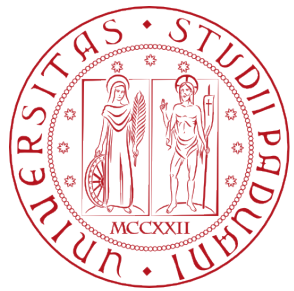
Automi e Linguaggi Formali

Macchine di Turing, problemi ricorsivi e ricorsivamente enumerabili



UNIVERSITÀ DEGLI STUDI DI PADOVA

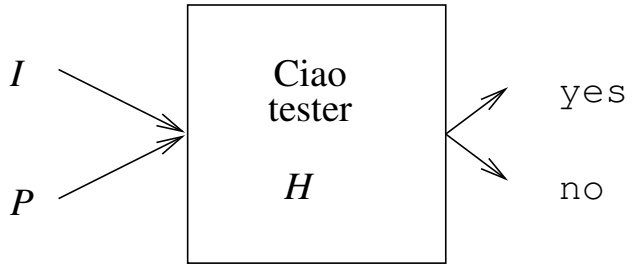
A.A. 2014-2015
Alessandro Sperduti
sperduti@math.unipd.it

Problemi che i calcolatori non possono risolvere

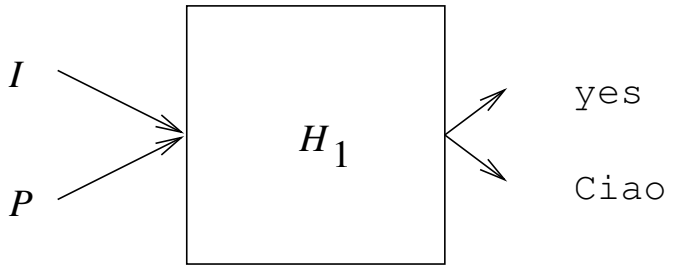
- È importante sapere se un programma è corretto, cioè fa quello che ci aspettiamo
- È facile vedere che il programma contente solo il comando `printf(‘‘Ciao’’)` stampa `Ciao`
- Ma il programma in Fig. 8.2 del libro?
- Stampa `Ciao`, dato un input n se e solo se l'equazione $x^n + y^n = z^n$ ha una soluzione dove x, y e z sono interi
- Sappiamo ora (ultimo teorema di Fermat) che stampa `Ciao`, con l'input $n = 2$, e cicla per sempre su input $n > 2$
- Ci sono voluti 300 anni per provarlo
- Possiamo sperare di avere un programma che prova la correttezza di programmi?

L'ipotetico programma H che testa ``Ciao''

Supponiamo che esista un programma H tale che, dato un qualunque programma P e un input I, dice `yes` se P con input I stampa ``Ciao'', altrimenti dice `no`

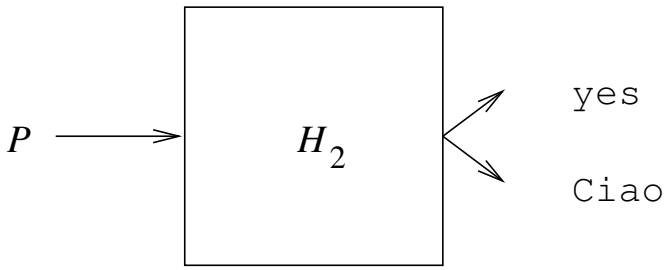


Modifichiamo il comando di stampa di `no` di H in `Ciao`.
Otteniamo il programma H_1

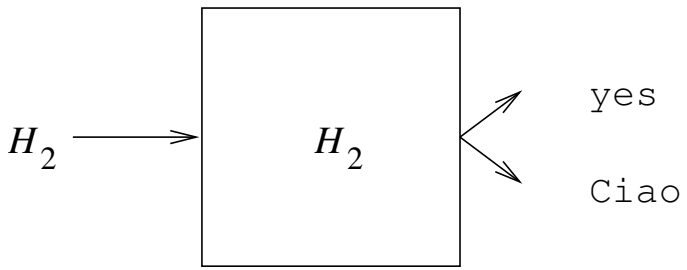


L'ipotetico programma H che testa ``Ciao''

Modifichiamo H_1 in modo che l'input I sia lo stesso P. Otteniamo il programma H_2:

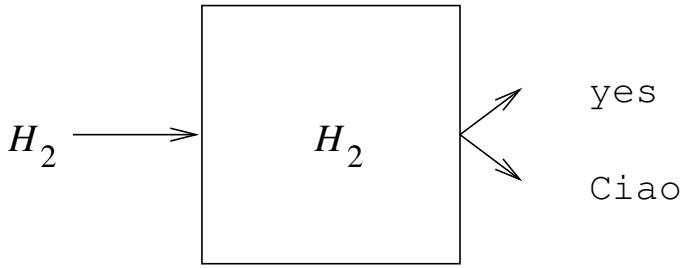


H_2 stampa `yes` se P con input P stampa `Ciao`, altrimenti stampa `Ciao`. Diamo H_2 in input ad H_2:



L'ipotetico programma H che testa ``Ciao''

Diamo H_2 in input ad H_2:



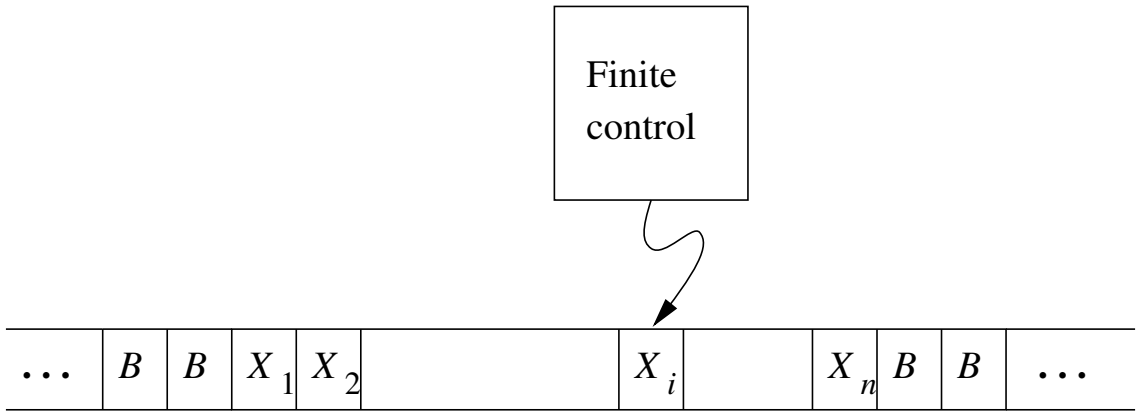
- Se H_2 stampa `yes`, avrebbe dovuto stampare `Ciao`
- Se H_2 stampa `Ciao`, avrebbe dovuto stampare `yes`
- Quindi H_2 non può esistere
- Quindi neanche H può esistere

Quindi il problema affrontato è *indecidibile*: non esiste nessun programma che, dato un programma qualsiasi e un input, sappia dire se quel programma con quell'input stampa `Ciao`

Problemi indecidibili

- Problemi per cui non esiste alcun programma che li possa risolvere
- Problema: appartenenza di una stringa ad un linguaggio
- Il numero di linguaggi diversi su un alfabeto non è numerabile
- I programmi (stringhe finite su un alfabeto) sono numerabili: si possono ordinare per lunghezza, e poi lessicograficamente $\Rightarrow$ primo programma, secondo programma, ecc.
- Quindi esistono infinitamente più linguaggi che programmi
- Quindi devono esistere problemi indecidibili (Godel 1931)

La macchina di Turing (Turing, 1936)



Una TM fa una **mossa** in funzione del suo stato, e del simbolo sotto la testina di lettura del nastro.
In una mossa, una TM

1. cambia stato
2. scrive un simbolo del nastro nella cella sotto la testina
3. muove la testina di una cella verso destra o verso sinistra

Definizione formale come automa

Una macchina di Turing deterministica è una 7-tupla

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F),$$

dove

1. Q è un insieme finito di stati,
2. Σ è un insieme finito di simboli di input,
3. Γ è un insieme finito di simboli di nastro,
4. δ è una funzione di transizione da Q a $Q \times \{L, R\}$,
5. q_0 è lo stato iniziale,
6. $B \in \Gamma$ è il simbolo blank, e
7. $F \subseteq Q$ è l'insieme di stati finali

Descrizioni istantanee

Una TM cambia configurazione dopo ogni mossa.
Usiamo le descrizioni istantanee (ID) per descrivere le configurazioni.
Una ID è una stringa della forma

$$X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n$$

dove

1. q è lo stato della TM
2. $X_1 X_2 \cdots X_n$ è la porzione non-blank del nastro
3. La testina è sopra il simbolo i-esimo

Le mosse e il linguaggio di una TM

Useremo $\vdash_M$ per indicare una mossa di M da una configurazione ad un'altra.

- Supponiamo $\delta(q, X_i) = (p, Y, L)$. Allora

$$X_1 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \ \vdash_M \ X_1 \cdots X_{i-2} p X_{i-1} Y X_{i+1} \cdots X_n$$

- Se $\delta(q, X_i) = (p, Y, R)$. Allora

$$X_1 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \ \vdash_M \ X_1 \cdots X_{i-1} Y p X_{i+1} \cdots X_n$$

Indichiamo la chiusura riflessiva e transitiva di $\vdash_M$ con $\vdash_M^*$.

- Una TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ **accetta** il linguaggio

$$L(M) = \{w \in \Sigma^* \ : \ q_0 w \vdash_M^* \alpha p \beta, p \in F, \alpha, \beta \in \Gamma^*\}$$

Una TM per $\{0^n1^n : n \geq 1\}$

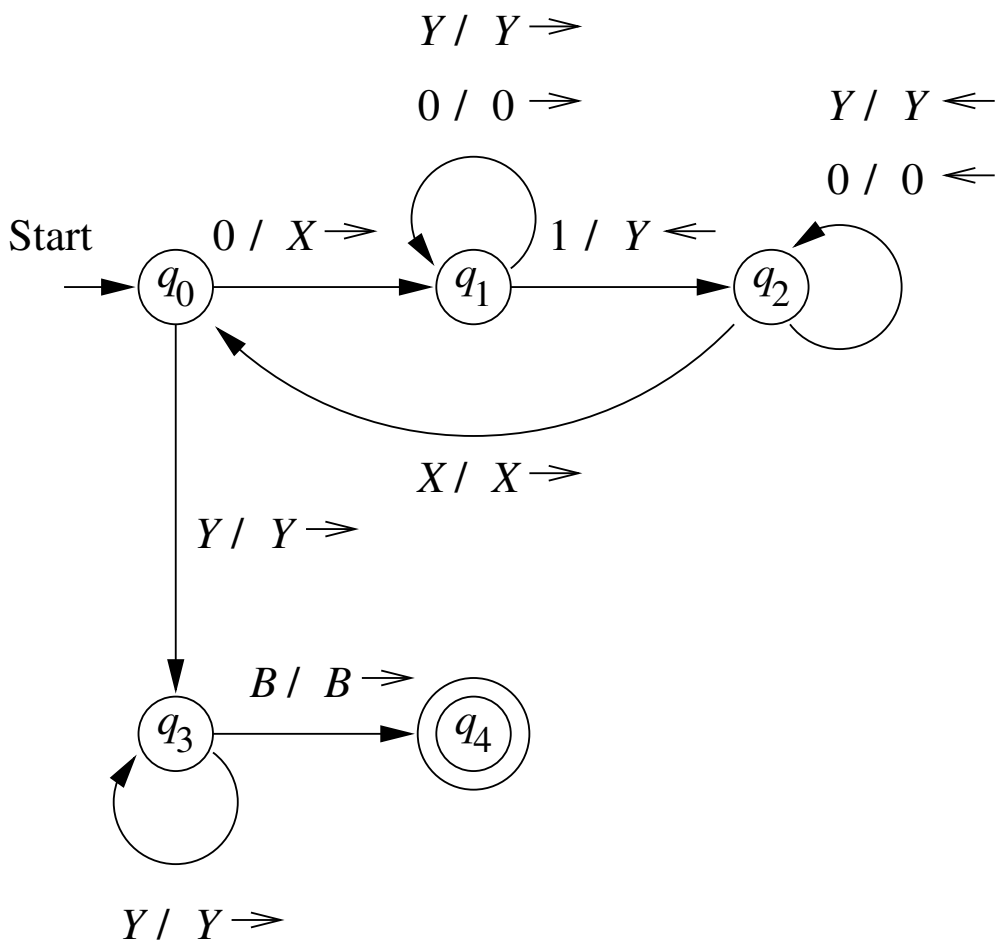
$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

dove δ è data dalla seguente tabella

	0	1	X	Y	B
$\rightarrow q_0$	(q_1, X, R)			(q_3, Y, R)	
q_1	$(q_1, 0, R)$	(q_2, Y, L)		(q_1, Y, R)	
q_2	$(q_2, 0, L)$		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	(q_4, B, R)
$*q_4$					

Una TM per $\{0^n 1^n : n \geq 1\}$

Possiamo rappresentare M con il seguente **diagramma di transizione**



Una TM con "output"

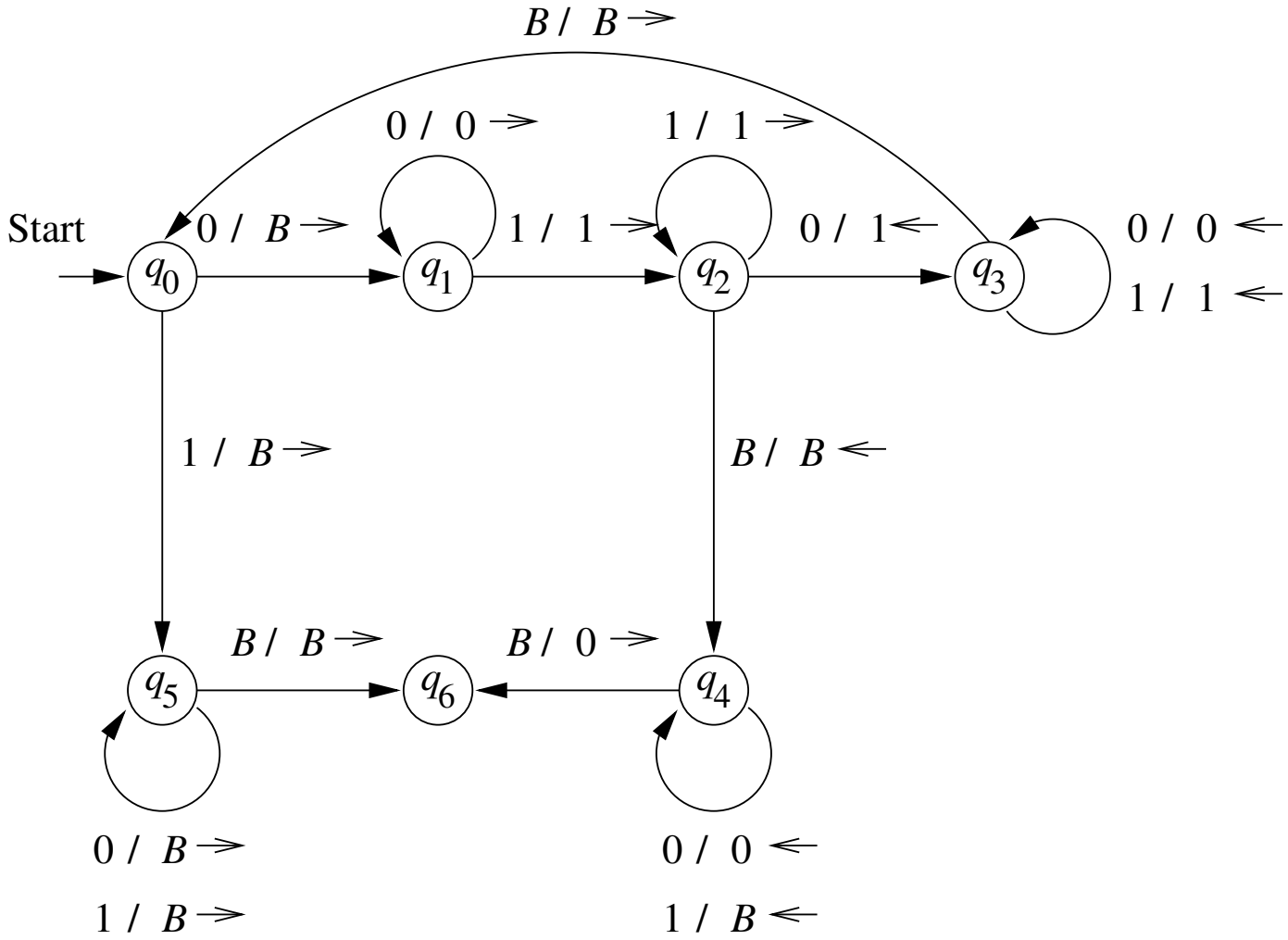
La seguente TM calcola

$$m \dot{-} n = \max(m - n, 0)$$

	0	1	B
$\rightarrow q_0$	(q_1, B, R)	(q_5, B, R)	
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	
q_2	$(q_1, 1, L)$	$(q_2, 1, R)$	(q_4, B, L)
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4	$(q_4, 0, L)$	(q_4, B, L)	$(q_6, 0, R)$
q_5	(q_5, B, R)	(q_5, B, R)	(q_6, B, R)
$*q_6$			

Una TM con "output"

Il diagramma di transizione è



Accettazione per arresto

1. Una TM **si arresta** se entra in uno stato q guardando un simbolo di nastro X e non ci sono mosse possibili, cioè $\delta(q, X)$ non è definita
2. Se una TM accetta una stringa, possiamo assumere che si arresti (basta rendere indefinito $\delta(q, X)$ per ogni q accettante)
3. Se non accetta, non possiamo fare in modo che si arresti
4. Linguaggi **ricorsivi**: esiste una TM che si arresta su ogni stringa (sia accettata che non accettata)
5. Linguaggi **ricorsivamente enumerabili**: esiste una TM che si arresta se la stringa è accettata
6. Problema **decidibile**: esiste una TM che si arresta sempre