

- 1) Costruire un PDA che riconosca per stack vuoto il linguaggio $L = \{w \mid w \text{ in } \{0,1\}^* \text{ e tale che contenga lo stesso numero di 0 e di 1}\}$.

Il PDA ha un solo stato q . Per semplicità chiameremo Z il simbolo del fondo dello stack. Le transizioni sono:

$\delta(q, 0, Z) = (q, 0Z)$, $\delta(q, 0, 0) = (q, 00)$, $\delta(q, 1, Z) = (q, 0Z)$, $\delta(q, 1, 0) = (q, \epsilon)$, $\delta(q, 0, 1) = (q, \epsilon)$, $\delta(q, \epsilon, Z) = (q, \epsilon)$.

L'ultima transizione è quella che toglie il fondo dello stack Z e quindi blocca l'automa. L'automa con stack vuoto accetta solo se l'input è completamente consumato.

- 2) Il PDA del punto (1) è deterministico o no? Spiegare la risposta. In caso fosse non deterministico cercate di renderlo deterministico oppure spiegate perché non è possibile farlo.

Il PDA del punto (1) è nondeterministico a causa del fatto che sono presenti le transizioni: $\delta(q, 0, Z) = (q, 0Z)$ e $\delta(q, \epsilon, Z) = (q, \epsilon)$ che sono entrambe applicabili nella situazione in cui il PDA è nello stato q e con Z in cima alla pila. Intuitivamente la prima corrisponde alla scelta di continuare a consumare l'input, mentre la seconda corrisponde alla scelta di assumere che l'input sia vuoto e di svuotare lo stack fermando il calcolo. Non è possibile trasformare l'automa in modo che diventi deterministico mantenendo l'accettazione per stack vuoto. Infatti L non ha la proprietà del prefisso (per esempio 01 e 0101 sono entrambe stringhe di L) e abbiamo dimostrato in classe, che i DPDA che accettano per stack vuoto riconoscono solo linguaggi che godono di questa proprietà.

- 3) Per affrontare l'argomento dell'indcidibilità, è indispensabile numerare da 1 in su tutte le stringhe binarie. Qual è il numero progressivo della stringa 001 ?

Ogni stringa binaria w ha un numero progressivo che è $1w$. Quindi per 001 , il numero progressivo è 1001 che è 9 in base 10.

- 4) Cos'è un linguaggio ricorsivo? Cos'è un linguaggio RE? Un linguaggio RE è certamente non ricorsivo? Spiegare la differenza tra ricorsivo e RE.

Un linguaggio (binario) L è detto ricorsivo se esiste una TM M che per ogni w binario si ferma in uno stato finale se w è in L e in uno stato non finale se w non è in L . Una tale M definisce un algoritmo per L .

Un linguaggio (binario) L è detto RE se esiste una TM M che per ogni w si ferma in uno stato finale se w è in L , mentre se w non è in L può fermarsi in uno stato non finale o continuare il calcolo per sempre. I linguaggi RE contengono propriamente i linguaggi ricorsivi. Un esempio di linguaggio RE che non sia ricorsivo è il linguaggio universale $L_u = \{(M, w) \mid M \text{ è la codifica binaria di una TM e } w \text{ una stringa binaria t. c. } w \text{ è in } L(M)\}$

5) i) Definire il linguaggio L_d e spiegare perché non è RE.

$L_d = \{w_i \mid w_i \text{ non è in } L(M_i)\}$ M_i è la codifica binaria di una TM e w_i è questa codifica. Quindi M_i e w_i sono identiche stringhe binarie, ma la prima è interpretata come TM e la seconda come input.

Che L_d non sia RE lo si dimostra per assurdo nel modo seguente:

assumiamo che esista una TM che riconosca L_d . Sia questa TM, M_i che, interpretata come stringa binaria, è w_i . Quindi assumiamo che $L(M_i) = L_d$. Allora w_i non può né essere in $L(M_i) = L_d$ né può non essere in $L(M_i) = L_d$, infatti,

a) supponiamo che w_i sia in $L(M_i)$ allora, per la def. di L_d , w_i non può essere in $L_d = L(M_i) \Rightarrow$ assurdo

b) supponiamo che w_i non sia in $L(M_i)$ allora, per def. di L_d , w_i deve essere in $L_d = L(M_i) \Rightarrow$ assurdo

da questi assurdi deriva che non esiste alcuna TM M_i t.c. $L(M_i) = L_d$.

ii) Spiegare perché il complemento di L_d non può essere ricorsivo.

Visto che sappiamo che, se L è ricorsivo, allora $\text{comp}(L)$ è pure ricorsivo, se $\text{comp}(L_d)$ fosse ricorsivo, anche L_d lo sarebbe, ma abbiamo appena visto che L_d non è RE e quindi, a maggior ragione, non è nemmeno ricorsivo (i ricorsivi sono contenuti in RE).