Lezione 5 – la macchina di Turing Universale

Lezione del 21/03/2023

Cosa è una macchina di Turing?

- Una macchina di Turing è la descrizione di un procedimento per risolvere un problema
  - decritto nel linguaggio delle quintuple
  - ossia, è un procedimento per il modello di calcolo Macchina di Turing
- Quindi, una macchina di Turing è un algoritmo
  - e, se la facciamo lavorare su qualche input, quella, in qualche modo, ci calcola la soluzione per l'istanza del problema che gli abbiamo dato in input
- e il dato in input, per una macchina di Turing, è una parola, costituita da caratteri di un certo alfabeto
  - l'input è una **parola** – che viene scritta sul nastro della macchina
- Uhmmm… Una macchina di Turing, però, è anche qualcos'altro

Macchine e parole

- Prendiamo una macchina di Turing:
  - cioè, un alfabeto Σ e un insieme degli stati Q
  - e, soprattutto, *l'insieme delle sue quintuple* P
  - osservate che è sufficiente avere l'insieme P per sapere tutto di T: da T possiamo ricavare sia Σ che Q
  - beh, in effetti P non ci dice proprio tutto tutto: per sapere tutto di T, oltre che P, dobbiamo conoscere anche quale sia lo stato iniziale e quali siano gli stati finali
  - "e questa cosa, quello che ci basta per sapere tutto di T, tenetelo a mente perché ci servirà nella prossima lezione", vi avevo detto
- Bene. Siamo alla "prossima lezione".

Macchine e parole

- Ebbene, data una macchina T
- se decidiamo di costruire una parola secondo le regole seguenti
  - il primo carattere della parola è 'q_0', che è seguito da un carattere non in Σ, diciamo '-'
  - che è seguito da 'q_A', poi da '-', poi da 'q_R',
  - e poi, seguono, una di seguito all'altra, tutte le quintuple
- la parola che abbiamo appena costruito definisce completamente T
- Facciamo un esempio

Macchine e parole

- Prendiamo una macchina T_{PAL} che termina in q_A se la parola scritta (composta da caratteri ‘a’ e ‘b’) sul suo nastro ha lunghezza pari ed è palindroma
- il suo stato iniziale è q_0, il suo stato di accettazione è q_A, il suo stato di rigetto è q_R, e le sue quintuple sono
  - $\langle q_0, a, \square, q_a, D \rangle$, $\langle q_0, b, \square, q_b, D \rangle$,
  - $\langle q_a, a, a, q_a, D \rangle$, $\langle q_a, b, b, q_a, D \rangle$, $\langle q_b, a, a, q_b, D \rangle$, $\langle q_b, b, b, q_b, D \rangle$,
  - $\langle q_a, \square, \square, q_{a1}, S \rangle$, $\langle q_b, \square, \square, q_{b1}, S \rangle$,
  - $\langle q_{a1}, a, \square, q_2, S \rangle$, $\langle q_{a1}, b, b, q_R, F \rangle$, $\langle q_{b1}, a, a, q_R, F \rangle$, $\langle q_{b1}, b, \square, q_2, S \rangle$,
  - $\langle q_2, a, a, q_2, S \rangle$, $\langle q_2, b, b, q_2, S \rangle$, $\langle q_2, \square, \square, q_0, D \rangle$,
  - $\langle q_0, \square, \square, q_A, F \rangle$.
- Ebbene, è T_{PAL} completamente descritta dalla parolona seguente:
  - $q_0 - q_A - q_R$ $\langle q_0, a, \square, q_a, D \rangle$ $\langle q_0, b, \square, q_b, D \rangle$ $\langle q_a, a, a, q_a, D \rangle$ $\langle q_a, b, b, q_a, D \rangle$ $\langle q_b, a, a, q_b, D \rangle$ $\langle q_b, b, b, q_b, D \rangle$ $\langle q_a, \square, \square, q_{a1}, S \rangle$ $\langle q_b, \square, \square, q_{b1}, S \rangle$ $\langle q_{a1}, a, \square, q_2, S \rangle$ $\langle q_{a1}, b, b, q_R, F \rangle$ $\langle q_{b1}, a, a, q_R, F \rangle$ $\langle q_{b1}, b, \square, q_2, S \rangle$ $\langle q_2, a, a, q_2, S \rangle$ $\langle q_2, b, b, q_2, S \rangle$ $\langle q_2, \square, \square, q_0, D \rangle$ $\langle q_0, \square, \square, q_A, F \rangle$.

Aperta parentesi

- Vi siete accorti che l'insieme delle quintuple di T_{PAL} non è una funzione totale?
- Infatti, non considera in alcun modo il caso in cui la parola in input ha lunghezza dispari. In questo caso, infatti, $T_{PAL}(x)$ termina
  - nello stato q_{a1} se x è una parola palindroma di lunghezza dispari ed ha 'a' al centro – per esempio, abbababba
  - nello stato q_{b1} se x è una parola palindroma di lunghezza dispari ed ha 'b' al centro – per esempio, abbbbba
- Naturalmente, possiamo completare P aggiungendo le quintuple
  - $\langle q_{a1}, \square, \square, q_R, F \rangle$, $\langle q_{b1}, \square, \square, q_R, F \rangle$
- Osservate che, poiché vogliamo che T_{PAL} termini in q_A solo se la parola scritta sul suo nastro, oltre ad essere palindroma, ha lunghezza pari, allora T_{PAL} rigetta le parole palindrome di lunghezza dispari
  - come abbiamo già visto!
- Chiusa parentesi (ma ci torneremo la prossima lezione). Ora proseguiamo con "Macchine e parole"

Macchine e parole

- Insomma, in definitiva, una macchina di Turing è… una parola!
  - costituita di caratteri dell'alfabeto $Q \cup \Sigma \cup \{-\} \cup \{\langle\} \cup \{\rangle\} \cup \{\square\}$
- Ma, se è una parola, allora possiamo ben pensare di scriverla sul nastro di un'altra macchina di Turing (chiamiamola A)
  - così che A lavori sulla nostra macchina di Turing come input
- Va bene, possiamo. Ma perché mai dovremmo?!
  - beh, per esempio, se sul nastro di A ci scriviamo, oltre alla parola che descrive la nostra macchina di Turing di partenza (chiamiamola T), anche un input x di T, allora A potrebbe simulare la computazione T(x)
  - e, dunque, se chiamiamo p_T la parola che descrive T, l'esito della computazione $A(p_T, x)$ sarebbe uguale all'esito della computazione T(x)
- Sì, carino, ma a che serve?!
- Eh, a che serve…

Oltre la macchina

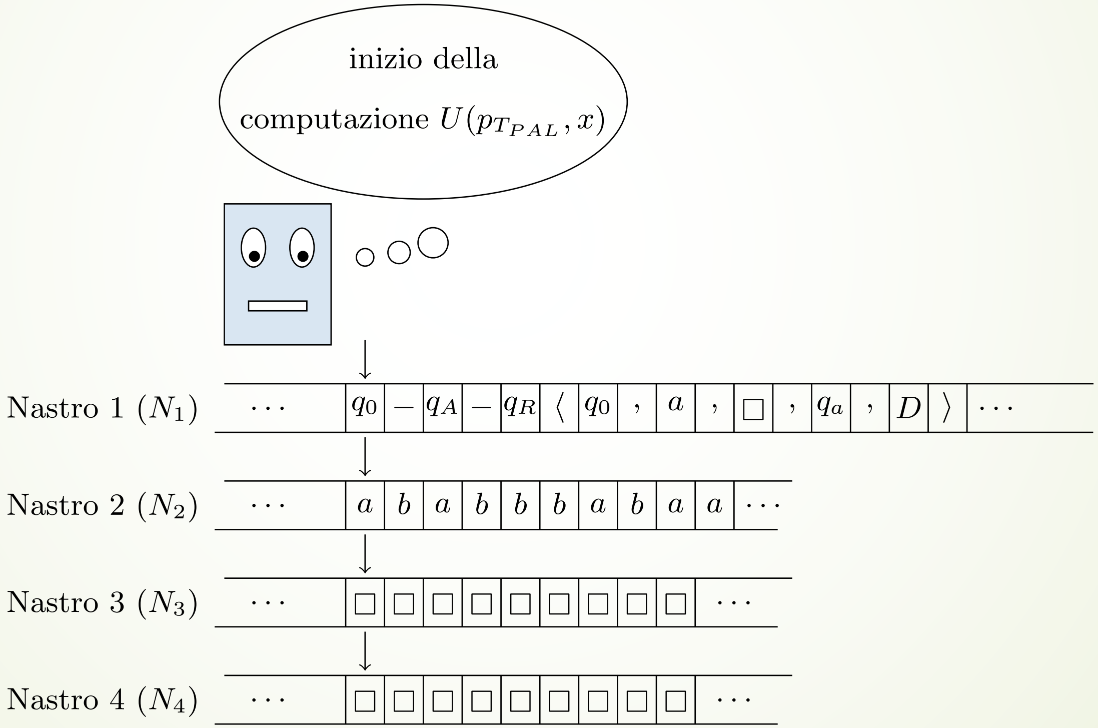
- Pensate se, per caso, riuscissimo a progettare una macchina di Turing U che prende in input due parole
  - una parola p_T che descrive una ***qualsiasi*** macchina di Turing T
  - una parola x, input di T
- e che riesce a simulare la computazione T(x) – ***qualunque sia T!!!!***
- Ossia, U sarebbe una macchina di Turing alla quale potrei comunicare un algoritmo (qualsiasi!) e un input per quell'algoritmo, e U eseguirebbe l'algoritmo su quell'input
- U sarebbe l'algoritmo che descrive il comportamento di un calcolatore!
- Turing ha progettato U – quella che ha preso il nome di *macchina di Turing Universale*
- descritta a fondo nel paragrafo 2.6 – ora vi illustro solo qualche idea
  - la notazione nel paragrafo 2.6 è un po' diversa da quella che vi ho descritto qui (troppo complicato replicare quella notazione in power point)

La macchina Universale U

- Intanto, progettiamo U in modo tale che sappia simulare soltanto macchine ad un nastro
  - tanto, lo sappiamo come simulare una qualunque macchina di Turing mediante una macchina ad un nastro (vero che lo sappiamo? 🙄🤔😠)
- Invece, dotiamo U di 4 nastri e testine indipendenti
  - sul primo nastro viene inizialmente scritta la parola p_T che descrive la macchina T la cui computazione deve essere simulata – e il contenuto di questo nastro non sarà mai modificato durante la computazione U(T, x)
  - sul secondo nastro viene scritto l'input x della macchina T – e questo sarà il nastro sul quale avverrà la simulazione vera e propria della computazione T(x)
  - sul terzo nastro, all'inizio della computazione, U copia lo stato iniziale di T – che, ricordiamo, è il primo simbolo di p_T
  - sul quarto nastro, all'inizio della computazione, U copia lo stato di accettazione di T – che, ricordiamo, è il simbolo di p_T a destra del primo '-'
  - vediamo con qualche figura…

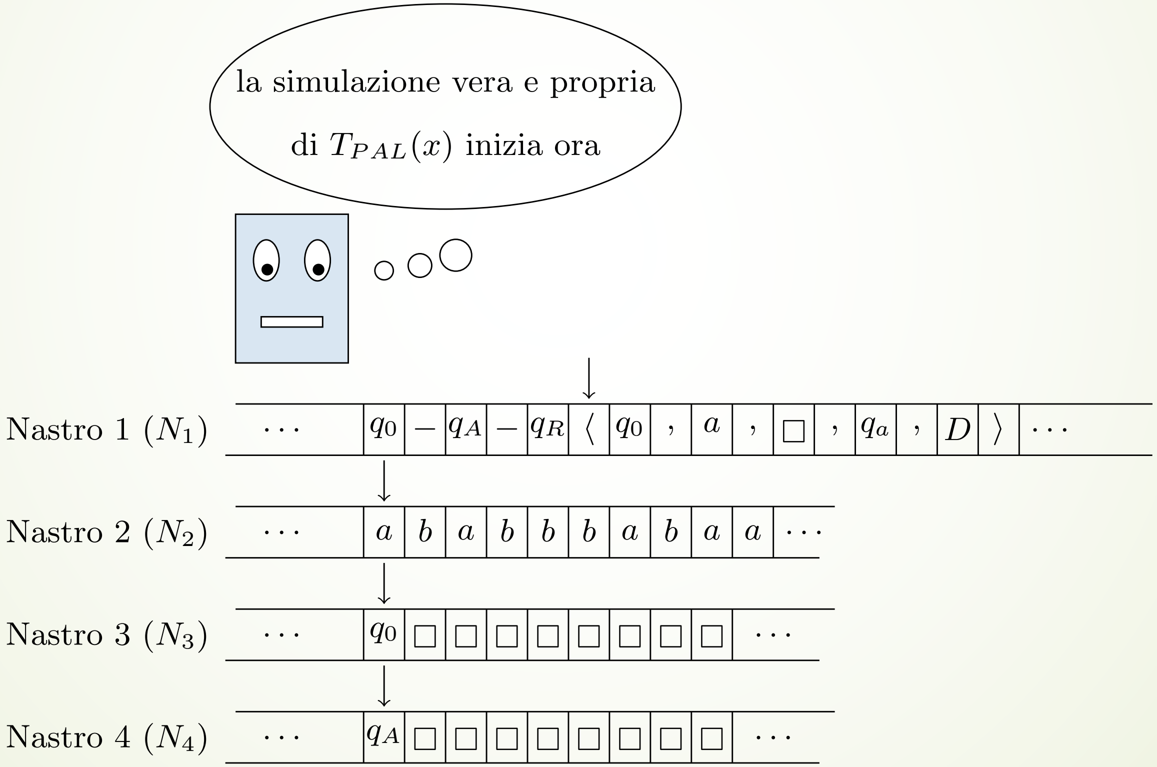
La macchina Universale U

- U prima che la computazione U($p_{T_{PAL}}$, ababbbabaa) abbia inizio



La macchina Universale U

- La computazione U($p_{T_{PAL}}$, ababbbabaa) procede: U ha copiato lo stato iniziale di T_{PAL} su N_3, lo stato di accettazione T_{PAL} su N_4, e si prepara a simulare $T_{PAL}(x)$



La macchina Universale U

- A questo punto, U ha copiato lo stato iniziale di T sul terzo nastro e lo stato di accettazione di T su quarto nastro. Per tutta la durata della simulazione che U sta per iniziare:
  - il contenuto di N_4 non verrà mai modificato
  - **N_3 conterrà sempre lo stato in cui si troverebbe T a quel punto della simulazione**
- U inizia la simulazione di T(x) vera e propria: che è una ripetizione dei passi seguenti
  - 1) U cerca la quintupla di T da eseguire
  - 2) se ha trovato la quintupla da eseguire, allora la esegue e torna al punto 1)
  - 3) se non ha trovato la quintupla da eseguire, allora confronta il carattere letto sul terzo nastro (lo stato in cui si troverebbe T a questo punto della computazione) con il carattere letto sul quarto nastro (lo stato di accettazione di T)
    - se sono uguali, allora accetta
    - se sono diversi, rigetta
- Vediamo i punti 1) e 2) in dettaglio

La macchina Universale U

- 1) U cerca la quintupla di T da eseguire: la testina su N_1 è posizionata sul primo carattere ' 〈' ; U esegue i passi seguenti
  - 1.1) muove a destra di una posizione la testina su N_1
  - 1.2) se legge lo stesso carattere su N_1 e su N_3, allora U sta scandendo una quintupla di T che inizia con lo stato in cui si troverebbe T a questo punto della computazione; in questo caso muove a destra la testina su N_1 per posizionarla sul carattere a destra di ','
    - 1.2.1) se legge lo stesso carattere su N_2 e N_1, allora ha trovato la quintupla da eseguire e passa al punto 2)
    - 1.2.1) se non legge lo stesso carattere su N_2 e N_1, allora non ha trovato la quintupla da eseguire: in questo caso, muove a destra la testina su N_1 alla ricerca del prossimo carattere ' 〈' : se lo trova allora torna al punto 1.1), se non lo trova allora ha scandito tutte le quintuple di T senza trovare quella da eseguire e passa al punto 3)
  - 1.3) se non legge lo stesso carattere su N_1 e su N_3, allora sta scandendo una quintupla di T che non inizia con lo stato in cui si troverebbe T a questo punto della computazione; in questo caso muove a destra la testina su N_1 alla ricerca del prossimo carattere ' 〈' : se lo trova allora torna al punto 1.1), se non lo trova allora ha scandito tutte le quintuple di T senza trovare quella da eseguire e passa al punto 3)

La macchina Universale U

- 2) se U ha trovato la quintupla da eseguire, allora la esegue e torna al punto 1); la testina su N_1 è posizionata sul carattere uguale a quello letto dalla testina su N_2:
  - 2.1) muove a destra di due posizioni la testina su N_1: ora è posizionata sul carattere che deve essere scritto
  - 2.2) copia su N_2 il carattere che legge su N_1
  - 2.3) muove a destra di due posizioni la testina su N_1: ora è posizionata sul carattere che corrisponde allo stato in cui T deve entrare
  - 2.4) copia su N_3 il carattere che legge su N_1
  - 2.5) muove a destra di due posizioni la testina su N_1 : ora è posizionata sul carattere che descrive il movimento della testina
  - 2.6) se su N_1 legge 'S' allora sposta a sinistra la testina su N_2, se su N_1 legge 'D' allora sposta a destra la testina su N_2, se su N_1 legge 'F' allora non compie alcuna azione
- Riferirsi alla figura relativa alla computazione U($p_{T_{PAL}}$, ababbbabaa)
  - e provare a simulare l'intera computazione
- E studiare il paragrafo 2.6) – che quello, poi, vi chiedo!

La macchina Universale U

- Attenzione: abbiamo tralasciato qualche dettaglio importante circa il funzionamento di U!
- Intanto, osservate che le testine sui nastri 3 e 4 non si muovono *mai*
  - inoltre, dopo che nella prima cella di N_4 è stato scritto lo stato di accettazione della macchina "scritta" su N_1, il contenuto di N_4 *non verrà mai più modificato*
- Poi, quale è l'alfabeto di U? Finora abbiamo usato l'insieme $Q \cup \Sigma \cup \{-\} \cup \{ \langle \} \cup \{ \rangle \} \cup \{ \square \} \cup \{ , \}$ come alfabeto
  - ma ogni macchina T ha un suo insieme degli stati Q e un suo alfabeto Σ
  - e noi vogliamo che U sappia simulare ***qualunque*** macchina di Turing T
  - allora, U dovrebbe essere definita su un alfabeto infinito!!!!
  - Ma noi sappiamo che l'alfabeto di una macchina di Turing deve avere cardinalità costante (e, quindi, finita che più finita non si può!!!!)
  - E, allora, codifichiamo tutto in binario
  - E utilizziamo anche la codifica usata nella dispensa…

La macchina Universale U

- Intanto, osserviamo che, senza perdita di generalità, possiamo assumere che sia $\Sigma = \{0,1\}$ (vero che lo sappiamo?!) – e con questo siamo a posto!
- Poi, a pag. 11 della dispensa 2, viene descritta una macchina di Turing T con alfabeto {0,1} e
  - insieme degli stati $Q_T = \{\omega_0, \ldots, \omega_{k-1}\}$, con stato iniziale ω_0, stato di accettazione ω_1, e stato di rigetto ω_2 – osservate: $|Q_T| = k$
  - e insieme delle quintuple $P = \{p_1, \ldots, p_h\}$, dove la sua *i*-esima quintupla è $p_i = \langle \omega_{i1}, b_{i1}, b_{i2}, \omega_{i2}, m_i \rangle$
- mediante la seguente parolona ρ_T i cui caratteri appartengono all'alfabeto $Q_T \cup \{0, 1, \oplus, \otimes, -, f, s, d\}$
  - $\rho_T = \omega_0 - \omega_1 \otimes \omega_{11} - b_{11} - b_{12} - \omega_{12} - m_1 \oplus \omega_{21} - b_{21} - b_{22} - \omega_{22} - m_2 \oplus \ldots \oplus \omega_{h1} - b_{h1} - b_{h2} - \omega_{h2} - m_h \oplus$
  - rispetto alla nostra rappresentazione, in ρ_T abbiamo il carattere '-' al posto di ',', il carattere '$\otimes$' per segnalare l'inizio dell'insieme delle quintuple, e il carattere '$\oplus$' per separare due quintuple e per terminare la parolona

La macchina Universale U

- Dunque, l'alfabeto usato per rappresentare ρ_T contiene Q_T
  - ma ogni macchina T ha un suo insieme degli stati Q_T
  - e noi vogliamo che U sappia simulare ***qualunque*** macchina di Turing T
  - allora, U dovrebbe essere definita su un alfabeto infinito!!!!
  - Ma noi sappiamo che l'alfabeto di una macchina di Turing deve avere cardinalità costante (e, quindi, finita che più finita non si può!!!!)
  - E, allora, codifichiamo Q_T in binario
- A pag. 13 vene introdotta una codifica binaria b^Q dell'insieme Q degli stati di T – invece di usare quella codifica ve ne propongo qui una più semplice
  - $b^Q : Q \rightarrow \{ 0,1 \}^k$, ossia, la codifica b^Q rappresenta uno stato di T mediante una parola di k bit
  - $b^Q(\omega_i)$ è la parola che ha un 1 in posizione i+1 e 0 altrove – esempio: se k=4, $b^Q(\omega_0)$=1000, $b^Q(\omega_1)$=0100, $b^Q(\omega_2)$=0010, $b^Q(\omega_3)$=0001
- a questo punto, rappresentiamo T mediante la seguente parolona nell'alfabeto $\Sigma_B = \{0, 1, \oplus, \otimes, -, f, s, d\}$:
  - $\beta_T = b^Q(\omega_0) - b^Q(\omega_1) \otimes b^Q(\omega_{11}) - b_{11} - b_{12} - b^Q(\omega_{12}) - m_1 \oplus b^Q(\omega_{21}) - b_{21} - b_{22} - b^Q(\omega_{22}) - m_2 \oplus \ldots \oplus b^Q(\omega_{h1}) - b_{h1} - b_{h2} - b^Q(\omega_{h2}) - m_h \oplus$

La macchina Universale U

- Quello che cambia, a questo punto, rispetto alla descrizione di U che abbiamo appena visto è la gestione del passo 1.2):
  - 1.2) se U legge lo stesso carattere su N_1 e su N_3, allora sta scandendo una quintupla di T che inizia con lo stato in cui si troverebbe T a questo punto della computazione; in questo caso muove a destra la testina su N_1 per posizionarla sul carattere a destra di ','
- Adesso, su N_3 non è scritto un singolo carattere, ma una parola di k bit
  - perché, naturalmente, all'inizio della computazione, sul terzo nastro U ha copiato non "lo stato iniziale" di T ma i k bit che codificano lo stato iniziale di T – che, in questo caso, sono i primi k bit di β_T
- Perciò, "se U legge lo stesso carattere su N_1 e su N_3" diventa ora "se la sequenza di k bit sul nastro N_1 che inizia dal punto in cui è posizionata la testina coincide con la sequenza di k bit sul nastro N_3"
  - quella che prima era una quintupla, deve essere ora trasformata in un insieme di quintuple che permettono di eseguire k confronti

La macchina Universale U

- La descrizione **completa** della macchina U che lavora con questa codifica binaria (che è un lavoraccio tecnico) la trovate nel paragrafo 2.6. E la studiate!!!! Perché ci servirà anche più avanti!
- Un'ultima questione: e se, putacaso, la parola scritta sul primo nastro di U non corrisponde alla descrizione di una macchina di Turing?
- Abbiamo due possibilità per gestire questa questione
  - prima di iniziare a copiare lo stato iniziale di T sul terzo nastro e lo stato di accettazione di T sul quarto nastro, U controlla che la parola scritta sul primo nastro sia effettivamente la descrizione di una macchina di Turing (ossia, soddisfi le specifiche descritte a pag. 11 della dispensa 2): se non è così, U termina nello stato di rigetto
  - oppure, utilizziamo la regola che abbiamo illustrato nel paragrafo 2.3: se l'input non rispetta le specifiche… beh, affaracci dell'utente malaccorto, noi ce ne laviamo le mani.
- Vi ho già detto che dovete <u>studiare il paragrafo 2.6</u>?

Una povera macchina Universale U

- Dunque, Turing ha progettato una macchina di U che prende in input
  - una parola p_T che descrive una ***qualsiasi*** macchina di Turing T e
  - una parola x, input di T
- e che riesce a simulare la computazione T(x) – ***qualunque sia T!!!!***
- Ossia, U è l'algoritmo che descrive il comportamento di un calcolatore!
- Bella assai, quindi, questa cosa della *macchina di Turing Universale*!
- Tuttavia...
- Tuttavia, diciamocelo, questa storia dell'andare avanti e indietro sul nastro perché, nel modello progettato da Turing le testine, ad ogni passo, sanno muoversi di una sola posizione, è proprio noiosa!
- Ma perché mai a Turing non è venuto in mente di assegnare un indirizzo a ciascuna cella del nastro e definire le sue quintuple nella forma $\langle q_1 , s_1, s_2, q_2 , \mathbf{x} \rangle$,
  - che significa: se sei nello stato q_1 e leggi il simbolo s_1, allora scrivi s_2, entra nello stato q_2 e sposta la testina nella cella di indirizzo **x**
- ???? Perché mai non lo ha fatto??!!

Arricchiamola noi, la Macchina di Turing

- Ma perché mai a Turing non è venuto in mente di assegnare un indirizzo a ciascuna cella del nastro e definire le sue quintuple nella forma ⟨ q_1 , s_1, s_2, q_2 , **x**⟩ ???? Perché mai non lo ha fatto??!!
- Diciamocelo, Turing sembra averci ragionato poco sul suo modello di calcolo, per non aver pensato ad una memoria ad accesso diretto!
- Beh, ma se non lo ha fatto lui, possiamo sempre farlo noi...
- Ossia, possiamo definire un modello di calcolo quasi identico alla Macchina di Turing
- ma, nel nostro modello, ogni cella del nastro ha un indirizzo
  - e, siccome il nastro è infinito, ci sarà una cella 0, una cella 1, una cella 2, ... ma anche una cella -1, una cella -2, ...

...						...
...	-2	-1	0	1	2	...

- e le quintuple hanno la forma ⟨ q_1 , s_1, s_2, q_2 , **x**⟩
  - che significa: se sei nello stato q_1 e leggi il simbolo s_1, allora scrivi s_2, entra nello stato q_2 e sposta la testina nella cella di indirizzo **x**

Arricchiamola noi, la Macchina di Turing

- In fondo, dunque, non era così difficile progettare un modello di calcolo con ***memoria ad accesso diretto***, no?
- E adesso che abbiamo definito una simil-Macchina di Turing dotata di ***memoria ad accesso diretto***
- divertiamoci un po' a vederlo all'opera, questo nuovo modello di calcolo!
- Ad esempio, proviamo a progettare una simil-macchina di Turing che esegue lo stesso compito di T_{PAL}
- Cioè: provateci voi!
- E poi fatemi sapere!
  - E, divertitevi ...