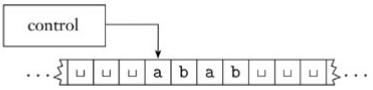
Argomenti trattati durante la lezione:

- Macchine di Turing. Introduzione ed esempi
- Esempi di Turing Machines. TM multinastro, semi-infinito, non-deterministiche ed esempi

Una macchina di Turing è un modello matematico di calcolo che descrive una macchina astratta che manipola i simboli su una striscia di nastro secondo una tabella di regole. Nonostante la semplicità del modello, è in grado di implementare qualsiasi algoritmo informatico.

La macchina funziona su un nastro di memoria infinito diviso in celle discrete, ciascuna delle quali può contenere un singolo simbolo tratto da un insieme finito di simboli chiamato alfabeto della macchina.



- un nastro infinito come memoria illimitata
- una testina che legge e scrive simboli sul nastro
- all'inizio il nastro contiene l'input
- per memorizzare informazione si scrive sul nastro
- la testina si può muovere ovunque sul nastro
- stati speciali per accetta e rifiuta

Una Macchina di Turing (o Turing Machine, TM) è una tupla $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$:

- Q è l'insieme finito di stati
- Σ è l'alfabeto di input che non contiene il simbolo blank $\sqcup$
- Γ è l'alfabeto del nastro che contiene $\sqcup$ e Σ
- $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$ è la funzione di transizione
- $q_0 \in Q$ è lo stato iniziale
- $q_{accept} \in Q$ è lo stato di accettazione
- $q_{reject} \in Q$ è lo stato di rifiuto (diverso da q_{accept})

1. Una TM può sia scrivere che leggere sul nastro
2. Una TM può muoversi sia a destra che a sinistra
3. Il nastro è infinito
4. Gli stati di rifiuto e accettazione hanno effetto immediato

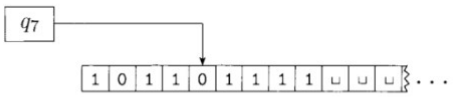


FIGURE 3.4
A Turing machine with configuration $1011q_701111$

1.1 Interpretazione della funzione di transizione

La funzione di transizione δ determina l'evoluzione della macchina in base allo stato interno attuale $q \in Q$ e al simbolo $X \in \Gamma$ letto dalla testina (quello presente nella cella del nastro attualmente visitata dalla testina):

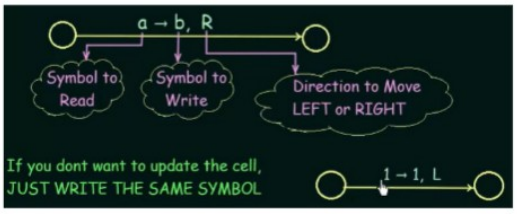
- Se $\delta(q, X)$ è indefinita, allora la macchina è **bloccata**, cioè la computazione della macchina si arresta.
- Se invece $\delta(q, X)$ è definita, e in particolare $\delta(q, X) = (p, Y, D)$, allora la macchina esegue la seguente mossa:
  - passa dallo stato interno q a p (si osservi che può essere $p = q$, dunque lo stato può non cambiare);
  - scrive il simbolo Y nella cella attualmente visitata, al posto di X (ma può essere $Y = X$);
  - sposta la testina a sinistra se $D = \mathtt{L}$, o a destra se $D = \mathtt{R}$ (in questa formalizzazione, la testina non può restare ferma).

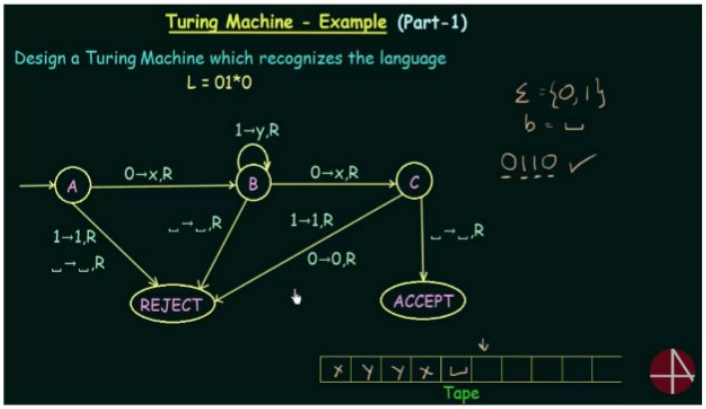
La configurazione C_1 produce C_2 se la TM può passare da C_1 a C_2 in un passo

Se $a, b, c \in \Gamma$, $u, v \in \Gamma^*$ e q_i, q_j sono stati, allora:

uaq_ibv produce uq_jacv se $\delta(q_i, b) = (q_j, c, L)$

uaq_ibv produce $uacq_jv$ se $\delta(q_i, b) = (q_j, c, R)$





- Un linguaggio è Turing-riconoscibile (o anche ricorsivamente enumerabile) se esiste una macchina di Turing che lo riconosce
- Se forniamo un input ad una TM, ci sono tre risultati possibili:
  - la macchina accetta
  - la macchina rifiuta
  - la macchina va in loop e non si ferma mai
  - una TM che termina sempre la computazione è un decisore
- Un linguaggio è Turing-decidibile (o anche ricorsivo) se esiste una macchina di Turing che lo decide

- Descrizione formale
  - Dichiara esplicitamente tutto quanto
  - Estremamente dettagliata
  - Da evitare a tutti i costi !!!
- Descrizione implementativa
  - Descrive a parole il movimento della testina e la scrittura sul nastro
  - Nessun dettaglio sugli stati
- Descrizione di alto livello
  - Descrizione a parole dell'algoritmo
  - Nessun dettaglio implementativo
  - Da utilizzare sempre, se non indicato altrimenti

1. Give an implementation-level description of a Turing machine that decides the language $B = \{ 0^n 1^n 2^n \mid n \geq 0 \}$.

M = "On input string w:

1. Scan the input from left to right to make sure that it is a member of $0^*1^*2^*$, and *reject* if it isn't.
2. Return tape head to left-hand end of tape.
3. Repeat the following until no more 0s left on tape.
4. Replace the leftmost 0 with x.
5. Scan right until a 1 occurs. If there are no 1s, *reject*.
6. Replace the leftmost 1 with x.
7. Scan right until a 2 occurs. If there are no 2s, *reject*.
8. Replace the leftmost 2 with x.
9. Return tape head to left-hand end of tape, and go to stage 3.
10. If the tape contains any 1s or 2s, *reject*. Otherwise, *accept*."

Exercise 5 : *Languages*.
Give implementation-level descriptions of Turing machines that decide the following languages:

1. $\{w \in \{a, b\}^* \mid w$ contains as many a as $b\}$;
2. $\{a^n b^n c^n \mid n \geq 0\}$;
3. $\{a^n b a^{2n} b a^{3n} \mid n \geq 0\}$.

Draw a formal-level implementation of one of those machines.
Answer:

1. M = *On input string w:*
   1. *Scan the tape and mark the first a that has not been marked. If no unmarked a has been found, go to step 3. Otherwise, bring the head back to the leftmost cell of the tape.*
   2. *Scan the tape and mark the first b that has not been marked. If no unmarked b has been found,* reject. *Otherwise, bring the head back to the leftmost cell of the tape, and go back to step 1.*
   3. *Bring the head back to the leftmost cell of the tape. Scan the tape: if there is an unmarked b,* reject*; otherwise,* accept.

2. *We use a machine with two tapes. As seen in the course, this is equivalent to a machine with one tape. The idea is to use the second tape to count n in unary.*

   $M =$ *On input string w:*

   1. *If the first cell of the first tape contains the blank character,* accept.
   2. *Scan the first tape to verify that the input is in $a^+b^+c^+$. Otherwise,* reject.
   3. *Bring the first head back to the leftmost cell of the first tape.*
   4. *Read all the a in the first tape. Each time an a is read, write a 1 in the second tape and move its head right.*
   5. *Bring the second head back to the leftmost cell of the second tape.*
   6. *Read all the b in the first tape. Each time a b is read, move the second head one step right.*
   7. *If the second head is not on a blank state,* reject *(we have less b than a). Otherwise, move the second head one step left. If it is not on a 1,* reject *(we have more b than a).*
   8. *Do steps 5-7, replacing the b by c.*
   9. Accept.

3. *We could again use a second tape to count n in unary, and then make sure that we count twice then thrice the second tape for the second and third series of consecutive a. However, another option is to use three tapes: one will count $2n$ and the other one $3n$. Again, this is equivalent to having only one tape.*

$M =$ *On input string w:*

1. *If the input is bb,* accept.
2. *Scan the first tape to verify that the input is in $a^+ba^+ba^+$. Otherwise,* reject.
3. *Bring the first head back to the leftmost cell of the first tape.*
4. *Read all the first consecutive a in the first tape (so, until you reach the first b). Each time an a is read, write a 1 in the second tape and move its head right before writing another 1 and moving again its head right; and on the third tape apply three times the following: write a 1, move the head right.*
5. *Bring the second head back to the leftmost cell of the second tape.*
6. *Read all the next consecutive a in the first tape (so, until you reach the second b). Each time an a is read, move the second head one step right.*
7. *If the second head is not on a blank state,* reject *(we have less than $2n$ a). Otherwise, move the second head one step left. If it is not on a 1,* reject *(we have more than $2n$ a).*
8. *Bring the third head back to the leftmost cell of the third tape.*
9. *Read all the final consecutive a in the first tape (so, until you reach the first blank character). Each time an a is read, move the third head one step right.*
10. *If the third head is not on a blank state,* reject *(we have less than $3n$ a). Otherwise, move the third head one step left. If it is not on a 1,* reject *(we have more than $3n$ a).*
11. Accept.

3.7 Explain why the following is not a description of a legitimate Turing machine.

M_{bad} = "On input $\langle p \rangle$, a polynomial over variables $x_1, \ldots, x_k$:

1. Try all possible settings of $x_1, \ldots, x_k$ to integer values.
2. Evaluate p on all of these settings.
3. If any of these settings evaluates to 0, *accept*; otherwise, *reject*."

Although this is quite an informal way of describing a Turing machine, I'd say the problem is one of the following:

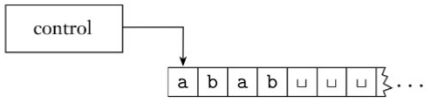
- `otherwise reject` - i agree with Welbog on that. Since you have a countably infinite set of possible settings, the machine can never know whether a setting on which it evaluates to 0 is still to come, and will loop forever if it doesn't find any - only when such a setting is encountered, the machine may stop. That last statement is useless and will never be true, unless of course you limit the machine to a finite set of integers.
- The code order: I would read this pseudocode as "first write all possible settings down, then evaluate p on each one" and there's your problem: Again, by having an infinite set of possible settings, not even the first part will ever terminate, because there never is a last setting to write down and continue with the next step. In this case, not even can the machine never say "there is no 0 setting", but it can never even start evaluating to find one. This, too, would be solved by limiting the integer set.

Anyway, i don't think the problem is the alphabet's size. You wouldn't use an infinite alphabet since your integers can be written in decimal / binary / etc, and those only use a (very) finite alphabet.

Varianti delle Macchine di Turing

Esistono definizioni alternative delle macchine di Turing. Chiamiamo varianti queste alternative e diciamo che tutte le varianti “ragionevoli” riconoscono la stessa classe di linguaggi, essendo robusti.

Macchine a nastro semi-infinito

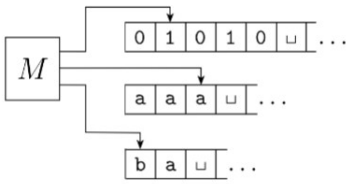


- è una TM con un nastro infinito solo verso destra
- l'input si trova all'inizio del nastro
- la testina parte dalla posizione più a sinistra del nastro
- se M tenta di spostare la testina a sinistra quando si trova nella prima cella del nastro, allora la testina rimane ferma

Theorem

1. *Per ogni TM a nastro semi-infinito esiste una TM a nastro infinito equivalente.*
2. *Per ogni TM a nastro infinito esiste una TM a nastro semi-infinito equivalente.*

Macchine Multinastro



- è una TM con k nastri semi-infiniti
- k testine di lettura e scrittura
- l'input si trova sul nastro 1
- ad ogni passo scrive e si muove simultaneamente su tutti i nastri

- funzione di transizione:

$$\delta : Q \times \Gamma^k \mapsto Q \times \Gamma^k \times \{L, R\}^k$$

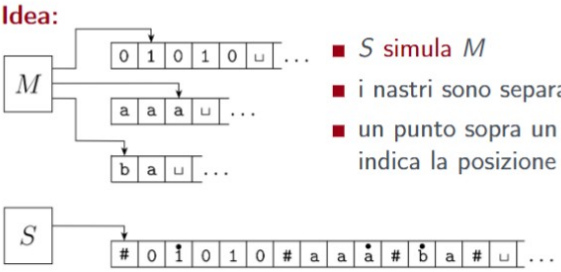
$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, L, R, \ldots, L) :$$

- se lo stato è q_i e le testine leggono $a_1, \ldots, a_k$
- allora scrivi $b_1, \ldots, b_k$ sui k nastri
- muovi ogni testina a sinistra o a destra come specificato

Theorem

Per ogni TM multinastro esiste una TM a singolo nastro equivalente.

Idea:



- S simula M
- i nastri sono separati da #
- un punto sopra un simbolo indica la posizione della testina

S = "Su input $w = w_1 \ldots w_n$:

1. Inizializza il nastro per rappresentare i k nastri:

$$\#\dot{w}_1 w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$$

2. Per simulare una mossa di M, scorri il nastro per determinare i simboli puntati dalle testine virtuali
3. Fai un secondo passaggio del nastro per aggiornare i nastri virtuali secondo la funzione di transizione di M.
4. Se S sposta una testina virtuale a destra su un #, allora M ha spostato la testina sulla parte vuota del nastro. Scrivi un $\sqcup$ e sposta il contenuto del nastro di una cella a destra
5. Se si raggiunge una configurazione di accettazione, accetta, se si raggiunge una configurazione di rifiuto, rifiuta, altrimenti ripeti da 2.

Corollary

Un linguaggio è Turing-riconoscibile se e solo se esiste una macchina di Turing multinastro che lo riconosce.

⇒ Un linguaggio è Turing-riconoscibile se è riconosciuto da una TM con un solo nastro, che è un caso particolare di TM multinastro

⇐ Costruzione precedente

Macchine non deterministiche

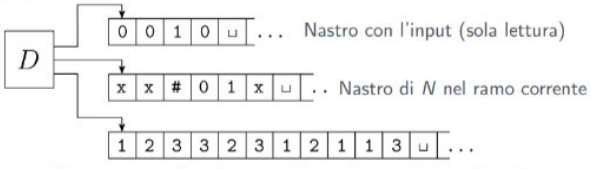
- Una TM non deterministica ha più strade possibili durante la computazione
- Consideriamo macchine con un solo nastro semi-infinito
- La funzione di transizione è:

$$\delta : Q \times \Gamma \mapsto 2^{(Q \times \Gamma \times \{L,R\})}$$

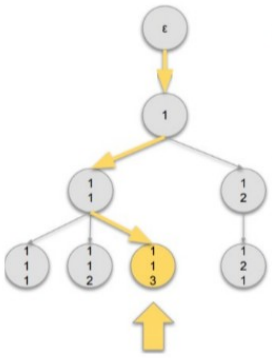
Theorem

Per ogni TM non deterministica N esiste una TM deterministica D equivalente.

Idea:



Il terzo nastro tiene traccia delle scelte non deterministiche



- Ad ogni nodo viene assegnato un indirizzo: una stringa sull'alfabeto $\Gamma_b = \{1, 2, \ldots, b\}$, dove b è il massimo numero di figli dei nodi dell'albero
- Il nodo 113 si raggiunge prendendo il primo figlio della radice, seguito dal primo figlio di quel nodo ed infine dal terzo figlio.
- Questo ordinamento può essere utilizzato per attraversare in modo efficiente l'albero in ampiezza.

1. Inizialmente il nastro 1 contiene l'input w e i nastri 2 e 3 sono vuoti
2. Copia il nastro 1 sul nastro 2 e inizializza la stringa sul nastro 3 a ε
3. Usa il nastro 2 per simulare N con input w su un ramo di computazione.
   Prima di ogni passo di N, consulta il simbolo successivo sul nastro 3 per determinare quale scelta fare (tra quelle consentite).

3. (cont.) Se non rimangono più simboli sul nastro 3, o se questa scelta non è valida, interrompi questo ramo e vai alla fase 4.
   Vai alla fase 4 anche se si incontra una configurazione di rifiuto.
   Se viene trovata una configurazione di accettazione, accetta.
4. Sostituire la stringa sul nastro 3 con la stringa successiva nell'ordine delle stringhe. Simula il ramo successivo di N andando alla fase 2.

Un linguaggio è Turing-riconoscibile se e solo se esiste una macchina di Turing non deterministica che lo riconosce.

(Come sopra: costruzione precedente)

Passiamo ora a vari esempi “reali” discussi in dettaglio.

3.12 A ***Turing machine with left reset*** is similar to an ordinary Turing machine, but the transition function has the form

$$\delta\colon Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{R}, \mathrm{RESET}\}.$$

If $\delta(q, a) = (r, b, \mathrm{RESET})$, when the machine is in state q reading an a, the machine's head jumps to the left-hand end of the tape after it writes b on the tape and enters state r. Note that these machines do not have the usual ability to move the head one symbol left. Show that Turing machines with left reset recognize the class of Turing-recognizable languages.

Solution: We can simulate a regular Turing Machine using the TM with left reset in the following way: When the regular TM would move right, it moves right; and when the regular TM would move left, it marks its current space, resets to the left, copies every space one space to the right (maintaining the position of the mark), then resets and moves to the position of the mark. The copying can be done in a single sweep, by simultaneously remembering the symbol that is being copied and the symbol that is being overwritten.

Bonus: we can also simulate the left-reset TM in the regular TM. When the left-reset TM would move right, regular TM moves right; and when the left-reset TM would reset to the left, the regular TM moves left repeatedly not changing any letters it sees until it gets to the left end of the tape.

3.11 A ***Turing machine with doubly infinite tape*** is similar to an ordinary Turing machine, but its tape is infinite to the left as well as to the right. The tape is initially filled with blanks except for the portion that contains the input. Computation is defined as usual except that the head never encounters an end to the tape as it moves leftward. Show that this type of Turing machine recognizes the class of Turing-recognizable languages.

3.12 A ***Turing machine with left reset*** is similar to an ordinary Turing machine, but the transition function has the form

$$\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{R}, \mathrm{RESET}\}.$$

If $\delta(q, a) = (r, b, \mathrm{RESET})$, when the machine is in state q reading an a, the machine's head jumps to the left-hand end of the tape after it writes b on the tape and enters state r. Note that these machines do not have the usual ability to move the head one symbol left. Show that Turing machines with left reset recognize the class of Turing-recognizable languages.

3.14 A ***queue automaton*** is like a push-down automaton except that the stack is replaced by a queue. A ***queue*** is a tape allowing symbols to be written only on the left-hand end and read only at the right-hand end. Each write operation (we'll call it a *push*) adds a symbol to the left-hand end of the queue and each read operation (we'll call it a *pull*) reads and removes a symbol at the right-hand end. As with a PDA, the input is placed on a separate read-only input tape, and the head on the input tape can move only from left to right. The input tape contains a cell with a blank symbol following the input, so that the end of the input can be detected. A queue automaton accepts its input by entering a special accept state at any time. Show that a language can be recognized by a deterministic queue automaton iff the language is Turing-recognizable.

(Si veda il file "(EN) Esempi svolti a lezione – Varianti di TM")

3. (9 punti) Dimostra che un linguaggio è decidibile se e solo se esiste un enumeratore che lo enumera seguendo l'ordinamento standard delle stringhe.

3.9 Let a k-PDA be a pushdown automaton that has k stacks. Thus a 0-PDA is an NFA and a 1-PDA is a conventional PDA. You already know that 1-PDAs are more powerful (recognize a larger class of languages) than 0-PDAs.

a. Show that 2-PDAs are more powerful than 1-PDAs.

b. Show that 3-PDAs are not more powerful than 2-PDAs.
(Hint: Simulate a Turing machine tape with two stacks.)

(Si veda il file "(EN) Esempi svolti a lezione – Potenza di PDA ed Enumeratori")

Le TM sono un mezzo potente e servono ad esprimere qualsiasi tipo di linguaggio o algoritmo:

2. (12 punti) Dati due DFA, considera il problema di determinare se esiste una stringa accettata da entrambi.

(a) Formula questo problema come un linguaggio $AGREE_{\text{DFA}}$.

(b) Dimostra che $AGREE_{\text{DFA}}$ è decidibile.

Soluzione.

(a) $AGREE_{\text{DFA}} = \{\langle A, B\rangle \mid A, B \text{ sono DFA, ed esiste una parola } w \text{ tale che } w \in L(A) \text{ e } w \in L(B)\}$

(b) La seguente macchina N usa la Turing machine M che decide E_{DFA} per decidere $AGREE_{\text{DFA}}$:

$N =$ "su input $\langle A, B\rangle$, dove A, B sono DFA:

1. Costruisci il DFA C che accetta l'intersezione dei linguaggi di A e B
2. Esegui M su input $\langle C\rangle$. Se M accetta, rifiuta, se M rifiuta, accetta."