Aniello Murano

Macchine di Turing non-deterministiche

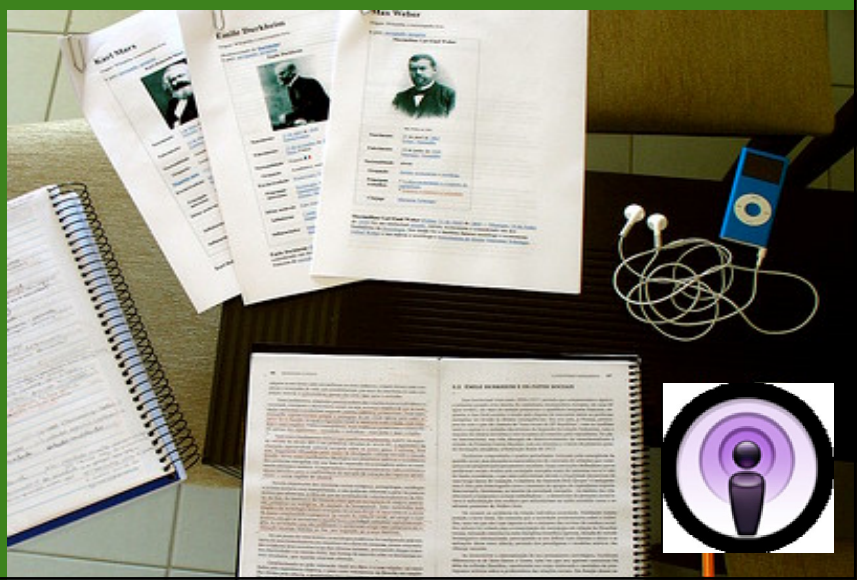
Lezione n.5
Parole chiave:
Nondeterministic Turing machine

Corso di Laurea:
Informatica

Codice:

Email Docente:
murano@ na.infn.it

A.A. 2008-2009



Riassunto delle lezioni precedenti

- Nelle lezioni precedenti abbiamo introdotto il concetto di macchina di Turing (deterministica) e abbiamo mostrato alcuni semplici linguaggi da essa accettati.
- Abbiamo mostrato alcune varianti di questa macchina e, tra le altre, la macchina di Turing multinastro.
- Abbiamo visto la macchina di Turing multinastro è equivalente alla macchina base con un trasformazione che chiede un salto al più quadratico.

Sommario di questa lezione

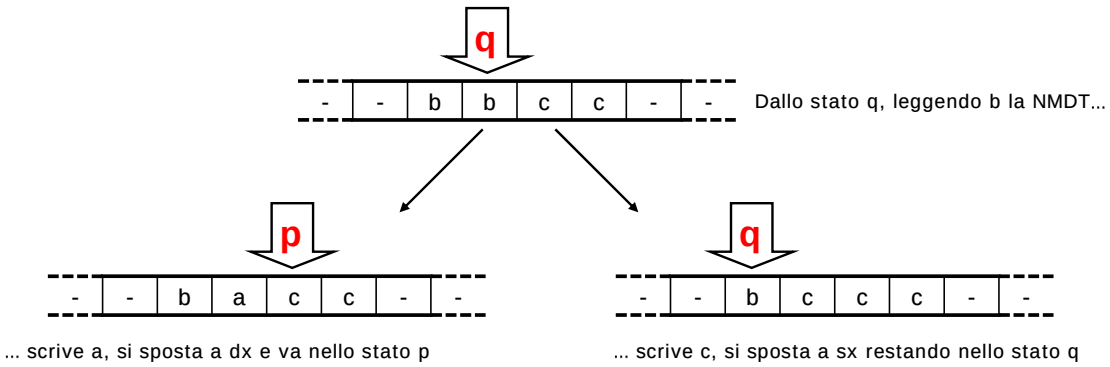
- In questa lezione parleremo di macchine di Turing non deterministiche.
- Vedremo che le macchine di Turing nondeterministiche sono equivalenti a quelle deterministiche, ma la simulazione può richiedere un numero di passi esponenziali nella taglia della prima macchina.

Motivazioni

- Con le macchine di Turing deterministiche è possibile caratterizzare la classe di complessità polinomiale P come la classe dei linguaggi che possono essere decisi da una macchina di Turing in tempo polinomiale.
- Con le macchine di Turing nondeterministiche, introdurremo le classi di complessità di tempo e spazio nondeterministiche. In particolare introdurremo la classe NP, come la classe dei linguaggi che
  - possono essere decisi in tempo esponenziale,
  - L'appartenenza di una stringa al linguaggio può essere verificata in tempo deterministico polinomiale

Macchine di Turing nondeterministiche

- In una macchina di Turing nondeterministica (NMdT), la funzione di transizione δ associa ad una coppia (stato, simbolo letto sul nastro della macchina) un insieme di (0, 1, o più) di triple (stato, nuovo simbolo da scrivere, direzione della testina)
- Per esempio, la MdT in figura può avere come configurazione successiva sia quella indicata a sinistra che quella a destra



Definizione formale

- Una macchina di Turing non deterministica (NMdT) è definita da tupla:

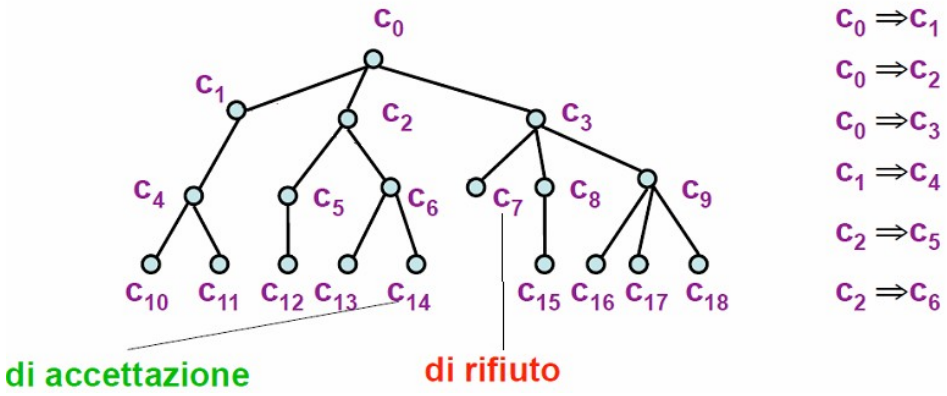
  $N = (\Sigma, \Gamma, Q, \delta, q_0, accept, reject)$

  dove **Γ** contiene i caratteri speciali "blank" (ed eventualmente lo "starting tape") e δ è la relazione di transizione definita nel seguente modo:
- **$\delta : (Q \setminus \{accept, reject\} \times \Gamma) \rightarrow P(Q \times \Sigma \times \{l, r\})$**
- Cioè, da ogni configurazione si può transire in una o più configurazioni simultaneamente.
- La configurazione successiva non è univocamente determinata. In realtà si possono avere 0, 1 o più mosse che permettono una possibile configurazione.
- La computazione non è più una successione di configurazioni ma un albero di configurazioni.
- Il grado di non determinismo corrisponde, dato una generica configurazione, al massimo numero di configurazioni generate dalla funzione .
- Una NMdT si comporta come se ad ogni passo instanziasse nuove NMdT, ognuna delle quali elabora una delle configurazione ottenute dalla funzione di transizione.
- Esercizio: esprimere formalmente cosa significa che C1→C2
- Chiaramente, si ha C1→* C2 se esiste una sequenza di mosse legali che portano da C1 a C2

Linguaggio accettato da una NMdT

- Una stringa in input è accetta da una NMdT M se dalla configurazione iniziale può raggiungere una configurazione di accettazione (cioè, una configurazione il cui stato è "accept")
- Il linguaggio L(M) accettato da una NMdT M è l'insieme delle stringhe in input che sono accettate da M
  - $x \in L(M)$ significa che c'è **una** computazione di M sulla stringa x che porta ad uno stato accept. Si noti che esiste anche la possibilità in questo caso che altre computazioni su x non portano a nessuno stato (interruzione della computazione per mancanza di mosse possibili) oppure ci sono computazioni che continuano per sempre.
  - $x \notin L(M)$ significa che **nessuna** computazione di M su x porta ad uno stato "accept", ovvero che **tutte** le computazioni sono non accettanti.
- Osservazioni
  - Si osservi che i due concetti sono molto "asimmetrici" tra di loro: il primo richiede che esista almeno una computazione accettante, quindi se abbiamo a disposizione una soluzione è facile certificarla come tale.
  - Viceversa una stringa è rifiutata se tutte le possibili computazioni sono rifiutanti e questo può essere molto più difficile da verificare.

Esempio di albero di computazione



Esempio: Problema del partizionamento

- Input: Una sequenza di interi $a_1,\ldots,a_m$ (in binary)
- Problema: Possiamo scrivere una loro partizione in due insiemi, tale che la somma degli elementi presenti nel primo insieme sia uguale alla somma degli elementi presenti nel secondo?
- Per la soluzione di questo problema possiamo usare una NMdT a 3 nastri.
  1. Il primo nastro contiene la sequenza di input
  2. Il secondo e il terzo nastro sono inizializzati con il valore 0.
  2. Si legge il nastro di input e per ogni numero a_i si scegli nondeterministicamente di aggiungere il numero al secondo o al terzo nastro.
  3. Al passo n-esimo si confrontano i numeri sul secondo e il terzo nastro. Se uguali si accetta la stringa di input altrimenti no.
- Nota: il numero di operazioni dell'algoritmo è proporzionale all'input e all'operazione di somma.

Esempio: Il problema del commesso viaggiatore

- **Input:** una rete di città, connesse tramite delle strade di cui si conosce la lunghezza, e un valore intero B.
- **Problema:** trovare il percorso che un commesso viaggiatore deve seguire per visitare tutte le città una e una sola volta, con distanza minore di B.
- Espresso nei termini della teoria dei grafi è così formulato: dato un grafo completo pesato, trovare il ciclo hamiltoniano con peso minore. La rete di città può essere rappresentata come un grafo in cui le città sono i nodi, le strade gli archi e le distanze i pesi sugli archi.
- Applicazioni pratiche del problema: logistica, trasporti, costruzione di circuiti stampati, ecc.
- Non esistono algoritmi efficienti per la risoluzione del problema. L'unico metodo di risoluzione è rappresentato dall'enumerazione totale, ovvero nell'elaborazione di tutti i possibili cammini sul grafo per la successiva scelta di quello migliore che ha tempo esponenziale: numero esponenziale di cammini possibili.
- Usando una NMdT con 3 nastri è possibile risolvere questo problema in un numero di passi polinomiale (quadratico) nella taglia dell'input:
  1. Il primo nastro contiene una rappresentazione del grafo in input e il secondo il valore B
  2. Sul terzo nastro scriviamo nondeterministicamente un percorso.
  3. Si controllano che, il contenuto del terzo nastro sia realmente un percorso, che sia aciclico e completo, e che il suo valore sia minore di B.
  4. Alla fine del percorso, se B è diventato 0 si accetta, altrimenti no.

Rappresentazione di un grafo

- Sia G un Grafo diretto.
- Assumiamo per semplicità che i nodi siano {1,…,n}.
- I nodi possono essere codificati in binario
- Analogamente gli archi possono essere rappresentati da una coppia binaria;
- Il grafo può così essere rappresentato dalla seguente stringa:

  #nodo1#nodo2#...###nodo1#nodo2##nodo2#nodo3##... ###

Esercizio

- Riconoscere con una MdTN il seguente linguaggio:

  $L = \{W1\#W2 \mid W1 \text{ sottoinsieme di } W2,\ W1,W2 \in \{0,1\}^*\}$

Nondeterminismo vs determinismo (1)

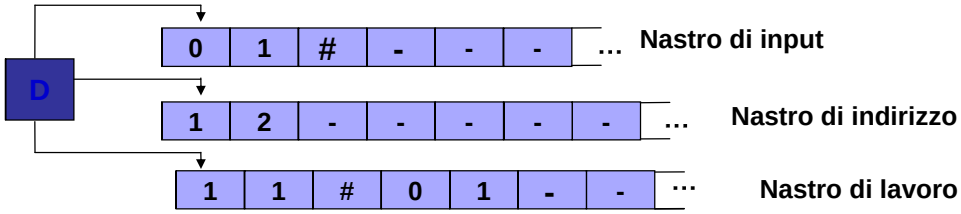
Teorema

- Per ogni NMdT M esiste una MdT (deterministica) $M^{(3)}$ a 3 nastri equivalente.
- La simulazione di M tramite $M^{(3)}$ si ottiene visitando l'albero delle computazione di M utilizzando l'algoritmo di **visita in ampiezza** degli alberi.
- **Nota:** Non si può usare la **visita in profondità** poiché si potrebbe visitare un ramo corrispondente ad una computazione infinita.
- Ad ogni passo di computazione di M si possono generare un numero massimo di scelte **d** pari al grado di non determinismo della macchina M.
- Supponiamo di numerare con numeri compresi tra 1 e d le scelte derivanti dalla funzione di transizione di M.
- In tal modo ogni computazione potrà essere identificata come una sequenza di numeri compresi tra 1 e d, ognuno dei quali identifica una delle possibili d scelte generate dalla funzione di transizione di M.
- Non tutte le combinazioni saranno valide poiché non è detto che ad ogni passo la funzione di transizione generi esattamente d scelte.
- Dopo "i" passi di computazione della macchina M, esistono al più d^i stringhe di lunghezza "i" che rappresentano particolari computazioni di M.

Nondeterminismo vs determinismo (2)

- La macchina $M^{(3)}$ è organizzata nel seguente modo:
  - primo nastro contiene la stringa di input
  - il secondo nastro contiene, in ordine lessicografico, **tutte** le sequenze finite composte da cifre comprese tra 1 e d. Ogni sequenza di lunghezza s sul secondo nastro è in corrispondenza con una computazione di M di **s** passi. Dunque per s passi avremo al più $\mathbf{d}^{\mathbf{s}}$ stringhe sul secondo nastro.
  - il terzo nastro eseguirà la simulazione vera e propria
- La simulazione avviene nel seguente modo:
  1. Al passo s-esimo di computazione di M, si generano sul secondo nastro stringhe di lunghezza s corrispondenti a possibili sequenze di scelte (nondeterministiche) per computazioni di lunghezza s
  2. per ogni sequenza di lunghezza s
     a. si copia il contenuto del primo nastro sul terzo nastro
     b. si scandisce il secondo nastro, e per ogni j corrispondente all'indice di una possibile scelta si applica la j-esima scelta al terzo nastro
- es. se s=4 e d=2 e la sequenza è 2112, $M^{(3)}$ sceglie per la prima mossa la seconda transizione disponibile, per la seconda mossa la prima, ecc....
- Se esiste un cammino di lunghezza s che porta M in uno stato finale, esiste sicuramente una fase di calcolo di $M^{(3)}$ che percorre tale cammino (in tempo finito).
- Se viceversa tale cammino non esiste allora anche $M^{(3)}$ non raggiungerà mai lo stato finale.

Passi della simulazione



- Tutte le regole di transizione si numerano da 1 a d.
- Visita livello 1:
  - L'input è sul nastro 1. Si genera la stringa di indirizzo sul nastro 2 e si copia l'input sul nastro 3.
  - Per la sequenza 1, c'è solo una configurazione possibile: quella iniziale. Si copia la stringa dal nastro 1 al nastro 3.
- Visita livello 2:
  - Si generano in sequenza le stringhe di lunghezza 2 sul nastro 2, per esempio 11 e 12.
  - Per ogni stringa prodotta, si copia l'input dal primo al terzo nastro e si modifica in base all'applicazione della giusta regola di transizione (prima la 1 e poi la 2). Poi si ricopia il tutto nel nastro 1
  - La stringa sul nastro 1 corrispondente alle configurazioni successive possibile della NMdT, separate da #
- Visita livello 3:
  - Si generano in sequenza le stringhe di lunghezza 3 sul nastro 2, per esempio 111, 112, 121 e 122.
  - Per ogni stringa prodotta, si copia la configurazione da modificare dal primo al terzo nastro (per 111 e 112 fino al # e poi per 121 e 122 la parte dopo #) e si modifica in base alla regola di transizione scelta.
  - La stringa sul nastro 3 corrisponde a quattro configurazioni della NMdT, tutte separate da #. Si sposta questa stringa dal terzo al primo nastro e si passa al livello successivo ….

Nondetermismo vs determinismo (3)

- Ad ogni passo j della computazione di M, la macchina deterministica $M^{(3)}$ deve compiere esattamente jd^j passi, cioè la lunghezza del cammino per il numero dei cammini.
- Se M termina in k passi allora $M^{(3)}$ esegue al più in un numero di passi pari al più a

$$\sum_{j=0}^{k} j\, d^j$$

- Che è esponenziale in k.
- Quindi una macchina di Turing non deterministica può essere simulata da una macchina di Turing deterministica multi-nastro in un tempo esponenziale.

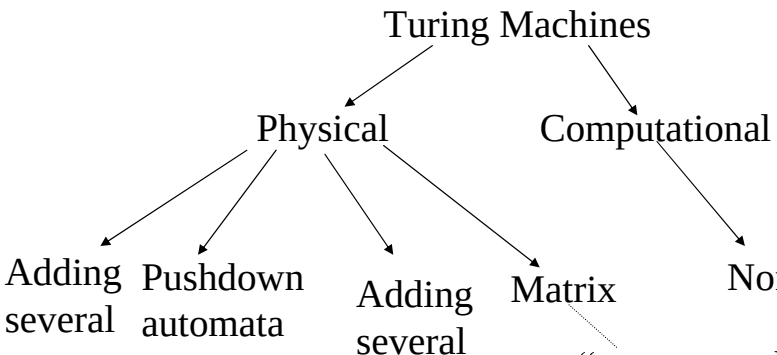
Osservazione

- Una macchina non deterministica può "risolvere" un problema (ad esempio, accettare un linguaggio) in tempo polinomiale rispetto alla lunghezza della stringa mentre la simulazione effettuata da una macchina deterministica richiede tempo esponenziale.
- **Problema aperto.** Esiste una simulazione più efficiente? E' possibile simulare una macchina di Turing non deterministica con una deterministica in tempo polinomiale?

Macchine di Turing Multinastro nondeterminstiche

- E' chiaramente possibile l'aggiunta simultanea di più nastri e del nondeterminismo alla definizione di base di una MdT.
- La macchina ottenuta ha sicuramente una rappresentazione più succinta della macchina classica, ma non ha un maggiore potere computazionale:
  - E' possibile trasformare una MdT multinastro non deterministica in una mono-nastro, applicando la trasformazione vista per le MdT deterministiche;
  - Poi si può rimuovere il nondeterminismo così come abbiamo visto nelle slide precedenti;
  - Le due trasformazioni richiedono un numero di passi esponenziali rispetto alla taglia della macchina di partenza.

Possible Extensions of Turing Machines



"no one can be told what the Matrix is, you have to see it for your self"

None of these extensions add *computational power* to the Turing Machines