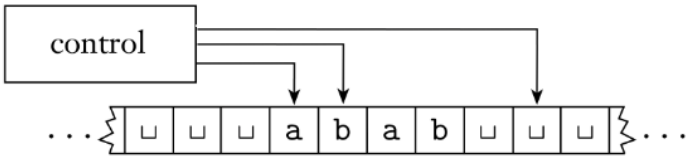
Una macchina di Turing a testine multiple è una macchina di Turing con un solo nastro ma con varie testine. Inizialmente, tutte le testine si trovano sopra alla prima cella dell'input. La funzione di transizione viene modificata per consentire la lettura, la scrittura e lo spostamento delle testine. Formalmente,

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

dove k è il numero delle testine. L'espressione

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

significa che, se la macchina si trova nello stato q_i e le testine da 1 a k leggono i simboli $a_1, \ldots, a_k$ allora la macchina va nello stato q_j, scrive i simboli $b_1, \ldots, b_k$ e muove le testine a destra e a sinistra come specificato.



Un esempio di TM con tre testine.

Mostriamo come simulare una TM a testine multiple B con una TM deterministica a nastro singolo S. Essendo una macchina che riporta molteplici testine su un nastro singolo, ciascuna di queste testine deve essere memorizzata in qualche modo posizionalmente parlando, per esempio assumiamo di separarle per mezzo di un pallino •, capendo dove è stata l'ultima scrittura, si sposta in avanti marcando la scrittura scrivendo l'insieme di simboli di b e prosegue in questo modo.
Ciascuna sequenza di stringhe memorizzata è separata da un # e tramite le marcature siamo in grado di capire dove e quando si sono avute scritture particolari.
La TM S a nastro singolo funziona come segue:

- si parte dalla prima cella e, avendo un certo numero di simboli da leggere, una delle testine si muove e non appena finisce la sua lettura, marca con un simbolo • la posizione
- le successive testine cominciano la lettura nello stesso modo, sapendo che le rispettive porzioni sono separate da # e quindi si ha la scansione di ciascuna porzione. In particolare, ogni testina, comincerà a riscrivere partendo da •
- la scrittura del nastro avviene secondo la funzione di transizione di B:
    - se δ(r,a) = (s,b,L), scrive *b* non marcato sulla cella corrente, dopodiché si sposta immediatamente a sinistra e sposta di una cella a sinistra tutte le marcature con il pallino, fino a che non raggiunge il #, al posto del quale aggiungerà un blank marcato, indicando che ha compiuto la sua scrittura a sinistra.
    - se δ(r,a) = (s,b,R) scrive *b* non marcato sulla cella corrente, dopodiché si sposta immediatamente a destra e sposta di una cella a destra tutte le marcature con il pallino, fino a che non raggiunge il #, al posto del quale aggiungerà un blank marcato, indicando che ha compiuto la sua scrittura a destra.

Se questa simulazione raggiunge lo stato di accettazione di B allora accetta, altrimenti se raggiunge lo stato di rifiuto di B, allora rifiuta. Se non dovesse essere nessuno dei due, si ripeterebbe la simulazione partendo dallo scorrimento del nastro.

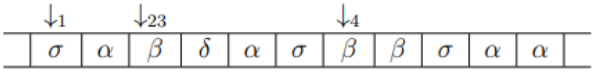
Una soluzione alternativa:

SOLUTION We must prove that the k head TM can be simulated by an ordinary TM. Since we already know that a multiple-tape TM can be simulated with an ordinary one, it suffices to simulate the k-head TM with a multiple-tape TM. We call K to the k-tape TM and M to the multiple-tape TM that simulates the former.

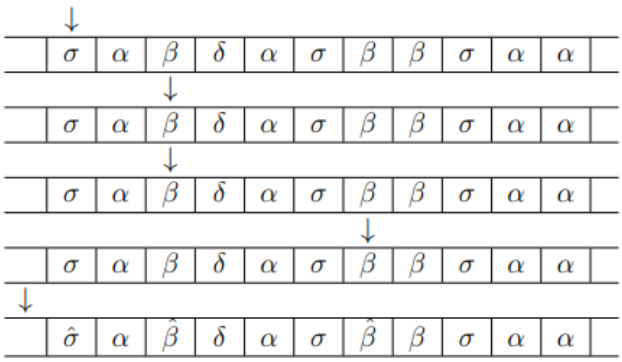
Assume we enumerate each head as h_i with $i = 1, \ldots, k$. We use a $k+1$-tape TM M to do the simulation. The first step is to copy the input tape of K into each tape of M. In the last tape, we call it 'extra', we add a mark to each symbol where there is a head in K (add a mark means that the alphabet of machine M is the union of the alphabet of the machine K plus the set of all the symbols in the alphabet, with a mark).

So, we get the following picture:

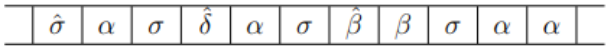
Let the machine K be in the following configuration:



Then the machine M is in the following configuration:



At each move in machine K, we do the same movement in the corresponding tape. Then, we check the last (extra) tape to see what are the changes with respect to each tape. We synchronize those changes one by one in the extra tape. For example, if in the tape 2th the 3th cell has been changed from β to σ and the head moves to the right, then after synchronizing such tape, the extra tape will looks as follows



Then when we have to synchronize the 3th tape we compare all the cells except those already synchronized (all except the 3th cell in our example), so the extra tape remain unchanged.

After having synchronized all the tapes. We replicate the extra tape in all the other tapes, which finish the synchronization process and the movement.

2. Dimostra che il seguente linguaggio è indecidibile:

$$L_2 = \{\langle M, w\rangle \mid M \text{ accetta la stringa } ww^R\}.$$

Partiamo dal costruire una TM M che effettivamente riconosca la stringa ww^R, quindi la stringa che abbiamo e similmente la stringa palindroma. Per fare ciò immaginiamo che:
M = Su input w:

- si ha il primo carattere della prima stringa, lo esaminiamo. Se appartiene alla stringa w, prosegue la computazione e si muove alla fine del nastro; altrimenti, rifiuta
- da questa parte si ha la stringa palindroma, dunque si esamina se tale carattere appartiene al palindromo, se così non fosse rifiuta.
- continuando a muoversi su e giù nella testina, la macchina *accetta* se esaurisce in questo modo tutta la computazione, *rifiuta* altrimenti.

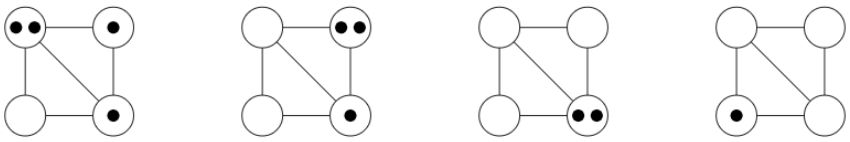
Abbiamo quindi creato un decisore. Ora creiamo una funzione di riduzione f, che prende in input la TM M appena creata ed una stringa *w*, dimostrando quindi che $A_{TM} \leq_m L_2$ e se e solo se f<M, w> = <M> $\in L_2$

La funzione di decisione f

- esegue M sull'input w
- se M accetta, *accetta*
- se M rifiuta, allora *rifiuta*

Se l'input appartiene correttamente al linguaggio, la TM rifiuta. La funzione calcola correttamente tutti gli input e si ferma se avesse una cosa del tipo (01), rispetto ad una stringa accettabile come (0110). Siccome f è computabile in un numero finito di passi, si fermerà su tutti gli input ed f rappresenta una riduzione del linguaggio, rifiutando tutte le stringhe che non centrano.
Tuttavia, sappiamo che L_2 è indecidibile ed è una contraddizione, dunque non può esistere un decisore.

3. Pebbling è un solitario giocato su un grafo non orientato G, in cui ogni vertice ha zero o più ciottoli. Una mossa del gioco consiste nel rimuovere due ciottoli da un vertice v e aggiungere un ciottolo ad un vertice u adiacente a v (il vertice v deve avere almeno due ciottoli all'inizio della mossa). Il problema PEBBLEDESTRUCTION chiede, dato un grafo $G = (V, E)$ ed un numero di ciottoli $p(v)$ per ogni vertice v, di determinare se esiste una sequenza di mosse che rimuove tutti i sassolini tranne uno.



Una soluzione in 3 mosse di PEBBLEDESTRUCTION.

Dimostra che PEBBLEDESTRUCTION è NP-hard usando il problema del Circuito Hamiltoniano come problema NP-hard noto (un circuito Hamiltoniano è un ciclo che attraversa ogni vertice di G esattamente una volta).

Per dimostrare che PEBBLEDESTRUCTION/PB è NP-Hard, dobbiamo dimostrare che è NP e che è riducibile attraverso l'idea del circuito hamiltoniano.
Descriviamo il certificato per questo problema, tale che si abbia SI in tempo polinomiale.
Esso deve verificare le seguenti condizioni:

- l'insieme dei ciottoli p deve essere collegato ad ogni vertice v
- per ogni vertice, verifica se esiste un arco collegato
- esiste almeno un ciottolo che contiene un sassolino, mentre tutti gli altri non ne contengono nessuno

Dovendo esaminare a coppie i vertici del grafo in PB si ha che la verifica viene eseguita in tempo polinomiale. Per dimostrare che PB è NP-Hard si deve descrivere un algoritmo per risolvere PB attraverso l'uso del Circuito Hamiltoniano/HAMPATH/HM e dimostrare che la riduzione è corretta, tale da trasformare istanze buone di HM in istanze buone di PB e dimostrare che l'istanza è corretta.

Partendo dal circuito hamiltoniano H, sappiamo che esso deve avere tutti i vertici collegati in un ciclo esattamente una volta. Supponiamo quindi che per ogni vertice si abbia almeno un sassolino, tranne un vertice che ne presenta 2, cioè p(v) = 2.
Se abbiamo una riduzione, quindi, l'istanza buona S deve verificare che ogni singolo vertice sia connesso agli altri e che presenti almeno un sassolino; se ciò accade siamo all'interno di un circuito hamiltoniano.
Detto ciò, è possibile che, aggiungendo all'ultimo vertice percorso un sassolino, siamo in un'istanza buona di HM.
Se invece vogliamo verificare se l'istanza S' sia buona per PB, allora prendiamo l'ultimo vertice, che presenta un grado di pebbles/sassolini uguale a 2. Allora è possibile, verificare, togliendo dall'ultimo vertice il sassolino in più e verificando che tutti gli altri vertici contengano un solo sassolino; nel qual caso, abbiamo percorso tutto il ciclo e correttamente abbiamo che si ha un circuito hamiltoniano.

Per verificare la correttezza dell'algoritmo rispetto all'idea iniziale, si ripercorre tutto il circuito, lasciando un solo sassolino all'ultimo vertice e togliendo tutti gli altri con la verifica hamiltoniana.
Se abbiamo che il vertice utilizzato era lo stesso di partenza (indichiamo il vertice u che discutevo prima e v quello attuale dopo l'ultimo ciclo), tale che u = v, allora abbiamo un circuito hamiltoniano che arriva nello stesso punto in cui iniziava il pebble.
Quindi, tutto viene eseguito in tempo polinomiale. Se avessimo anche solo un vertice di troppo o rimanesse un sassolino, correttamente non si avrebbe più un circuito hamiltoniano e neanche il pebble sarebbe risolto. Dunque, PB è un problema NP-Completo e può essere correttamente ridotto ad H.