Linguaggi di Programmazione e Modelli Computazionali
Esame del XX YYYYYYYY KKKK

Esercizio 1:
Progettare un automa a stati finiti deterministico che riconosce il linguaggio delle stringhe binarie che contengono un numero pari di 0 oppure che interpretate come numero in base 2 rappresentano un numero multiplo di 3.

Esercizio 2:
Definire una grammatica libera dal contesto che genera il linguaggio delle stringhe binarie con un ugual numero di ***0*** ed ***1***.

Esercizio 3:
Progettare un automa a pila che riconosce il linguaggio sull'alfabeto binario ***{0,1}*** in cui il numero di occorrenze di ***0*** è il doppio del numero di occorrenze di ***1***.

Esercizio 4:
Si consideri il linguaggio delle stringhe di parentesi tonde bilanciate del tipo ***ww*** (cioè costituite dalla concatenazione di due stringhe identiche). Classificare il linguaggio dicendo se è un linguaggio regolare, libero, ricorsivo, ricorsivamente enumerabile, o nemmeno ricorsivamente enumerabile. Giustificare la risposta.

Esercizio 5:
Si consideri il seguente linguaggio sull'alfabeto ***{0,1,2}***:
L = { x2y | x** e **y** sono numeri binari tali che **L(M)

$L = \{\, x2y \mid x \text{ e } y \text{ sono numeri binari tali che } L(M_x) \text{ e } L(M_y) \text{ hanno una stringa in comune} \,\}$
dove M_x e M_y sono la x-esima e la y-esima macchina di Turing secondo una qualche enumerazione delle macchine di Turing. Dire se L è ricorsivo, ricorsivamente enumerabile, o nemmeno ricorsivamente enumerabile. Giustificare la risposta.

Esercizio 6:
Descrivere come si può ottenere un DFA da un ε-NFA.

Esercizio 7:
Riportare enunciato e dimostrazione del Pumping Lemma per linguaggi liberi.

Esercizio 8:
Si consideri la seguente grammatica:

If → ***if cond then*** Block Else	Block → ***stm*** | ε
Else → ***else*** ElseBlock | ε	ElseBlock → If | Block

Dire se si tratta di grammatica di tipo LL(1) e nel caso presentare la corrispondente tabella di parsing.

Esercizio 1

Risolvibile "simbolicamente"
Costruisco i due DFA e poi faccio una simulazione in parallelo (costruzione di unione linguaggi) che fa "or" degli stati finali.

Stati nella forma (x,y) dove:

x \in {0,1}   (stati DFA n. pari di 0)
y \in {0,1,2} (stati DFA multipli di 3)

Formalmente:

stati: Q=Q_1 x Q_2 (prodotto cartesiano)

stato iniziale: (q_1,q_2) dove q_1, q_2 sono gli stati iniziali dei due automi

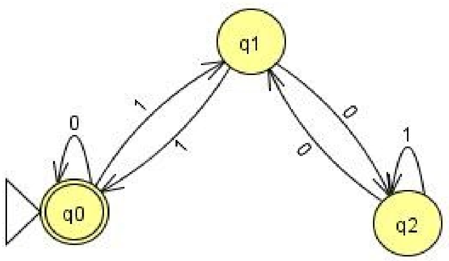
stati finali: F={(q_1,q_2) | q_1 \in F_1 oppure q_2 \in F_2}

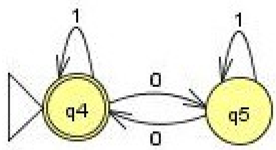
transizioni: \delta((x,y),a)=(\delta_1(x,a),\delta_2(y,a))

--
cioè tupla DFA simulazione = (Q_1 x Q_2, \Sigma, \delta, (q_1,q_2), (F_1 x Q_2) \cup (Q_1 x F_2))
dove:
\delta((x,y),a)=(\delta_1(x,a),\delta_2(y,a))

----

modo alternativo: aggiungo uno stato nuovo iniziale, connetto a tale stato con transizioni \epsilon gli ex stati iniziali dei due automi (simile a costruzione di "+" in trasf da RE a NFA) e poi trasformo in deterministico





Esercizio 2

S → 0AS | 1BS | epsilon
A → 1 | 0AA
B → 0 | 1BB

Esercizio 3

numero 0 doppio di numero 1

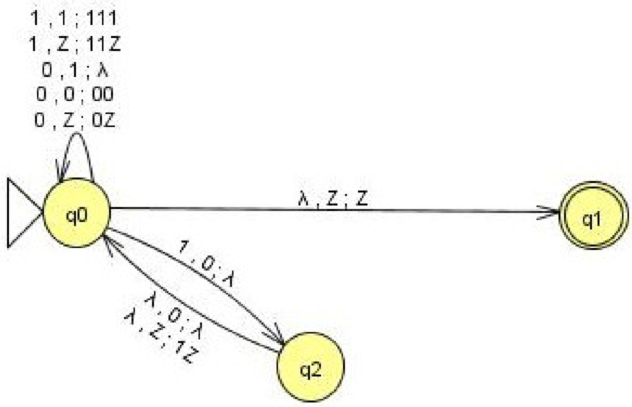
idea:

ECCEDENZA DI 1 (per compensare devo produrre 0 quanto il contenuto della pila)

ogni 1 impilo due 1
ogni 0 spilo un 1

ECCEDENZA DI 0 (per compensare devo produrre 1 quanto la meta' della pila)

ogni 0 impilo uno 0
ogni 1 spilo due 0 (se ce n'e' uno solo indico una eccedenza di 1)



Esercizio 4

L e' ricorsivo

Considero TM DETERMINISTICA con multipli nastri che opera come segue
(inizialmente l'input "x" si trova sul primo nastro):
1) individua (es. tramite una traccia di marcatura) la posizione a metà della stringa:
   se non è di lunghezza pari si blocca
2) copia metà della stringa in un secondo nastro
3) confronta le due stringhe: se non sono uguali si blocca
4) esegue l'algoritmo CYK per la grammatica libera G sotto sul secondo nastro

S->(S)|\epsilon|SS

L non e' libero

Data n costante del pumping lemma dei ling liberi considero

(^n)^n (^n)^n

|vwx| <=n allora vwx comprende simboli soltanto di uno o, al piu', due gruppi adiacenti dei 4 (di cui un gruppo è di "(" ed un gruppo è di ")").
Quindi se vx include "(" essi appartengono tutti ad uno stesso gruppo di "(" ed, analogamente, se vx

include ")" essi appartengono tutti ad uno stesso gruppo di ")".
Siccome |vx|>0, pompando v ed x 2 volte, c'e' almeno un gruppo di "(" o un gruppo di ")" che assume lunghezza >n, mentre l'altro gruppo dei 4 dello stesso tipo ("(" o ")") rimane invariato (di lunghezza n).

Quindi uvvwxxy non appartiene a L: assurdo.

Esercizio 5

L e' ricorsivamente enumerabile

Considero TM NON-DET M con multipli nastri che opera come segue
(inizialmente l'input "w" si trova sul primo nastro):
1) se w non è nella forma "x2y" con x e y non nulle mi blocco
2) decodifico x -> M_x ed y -> M_y
3) creo non-deterministicamente una stringa qualsiasi w' in un nastro non ancora usato
4) copio w' in modo da averla in due nastri
5) uso i due nastri per simulare in parallelo M_x con input w' ed M_y con input w'
   (come per le costruzioni di chiusura rispetto a unione e intersezione)
5) se entrambe le simulazioni hanno raggiunto l'accettazione allora accetto

L non e' ricorsivo

Mi chiedo se ci sia un linguaggio che so non essere ricorsivo che si riduce a questo
(cioè ipotizzando per assurdo che ci sia un algoritmo per questo allora otterrei un algoritmo anche per il linguaggio noto)

In questo caso basta considerare L_u!
L_u= {w_k | w_z \in L(M_x) e decodifica di k è (x,z)}

La riduzione da L_u ad L è la seguente.
Per stabilire se w_k sta in L_u considero (x,z) decodifica di k e pongo:
y=indice di una qualsiasi TM tale che L(M_y)={w_z}
e poi uso un ipotetico algoritmo per stabilire se "x2y" sta in L

Ho che:
x2y \in L se e solo se L(M_x) e L(M_y)={w_z} hanno una stringa in comune, cioè w_z \in L(M_x)

Quindi ipotizzando di avere un algoritmo per L avrei un algoritmo per L_u: assurdo!

Un altro modo per fare la dimostrazione di non ricorsività è considerare il linguaggio
L'= {w_x | L(M_x) diverso dal vuoto}
(per il teorema di Rice è non ricorsivo, basta prendere L_P con P tutti i linguaggi RE meno il vuoto)

La riduzione da L' ad L è la seguente.
Per stabilire se w_x sta in L' considero:
y=indice di una qualsiasi TM tale che L(M_y) = tutte le stringhe
e poi uso un ipotetico algoritmo per stabilire se "x2y" sta in L

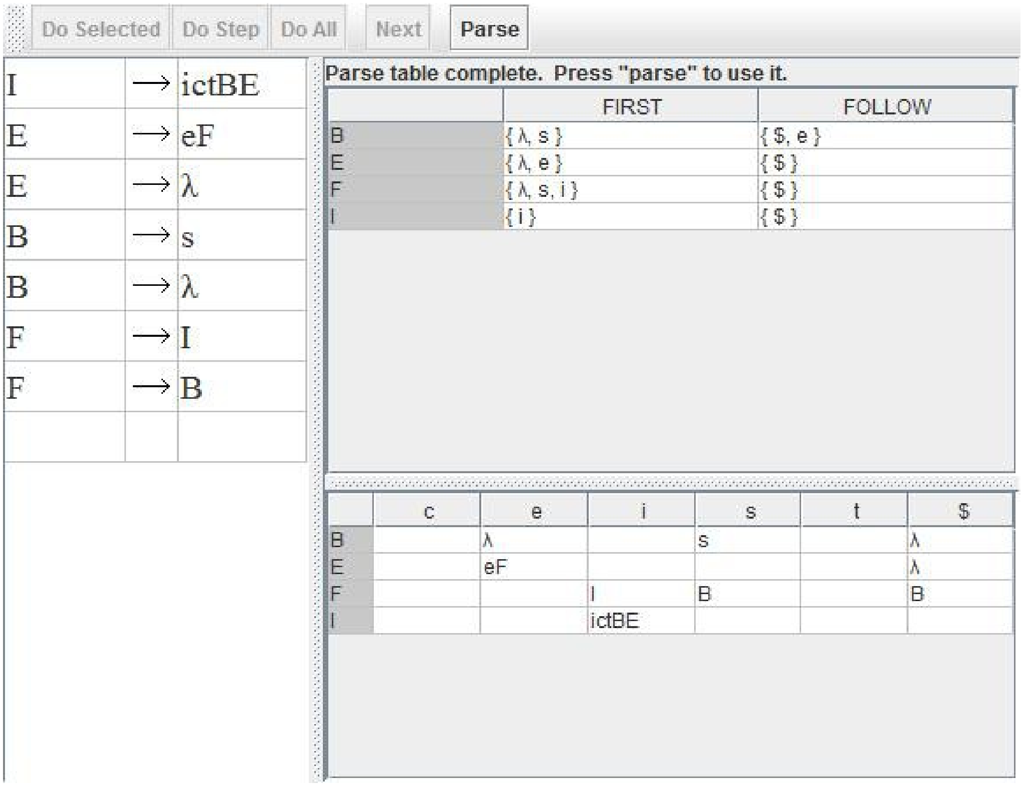
Ho che:
x2y \in L se e solo se L(M_x) e L(M_y) hanno una stringa in comune, cioè L(M_x) diverso dal vuoto

Quindi ipotizzando di avere un algoritmo per L avrei un algoritmo per L': assurdo!

Esercizio 8

la soluzione va la da' JFLAP facendogli costruire la tabella LL(1) su es8.jff
(in JFLAP simboli terminali e non terminali devono essere rappresentati da un unico carattere):
siccome non vi sono caselle della tabella con più di una entry allora è LL(1)

I $\rightarrow$ ictBE
E $\rightarrow$ eF | epsilon
B $\rightarrow$ s | epsilon
F $\rightarrow$ I | B



I	$\rightarrow$	ictBE
E	$\rightarrow$	eF
E	$\rightarrow$	λ
B	$\rightarrow$	s
B	$\rightarrow$	λ
F	$\rightarrow$	I
F	$\rightarrow$	B

	FIRST	FOLLOW
B	{ λ, s }	{ $, e }
E	{ λ, e }	{ $ }
F	{ λ, s, i }	{ $ }
I	{ i }	{ $ }

	c	e	i	s	t	$
B		λ		s		λ
E		eF				λ
F			I	B		B
I			ictBE			