Automi e Linguaggi Formali
Soluzione della prima prova intermedia

1. Per risolvere l'esercizio dobbiamo dimostrare che *(a)* ogni linguaggio riconosciuto da una TM con reset a sinistra è Turing-riconoscibile e *(b)* ogni linguaggio Turing-riconoscibile è riconosciuto da una TM con reset a sinistra.

(a) Mostriamo come convertire una TM con reset a sinistra M in una TM standard S equivalente. S simula il comportamento di M nel modo seguente. Se la mossa da simulare prevede uno spostamento a destra, allora S esegue direttamente la mossa. Se la mossa prevede un $RESET$, allora S scrive il nuovo simbolo sul nastro, poi scorre il nastro a sinistra finché non trova il simbolo $\triangleright$, e riprende la simulazione dall'inizio del nastro. Per ogni stato q di M, S possiede uno stato q_{RESET} che serve per simulare il reset e riprendere la simulazione dallo stato corretto.

S = "Su input w:

1. scrive il simbolo $\triangleright$ subito prima dell'input, in modo che il nastro contenga $\triangleright w$.
2. Se la mossa da simulare è $\delta(q, a) = (r, b, R)$, allora S la esegue direttamente: scrive b sul nastro, muove la testina a destra e va nello stato r.
3. Se la mossa da simulare è $\delta(q, a) = (r, b, RESET)$, allora S esegue le seguenti operazioni: scrive b sul nastro, poi muove la testina a sinistra e va nello stato r_{RESET}. La macchina rimane nello stato r_{RESET} e continua a muovere la testina a sinistra finché non trova il simbolo $\triangleright$. A quel punto la macchina sposta la testina un'ultima volta a sinistra, poi di una cella a destra per tornare sopra al simbolo di fine nastro. La computazione riprende dallo stato r.
4. Se non sei nello stato di accettazione o di rifiuto, ripeti da 2."

(b) Mostriamo come convertire una TM standard S in una TM con reset a sinistra M equivalente. M simula il comportamento di S nel modo seguente. Se la mossa da simulare prevede uno spostamento a destra, allora M può eseguire direttamente la mossa. Se la mossa da simulare prevede uno spostamento a sinistra, allora M simula la mossa come descritto dall'algoritmo seguente. L'algoritmo usa un nuovo simbolo $\triangleleft$ per identificare la fine della porzione di nastro usata fino a quel momento, e può marcare le celle del nastro ponendo un punto al di sopra di un simbolo.

M = "Su input w:

1. Scrive il simbolo $\triangleleft$ subito dopo l'input, per marcare la fine della porzione di nastro utilizzata. Il nastro contiene $\triangleright w \triangleleft$.
2. Simula il comportamento di S. Se la mossa da simulare è $\delta(q, a) = (r, b, R)$, allora M la esegue direttamente: scrive b sul nastro, muove la testina a destra e va nello stato r. Se muovendosi a destra la macchina si sposta sulla cella che contiene $\triangleleft$, allora questo significa che S ha spostato la testina sulla parte di nastro vuota non usata in precedenza. Quindi M scrive un simbolo blank marcato su questa cella, sposta $\triangleleft$ di una cella a destra, e fa un reset a sinistra. Dopo il reset si muove a destra fino al blank marcato, e prosegue con la simulazione mossa successiva.
3. Se la mossa da simulare è $\delta(q, a) = (r, b, L)$, allora S esegue le seguenti operazioni:
   - 3.1 scrive b sul nastro, marcandolo con un punto, poi fa un reset a sinistra
   - 3.2 Se il simbolo subito dopo $\triangleright$ è già marcato, allora vuol dire che S ha spostato la testina sulla parte vuota di sinistra del nastro. Quindi M scrive un blank e sposta il contenuto del nastro di una cella a destra finché non trova il simbolo di fine nastro $\triangleleft$. Fa un reset a sinistra e prosegue con la simulazione della prossima mossa dal nuovo blank posto subito dopo l'inizio del nastro. Se il simbolo subito dopo $\triangleright$ non è marcato, lo marca, resetta a sinistra e prosegue con i passi successivi.
   - 3.3 Si muove a destra fino al primo simbolo marcato, e poi a destra di nuovo.
   - 3.4 se la cella in cui si trova è marcata, allora è la cella da cui è partita la simulazione. Toglie la marcatura e resetta. Si muove a destra finché non trova una cella marcata. Questa cella è quella immediatamente precedente la cella di partenza, e la simulazione della mossa è terminata
   - 3.5 se la cella in cui si trova non è marcata, la marca, resetta, si muove a destra finché non trova una marcatura, cancella la marcatura e riprende da 3.3.
4. Se non sei nello stato di accettazione o di rifiuto, ripeti da 2."

2. (a) Dimostriamo separatamente i due versi del se e solo se.

- Supponiamo che A sia Turing-riconoscibile. Allora esiste una Macchina di Turing M che riconosce A. Consideriamo la funzione f tale che $f(w) = \langle M, w \rangle$ per ogni stringa $w \in \Sigma^*$. Questa funzione è calcolabile ed è una funzione di riduzione da A a A_{TM}. Infatti, se $w \in A$ allora anche $\langle M, w \rangle \in A_{TM}$ perché la macchina M accetta le stringhe che appartengono ad A. Viceversa, se $w \notin A$, allora $\langle M, w \rangle \notin A_{TM}$ perché la macchina M rifiuta le stringhe che non appartengono ad A.
- Supponiamo che $A \leq_m A_{TM}$. Sappiamo che A_{TM} è un linguaggio Turing-riconoscibile. Per le proprietà delle riduzioni mediante funzione, possiamo concludere che anche A è Turing-riconoscibile.

(b) Dimostriamo separatamente i due versi del se e solo se.

- Supponiamo che A sia decidibile. Allora esiste una Macchina di Turing M che decide A. Consideriamo la funzione f definita nel modo seguente:

$$f(w) = \begin{cases} 01 & \text{se } M \text{ accetta } w \\ 10 & \text{se } M \text{ rifiuta } w \end{cases}$$

  M è un decisore e la sua computazione termina sempre. Quindi la funzione f può essere calcolata dalla seguente macchina di Turing:

  $F =$ "su input w:
  1. Esegui M su input w.
  2. Se M accetta, restituisci 01, se M rifiuta, restituisci 10."

  f è anche funzione di riduzione da A a 0^*1^*. Infatti, se $w \in A$ allora $f(w) = 01$ che appartiene al linguaggio 0^*1^*. Viceversa, se $w \notin A$, allora $f(w) = 10$ che non appartiene a 0^*1^*.
- Supponiamo che $A \leq_m 0^*1^*$. Sappiamo che 0^*1^* è un linguaggio decidibile. Per le proprietà delle riduzioni mediante funzione, possiamo concludere che anche A è decidibile.

3. Per mostrare che *LPATH* è NP-completo, dimostriamo prima che *LPATH* è in NP, e poi che è NP-Hard.

- *LPATH* è in NP. Il cammino da s a t di lunghezza maggiore o uguale a k è il certificato. Il seguente algoritmo è un verificatore per *LPATH*:

  V = "Su input $\langle\langle G, s, t, k\rangle, c\rangle$:
  1. Controlla che c sia una sequenza di vertici di G, $v_1, \ldots, v_m$ e che m sia minore o uguale al numero di vertici in G più uno. Se non lo è, rifiuta.
  2. Se la sequenza è di lunghezza minore o uguale a k, rifiuta.
  3. Controlla se $s = v_1$ e $t = v_m$. Se una delle due è falsa, rifiuta.
  4. Controlla se ci sono ripetizioni nella sequenza. Se ne trova una diversa da $v_1 = v_m$, rifiuta.
  5. Per ogni i tra 1 e $m - 1$, controlla se (p_i, p_{i+1}) è un arco di G. Se non lo è rifiuta.
  6. Se tutti i test sono stati superati, accetta."

  Per analizzare questo algoritmo e dimostrare che viene eseguito in tempo polinomiale, esaminiamo ogni sua fase. Ognuna delle fasi è un controllo sugli m elementi del certificato, e quindi richiede un tempo polinomiale rispetto ad m. Poiché l'algoritmo rifiuta immediatamente quando m è maggiore del numero di vertici di G più uno, allora possiamo concludere che l'algoritmo richiede un tempo polinomiale rispetto al numero di vertici del grafo.

- Dimostriamo che *LPATH* è NP-Hard per riduzione polinomiale da *HAMILTON* a *LPATH*. La funzione di riduzione polinomiale f prende in input un grafo $\langle G\rangle$ e produce come output la quadrupla $\langle G, s, s, n\rangle$ dove s è un vertice arbitrario di G e n è uguale al numero di vertici di G. Dimostriamo che la riduzione polinomiale è corretta:
  - Se $\langle G\rangle \in HAMILTON$, allora esiste un circuito Hamiltoniano in G. Dato un qualsiasi vertice s di G, possiamo costruire un cammino che parte da s e segue il circuito Hamiltoniano per tornare in s. Questo cammino attraversa tutti gli altri vertici di G prima di tornare in s e sarà quindi di lunghezza n. Di conseguenza $\langle G, s, s, n\rangle \in LPATH$.
  - Se $\langle G, s, s, n\rangle \in LPATH$, allora esiste un cammino semplice nel grafo G che inizia e termina in s ed è di lunghezza maggiore o uguale a n. Un cammino semplice non ha ripetizioni, ad eccezione dei vertici iniziali e finali. Un cammino semplice da s ad s di lunghezza n deve attraversare tutti gli altri nodi una sola volta prima di tornare in s, ed è quindi un circuito Hamiltoniano per G. Cammini semplici più lunghi non possono esistere perché dovrebbero ripetere dei vertici. Quindi l'esistenza di un cammino semplice nel grafo G che inizia e termina in s ed è di lunghezza maggiore o uguale a n implica l'esistenza di un circuito Hamiltoniano in G, ed abbiamo dimostrato che $\langle G\rangle \in HAMILTON$.

  La funzione di riduzione si limita ad aggiungere tre nuovi elementi dopo la codifica del grafo G: due vertici ed un numero, operazione che si può fare in tempo polinomiale.