo è.



sottoproblema

TM → SE ∃ "xy zzy" SUL NASTRO

(a) FORMULA COME L $\langle M, w \rangle$

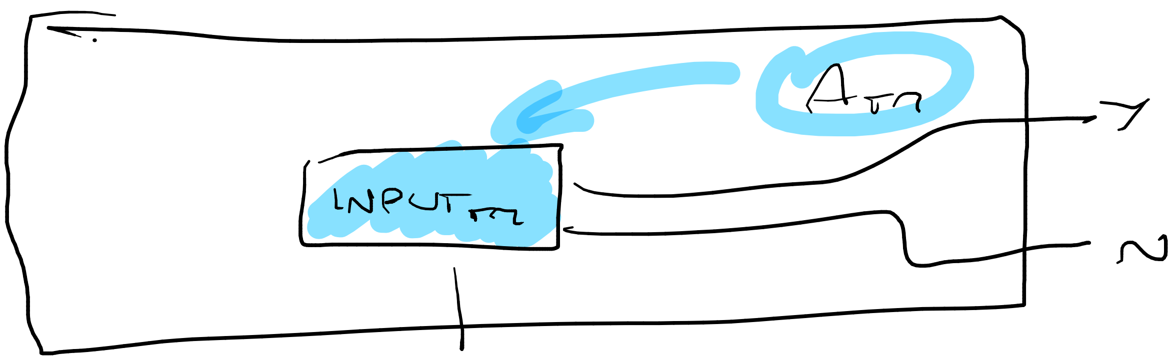
$INPUT_{TM}$ = { M è una TM che riconosce "xy zzy" sul nastro con "w" input }

(b) DIMOSTRA $INPUT_{TM}$ SIA INDECIDIBILE

PROBLEMA INDECIDIBILE $\leq_m$ $INPUT_{TM}$

A_{TM} → RICONOSCI w COME INPUT

$A_{TM} \leq_m INPUT_{TM}$



SUBPROBLEMA = ESERCIZIO

3. **(12 punti)** Data una Turing Machine M, considera il problema di determinare se esiste un input tale che M scrive "$xyzzy$" su cinque celle adiacenti del nastro. Puoi assumere che l'alfabeto di input di M non contenga i simboli x, y, z.

(a) Formula questo problema come un linguaggio $MAGIC_{TM}$.

(b) Dimostra che il linguaggio $MAGIC_{TM}$ è indecidibile.

Soluzione.

(a) $MAGIC_{TM} = \{\langle M\rangle \mid M$ è una TM che scrive $xyzzy$ sul nastro per qualche input $w\}$

(b) La seguente macchina F calcola una riduzione $\boxed{A_{TM} \leq_m MAGIC_{TM}}$:

F. DI RIDUZIONE

F = "su input $\langle M, w\rangle$, dove M è una TM e w una stringa:

1. Verifica che i simboli x, y, z non compaiano in w, né nell'alfabeto di input o nell'alfabeto del nastro di M. Se vi compaiono, sostituiscili con tre nuovi simboli X, Y, Z nella parola w e nella codifica di M.
2. Costruisci la seguente macchina M':

   M' = "Su input x:

   ATM ACCETTA

   1. Simula l'esecuzione di M su input w, senza usare i simboli x, y, z.
   2. Se M accetta, scrivi $xyzzy$ sul nastro, altrimenti rifiuta senza modificare il nastro.

   MAGIC ESISTE

3. Ritorna $\langle M'\rangle$."

Mostriamo che F calcola una funzione di riduzione da A_{TM} a $MAGIC_{TM}$, cioè una funzione tale che

SOTTOPROBLEMA

$$\langle M, w\rangle \in A_{TM} \text{ se e solo se } \langle M'\rangle \in MAGIC_{TM}.$$

- Se $\langle M, w\rangle \in A_{TM}$ allora la macchina M accetta w. In questo caso la macchina M' scrive $xyzzy$ sul nastro per tutti gli input. Di conseguenza $\langle M'\rangle \in MAGIC_{TM}$.
- Viceversa, se $\langle M, w\rangle \notin A_{TM}$ allora la macchina M rifiuta o va in loop su w. Per tutti gli input, la macchina M' simula l'esecuzione di M su w senza usare i simboli x, y, z (perché sono stati tolti dalla definizione di M e di w se vi comparivano), e rifiuta o va in loop senza scrivere mai $xyzzy$ sul nastro. Di conseguenza $\langle M'\rangle \notin MAGIC_{TM}$.

[ATM ACCETTA SSE ∃ "xyzzy"]

[SSE ATM INDEC. → MAGIC INDECIDIBILE]

$\boxed{P = NP}$

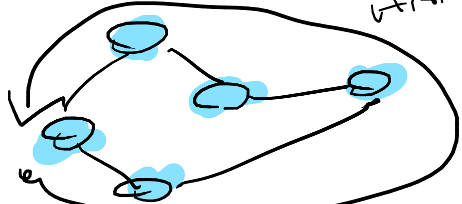
RISOLVO IN T. POLY

NON-POLYNOMIAL

RICONOSCO IN T. FINITO MA RISOLVO?

GRAFO = $G = \langle V, E\rangle$

CIRCUITO HAM.



RICONOSCO MA NON A RISOLVERLO...

NP-HARD

DIMOSTRAZIONI → REGOLARE
→ CONTEXT FREE
→ $\boxed{\text{NON REGOLARE}}$

PUMPING-LEMMA → $\underline{w \neq \varepsilon}$ → xy^iz
$i \geq 0$

0^{k-p+q} $\underline{xy^2z}$ / xy^0z 0^{k-p-q}

2. (12 punti) Considera il linguaggio

$$L_2 = \{1^{2n^2+3} \mid n \geq 0\}$$

Dimostra che L_2 non è regolare.

$\underline{1^3} = 111$ $(n=0)$
$\underline{1^5} = 11111$ $(n=1)$
1^{11} ......

ASSUMENDO L REG.

$w = xyz$ → $y \neq \varepsilon$, $i \geq 0$ → $\underline{|xy| \leq k}$

xy^iz

$0^p 0^q k$ ....

$x = \varepsilon$
→ $y = 1^3$ $(i \geq 0)$
$z = 1^{2n^2+6}$

$i = 2 \Rightarrow xy^2z = 1^6 1^{2n^2} = 1^{2n^2+6} \geq \underline{(2n^2+3)}$

$\# 1 \Rightarrow z \geq y$

$1^{2n^2+3} \rightarrow 1^{3} \rightarrow x$

$\rightarrow 1^{n^2} \rightarrow p > 0 \Big] \rightarrow y$

$\rightarrow 1^{n^2}$

$z = 1^{k-p}$

$xy^0z \rightarrow 1^3 \, 1^{k-p}$

$= 1^{k-p+3} \rightarrow$ ROSSA $\Big] \rightarrow |xy| \leq k$

GENERALE

---

1. (12 punti) Dati due linguaggi $L, M \subseteq \Sigma^*$, definiamo il linguaggio

$$L \triangleleft M = \{w \in L \mid \text{esiste } v \in M \text{ tale che } |v| = |w|\}.$$

STESSO NUMERO DI ELEMENTI

Dimostra che se L ed M sono linguaggi regolari allora anche $L \triangleleft M$ è regolare.

DFA / NFA / RE

DFA $\rightarrow L$

DFA $\rightarrow M$

UGUALE

$DFA_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$

$DFA_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$

NFA

$\downarrow$

ε-TRANSIZIONI

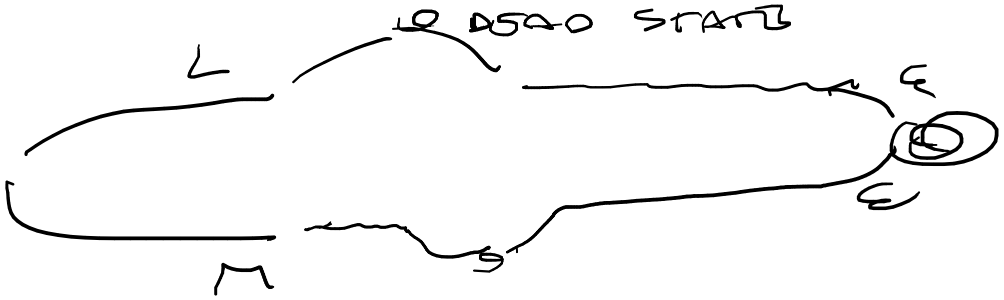
$L \triangleleft M \rightarrow Q = Q_L \times Q_M$

$\delta_L(r, a) = \delta_L(r_{i-1} \ldots)$

$\delta_M(r, a) = \delta_M(r_{i-1} \ldots)$

CONCATENAZIONE —> AVANZANDO SEGUENDO LA COMPUT.

SE ∃ UN SIMBOLO NON APPARTENENTE ALL'ALTRO —> STATO POZZO



— $|L| = |M|$

-> STESSO STATO FINALE -> $Q_{F_L} = Q_{F_M}$

-> $F = \{F_L \times F_M\}$

1. *Dimostra che se L ed M sono linguaggi regolari sull'alfabeto $\{0,1\}$, allora anche il seguente linguaggio è regolare:*

$$L \sqcap M = \{x \sqcap y \mid x \in L, y \in M \text{ e } |x| = |y|\},$$

REGOLARI ...

dove $x \sqcap y$ rappresenta l'and bit a bit di x e y. Per esempio, $0011 \sqcap 0101 = 0001$.

Poiché L e M sono regolari, sappiamo che esiste un DFA $A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$ che riconosce L e un DFA $A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ che riconosce M.

Costruiamo un NFA A che riconosce il linguaggio $L \sqcap M$:

- L'insieme degli stati è $Q = Q_L \times Q_M$, che contiene tutte le coppie composte da uno stato di A_L e uno stato di A_M.
- L'alfabeto è lo stesso di A_L e di A_M, $\Sigma = \{0,1\}$.
- La funzione di transizione δ è definita come segue:

$$\delta((r_L, r_M), 0) = \{(\delta_L(r_L, 0), \delta_M(r_M, 0)), (\delta_L(r_L, 1), \delta_M(r_M, 0)), (\delta_L(r_L, 0), \delta_M(r_M, 1))\}$$
$$\delta((r_L, r_M), 1) = \{(\delta_L(r_L, 1), \delta_M(r_M, 1))\}$$

—> COMBO INPUT ...

  La funzione di transizione implementa le regole dell'and tra due bit: l'and di due 1 è 1, mentre è 0 se entrambi i bit sono 0 o se un bit è 0 e l'altro è 1.
- Lo stato iniziale è (q_L, q_M).
- Gli stati finali sono $F = F_L \times F_M$, ossia tutte le coppie di stati finali dei due automi.

---

3. (12 punti) Dato un linguaggio $L \subseteq \Sigma^*$, definiamo il linguaggio

$$delete\#(L) = \{xy \mid x\#y \in L\}.$$

GRAMMATICA FINISCE ?

Dimostra che la classe dei linguaggi context free è chiusa per l'operazione *delete#*.

CFG —> CHOMSKY -> $|2n-1|$

$G = (V, \Sigma, R, S) \longrightarrow$ CHOMSKY

$G' = (V', \Sigma', R', S')$

$\begin{bmatrix} A \to BC \\ A \to a \end{bmatrix}$

$\exists\ G'$ IN CHOMSKY

$\boxed{G = (V, \Sigma, R, S)}$

$\to S' = S$ (STESSO INIZIO)

$\to \Sigma' = \Sigma$

$\to V' = V$

$S \to A$ SKIP $\cancel{\#}$

$\to R' = R \cup \longrightarrow A \to \# \;/\; A \to \varepsilon$

$\{A \to BC\}$

$(\Longleftrightarrow)$

$W \longrightarrow G \Longrightarrow$ TOGLIO $\#$

G'

CHEATSHEET - RECAP

$A \to BC$ $\quad B \to b$ $\quad C \to c$

$\left(\frac{\underline{A \to BC}}{A \to E}\right)$

① DIMOSTRA CF → PREFIX

PREFISSO

FORMALE [$G = (\Sigma, V, R, S)$

$\underline{S \to S}$
$A \to BC$
$B \to b$
$C \to c$

② DIMOSTRA REGOLARE

DFA
NFA

TRANSIZIONI

$A = (Q, \Sigma, \underline{\delta}, q_0, F)$

- L. SINGOLO
- OPERAZIONI

③ PL = PUMPING LEMMA

ENUNCIATO
PUMPING UP
PUMPING DOWN

2° PARTE

① TM → VARIANTI - IMPLEMENTAZIONE AD ALTO LIVELLO

TM SINGOLO → VARIANTE

(⟸⟹)

δ

② INDECIDIBILITÀ / NP-HARD → RIDUZIONI

$(A \leq_m B)$

decidibilità

indecidibilità

_ _ _ _ _