Calcolabilità e complessità (corso B) - Laurea in Informatica - a. a. 2021-2022
Prova scritta (2 ore) – 15 luglio 2022

1. Dimostrare che la classe dei linguaggi regolari è chiusa rispetto all'operazione di unione. (senza usare automi non deterministici - 4 punti).
2. Dato il linguaggio { ww | w in {0,1}* }, dimostrare che non è regolare (4 punti).
3. Fornire un esempio di linguaggio non Turing-riconoscibile e dimostrare perchè (4 punti).
4. Dimostrare che SAT è NP-completo (4 punti).
5. Un triangolo in un grafo non orientato è una clique di dimensione 3. Dimostrare che TRIANGLE è in P, con TRIANGLE = {<G> | G contiene un triangolo} (4 punti).
6. Scrivere il DFA che accetta il linguaggio {w in {0,1}* | w non contiene mai più di due 0 consecutivi} (4 punti).

1) Supponiamo che M_1 riconosca A_1, dove $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$, e che M_2 riconosca A_2, dove $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$. Costruiamo quindi $M = (Q, \Sigma, \delta, q_0, F)$ che riconosce $A_1 \cup A_2$.

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
   Questo è il prodotto cartesiano degli insiemi Q_1 e Q_2 ed è l'insieme di tutte le coppie di stati, il primo in Q_1 e il secondo in Q_2.
2. Σ, l'alfabeto è lo stesso sia per M_1 che per M_2 assumendo, per semplicità, che abbiano uguale alfabeto.
3. δ, la funzione di transizione è definita come segue. Per ogni $r_1, r_2 \in Q$ e ogni $a \in \Sigma$, sia $\delta((r_1, r_2)\ a) = (\delta_1(r_1, a), \delta_2(r_2, a))$. Quindi δ riceve uno stato di M, con un simbolo di input, e restituisce lo stato successivo di M.
4. q_0 è la coppia (q_1, q_2).
5. F è l'insieme delle coppie in cui l'uno o l'altro elemento è uno stato accettante di M_1 o di M_2. Possiamo scriverlo come $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$. Questa espressione è uguale a $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

Questo conclude la costruzione dell'automa finito M che riconosce l'unione di A_1 e A_2.

2) Dobbiamo dimostrare con il pumping lemma per i linguaggi regolari che il linguaggio A = {ww | w $\in$ {0, 1}} non è regolare.

Il pumping lemma dice che se A è un linguaggio regolare allora esiste un numero p tale che se s è una qualsiasi stringa in A di lunghezza almeno p, allora s può essere divisa in tre parti, s = xyz, soddisfacenti le seguenti condizioni:

1. $\forall i \geq 0, xy^iz \in A$;
2. $|y| > 0$;
3. $|xy| \leq p$.

Assumiamo quindi per assurdo che A sia regolare. Sia p la lunghezza del pumping data dal pumping lemma. Sia s la stringa 0^p10^p1. Poiché s è un elemento di A ed s ha lunghezza maggiore di p, il pumping lemma assicura che s può essere divisa in tre parti, s = xyz, che verificano le tre condizioni del lemma. Mostriamo che questo risultato è impossibile considerando quattro casi distinti.

1. La stringa y contiene solo 0 allora ci sarà un numero troppo elevato di 0 e la stringa risultante xyyz non farà parte di A perché non divisibile in due stringhe uguali;
2. La stringa y contiene 1 (ponendo z uguale a stringa vuota) e in questo caso ci sarà un 1 in più per ogni volta che si pompa la y, facendo sì che anche in questo caso la stringa risultante xyyz non faccia parte di A;
3. La stringa y contiene 1 seguito da p 0 e in questo caso la stringa risultante xyyz farà parte di A ma continuando ad iterare ci si accorgerà che xyyyz non fa parte di A, deducendo che i deve essere necessariamente dispari.
4. La stringa y è uguale a 0^p10^p1 (ponendo x e z stringhe vuote) osservando che effettivamente $\forall i \geq 0, xy^iz \in A$ però questo viola la condizione numero 3.

Osservando la condizione 3 ci si accorge che ci saremmo potuti fermare direttamente al primo caso, visto che è l'unico a non violarla. Quindi il linguaggio A non è regolare.

3) Un esempio di linguaggio non Turing-riconoscibile è il complemento del linguaggio A_{TM} = {(M, w), M è una mdT e che accetta w}.

A_{TM} è Turing-riconoscibile visto che è riconosciuto dalla seguente mdT:
"Su input <M, w>, dove M è una TM e w è una stringa:

1. Simula M su input w;
2. Se durante la computazione M entra nello stato di accettazione, accetta; se M entra nello stato di rifiuto, rifiuta."

Essendo Turing-riconoscibile ed essendo non decidibile, il suo complemento è necessariamente non Turing-riconoscibile.

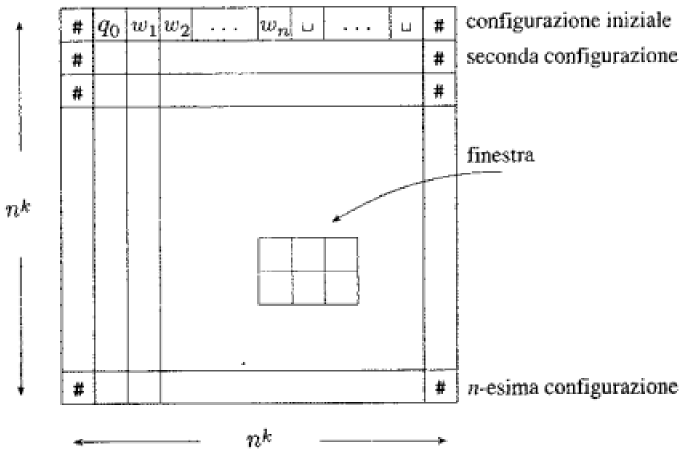
4) Il teorema di Cook-Levine dice che SAT è un linguaggio NP-completo.

Per dimostrarlo deve risultare che:

- SAT è in NP;
- Devo ridurre tutti i linguaggi al problema SAT in tempo polinomiale.

Allora:

- SAT è in NP sia con il verificatore che in maniera non deterministica; infatti, con il verificatore la stringa c rappresentava i valori di verità che rendevano la formula vera;
- Sia dato un linguaggio L $\in$ NP.
  Devo definire una riduzione polinomiale a SAT.
  Dato che L $\in$ NP, allora esiste una mdT non deterministica che decide L in tempo polinomiale. Per ogni stringa w costruisco in tempo polinomiale un'espressione booleana tale che se w $\in$ L allora l'espressione è vera, quindi: ϕ(M, w) è soddisfacibile se w $\in$ L e non soddisfacibile se w $\notin$ L.
  Adesso vediamo come sono le computazioni della macchina non deterministica M.
  Dato che M non è deterministica ci saranno molte computazioni e dato che L $\in$ NP, allora l'albero delle computazioni avrà profondità al massimo n^k, dove n è la lunghezza della stringa in input.
  Adesso considero una computazione accettante che ovviamente sarà al massimo n^k e l'input occuperà al massimo n^k celle di memoria a sinistra; quindi, la massima area di calcolo sul nastro sarà $2n^k$.
  Adesso l'albero delle computazioni si può vedere sottoforma di una matrice chiamata tableaux delle configurazioni.



Quindi l'alfabeto del tableau sarà: C = {{#} $\cup$ {insieme degli stati} $\cup$ {alfabeto del nastro}}.
Per ogni cella con posizione i, j e per ogni simbolo dell'alfabeto $s \in C$, creo la variabile $x_{i, j, s}$ in cui se nella posizione i e j vi è il simbolo s allora $x_{i, j, s} = 1$, altrimenti è uguale a 0.
Quindi $\phi(M, w)$ è costruita mediante la variabile $x_{i, j, s}$.
$\phi(M, w) = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$

ϕ_{cell} indica che ogni cella del tableau delle configurazioni contiene uno e un solo simbolo. Pertanto, per tutte le coppie di i e j esiste almeno un simbolo in ogni cella AND per tutti gli s e t, simboli dell'alfabeto, con $s \neq t$ deve succedere che o è presente s o è presente t, quindi in ogni cella vi è solo un simbolo e vi è la certezza che questo simbolo c'è. Ha complessità $O(n^{2k})$.
Per quanto riguarda la dimensione si ha $2n^k + 3$ perché la massima area di calcolo è pari a $2n^k$, poi vi sono due #, uno all'inizio della riga e uno alla fine della riga e in più vi è lo stato, quindi $2n^k + 3$.

$$\phi_{\text{cell}} = \bigwedge_{1 \leq i,j \leq n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} \left(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}} \right) \right) \right].$$

ϕ_{start} indica come è formata la prima riga del tableau, quindi come inizia la computazione. La ϕ_{start} ha dimensione $2n^k + 3$, in quanto lo spazio di ogni singola riga è pari a $2n^k + 3$ e, quindi, ha complessità pari a $O(n^k)$.

$$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge$$
$$x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge$$
$$x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}\,.$$

ϕ_{accept} dà certezza che la computazione raggiunge uno stato di accettazione. La ϕ_{accept} ha dimensione $(2n^k + 3)^2$, quindi ha complessità pari a $O(n^{2k})$.

$$\phi_{\text{accept}} = \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{\text{accept}}}\,.$$

ϕ_{move} dà la sicurezza che ci sono computazioni valide ed è espressa mediante windows legali. Quindi la ϕ_{move} è pari a tutte le finestre(i, j) legali.
La dimensione di una window legale è pari a 6, il numero di possibili windows legali è pari a $|C|^6$ e il numero di possibili celle è dato dal numero di righe per il numero di colonne del tableau delle configurazioni, quindi $(2n^k + 3) * n^k$; quindi, risulta che la ϕ_{move} ha complessità pari a $O(n^{2k})$.

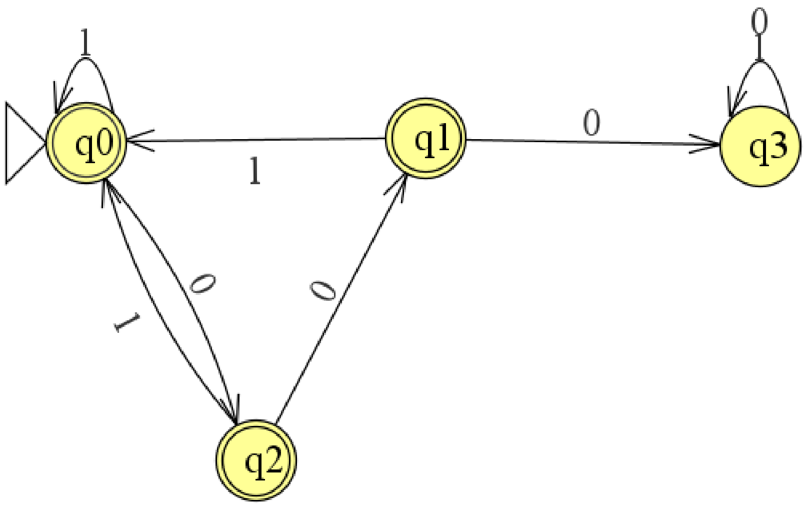
$$\phi_{\text{move}} = \bigwedge_{1 \le i < n^k,\ 1 < j < n^k} \big(\text{la finestra } (i,j) \text{ è lecita}\big).$$

Risulta che la complessità di $\phi(M, w) = O(n^{2k}) + O(n^k) + O(n^{2k}) + O(n^{2k}) = O(n^{2k})$.

Quindi vi è complessità polinomiale in n e pertanto, data una mdT non deterministica, costruisco una formula $\phi(M, w)$ e ho che se $w \in L$ allora la formula risulta soddisfacibile.

5) TRIANGLE è in P perché può essere deciso in tempo polinomiale. Si può utilizzare un algoritmo che, con input <G>, esamina tutte le terne di nodi. Se $n = |V|$, quindi n uguale al numero dei vertici, allora ci sono $\frac{n(n-1)(n-2)}{3 \cdot 2 \cdot 1}$ triple. Per ciascuna di queste triple, controlla se tutti e tre i bordi sono collegati, cioè formano un triangolo. Appena viene trovato un triangolo, accetta. Se alla fine non viene trovato nessun triangolo, rifiutare. L'algoritmo richiede passi $O(n^3)$, quindi è polinomiale.

6)


q0
q1
q2
q3
1
1
0
0
1
0
0
0