Espressioni regolari:

Identità tra espressioni regolari:

NON valide:

$(r+s)^* = r^* + s^*$

$(rs+r)^*rs = (rr^*s)^*$

$s(rs+s)^*r = rr^*s(rr^*s)^*$

Valide:

$\varepsilon^* = \emptyset^*$

$r^*r^* = rr^* + \varepsilon$

$(rs)^*r = r(sr)^*$

$(s^*r)^*s^* = (r^*s)^*r^*$

$(\varepsilon + \emptyset)^* = \emptyset^*$

$(\varepsilon + r^*r) = r^*$

$(r+s)^* = (r^*s)^*r^*$

$(r^* + s^*)^* = (r^*s^*)^*$

Quale delle seguenti espressioni regolari **non denota** il linguaggio L su $\Sigma = \{0,1\}$ delle stringhe in cui ogni occorrenza di 00 precede tutte le occorrenze di 11?

Tutte NON denotano:

$(10+0)^*(1+10)^*$

$(10+0)^*(1+\epsilon)(01+1)^*0$

$(10+0)^*(1+\epsilon)(01+1)^*(0+\epsilon)$

Quale delle seguenti espressioni regolari **denota** il linguaggio su $\{0,1\}$ delle stringhe che contengono un numero di '0' divisibile per 3 ?

Denota:

$(1^*01^*01^*01^*)^* + 1^*$

NON denota:

$(1^*01^*01^*0)^* + 1^*$

$((0+1)^*0(0+1)^*0(0+1)^*0(0+1)^*)^* + 1^*$

$(1^*01^*01^*01^*)^*$

Si considerino le espressioni regolari su $\{0,1\}$

$r_1 = (0+1)^*(0011+1010)(0+1)^*$

$r_2 = \epsilon + (0+10+110)^*(\epsilon + 1 + 11)$

Tutte NON valide:

$[\![r_1]\!] \supset [\![r_2]\!]$

$[\![r_1]\!] = [\![r_2]\!]$

$[\![r_1]\!] \cap [\![r_2]\!] = \emptyset$

$[\![r_1]\!] \subset [\![r_2]\!]$

Quale delle seguenti espressioni regolari su $\Sigma = \{a,b,c\}$ **denota** il linguaggio $\{\epsilon\} \cup \{w \in \Sigma^* \mid \text{il numero di occorrenze di } a \text{ in } w \text{ è } \textbf{pari e positivo}\}$?

Denota:

$((b+c)^*a(b+c)^*a(b+c)^*)^*$

NON denota:

$((b+c)^*a(b+c)^*a)^*$

$(a(b+c)^*a(b+c)^*)^*$

$\left((b^*c^*)a(b^*c^*)a(b^*c^*)\right)^*$
$\left((b^* + c^*)a(b^* + c^*)a(b^* + c^*)\right)^*$

Quale delle seguenti espressioni regolari su $\Sigma = \{a, b, c\}$ **denota** il linguaggio $\{w \in \{a, b, c\}^* \mid$ il numero di occorrenze di a in w è **dispari**$\}$?

Denota:

$((b + c)^*a(b + c)^*a)^*((b + c)^*a(b + c)^*)$

NON denota:

$((b + c)^*a(b + c)^*)((b + c)^*a(b + c)^*a)^*$
$((b + c)^*a(b + c)^*a(b + c)^*)^*(a(b + c)^*)$
$((b + c)^*a)((b + c)^*a(b + c)^*a(b + c)^*)^*$
$(a(b + c)^*)((b + c)^*a(b + c)^*a(b + c)^*)^*$

Cardinalità:

Cardinalità dell'insieme di *

Linguaggi: $|\wp(\mathbb{N})|$
Linguaggi non acontestuali: $|\wp(\mathbb{N})|$
Linguaggi acontestuali: $|\mathbb{N}|$
Linguaggi regolari: $|\mathbb{N}|$
su alfabeto di $n > 0$ simboli

Stringhe lunghe n su un alfabeto Σ: $|\Sigma|^n$
Macchine di Turing: $|\mathbb{N}|$

Qual è la cardinalità delle *

Funzioni parziali o totali: $|\wp(\mathbb{N})|$
Funzioni ricorsive totali e parziali: $|\mathbb{N}|$
Tutte le altre funzioni: $|\mathbb{N}|$

Insiemi ricorsivamente enumerabili:

Gli **insiemi ricorsivamente enumerabili** non sono chiusi rispetto a:

Chiusi rispetto:

differenza
completamento

NON chiusi rispetto:

rimozione di un elemento
unione
intersezione

Linguaggio su alfabeto:

Quali dei seguenti linguaggi sull'alfabeto * sono regolari?

$\boldsymbol{\Sigma = \{a, b\}}$

Regolare:

$\{a^n a^{(n+1)^2 - n^2} \in \Sigma^* \mid n \geq 0\}$

NON è regolare:

$\{a^n b^n \in \Sigma^* \mid n \geq 1\}$
$\{x \in \Sigma^* \mid x \text{ ha più a che b}\}$
$\{x \in \Sigma^* \mid x \text{ ha tante a quante b}\}$

$\boldsymbol{\Sigma = \{0, 1, 2\}}$

NON è regolare:

$\{0^{2^{n+1}} \in \Sigma^* \mid n \geq 1\}$

$\{0^n \in \Sigma^* \mid n \geq 1 \text{ è primo}\}$

$\{0^m 1^n \mathbf{1}^{n+m} \in \Sigma^* \mid n \geq 1, m \geq 1\}$

$\{0^m 1^n \mathbf{2}^{n+m} \in \Sigma^* \mid n \geq 1, m \geq 1\}$

Complemento – Sottoinsieme – Differenza – Alfabeto:

Il complemento di un linguaggio **finito / regolare**

è regolare

NON è:

finito

irregolare

acontestuale non regolare

Un sottoinsieme di un linguaggio **regolare / acontestuale**

è non decidibile

NON è:

monotono

acontestuale

regolare

decidibile

La differenza insiemistica di due linguaggi regolari?

è regolare

NON è:

è finita

è acontestuale non regolare

è monotona non acontestuale

Quante sono le sottostringhe di una stringa di lunghezza n su di un alfabeto di $m > 0$ simboli?

sono $1 + n(n+1)/2$

NON sono:

mn

$n(n+1)/2$

$m(m+1)/2$

$1 + m(n+1)/2$

Chiusura:

La chiusura di Kleene di un linguaggio *

finito

è regolare

NON è:

finita

monotona acontestuale

acontestuale non regolare

monotono

è infinita

NON è:

regolare

acontestuale

monotona non acontestuale

acontestuale

è infinita

NON è:

regolare

monotona non acontestuale

è acontestuale

Grammatica

Si considerino le seguenti grammatiche espressive in forma concisa e si dica quale di queste è ambigua o se nessuna lo è:

Ambigua:

$S \rightarrow SS|a$

$S \rightarrow SaS|\epsilon$

NON ambigua:

$S \rightarrow aS|a$

$S \rightarrow aSa|\epsilon$

Si consideri la più semplice grammatica libera contenente le produzioni $S \rightarrow Sb$, $S \rightarrow aSb$, $S \rightarrow abb$. Si dica quale linguaggio viene generato da tale grammatica.

Generato:

$\{a^n b^m \mid 0 < n < m\}$

NON generato:

$\{a^n b^m \mid 0 \leq n \leq m\}$

$\{a^n b^m \mid 0 < n \leq m\}$

$\{a^n b^m \mid 0 \leq n < m\}$

Altri

L'affermazione "Se $I \subseteq \mathbb{N}$ è un insieme X e $\bar{I} = \mathbb{N} \backslash I$ allora anche $\bar{I}$ è X" è vera, se al posto di X scrivo:

'ricorsivo'

NON scrivo:

'ricorsivamente enumerabile'

'non ricorsivamente enumerabile'

'ricorsivamente enumerabile non ricorsivo'

Quale combinazione delle variabili intere x, y, z fa si che la seguente espressione C++ valga 4 ?

$(x > y) \,?\, \big((x > z) \,?\, x : z\big) : \big((y > z) \,?\, y : z\big)$

$x = 3\,, y = 4\,, z = 2$

NON queste:

$x = 5\,, y = 5\,, z = 4$

$x = 4\,, y = 5\,, z = 5$

$x = 5\,, y = 4\,, z = 5$

Quali tra i seguenti sono problemi decidibili?

1. Se l'intersezione di due linguaggi regolari è infinita. (decidibile)
2. Se una data grammatica è ambigua. (semidecidibile)
3. Se due APND accettano lo stesso linguaggio. (non decidibile)
4. Se una data grammatica è acontestuale. (decidibile)

(5. Se l'insieme di stringhe accettate da un DFA è vuoto/infinito. (decidibile))

(6. Se dati due DFA, accettano il medesimo linguaggio. (decidibile))

Decidibili:

1 e 4

NON decidibili:

2 e 3

2 e 4

1 e 2

Automi: DFA, MdT APD, APND

- Quale dei seguenti automi può dar luogo a sequenze infinite di transizioni?
- Quali dei seguenti automi può accettare $\{x \in \{0,1\}^* \mid n_0(x) = n_1(x)\}$ con $n_b(\varepsilon) = 0$ e $n_b(aw) = 1 - |a - b| + n_b(w)$?

APND

NON:

NFA

DFA

$\varepsilon -$NFA

Quali dei seguenti automi può accettare $\{x \in \{0,1\}^* \mid x = r(x)\}$ con $r(\varepsilon) = \varepsilon$ e $r(aw) = r(w) \cdot a$?

APND

NON può accettare:

APD

DFA

$\varepsilon -$NFA

Quali dei seguenti automi si arresta sempre dopo aver effettuato un numero finito di transizioni se riceve in input una sequenza finita di simboli (non black per la MdT)?

DFA

NON si arresta sempre:

MdT

APD

APND

Quali delle seguenti coppie hanno diverso potere (peso) espressivo?

APD e APND (APND > APN)

NON hanno diverso potere espressivo:

DFA e NFA

ER ordinarie ed ER senza ε

MdT ordinarie e MdT con più nastri

Si dica quanti stati ha un DFA minimo che accetta il linguaggio sull'alfabeto {a, b} denotato dall'espressione regolare $\big(\underbrace{(a+b)(a+b)\dots(a+b)}_{n}\big)^*$:

n

NON ha:

$2n$

$n+1$

$n+2$

Si dica quanti stati ha un DFA minimo che accetta il linguaggio sull'alfabeto {a,b,c} denotato dall'espressione regolare $\varepsilon + \underbrace{(a+b)(a+b)\dots(a+b)}_{n}$:

$n+2$

NON ha:

n

$2n$

$n+1$

Si consideri l'automa a pila

$M = \langle \{q\}, \{a, b\}, \{a, b, S\}, \delta, q, S, \emptyset \rangle$

dove

$\delta(q, \epsilon, S) = \{(q, bSa), (q, bS), (q, SS), (q, \epsilon)\}$

$\delta(q, a, a) = \{(q, \epsilon)\}$

$\delta(q, b, b) = \{(q, \epsilon)\}$

si dica quale delle seguenti stringhe non è accettata per pila vuota:

NON accettata:

ba

Accettata:

ϵ

babb

bbaa

Si consideri un APND a uno stato q con alfabeto dello stack {a,b,S}, simbolo iniziale dello stack S e la funzione di transizione definita da

$\delta(q, \epsilon, S) = \{(q, Sb), (q, aSb), (q, abb)\}$

$\delta(q, a, a) = \{(q, \epsilon)\}$

$\delta(q, b, b) = \{(q, \epsilon)\}$

Si dica quale linguaggio viene generato da tale automa per pila vuota:

Viene generato:

$\{a^n b^m \mid 0 < n < m\}$

NON viene generato:

$\{a^n b^m \mid 0 \leq n < m\}$

$\{a^n b^m \;\; 0 < n \leq m\}$

$\{a^n b^m \;\; 0 \leq n \leq m\}$

Scriviamo $dfa(x)$ e $apnd(y)$ a significare che x è un DFA e y un APND; scriviamo $x \equiv y$ per dire che x e y sono equivalenti. Quali delle seguenti formule logiche rappresenta il fatto che, dato comunque un DFA, esiste un APND equivalente?

$\forall x : dfa(x) \Rightarrow (\exists y \,.\, apnd(y) \wedge x \equiv y)$

NON rappresenta ciò:

$\forall x : \exists y \,.\, (dfa(\mathbf{x}) \wedge apnd(\mathbf{y}) \wedge x \equiv y)$

$\forall x : \exists y \,.\, (dfa(\mathbf{y}) \wedge apnd(\mathbf{x}) \wedge x \equiv y)$

$\neg\forall x : (\exists y \,.\, dfa(x) \Rightarrow apnd(y) \wedge x \equiv y)$

Identificare le eventuali affermazioni vere tra le seguenti, che riguardano l'uso delle MdT come riconoscitori di linguaggi formali:

Vere:

una MdT che muova la testina solo a destra è tanto potente quanto un $\varepsilon-$NFA

False:

più di una delle altre

una MdT è più potente di $\varepsilon-$NFA perché il controllo della MdT non è a stati finiti

un $\varepsilon-$NFA++ che può riavvolgere il nastro (di input) è tanto potente quanto una MdT

Si consideri la MdT definita dal seguente schema:

Q	0	1	$
q_0			q_1 $ R
q_1	q_2 1 L	q_1 0 R	
q_2		q_2 1 L	

Si supponga che la MdT cominci la computazione nello stato q_0, avendo per input sul nastro la stringa "111010", con la testina posizionata sul primo simbolo $ alla sinistra della stringa stessa. Allora la computazione suddetta termina:

dopo 5 passi

NON termina dopo:

dopo 6 passi

dopo 3 passi

dopo 4 passi

Compilatore, Interprete, Programma

Se $C^{L_2}_{L_0,L_1}$ è un compilatore da L_0 a L_1 scritto in L_2 e $I^{L_2}_{L_0}$ è un interprete per L_0 scritto in L_2, si dica quale delle seguenti espressioni meglio la più comune implementazione del linguaggio Java (L_j). Con L_c si indica il linguaggio C.

$\lambda P.D, [\![I^{L_n}_{L_m}]\!] \left([\![C^{L_j}_{L_j,L_m}]\!] (P), D \right)$

NON:

$\lambda P.D, [\![I^{L_n}_{L_m}]\!] \left(I^{L_m}_{L_j}, (P, D) \right)$

$\lambda P.D, [\![I^{L_n}_{L_m}]\!] \left(I^{L_m}_{L_n}, \left([\![C^{L_j}_{L_j,L_?}]\!] (P), D \right) \right)$

$\lambda P.D, [\![I^{L_n}_{L_m}]\!] \left([\![C^{L_c}_{L_c,L_m}]\!] \left([\![C^{L_j}_{L_j,L_c}]\!] (P) \right), D \right)$

Se $C^{L_2}_{L_0,L_1}$ è un compilatore da L_0 a L_1 scritto in L_2 e $I^{L_2}_{L_0}$ è un interprete per L_0 scritto in L_2, si dica quale delle seguenti espressioni meglio la più comune implementazione del linguaggio C (L_c). Con L_j si indica il linguaggio Java.

$\lambda P.D, [\![I^{L_n}_{L_m}]\!] \left([\![C^{L_c}_{L_c,L_m}]\!] (P), D \right)$

NON:

$\lambda P.D, [\![I^{L_n}_{L_m}]\!] \left(I^{L_m}_{L_c}, (P, D) \right)$

$\lambda P.D, [\![I^{L_n}_{L_m}]\!] \left(I^{L_m}_{L_n}, \left([\![C^{L_c}_{L_c,L_?}]\!] (P), D \right) \right)$

$\lambda P.D, [\![I^{L_n}_{L_m}]\!] \left([\![C^{L_j}_{L_j,L_m}]\!] \left([\![C^{L_c}_{L_c,L_j}]\!] (P) \right), D \right)$

Se $C = C^{L}_{L,L}$ è un compilatore da L a L scritto in L, $I = I^{L}_{L}$ è un interprete per L scritto in L, $P = P^{L}$ è un programma scritto in L, allora delle seguenti asserzioni è falsa:

Falsa:

$\lambda D.[\![C]\!](P, D) = [\![P]\!]$

Vera:

$[\![P]\!] = [\![[\![C]\!](P)]\!]$

$\lambda D.[\![[\![C]\!](I)]\!](P, D) = [\![P]\!]$

$[\![[\![C]\!](P)]\!] = [\![[\![[\![C]\!](C)]\!](P)]\!]$

Se $C^{L_2}_{L_0,L_1}$ è un compilatore da L_0 a L_1 scritto in L_2 e scriviamo $L_x \prec L_y$ a significare "L_x è più semplice di L_y", allora vale:

NON vale:

$L_1 \prec L_0$

$L_0 \prec L_2$

$L_0 \prec L_1$

$L_1 \prec L_2$

Teoria Varia:

Funzioni:

Cardinalità: $|\wp(\mathbb{N})|$

Funzioni parziali (tutte le funzioni) $\supset$ Totali $\supset$

Cardinalità: $|\mathbb{N}|$

$\supset$ Calcolabili (parziali ricorsive) $\supset$ Ricorsive (R) $\supset$ Primitive ricorsive

Linguaggi:

Cardinalità: $|\wp(\mathbb{N})|$

$\Sigma^* \supset$

Cardinalità: $|\mathbb{N}|$

$\supset$ Linguaggi a struttura di fase (Tipo 0) $\supset$

$\supset$ Linguaggi dipendenti dal contesto (Tipo 1 – Linguaggi ricorsivi) $\supset$

$\supset$ Linguaggi acontestuali $\supset$ Linguaggi regolari $\supset$ Linguaggi finiti.

Linguaggi finiti

Unione
Intersezione
Concatenazione
Differenza

Linguaggi regolari (lineari destra e sinistra)

Unione
Intersezione
Concatenazione
Differenza
Complemento
Stella di Kleene

Linguaggi acontestuali

Unione
Concatenazione
Stella di Kleene

Linguaggi dipendenti dal contesto (Tipo 1, monotoni, ricorsivi)

Unione
Intersezione
Concatenazione
Stella di Kleene

Linguaggi a struttura di frase (Tipo 0, R.E.)

Unione
Intersezione
Concatenazione
Stella di Kleene

Linguaggi regolari:

$L = \{a^n a^m a^{n+m} \mid n \geq 3, m \geq 4\}$

$L = \{a^n b^m c^n \mid n^2 + m^2 \leq 10\ m\}$

$L = \{a^n b^m c^n \mid 1 \leq n \leq 9, m \geq 2n + 1\}$

Linguaggi non regolari:

$L = \{a^n b^m b^n \mid n \geq 1, m \geq 1\}$

$L = \{a^n b^m c^n \mid n \geq 1, m = 5\}$

Vere:

$\forall$ NFA esiste la possibilità di convertirlo in APD.
Il completamento acontestuale è ricorsivo.
$\exists$ DFA minimo $\forall$ linguaggio regolare.
Con un while senza assegnamento, la terminazione è decidibile.
Il while in programmazione, hanno memoria illimitata.
Il while è turing equivalente.
Il while senza skip è turing equivalente.

False:

Ogni APDN può essere convertito in APD equivalente.
E' possibile configurare un while semplice con più successi.