Macchina Turing Universale

Una limitazione delle macchine di Turing

Turing Machines sono "hardwired"

Eseguono un solo programma

Computer Reali sono ri-programmabili

Soluzione: Universal Turing Machine

Attributi:

- macchina Riprogrammabile
- Simula ogni altra Macchina di Turing

Universal Turing Machine

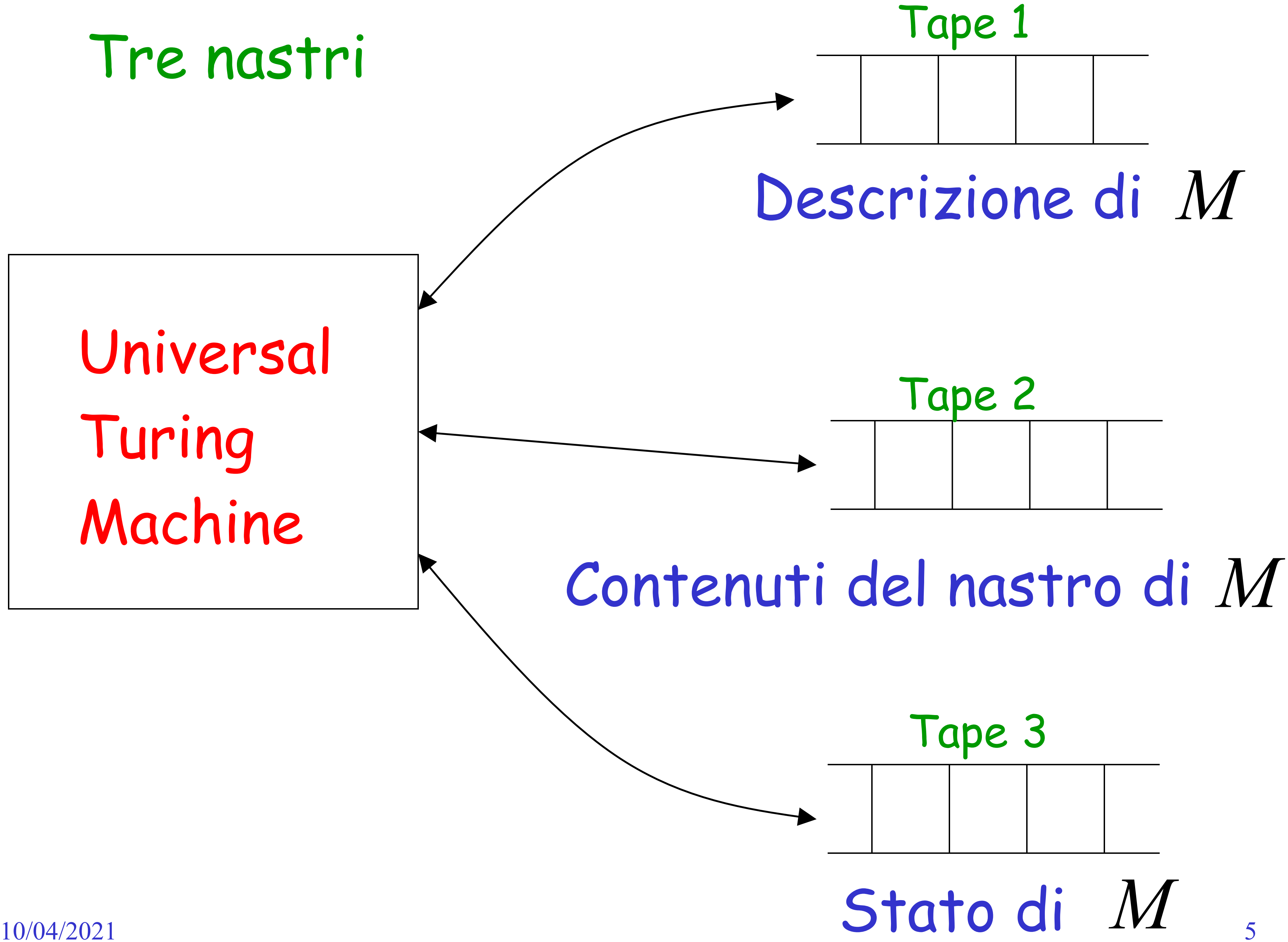
- Simula ogni altra Macchina di Turing M

Input della Universal Turing Machine:

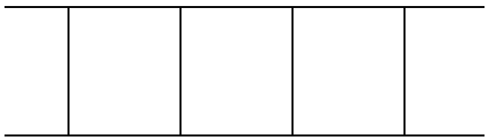
Descrizione delle transizioni di M

stringa di input di M

Tre nastri


Universal
Turing
Machine
Tape 1
Descrizione di M
Tape 2
Contenuti del nastro di M
Tape 3
Stato di M

Tape 1



Descrizione di M

Descriviamo le Turing machines M
Come una stringa di simboli:

codifichiamo M Come una
stringa di simboli

codice Alfabeto

Simboli:	a	b	c	d	$\cdots$
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
codifica:	1	11	111	1111	

Codifica degli stati

Stati:	q_1	q_2	q_3	q_4	$\cdots$
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
codifica:	1	11	111	1111	

Codifica dei movimenti della Head

Mossa:	L	R
	$\downarrow$	$\downarrow$
codifica:	1	11

codifica delle transizione

Transizione: $\delta(q_1, a) = (q_2, b, L)$

codifica: $1\,0\,1\,0\,1\,1\,0\,1\,1\,0\,1$

separatore

Codifica Turing Machine

Transizione:

$$\delta(q_1, a) = (q_2, b, L) \qquad \delta(q_2, b) = (q_3, c, R)$$

Codifica:

$$1010110110 1\ 00\ 1101101110111011$$

separatore

Tape 1 della Universal Turing Machine contiene:

Codifica binaria
della macchina da simulare M

Tape 1

1 0 1 0 11 0 11 0 10011 0 1 10 111 0 111 0 1100…

Nastro due, tre : simuliamo

(q_1,c, q_new, c_new, op)

Consideriamo tutti gli stati e tutti i caratteri (perché? Si può fare diversamente?)

Una lista
{(q,c, q_new, c_new, op) Per tutti gli stati q, Per tutti i caratteri c
}
Al primo posto

q_0 per tutti i caratteri
q_1 per tutti i caratteri
----
q_n per tutti i caratteri.

In rosso i codici

Primo nastro gli input: Macchina $ input
Copiamo Macchina secondo nastro ,
Copiamo input terzo nastro

Carattere osservato terzo nastro,
Stato da applicare secondo nastro
I=0
puntatore primo nastro su q_I
puntatore secondo nastro su carattere osservato C
QUI Guardo C e prendo la stringa che comincia co q_I,C
Copio nel terzo nastro (q_I,C,q_new,c_new,op)
/* eseguire l'operazione*/
C diventa c-new nel secondo nastro
Eseguo op sul secondo nastro che sposta la testina
/*Ora cosa devo fare? Da q_I devo passare a q-new */
I=new
Andare su q_I nel primo nastro
/*Come farlo? Devo spostare, sul primo nastro fino a che non trovo q_new*/
Andare Qui se q_I non è finale

Carattere osservato terzo nastro,
Stato da applicare secondo nastro
/ puntatore secondo nastro su q_0, puntatore terzo nastro primo carattere C */*

I=0
c=primo carattere
Label:
il valore di I ci dà lo stato q_I
il valore del carattere osservato, c, ci dà (q_I, c)
Eseguiamo l'operazione (q_I, c, q_n, nuovoc, op);
Ovvero I=n; c=nuovoc
c=carattere a op dell'osservato op=L,R
Spostiamo il puntatore del terzo nastro secondo op (L,R)
Spostiamo il puntatore del secondo nastro su q_n;
Vai a Label

Una Turing Machine è descritta
Come una stringa di 0's e 1's

quindi:

L'insieme delle Turing machines
Forma un linguaggio:

Ogni stringa di questo linguaggio è
la Codifica binaria di una Turing Machine

Linguaggio delle Turing Machines

L = { 010100101011, (Turing Machine 1)

00100100101111, (Turing Machine 2)

111010011110010101, ……

…… }

Insiemi contabili

Countable sets

Insieme infiniti sono:

Countable (enumerabili)

or

Uncountable (non enumerabili)

Countable set:

Esiste una corrispondenza uno a uno

tra

Gli elementi dell'insieme

e

Numeri naturali (interi positivi)

(ogni elemento dell'insieme è associato ad un numero naturale tale che non esistono due elementi che sono associati allo stesso numero

Esempio: L'insieme dei numeri pari

Intereri pari: $0,\ 2,\ 4,\ 6,\ \ldots$

(positive)

corrispondenza: $\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

Interi positivi: $1,\ 2,\ 3,\ 4,\ \ldots$

$2n$ Corrisponde a $n+1$

Esempio: L'insieme dei numeri razionali

Numeri razionali: $$\frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$$

approccio banale

Nominatore 1

Numeri razionali: $\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \ldots$

Corrispondenza: $\downarrow \quad \downarrow \quad \downarrow$

interi positivi: $1,\ 2,\ 3,\ \ldots$

Non funziona:

Non analizeremo mai

Numeri con nominatore 2: $\frac{2}{1}, \frac{2}{2}, \frac{2}{3}, \ldots$

Approccio migliore

P_1 $\frac{1}{1}$ $\frac{1}{2}$ $\frac{1}{3}$ $\frac{1}{4}$ $\cdots$

P_2 $\frac{2}{1}$ $\frac{2}{2}$ $\frac{2}{3}$ $\cdots$

$\frac{3}{1}$ $\frac{3}{2}$ $\cdots$

$\frac{4}{1}$ $\cdots$

$$
\begin{array}{lllllll}
P\_1(1) & \frac{1}{1} & \longrightarrow & \frac{1}{2} \; P\_1(2) & \frac{1}{3} \; P\_1(3) & \frac{1}{4} & \cdots \\
P\_2(1) & \frac{2}{1} & & \frac{2}{2} \; P\_2(2) & \frac{2}{3} & \cdots & \\
P\_3(1) & \frac{3}{1} & & \frac{3}{2} & \cdots & & \\
 & \frac{4}{1} & & \cdots & & &
\end{array}
$$

$$
\begin{array}{cccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\
\frac{3}{1} & \frac{3}{2} & \cdots & & \\
\frac{4}{1} & \cdots & & &
\end{array}
$$

Oppure?

$$
\begin{array}{ccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\
\frac{3}{1} & \frac{3}{2} & \cdots & & \\
\frac{4}{1} & \cdots & & &
\end{array}
$$

P_1(1)
P_1(2) P_2(1)
P_1(3) P_2(2) P_3(1)
P_1(4) P_2(3) P_3(2) P_4(1)

I=1
J=1,I
P_I(J)

Numeri razionali: $\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{2}{2}, \ldots$

Corrispondenza:

interi positivi: $1, \; 2, \; 3, \; 4, \; 5, \; \ldots$

un insieme è countable se esiste

un enumeration procedure

(enumeratore)

che definisce la corrispondenza con i numeri naturali

Definizione

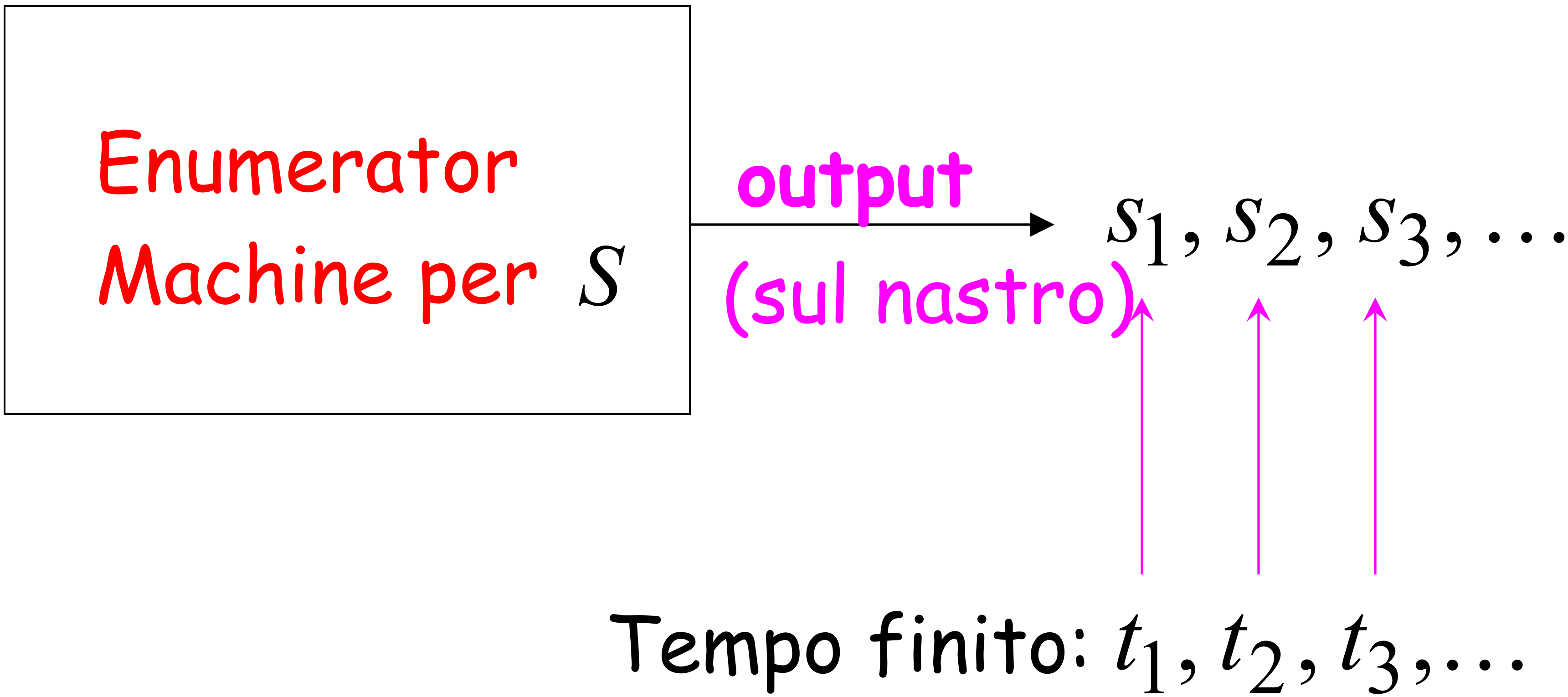
Sia S un insieme di stringhe (linguaggio)

Un **enumerator** per S è una Turing Machine che genera (scrive sul nastro) tutte le stringhe S una per una

e

Ogni stringa è generata in tempo finito

stringhe $s_1, s_2, s_3, \ldots \in S$



Osservazione:

Se esiste per S un enumeratore,
Allora l'insieme è countable

L' enumeratore descrive la corrispondenza
di S con i numeri naturali

Esempio: L'insieme delle stringhe è countable $S = \{a,b,c\}^{+}$

Approccio:

Descriviamo un enumeratore per S

Enumeratore banale:

Produrre le stringhe in ordine lessicografico

$$
\begin{aligned}
s_1 &= a \\
s_2 &= aa \\
\vdots \quad & aaa \\
& aaaa \\
& \ldots\ldots
\end{aligned}
$$

No buono:

le stringhe con primo carattere b
non vengono fuori

Procedura migliore : **Ordine proprio** (Canonical, proper, Order)

1. Produci tutte le stringhe di lunghezza 1
2. Produci tutte le stringhe di lunghezza 2
3. Produci tutte le stringhe di lunghezza 3
4. Produci tutte le stringhe di lunghezza 4

……….

Produce stringhe in

Proper Order:

$$\left.\begin{array}{l} s_1 = a \\ s_2 = b \\ \vdots \quad\; c \end{array}\right\} \text{lunghezza 1}$$

$$\left.\begin{array}{l} aa \\ ab \\ ac \\ ba \\ bb \\ bc \\ ca \\ cb \\ cc \end{array}\right\} \text{lunghezza 2}$$

$$\left.\begin{array}{l} aaa \\ aab \\ aac \\ \ldots\ldots \end{array}\right\} \text{lunghezza 3}$$

Codifica degli stati

Stati:	q_1	q_2	q_3	q_4	$\cdots$
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
codifica:	1	11	111	1111	

Codifica dei movimenti della Head

Mossa:	L	R
	$\downarrow$	$\downarrow$
codifica:	1	11

codifica delle transizione

Transizione: $\delta(q_1, a) = (q_2, b, L)$

codifica: $1\,0\,1\,0\,11\,0\,11\,0\,1$

separatore

Teorema: L'insieme di tutte le Turing Machines è countable

Proof: Ogni macchina di Turing può essere codificata

Con una stringa binaria di 0's e 1's

Definite un enumeration procedure

Per l'insieme delle stringhe che

Descrivono le Turing Machine

Enumerator:

Repeat

1. Genera le stringhe binarie di 0's e 1's in proper order

2. Check se la stringa generate descrive una Turing Machine

   if **YES:** print string sull'output tape

   if **NO:** ignora string

Binary strings | Turing Machines

0

1

00

01

⋮

1 0 1 0 11 0 11 0 0

1 0 1 0 11 0 11 0 1 $\xrightarrow{s_1}$ 1 0 1 0 11 0 11 0 1

⋮

1 0 11 0 1010010101101 $\xrightarrow{s_2}$ 1 0 11 0 1010010101101

⋮

End of Proof

Uncountable Sets

definizione: Un insieme è uncountable L'

se non è countable, ovvero

se non esiste un enumeratore

che lo enumera

Vogliamo provare che vi è un linguaggio che non è accettato da nessuna macchina di Turing

Tecnica:

1_ Turing machines sono countable

2_ Linguaggi sono uncountable

(Vi sono più linguaggi che Turing Machines)

Teorema:

Se S è un infinito enumerabile, allora l'insieme delle parti 2^S di S non è enumerabile.

Proof:

Poichè S è enumerabile, possiamo scrivere

$$S = \{s_1, s_2, s_3, \ldots\}$$

Elementi di S

Gli elementi dell'insieme delle parti 2^S hanno la forma:

$$\emptyset$$

$$\{s_1, s_3\}$$

$$\{s_5, s_7, s_9, s_{10}\}$$

......

Assurdo: Sia l'insieme delle parti 2^S enumerabile

Codifichiamo ogni insieme

con una stringa binaria di 0 e 1.

Elementi Insieme delle parti (in ordine arbitrario)	Codifica Binaria				
	s_1	s_2	s_3	s_4	$\cdots$
$\{s_1\}$	1	0	0	0	$\cdots$
$\{s_2, s_3\}$	0	1	1	0	$\cdots$
$\{s_1, s_3, s_4\}$	1	0	1	1	$\cdots$

Osservazione:

Ogni stringa binaria infinita corrisponde a un elemento dell' insieme delle parti:

esempio: $1\,0\,0\,1\,1\,1\,0 \;\cdots$

Corrispondente a : $\{s_1, s_4, s_5, s_6, \ldots\} \in 2^S$

assumiamo (per assurdo)

Che l' Insieme delle parti 2^S

è enumerabile

allora: possiamo enumerare gli elementi dell' insieme delle parti

$$2^S = \{t_1, t_2, t_3, \ldots\}$$

Insieme delle Parti elementi

Supponiamo che la seguente sia

Codifica Binaria

t_1	1	0	0	0	0	$\cdots$
t_2	1	1	0	0	0	$\cdots$
t_3	1	1	0	1	0	$\cdots$
t_4	1	1	0	0	1	$\cdots$
$\cdots$			$\cdots$			

Prendiamo la diagonale e complementiamola

$$\begin{array}{cccccccc}
t_1 & & 1 & 0 & 0 & 0 & 0 & \cdots \\
t_2 & & 1 & 1 & 0 & 0 & 0 & \cdots \\
t_3 & & 1 & 1 & 0 & 1 & 0 & \cdots \\
t_4 & & 1 & 1 & 0 & 0 & 1 & \cdots
\end{array}$$

Stringa binaria: $\boldsymbol{t} = 0011\ldots$

Stringa binaria

$$\boldsymbol{t} = 0011\ldots$$

Corrisponde ad un elemento dell'Insieme delle parti 2^S :

$$\boldsymbol{t} = \{s_3, s_4, \ldots\} \in 2^S$$

allora, t deve essere uguale a qualche t_i

$$t = t_i$$

ma,

il i-th bit nella codifica t è
il complemento i-th del bit di t_i, allora:

$$t \neq t_i \qquad \text{Per ogni i}$$

Contradizione!!!

Poichè abbiamo ottenuto una contradizione a partire dall'ipotesi che 2^S è contabile:

L' insieme delle Parti 2^S di S è uncountable

FINE

Una applicazione: Linguaggi

Considera l'alfabeto : $A = \{a, b\}$

L'insieme delle stringhe:

$$S = \{a, b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

infinito e countable

(possiamo enumerare le stringhe
in ordine proprio)

Considera alfabeto : $A = \{a, b\}$

L'insieme delle stringhe:

$$S = \{a, b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

infinito e countable

Ogni linguaggio è un sottoinsieme di S :

$$L = \{aa, ab, aab\}$$

Considera l'alfabeto : $A = \{a, b\}$

L'insieme delle stringhe:

$$S = A^* = \{a, b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

infinito e countable

Ricorda: L'insieme delle parti di S contiene tutti i linguaggi:

$$2^S = \{\emptyset, \{\lambda\}, \{a\}, \{a, b\}, \{aa, b\}, \ldots, \{aa, ab, aab\}, \ldots\}$$

uncountable

Considera alfabeto : $A = \{a, b\}$

countable

Turing machines: $M_1 \quad M_2 \quad M_3 \quad \cdots$

accetta

Linguaggi accettati da Turing Machines: $L_1 \quad L_2 \quad L_3 \quad \cdots$

countable

Denota: $X = \{L_1, L_2, L_3, \ldots\}$
countable

Nota: $X \subseteq 2^S$

$(S = \{a, b\}^*)$

Linguaggi accettati da Turing machines: X countable

Tutti i possibili linguaggi: 2^S uncountable

quindi: $X \neq 2^S$

$$\left(\text{since } X \subseteq 2^S, \text{ we have } X \subset 2^S\right)$$

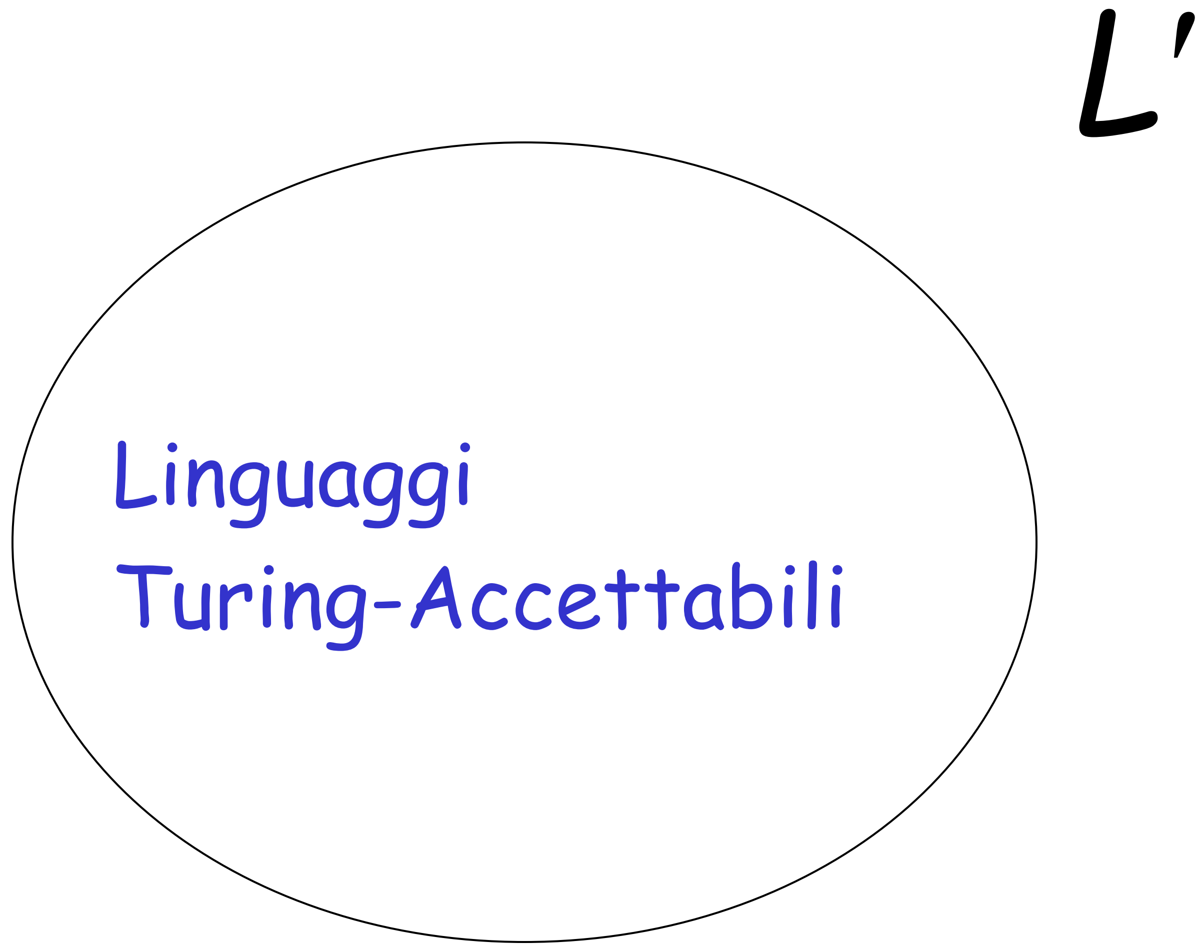
Conclusione:

Esiste un linguaggio L' non accettato da nessuna Turing Machine:

$$X \subset 2^S \Longrightarrow \exists L' \in 2^S \text{ and } L' \notin X$$

(linguaggio L' non può essere descritto da nessun algoritmo)

Linguaggi Non Turing-Accettabili



Nota che: $X = \{L_1, L_2, L_3, \ldots\}$

È un *multi-set* (elementi possono ripetersi)
Poichè un linguaggio può essere riconosciuto da più di una Turing machine

Anche se esaminiamo i doppioni il risultato è di nuovo un insieme countable poichè ogni elemento corrisponde a un intero positivo

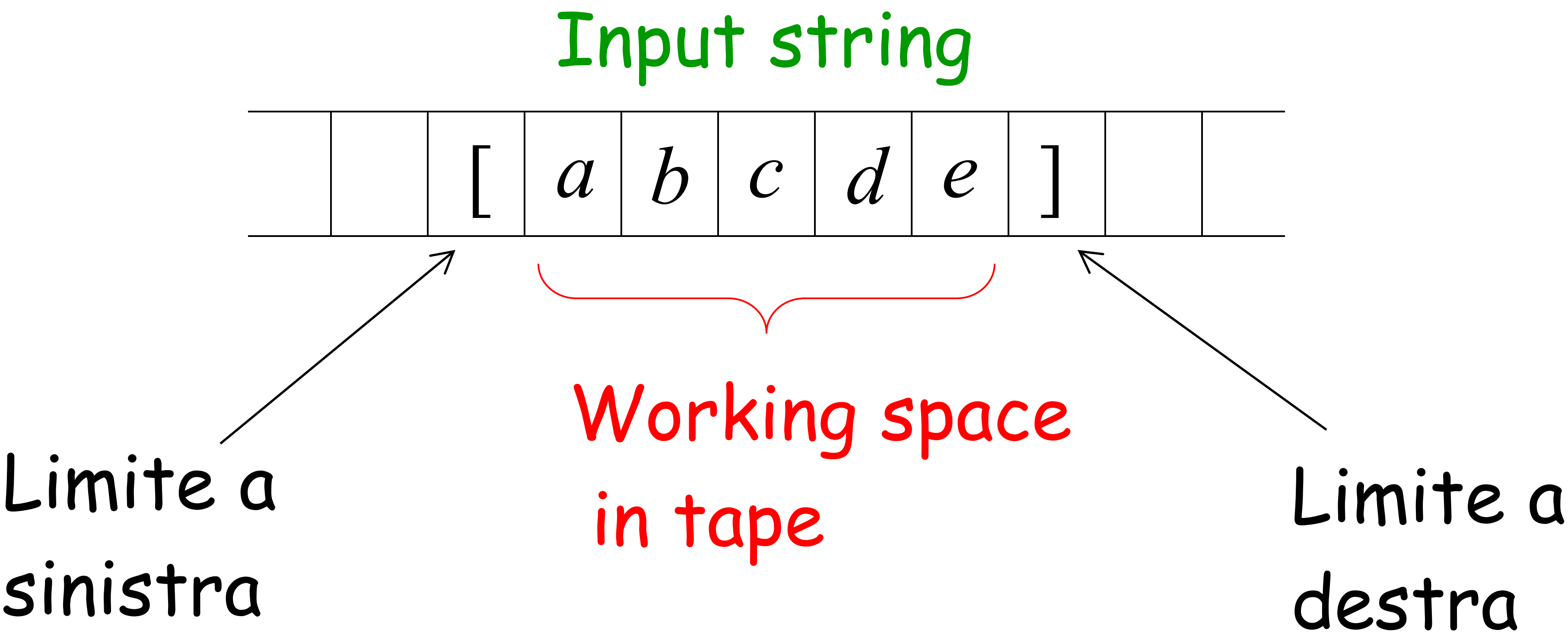
La gerarchia di Chomsky

Linear-Bounded Automa:

Come una Macchina di Turing
con una differenza:

Lo spazio dove è memorizzato l'input
è il solo spazio che può essere
utilizzato

Linear Bounded Automa (LBA)



Tutta la computazione si svolge tra i due limiti

Definiamo i LBA come macchine non deterministiche

Problema aperto:

LBA NonDeterministici
hanno lo stesso potere dei
LBA Deterministici ?

Esempio linguaggio accettato da un LBA:

$$L = \{a^n b^n c^n\} \qquad\qquad L = \{a^{n!}\}$$

LBA hanno più potere dei PDA (pushdown automata)

LBA hanno meno potere delle Turing Machines

Grammatica Context-Sensitive :

produzioni

$$u \rightarrow v$$

Stringhe di variabili
e terminali

Stringhe di variabil
e terminali

e: $$|u| \leq |v|$$

Il linguaggio $\{a^n b^n c^n\}$

è context-sensitive:

$$S \to abc \mid aAbc$$

$$Ab \to bA$$

$$Ac \to Bbcc$$

$$bB \to Bb$$

$$aB \to aa \mid aaA$$

Teorema:

Un linguaggio L è context sensitive

$\longrightarrow$

è accettato da Linear-Bounded automa

Devo provare il contrario

Dato L (data la grammatica per L) devo costruire una LB che riconosce tutte e solo le parole di L stringa elemento di L la macchina mi deve dire si

stringa non è elemento di L la macchina mi deve dire no

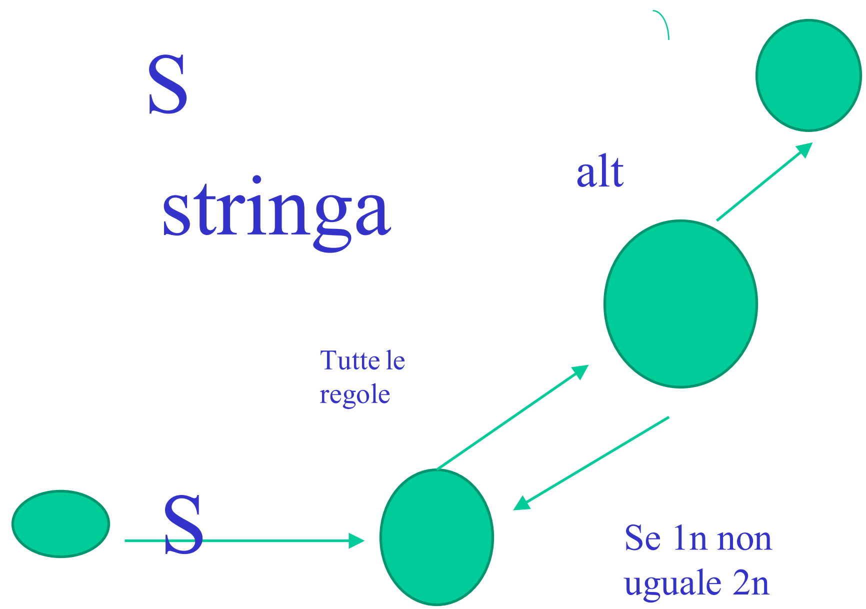
$$S \rightarrow abc \mid aAbc$$

$$Ab \rightarrow bA$$

$$Ac \rightarrow Bbcc$$

$$bB \rightarrow Bb$$

$$aB \rightarrow aa \mid aaA$$



Grammatiche senza limitazioni:

Produzioni

$$u \rightarrow v$$

Stringhe di variabili e terminali

Stringhe di variabili e terminali

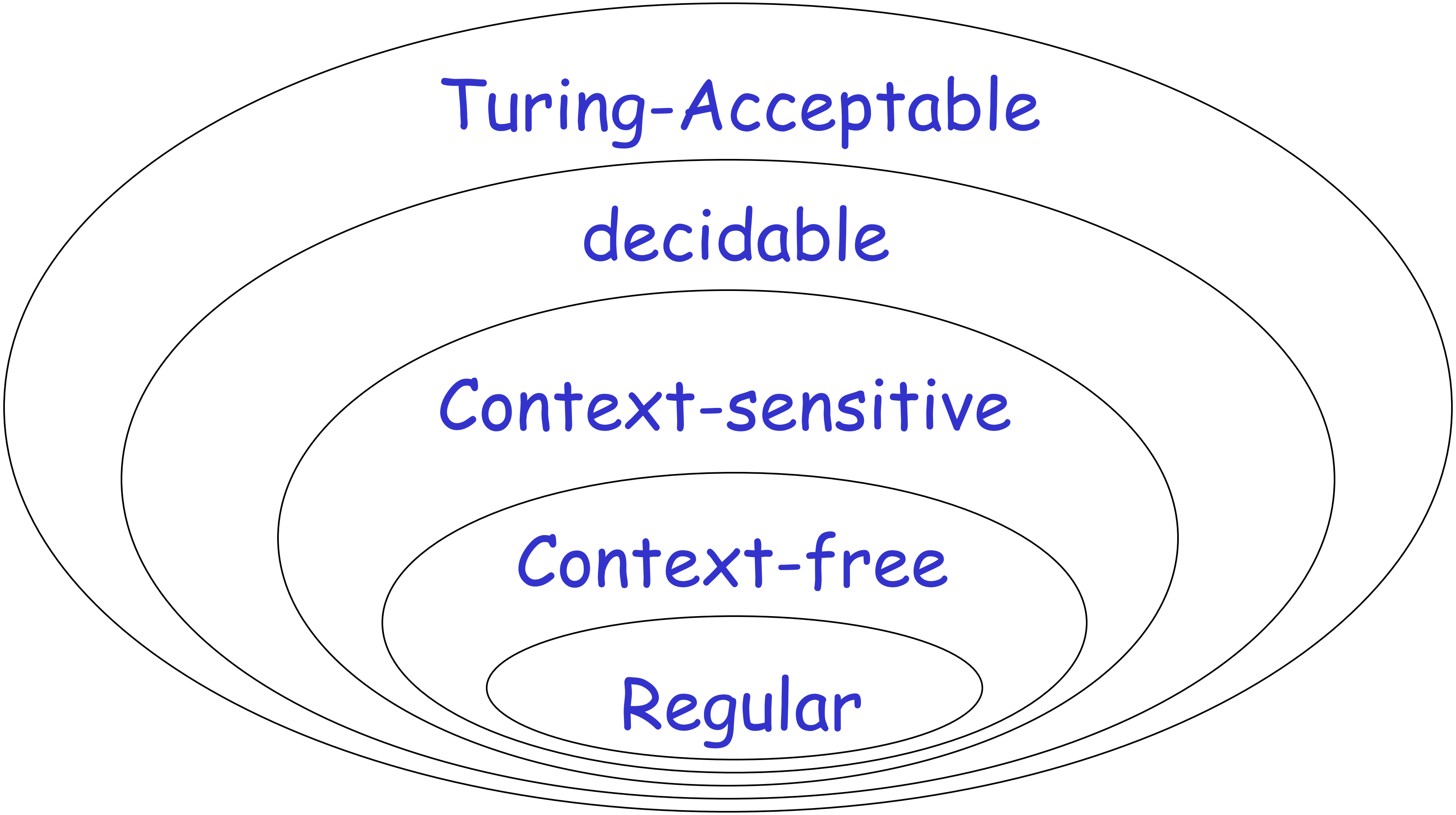
Teorema:

Un linguaggio L è Turing-Acceptable
Se L è generato da
una grammatica senza restrizione

Perchè? Come fare?

The Chomsky gerarchia

Non Turing-Acceptable



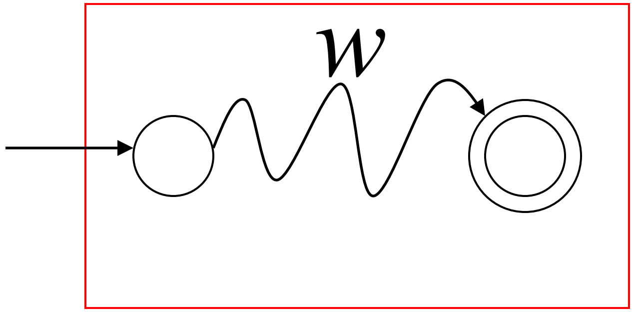
problemi decidibili (decidibile?) per linguaggi regolari

appartenenza

Domanda: dato un linguaggio regolare L
e una stringa w
Possiamo verificare se $w \in L$ **?**

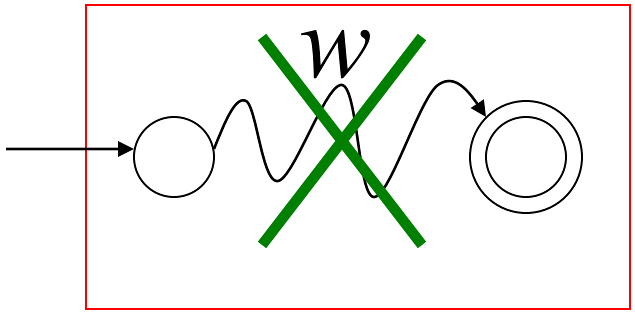
Risposta: Prendiamo un DFA che
accetta L
e verifichiamo se w è
accettato.

DFA



$w \in L$

DFA



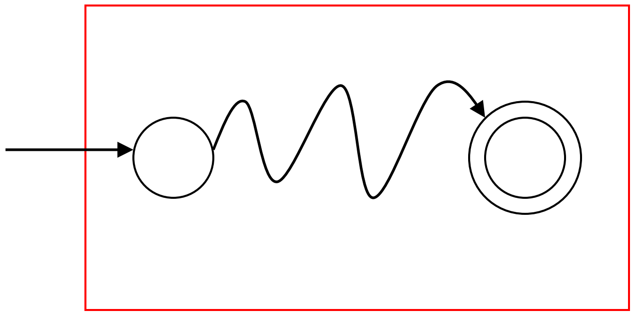
$w \notin L$

Domanda: dato un linguaggio regolare L
Come possiamo verificare
se L è vuoto: $(L = \emptyset)$?

Risposta: Prendiamo il DFA che accettà L

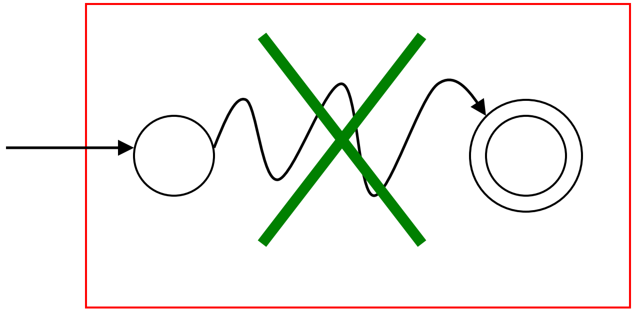
Verifichiamo se esiste almeno
un cammino da uno stato iniziale
ad uno stato di accettazione

DFA



$$L \neq \emptyset$$

DFA



$$L = \emptyset$$

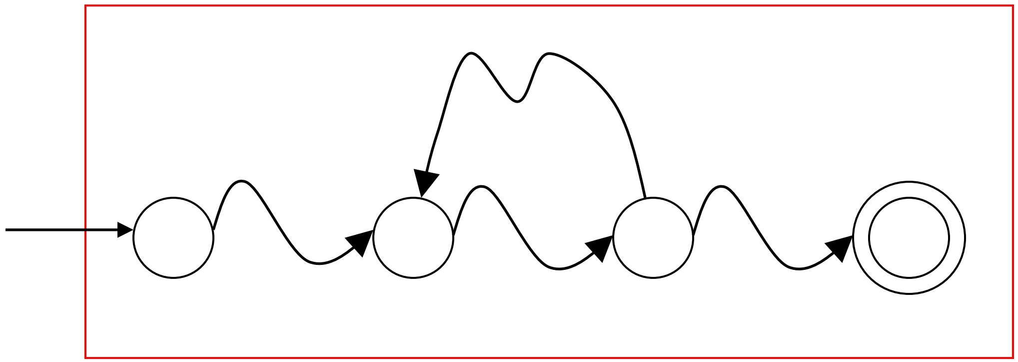
Domanda: Dato un linguaggio regolare L
Come possiamo verificare
Se L è finito?

Prendiamo il DFA che accetta L

Risposta:

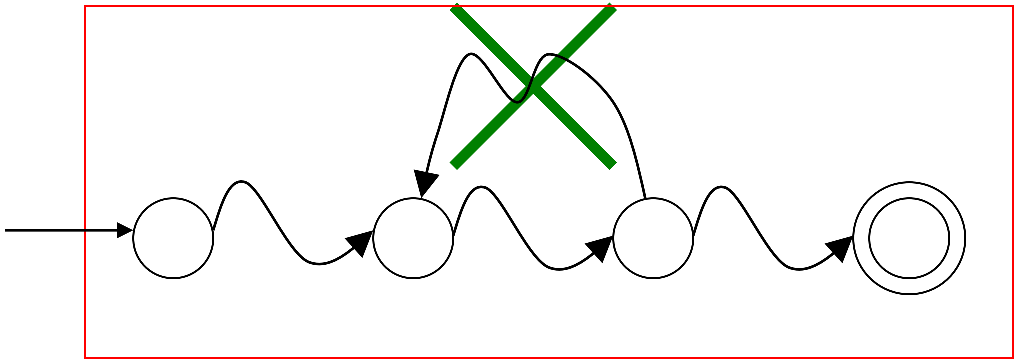
Verifichiamo se vi è un cammino con un loop.

DFA



L è infinito

DFA



L è finito

Domanda: dato linguaggi regolari L_1 e L_2
Come possiamo verificare che

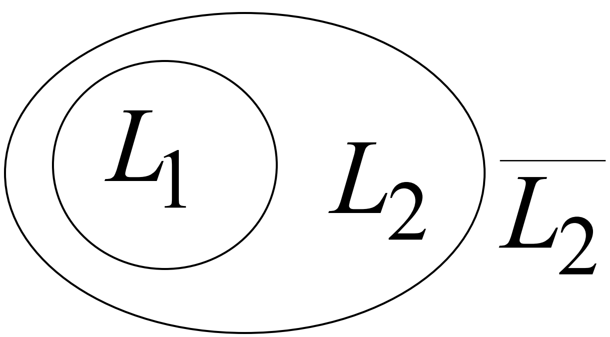
$$L_1 = L_2$$

Risposta: Trova se

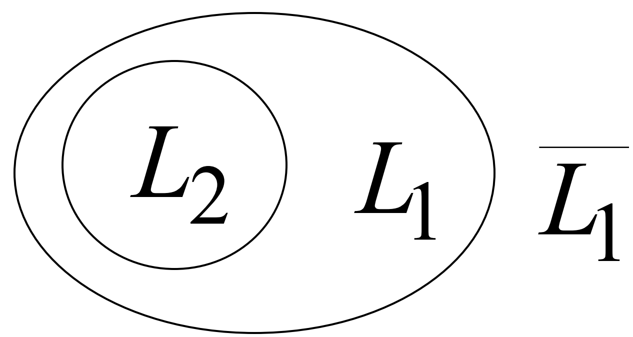
$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

$L_1 \cap \overline{L_2} = \emptyset$ **and** $\overline{L_1} \cap L_2 = \emptyset$



$$L_1 \subseteq L_2$$

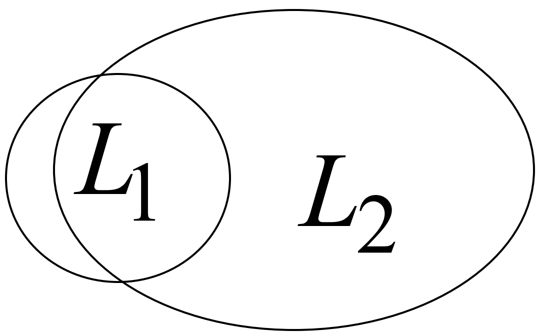


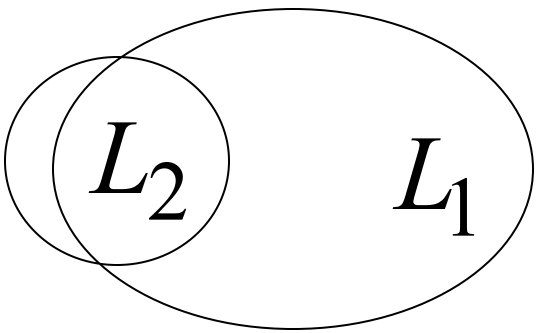
$$L_2 \subseteq L_1$$

$$L_1 = L_2$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) \neq \emptyset$$

$$L_1 \cap \overline{L_2} \neq \emptyset \quad \text{or} \quad \overline{L_1} \cap L_2 \neq \emptyset$$





$$L_1 \not\subset L_2 \qquad L_2 \not\subset L_1$$

$$L_1 \neq L_2$$

problemi decidibili
per
linguaggi Context-Free

appartenenza:

per grammatiche context-free G

Se la stringa $w \in L(G)$

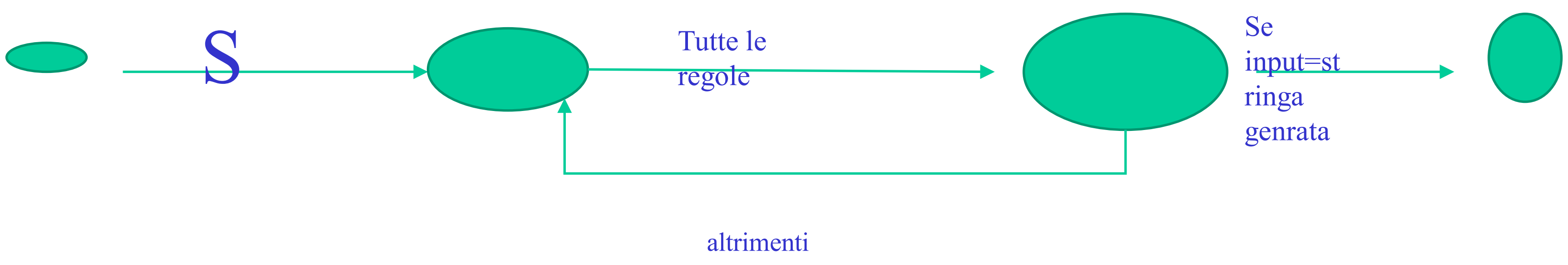
appartenenza : Parsers

- "Exhaustive search parser" o "non determinismo"
  - **CYK** parsing algorithm

Teorema:

Un linguaggio L è Turing-Acceptable
Se L è generato da
una grammatica senza restrizione

Grammatica definire Turing machine
(linear bounded uguale)



Turing machine → una grammatica senza restrizione

? =Qualsiasi carattere

Stati= non terminali, car =terminali

Q c:= Q ? c   Q c= c Q

Delta(Q, c)=(Q, c, L)   Q c = Q ? c

Delta(Q, c)=(Q, c, R)

Spostandosi trovo il blank Delta(Q, c)=(Q, c, L)
Delta(Q, b)=(Q, c, R|L)

F:= fermo la computazione