Macchine di Turing per calcolare funzioni

8 aprile 2022

Oggi

- Obiettivo: diversi «usi» della MdT
  - MdT riconosce un linguaggio
  - MdT decide un linguaggio
  - MdT calcola una funzione
  - MdT enumera stringhe di un linguaggio

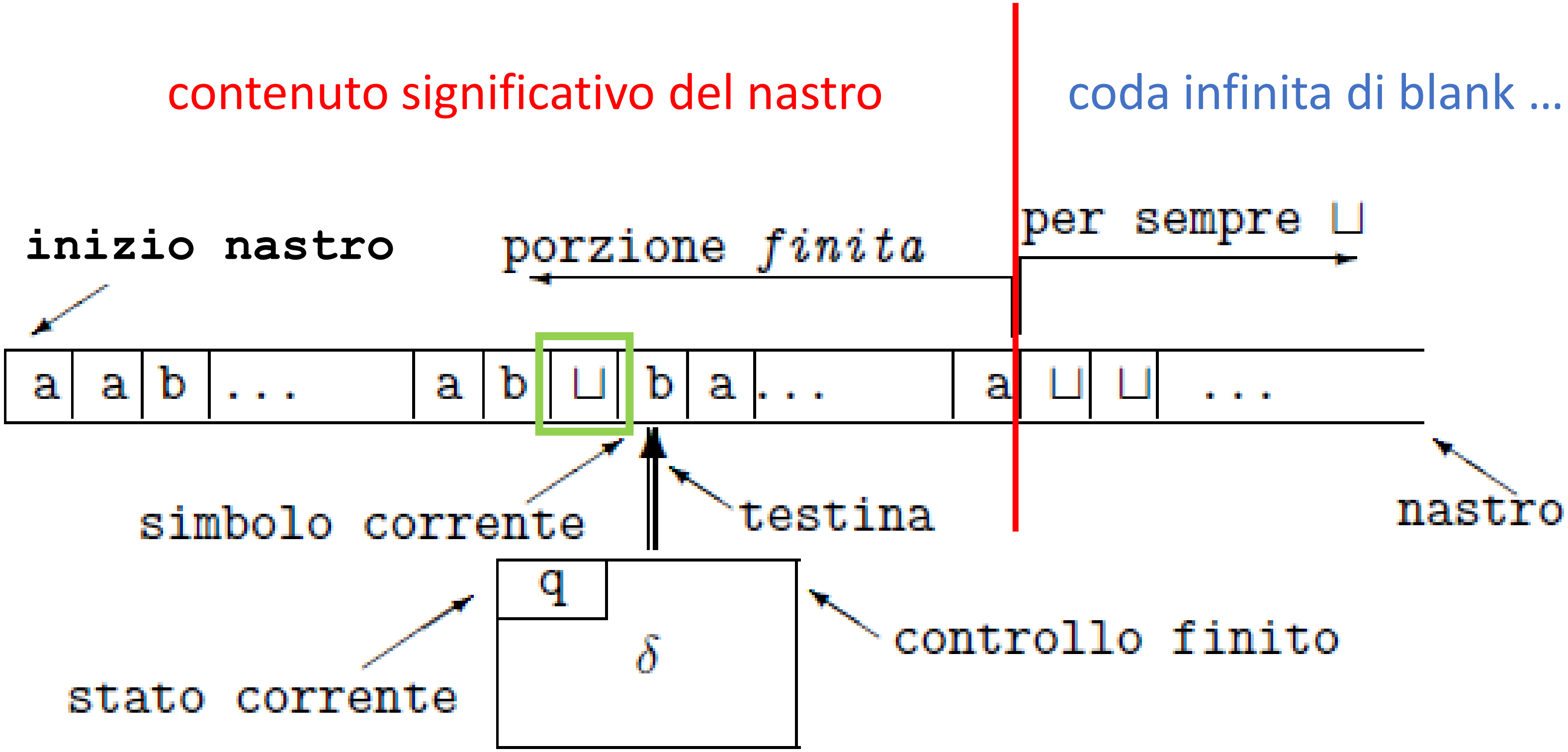
Descrizione formale MdT

Una Macchina di Turing è una settupla

$$(Q, \Sigma, \Gamma, \delta, q_o, q_{accept}, q_{reject})$$

- **Insieme Stati** Q
- **Alfabeto di lavoro** Σ ($\_ \notin \Sigma$)
- Γ: **Alfabeto del nastro** ($\_ \in \Gamma$, $\Sigma \subset \Gamma$)
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$: funzione transizione
- q_0: stato **iniziale**
- q_{accept}: stato **accept**
- q_{reject}: stato **reject**

Una MdT



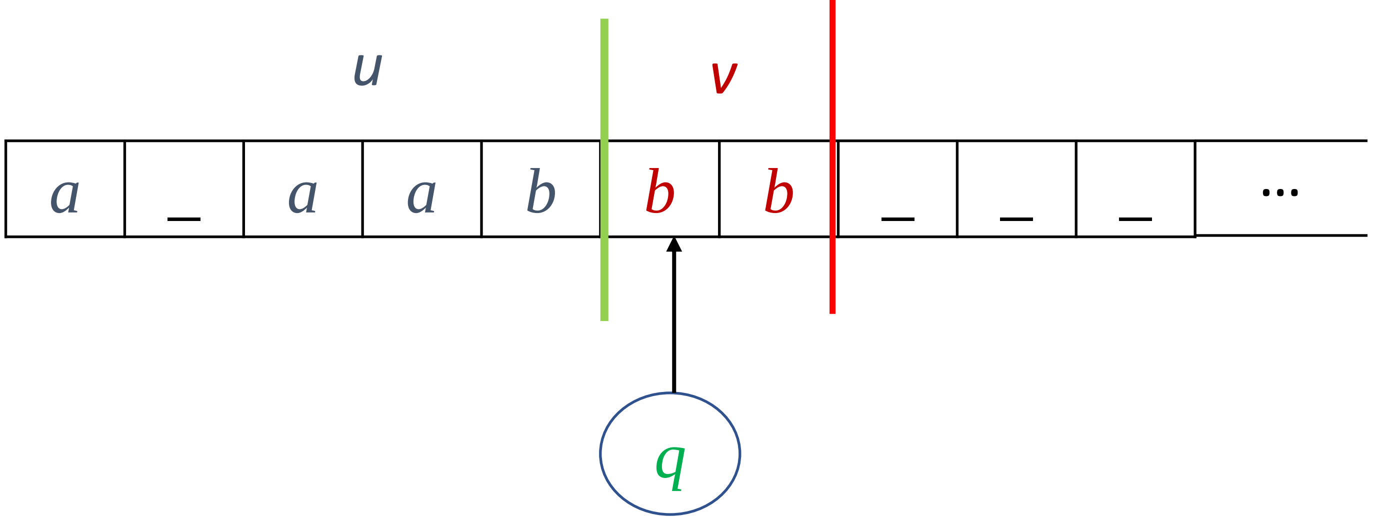
Il contenuto significativo del nastro è una stringa $w \in \Gamma^*$, con la convenzione che il suo ultimo carattere (se $w \neq \varepsilon$) non sia blank.
La stringa w PUO' contenere blank al suo interno.
A volte nel progetto di una MdT si definiscono delle transizioni per scrivere un carattere speciale nella prima cella del nastro, per meglio individuarla.

Configurazione di una MdT

La configurazione $C = u\ q\ v$ corrisponde a

contenuto significativo del nastro = $u\ v$

coda infinita di blank ...



Configurazione di una MdT

Descrizione concisa della situazione del calcolo di una MdT ad un certo istante, anche detta descrizione istantanea.

Configurazione di una MdT $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

$$C = u\ q\ v$$

- $q \in$ Q è lo stato corrente
- $u\ v \in \Gamma^*$ è il contenuto significativo del nastro (senza _ finali, se $u\ v \neq \varepsilon$)
- La testina è posizionata sul primo simbolo di v, se $v \neq \varepsilon$, su _ altrimenti

Passo di computazione

Siano C_1, C_2 due configurazioni di una MdT M.

Se C_1 produce C_2, scriveremo

$$C_1 \to C_2$$

La trasformazione $\to$ di C_1 in C_2 prende il nome di **passo di computazione**.

Corrisponde a un'applicazione della funzione di transizione di M.

Computazione di una MdT

Siano C, C' *configurazioni.*
$C \to^* C'$ *se esistono configurazioni* $C_1, \ldots, C_k$, $k \geq 1$ *tali che*

1. $C_1 = C$,
2. $C_i \to C_{i+1}$, *per* $i \in \{1, \ldots, k-1\}$,
   (ogni C_i *produce* C_{i+1}*)*
3. $C_k = C'$.

Diremo che $C \to^* C'$ *è una* **computazione** *(di lunghezza* $k-1$*).*

Quando k=1?

Configurazioni

Una configurazione C si dice:

- iniziale su input w se $C = q_0 w$, con w $\in \Sigma^*$
- di accettazione se $C = u\ q_{accept}\ v$
- di rifiuto se $C = u\ q_{reject}\ v$

Poiché non esistono transizioni da q_{accept} e da q_{reject}, allora le configurazioni di accettazione e di rifiuto sono dette configurazioni di arresto.

Parola accettata o rifiutata

Definizione

Una MdT M accetta una parola $w \in \Sigma^*$ se esiste una computazione $C \to^* C'$, dove $C = q_0 w$ è la configurazione iniziale di M con input w e $C' = uq_{accept}v$ è una configurazione di accettazione.

Una MdT M rifiuta una parola $w \in \Sigma^*$ se esiste una computazione $C \to^* C'$, dove $C = q_0 w$ è la configurazione iniziale di M con input w e $C' = uq_{reject}v$ è una configurazione di rifiuto.

Risultati di una computazione

Tre possibili Risultati computazione:

1. M accetta – se si ferma in q_{accept}
2. M rifiuta – se si ferma in q_{reject}
3. M cicla/loop – se non si ferma mai

Mentre M funziona non si può dire se è in loop; si potrebbe fermare in seguito oppure no.

Linguaggio riconosciuto da una MdT

Definizione

Sia $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ *una MdT. Il linguaggio* $L(M)$ *riconosciuto da* M*, è l'insieme delle stringhe che* M *accetta:*

$$L(M) = \{w \in \Sigma^* \mid \exists u, v \in \Gamma^* \; q_0 w \to^* u q_{accept} v\}.$$

Quindi

$$L(M) = \{w \in \Sigma^* \mid M \text{ accetta } w\}.$$

Decidere

$$L(M) = \{w \in \Sigma^* \mid M \text{ accetta } w\}$$

$$R(M) = \{w \in \Sigma^* \mid M \text{ rifiuta } w\}$$

In generale $L(M) \cup R(M)$ non coincide con Σ^*.

Se $L(M) \cup R(M) = \Sigma^*$, allora M si arresta su ogni input.

In tal caso M è chiamata un decisore (o decider) ed $L(M)$ è il linguaggio deciso da M.

Dal punto di vista delle macchine

Definizione

Una MdT $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ *è un* **decisore** *(o* **decider***) se, per ogni* $w \in \Sigma^*$*, esistono* $u, v \in \Gamma^*$ *e* $q \in \{q_{accept}, q_{reject}\}$ *tali che*

$$q_0 w \rightarrow^* uqv$$

Definizione

Una MdT M **decide** *un linguaggio L se M è un decisore e L = L(M).*

In tal caso L è **deciso** *da M.*

Dal punto di vista dei linguaggi

Definizione

Un linguaggio $L \subseteq \Sigma^*$ è Turing riconoscibile se esiste una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ tale che:

1. *M riconosce L*
   (cioè $L = L(M) = \{w \in \Sigma^* \mid \exists u, v \in \Gamma^* \ q_0 w \to^* u q_{accept} v\}$).

Definizione

Un linguaggio $L \subseteq \Sigma^*$ è decidibile se esiste una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ tale che:

1. *M riconosce L*
   (cioè $L = L(M) = \{w \in \Sigma^* \mid \exists u, v \in \Gamma^* \ q_0 w \to^* u q_{accept} v\}$).
2. *M si arresta su ogni input (cioè per ogni $w \in \Sigma^*$, $q_0 w \to^* uqv$ con $q \in \{q_{accept}, q_{reject}\}$).*

Se L è decidibile in particolare è riconoscibile; ma non viceversa.

Riconosce o calcola?

Sia $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ *una MdT. Il linguaggio* $L(M)$ *riconosciuto da* M*, è l'insieme delle stringhe che* M *accetta:*

$$L(M) = \{w \in \Sigma^* \mid \exists u, v \in \Gamma^* \ q_0 w \rightarrow^* u q_{accept} v\}.$$

In questa definizione ciò che conta è il raggiungimento dello stato q_{accept} , più che il contenuto finale del nastro, cioè *uv*.

Ribaltando questa visione, ci interesseremo adesso al contenuto finale del nastro, più che allo stato di arrivo, in modo che la MdT calcoli qualcosa.

Funzioni calcolabili

Le MdT possono essere utilizzate per il calcolo di funzioni.

Definizione

Una funzione $f : \Sigma^* \to \Sigma^*$ *è calcolabile se esiste una macchina di Turing* $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ *tale che*

$$\forall w \in \Sigma^* \quad q_0 w \to^* q_{accept} f(w)$$

Funzioni calcolabili

Quindi, una funzione $f : \Sigma^* \to \Sigma^*$ è calcolabile se esiste una macchina di Turing M che, su qualsiasi input w, si ferma avendo solo $f(w)$ sul nastro.

- Nota: in questo caso la MdT deve arrestarsi su ogni input.
- Possibile variante: nessun vincolo sulla posizione della testina nella configurazione di arresto.
- Possibile variante: nessuna distinzione tra q_{accept} e q_{reject}.

Funzioni calcolabili

- Definire una macchina di Turing deterministica M che calcoli la funzione $f(x, y) = x + y$, con x, y interi positivi.
- Si assuma che l'input sia della forma

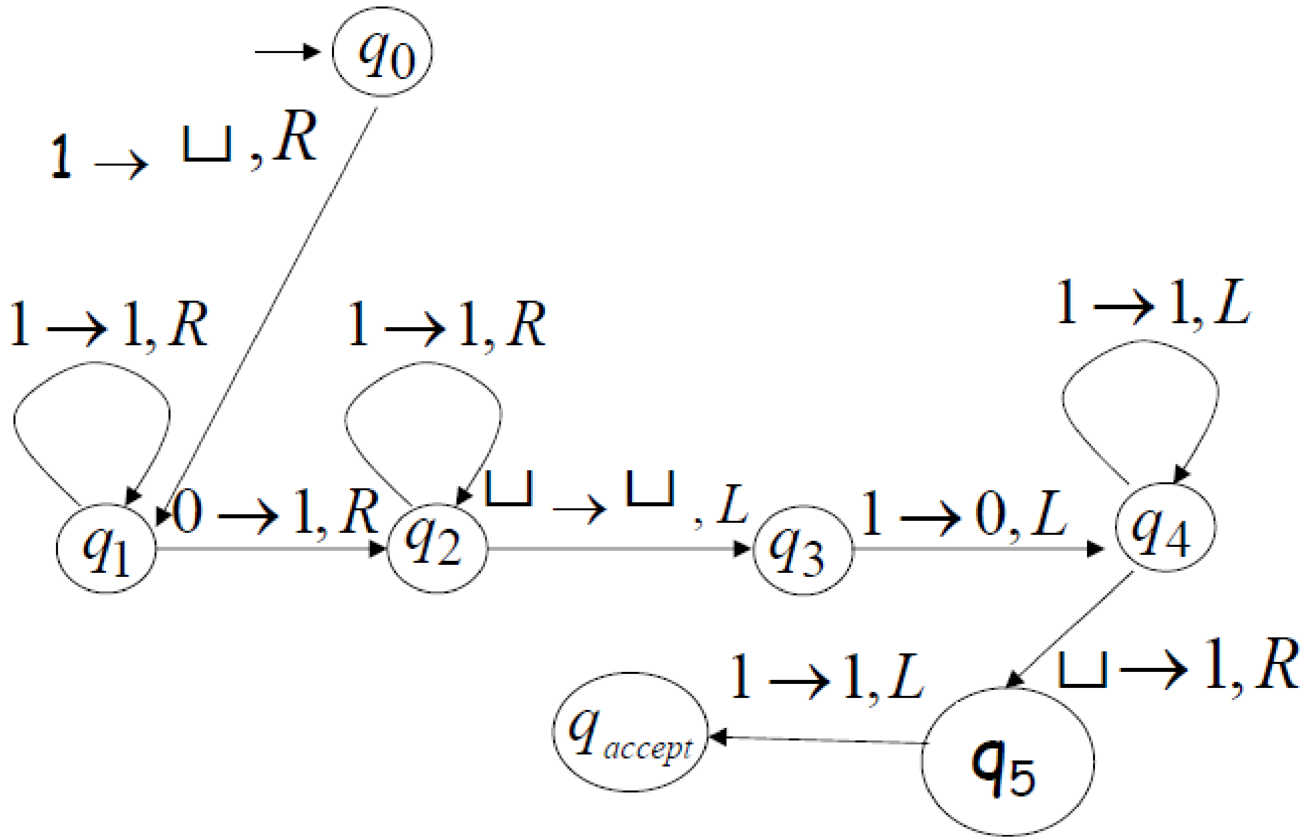
$$w0z$$

  dove w è la rappresentazione unaria di x e z è la rappresentazione unaria di y.
- Si definisca M in modo che l'output sia della forma

$$wz0$$

Per esempio se $x = 3$ e $y = 2$, l'input sarà 111011 e M dovrà fermarsi con 111110 sul nastro.

Macchina di Turing per $f(x, y) = x + y$



Rappresentazioni di MdT

- Settupla
- Diagramma di stato
- Tabella

Rappresentazione tabellare di una MdT

- 5 colonne e 1 riga per ogni etichetta
  - Prima colonna: stato attuale
  - Seconda colonna: simbolo letto
  - Terza colonna: nuovo stato
  - Quarta colonna: simbolo da scrivere
  - Quinta colonna: movimento testina
- Esempio:

stato	**simbolo**	**stato**	**simbolo**	**movimento**
q0	0	q0	1	R
q0	1	q0	0	R
q0	_	q1	_	L
q1	0	q1	0	L
q1	1	q1	1	L
q1	_	q2	_	R

COSA FA?

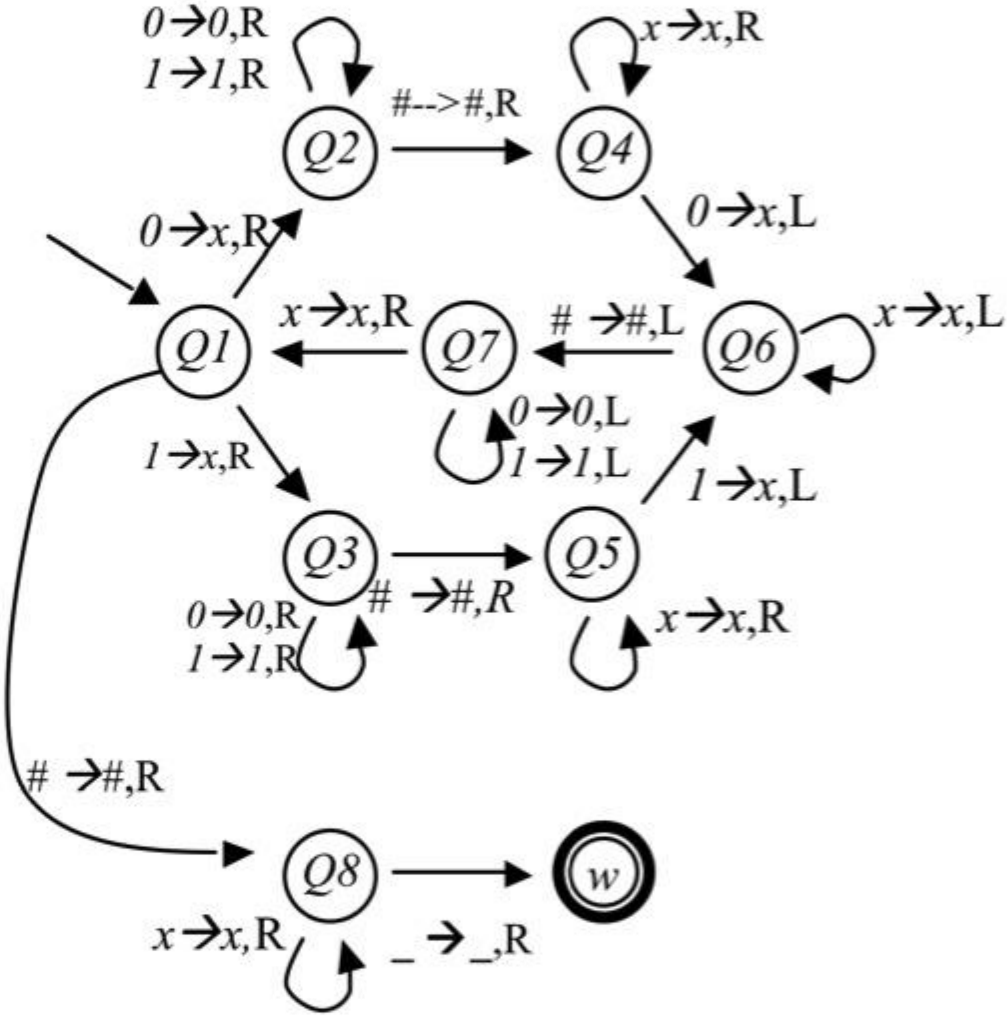
MdT per w#w

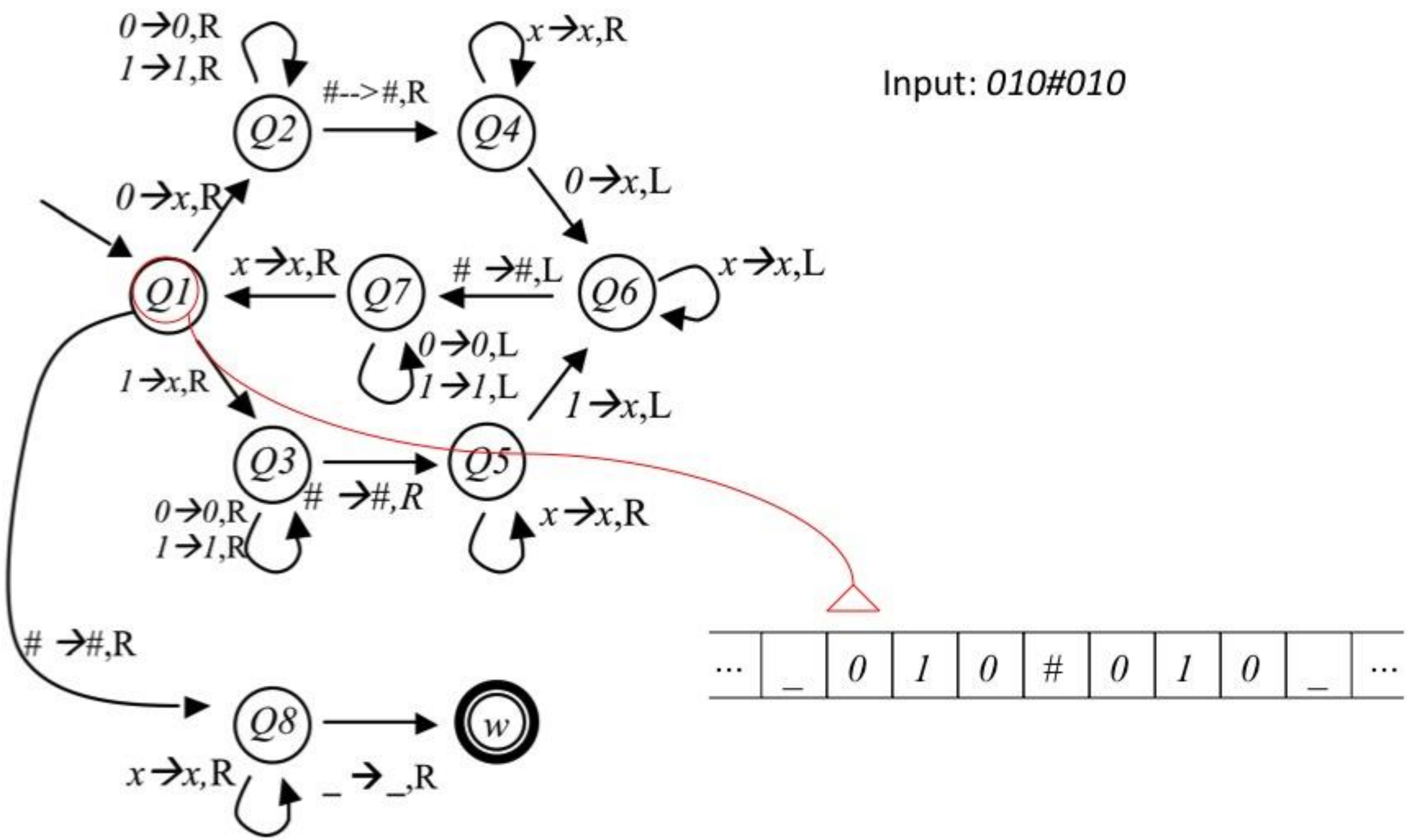
Esercizio

Progettare una MdT che decida il linguaggio

L = { w#w | w è una stringa sull'alfabeto { 0, 1 } }

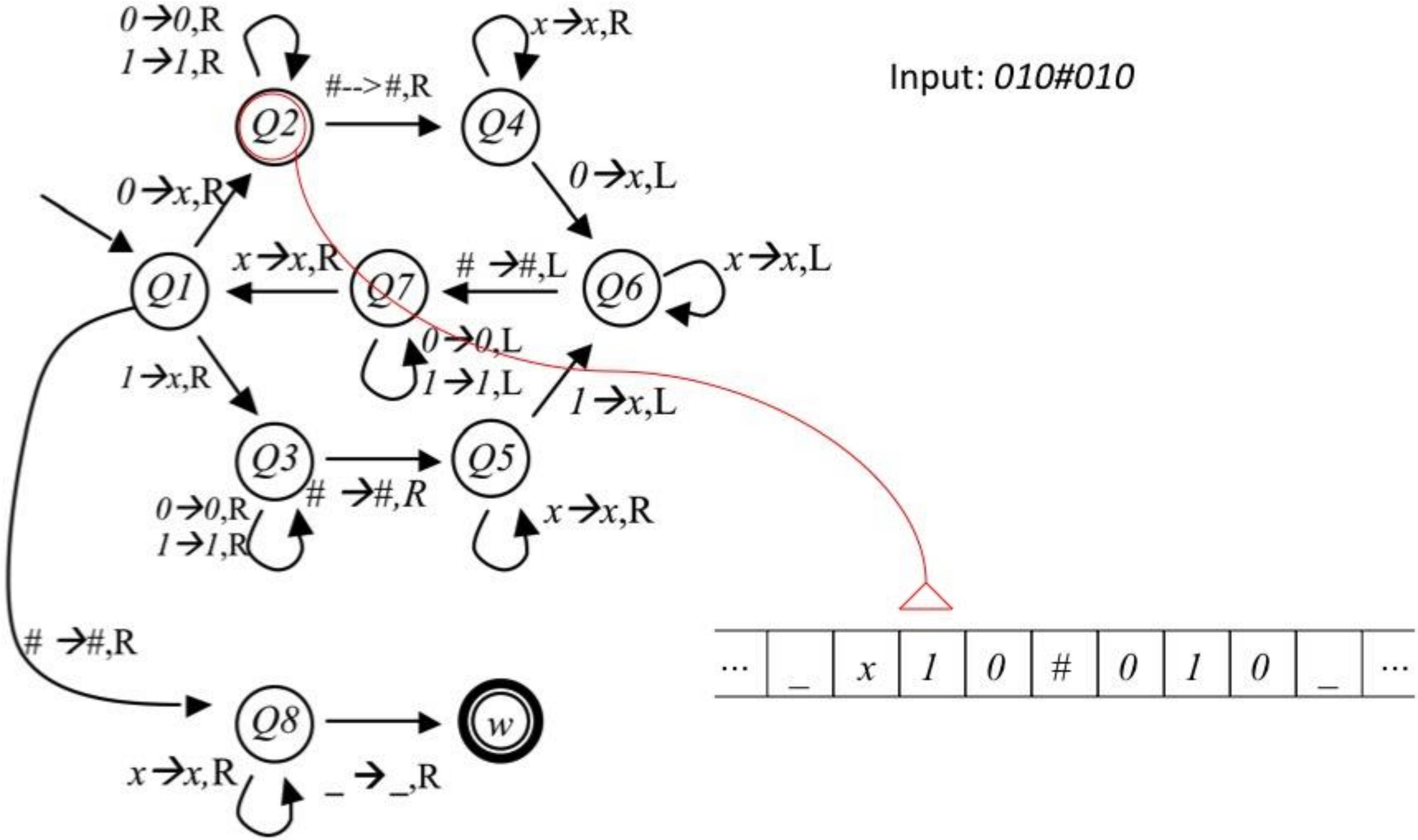
MdT per w#w

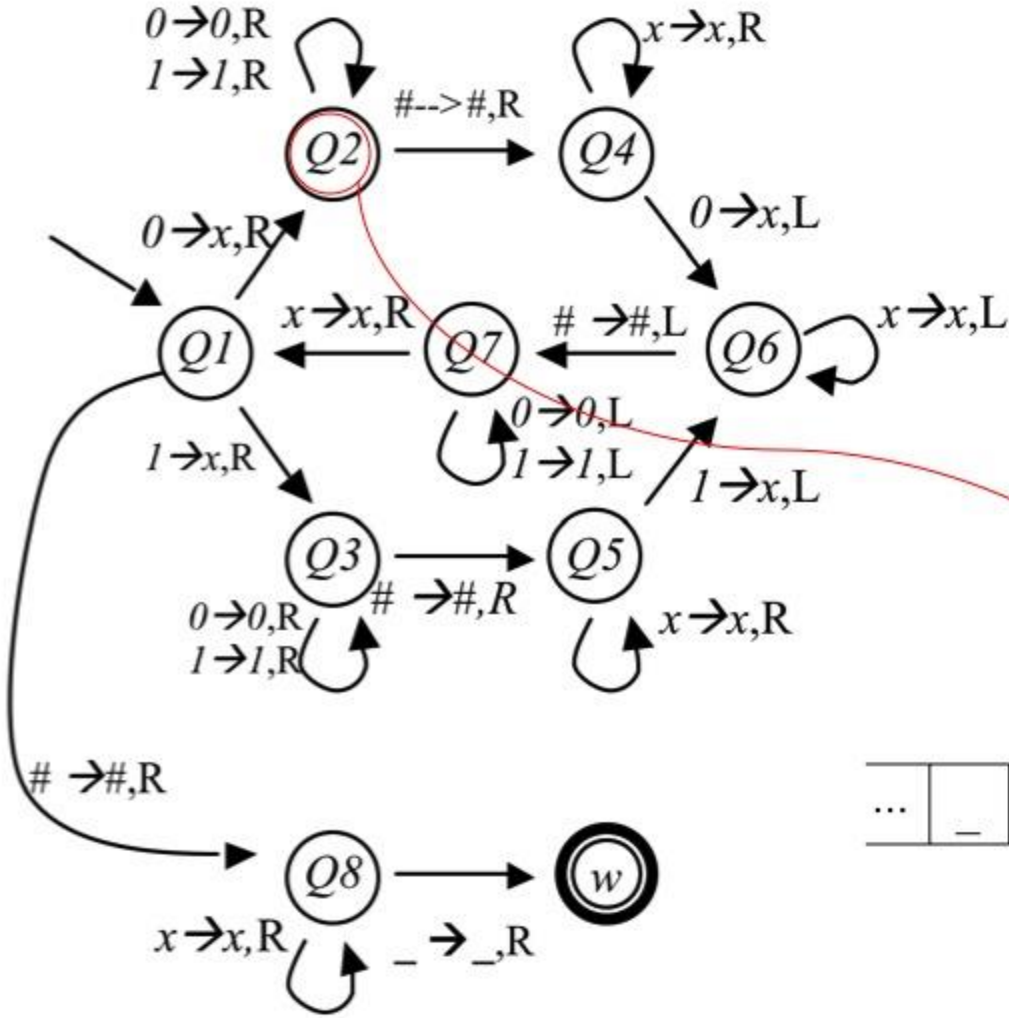

0→0,R
1→1,R
#-->#,R
x→x,R
Q2
Q4
0→x,R
0→x,L
Q1
x→x,R
Q7
→#,L
Q6
x→x,L
0→0,L
1→1,L
1→x,R
1→x,L
Q3
Q5
0→0,R
1→1,R
→#,R
x→x,R
→#,R
Q8
w
x→x,R
_ →_,R



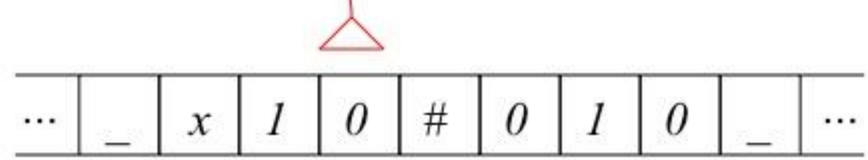
0→0,R
1→1,R
x→x,R
#-->#,R
Q2
Q4
0→x,R
0→x,L
x→x,R
Q1
Q7
→#,L
Q6
x→x,L
0→0,L
1→1,L
1→x,R
1→x,L
Q3
Q5
→#,R
0→0,R
1→1,R
x→x,R
→#,R
Q8
w
x→x,R
_ →_,R
Input: 010#010
... _ 0 1 0 # 0 1 0 _ ...

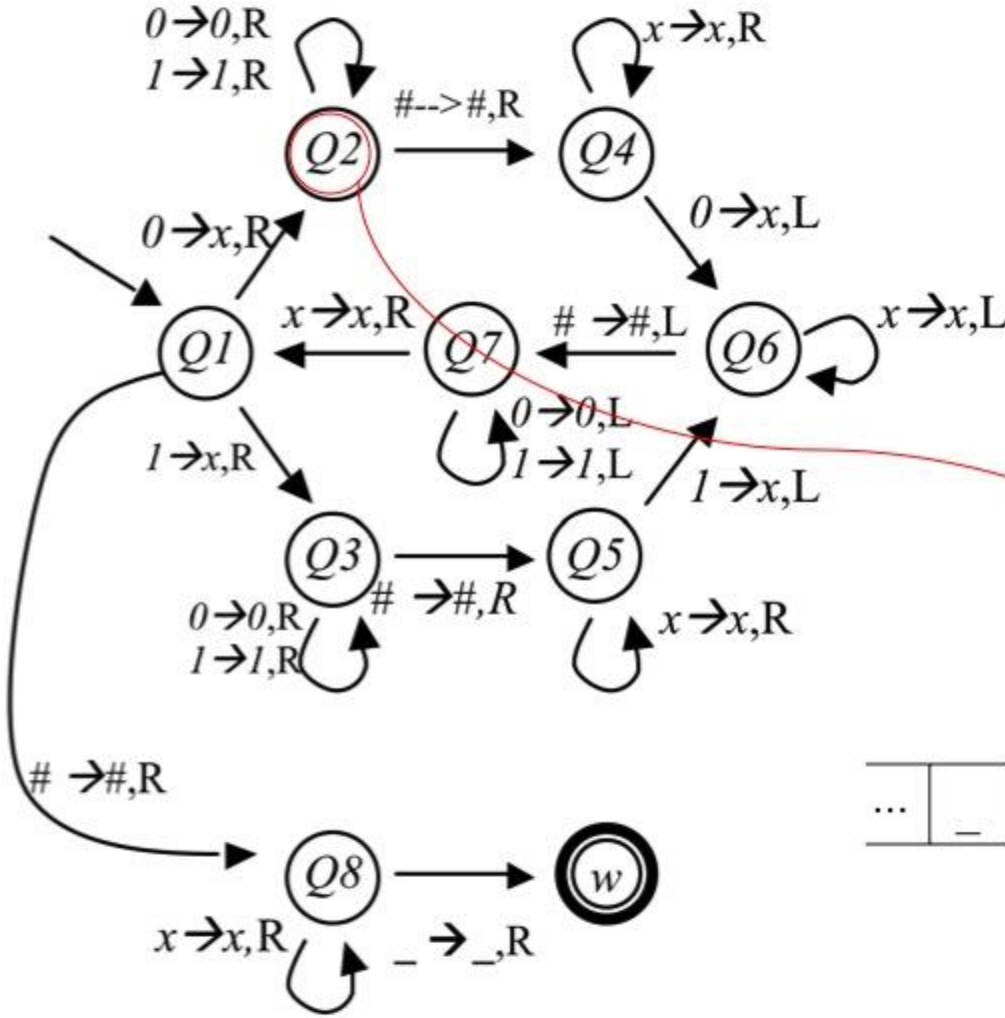
Input: *010#010*



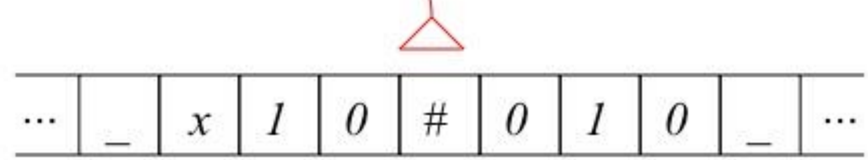


Input: *010#010*

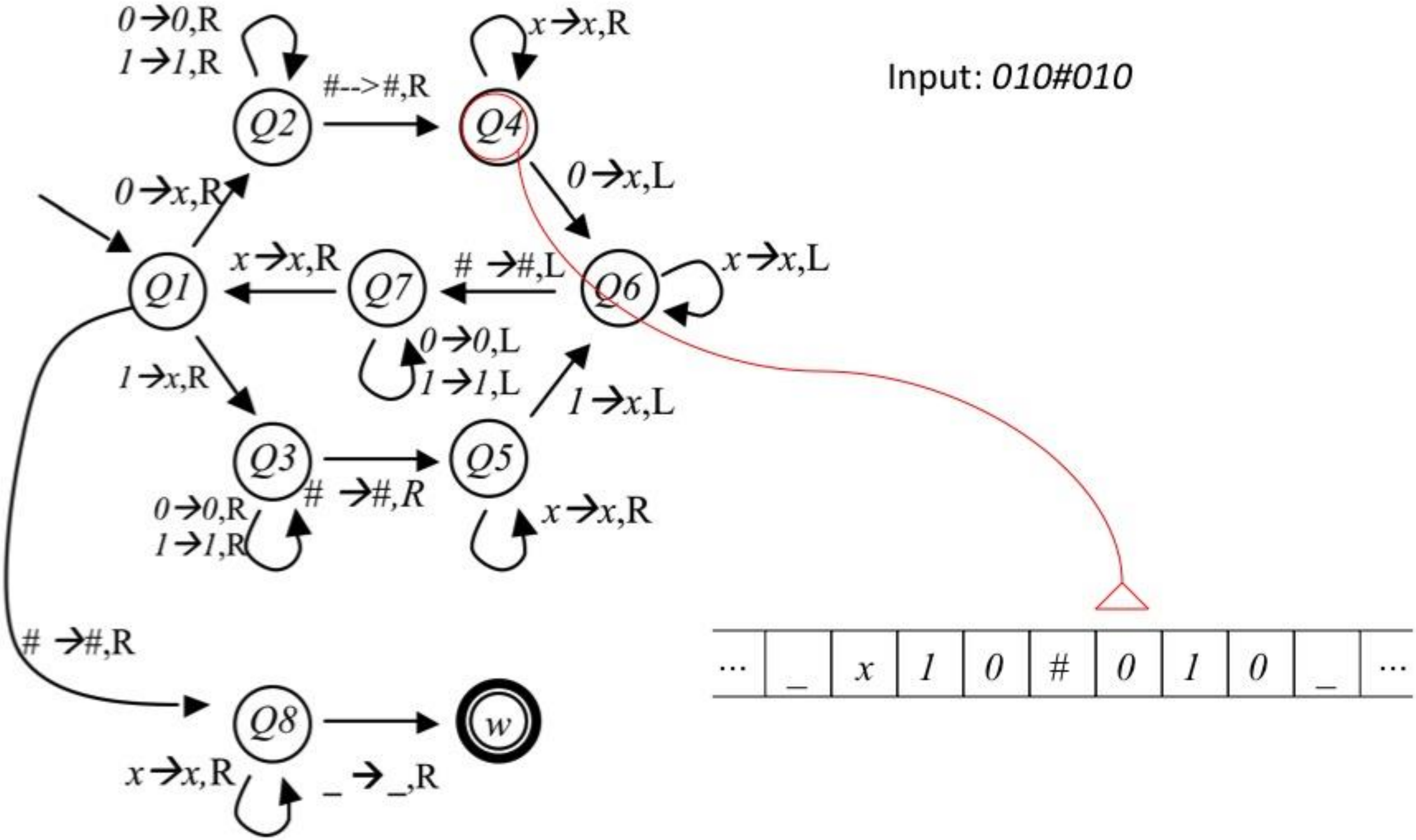


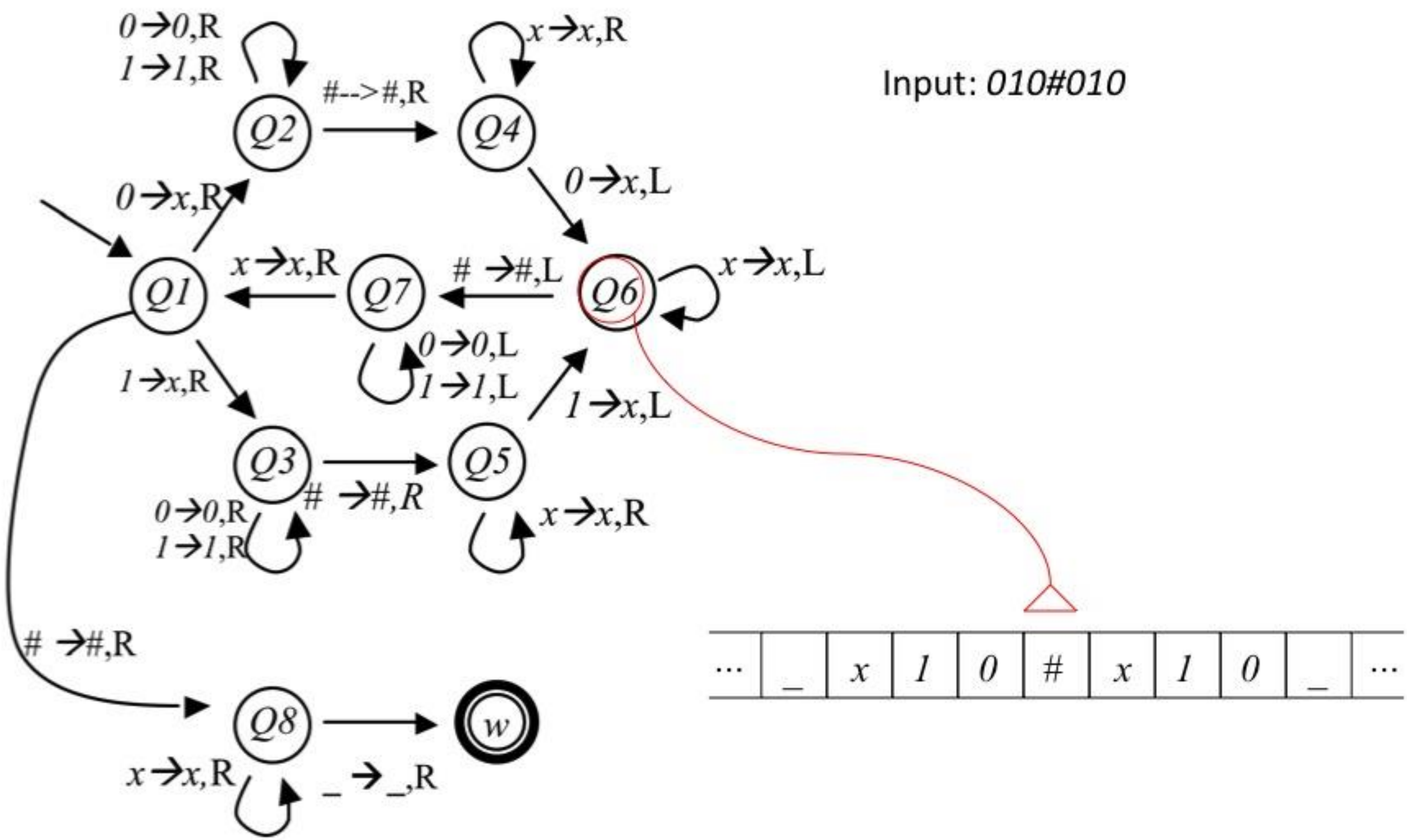


Input: *010#010*

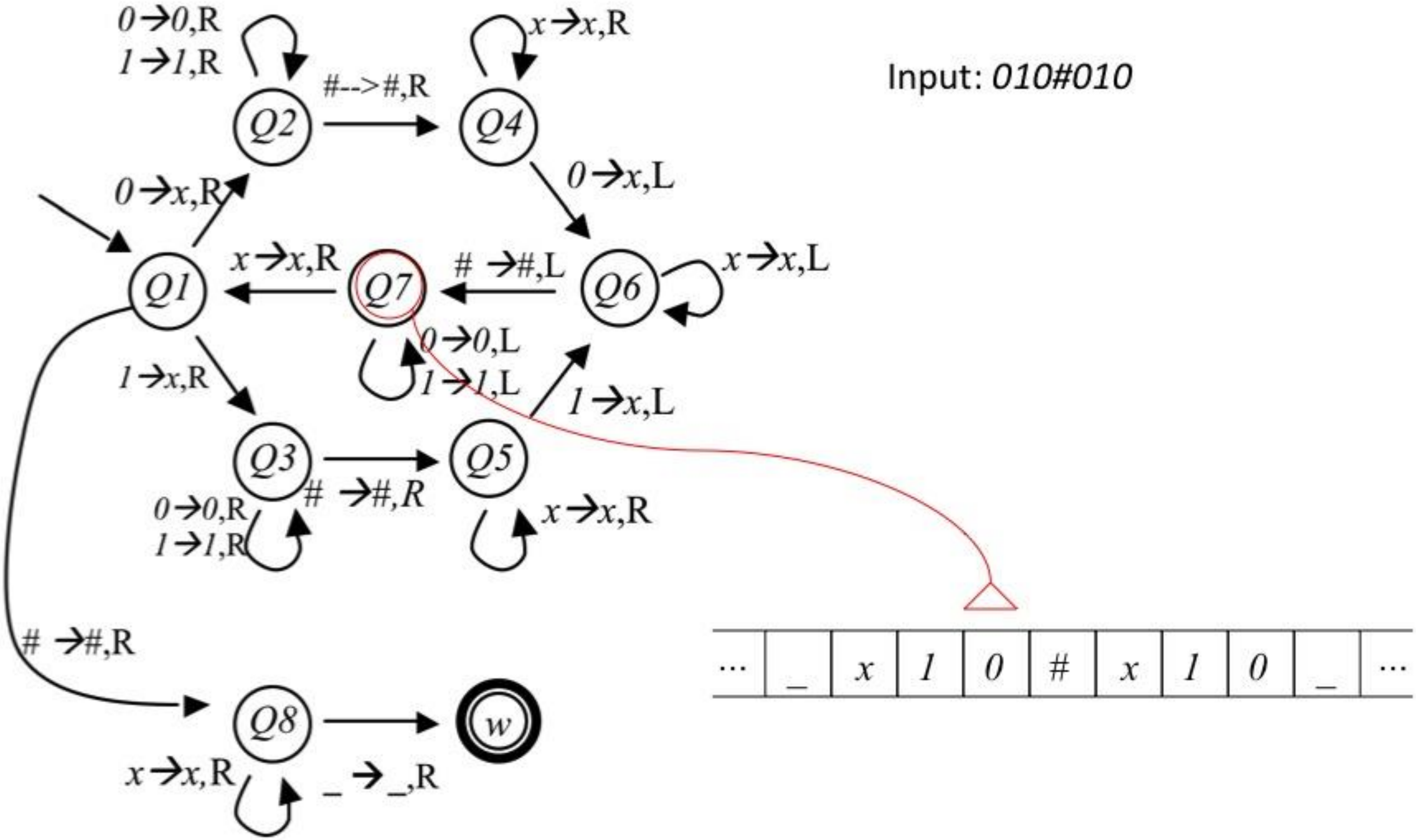


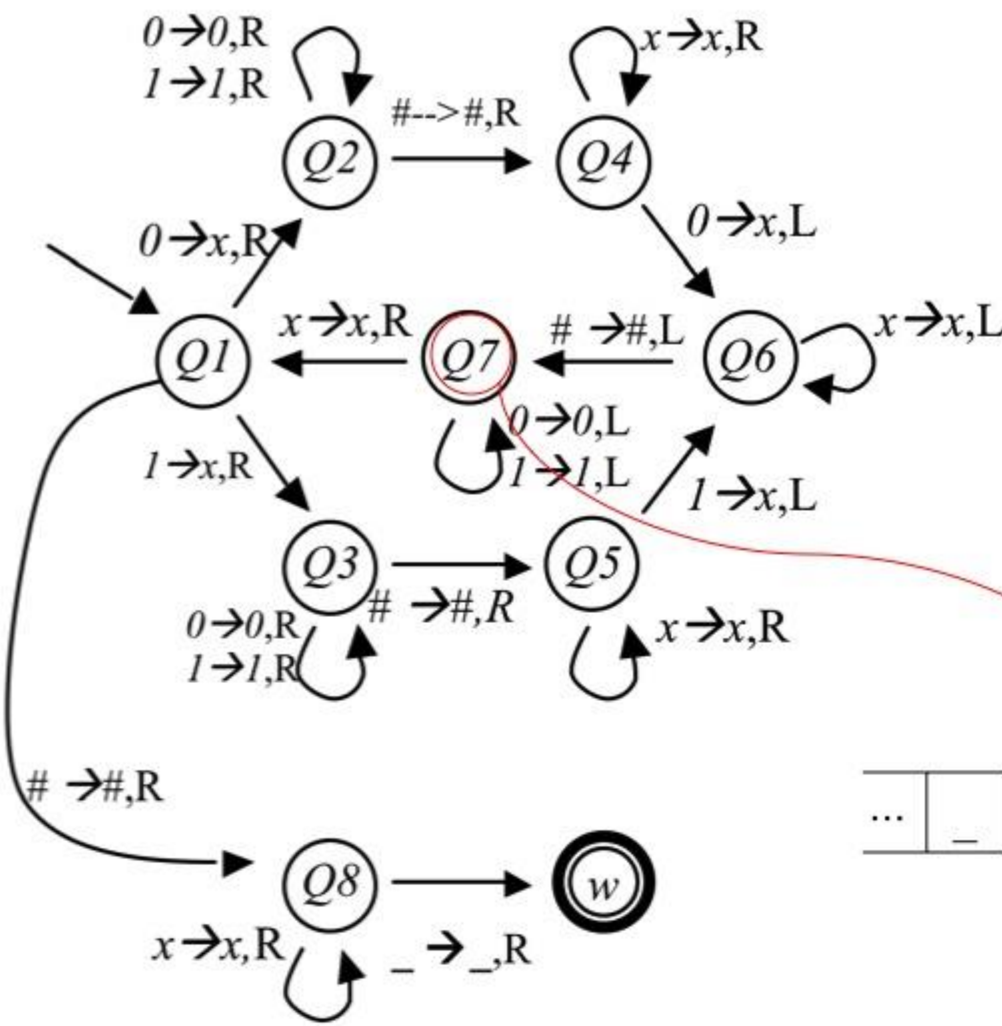
Input: *010#010*




Input: 010#010
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
w
0→0,R
1→1,R
#-->#,R
x→x,R
0→x,R
0→x,L
x→x,R
→#,L
x→x,L
0→0,L
1→1,L
1→x,R
1→x,L
→#,R
0→0,R
1→1,R
x→x,R
→#,R
x→x,R
_ →_,R
...
_
x
1
0
#
x
1
0
_
...

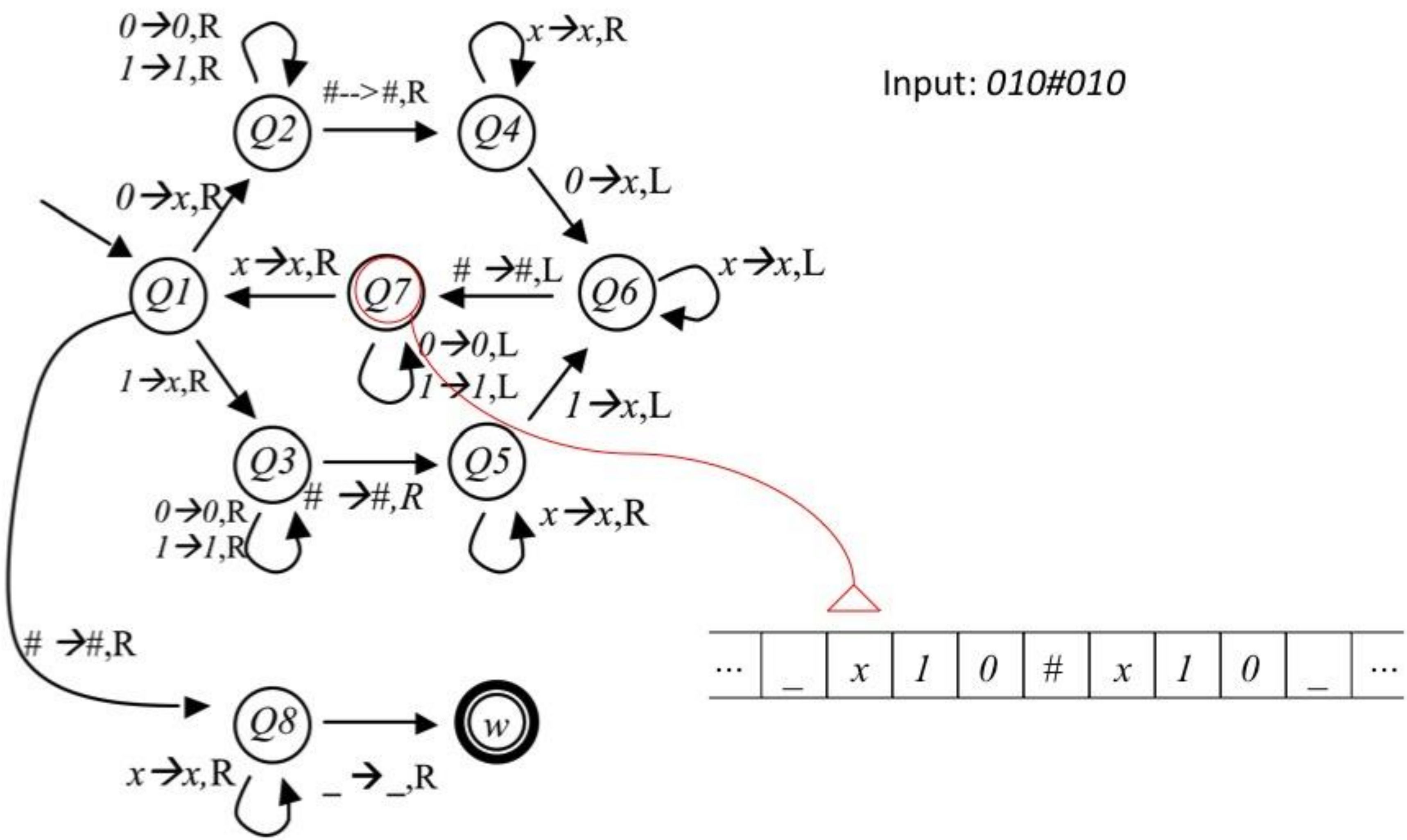
Input: *010#010*

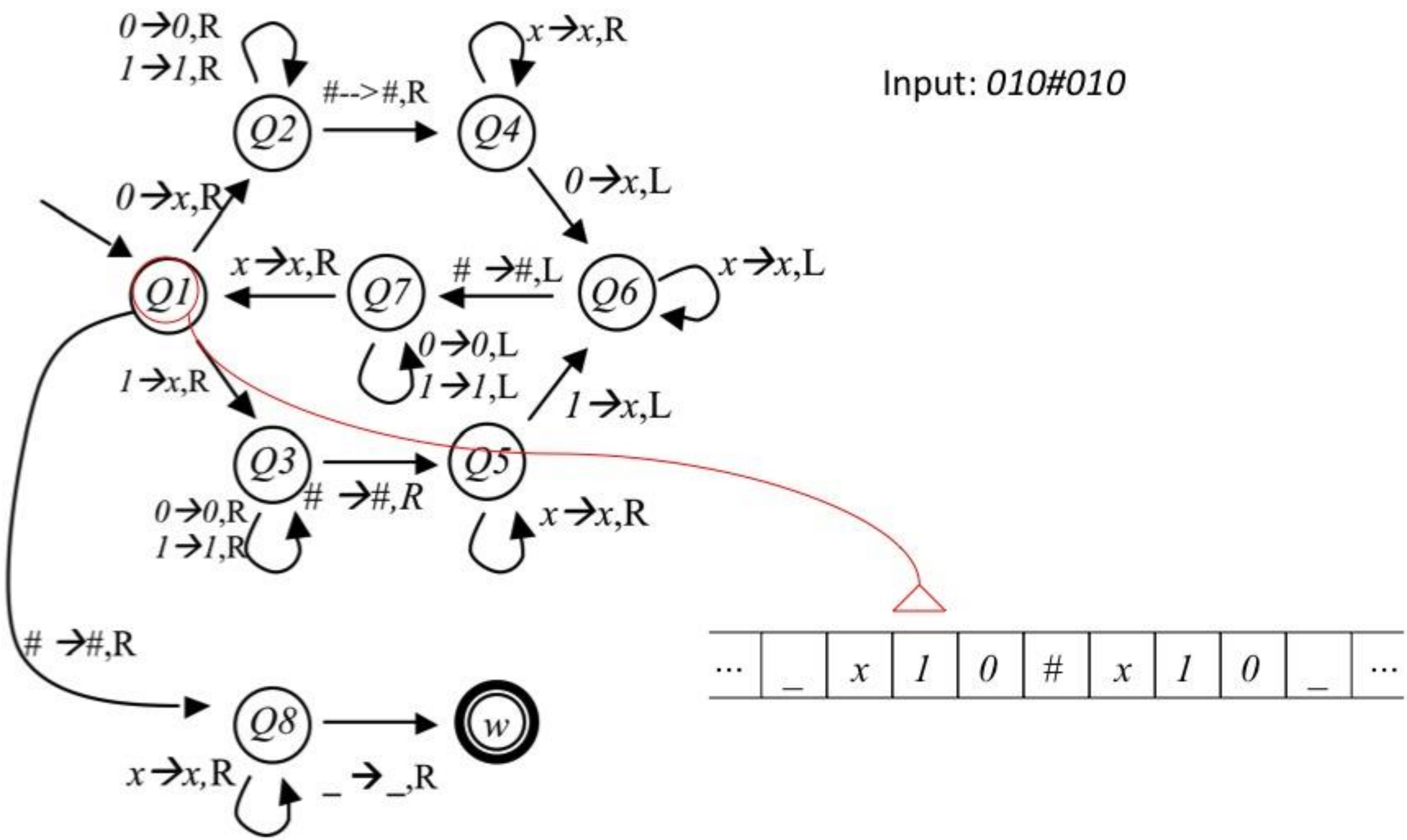




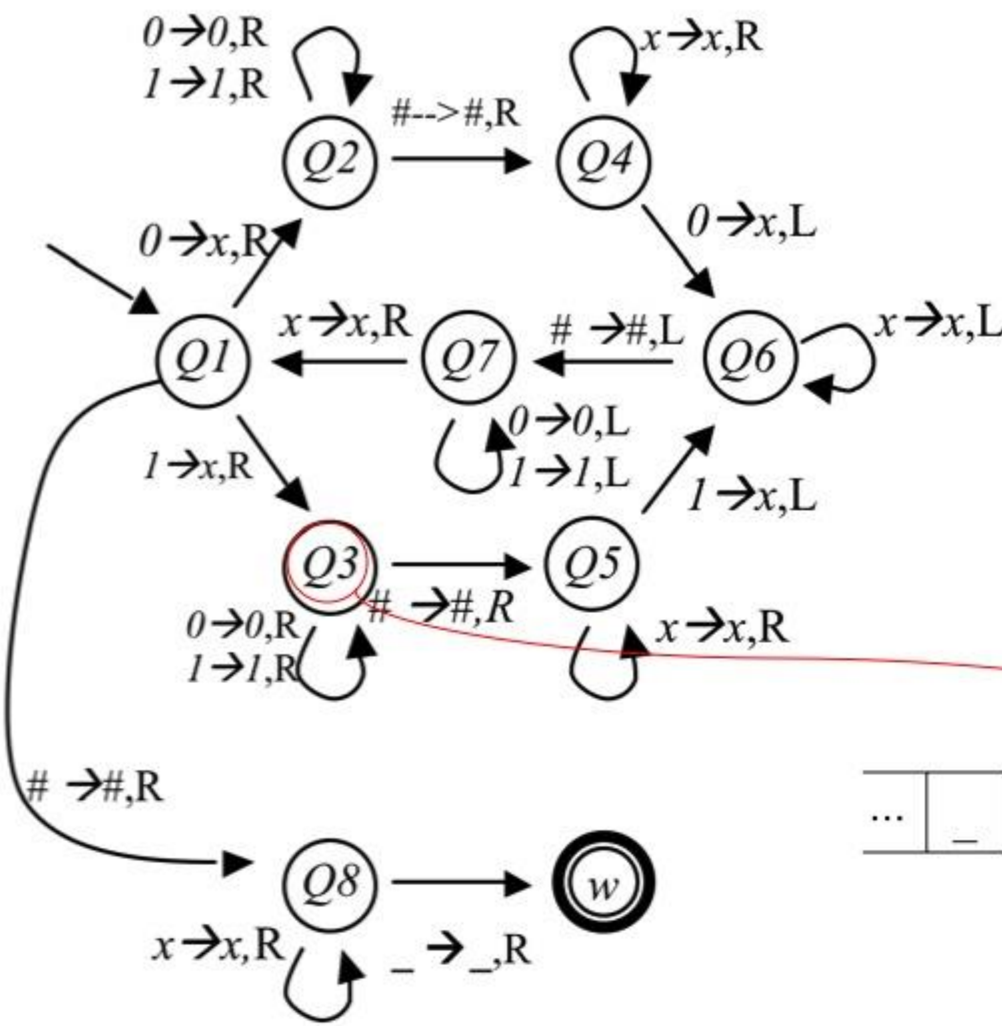
Input: *010#010*

Input: *010#010*

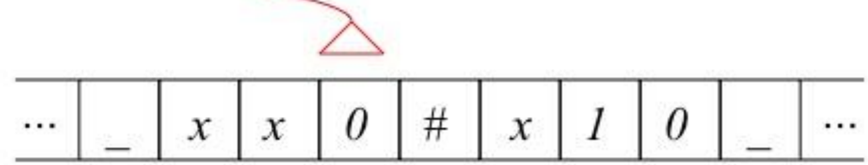




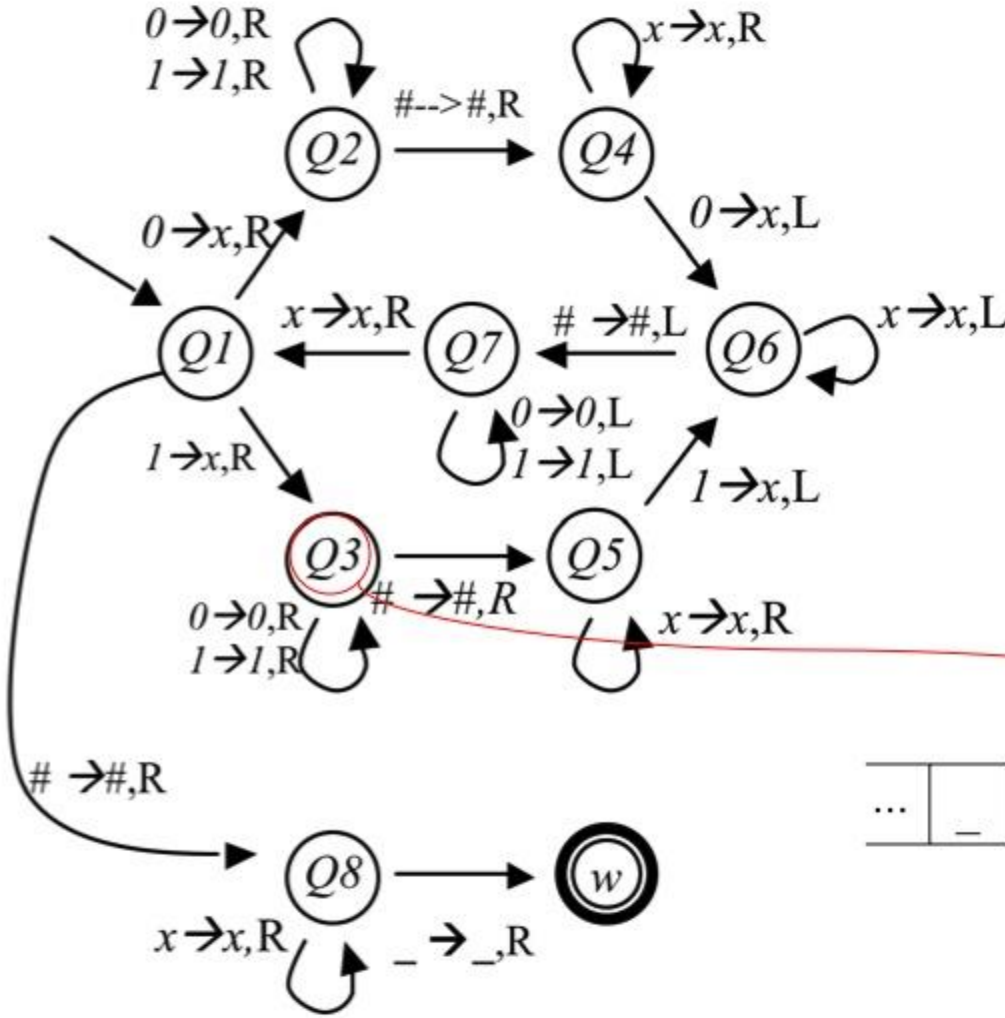
Input: 010#010
0→0,R
1→1,R
#-->#,R
x→x,R
0→x,R
0→x,L
x→x,R
→#,L
x→x,L
0→0,L
1→1,L
1→x,R
1→x,L
0→0,R
1→1,R
→#,R
x→x,R
→#,R
x→x,R
_ →_,R
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
w
... _ x 1 0 # x 1 0 _ ...

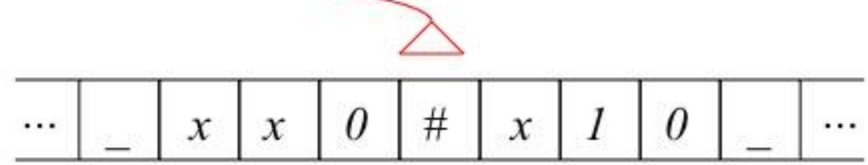


Input: *010#010*

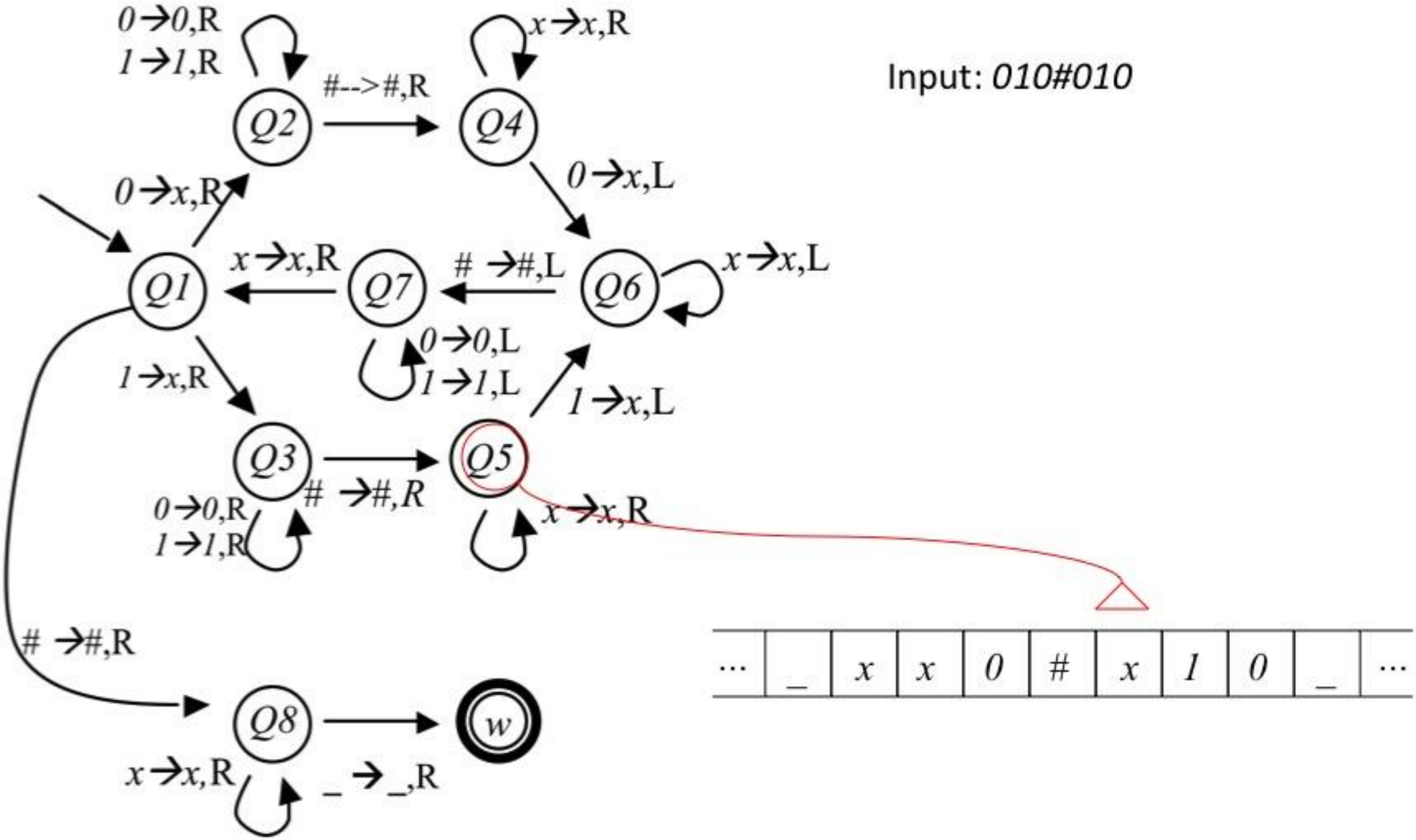


Input: *010#010*

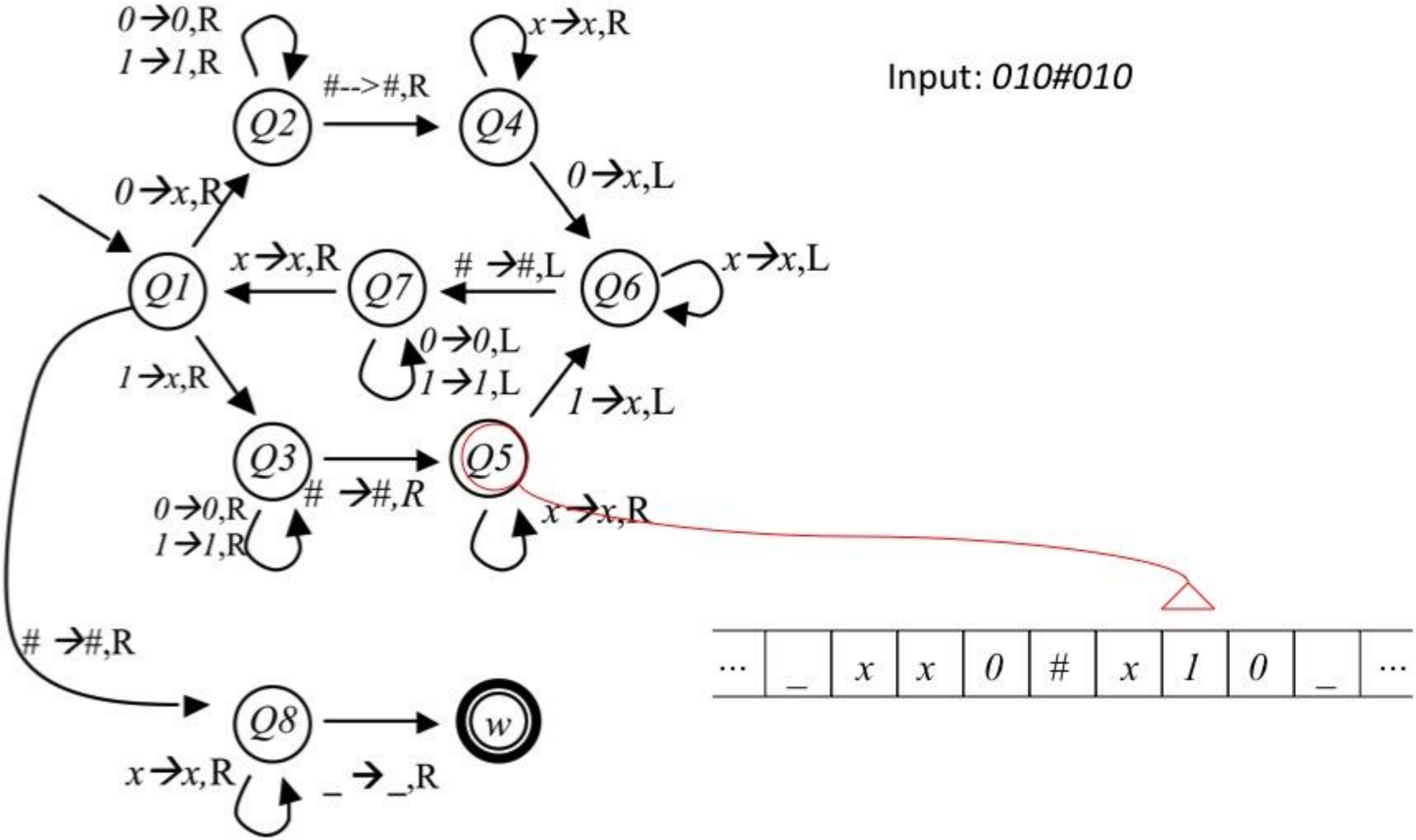




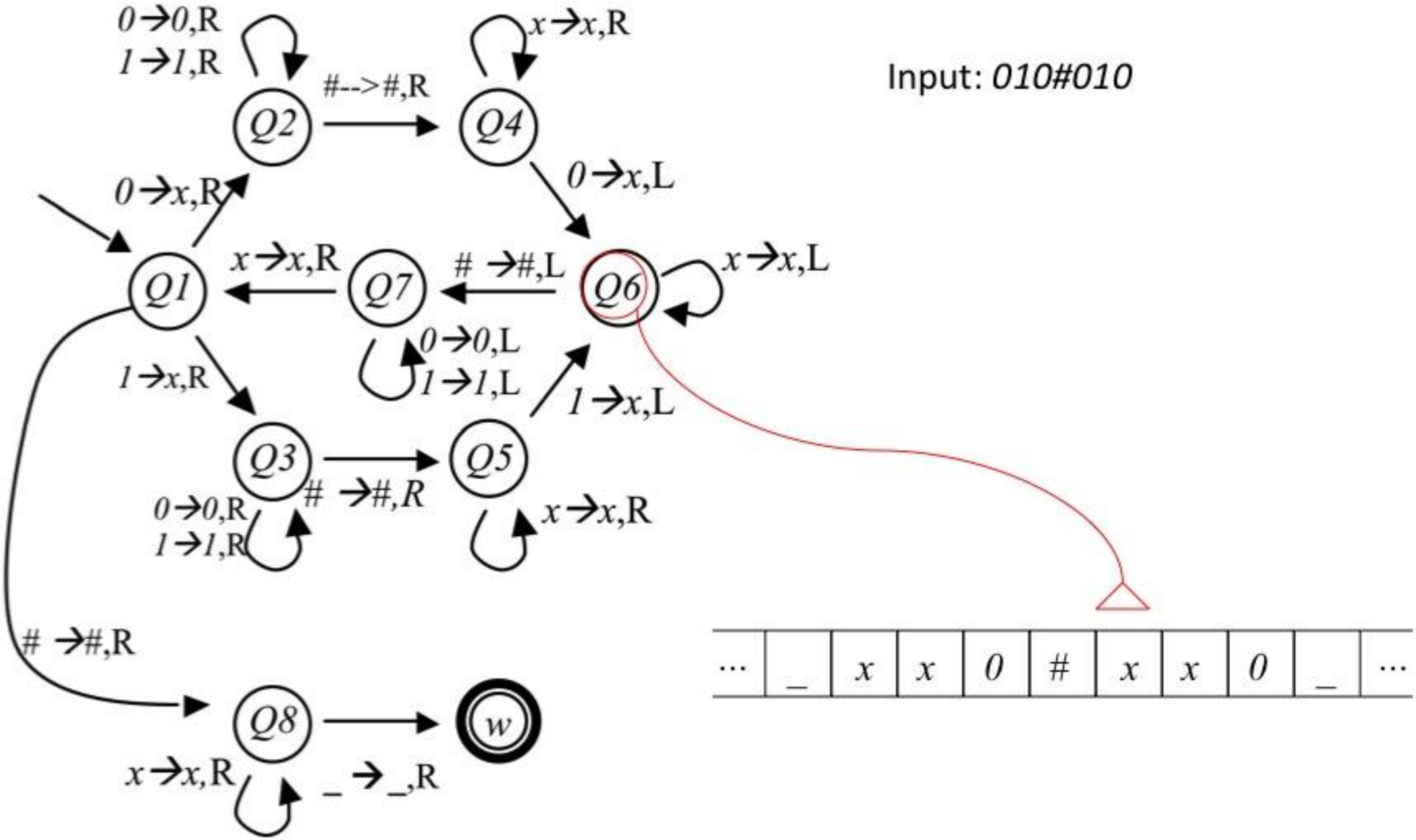
Input: *010#010*

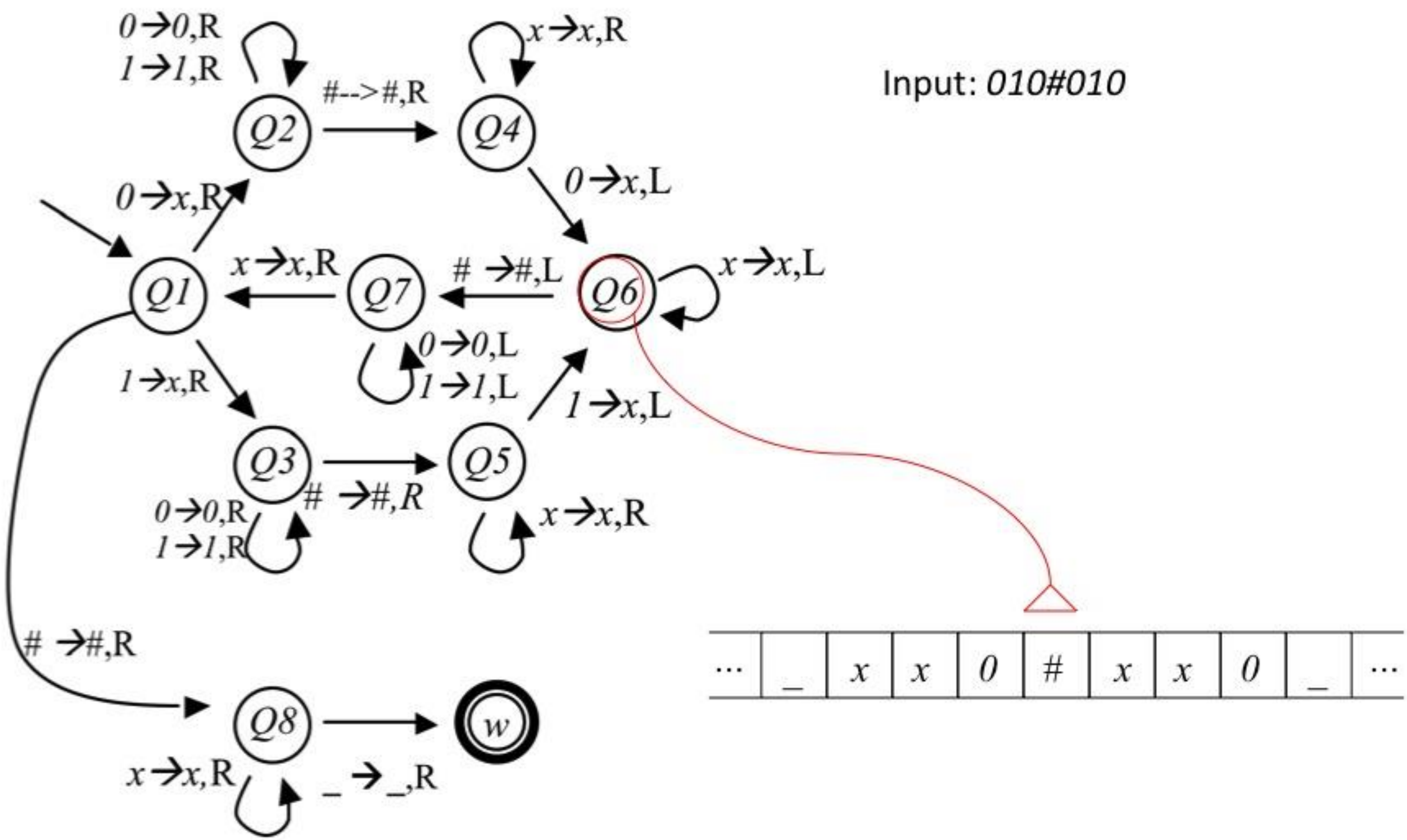


Input: *010#010*

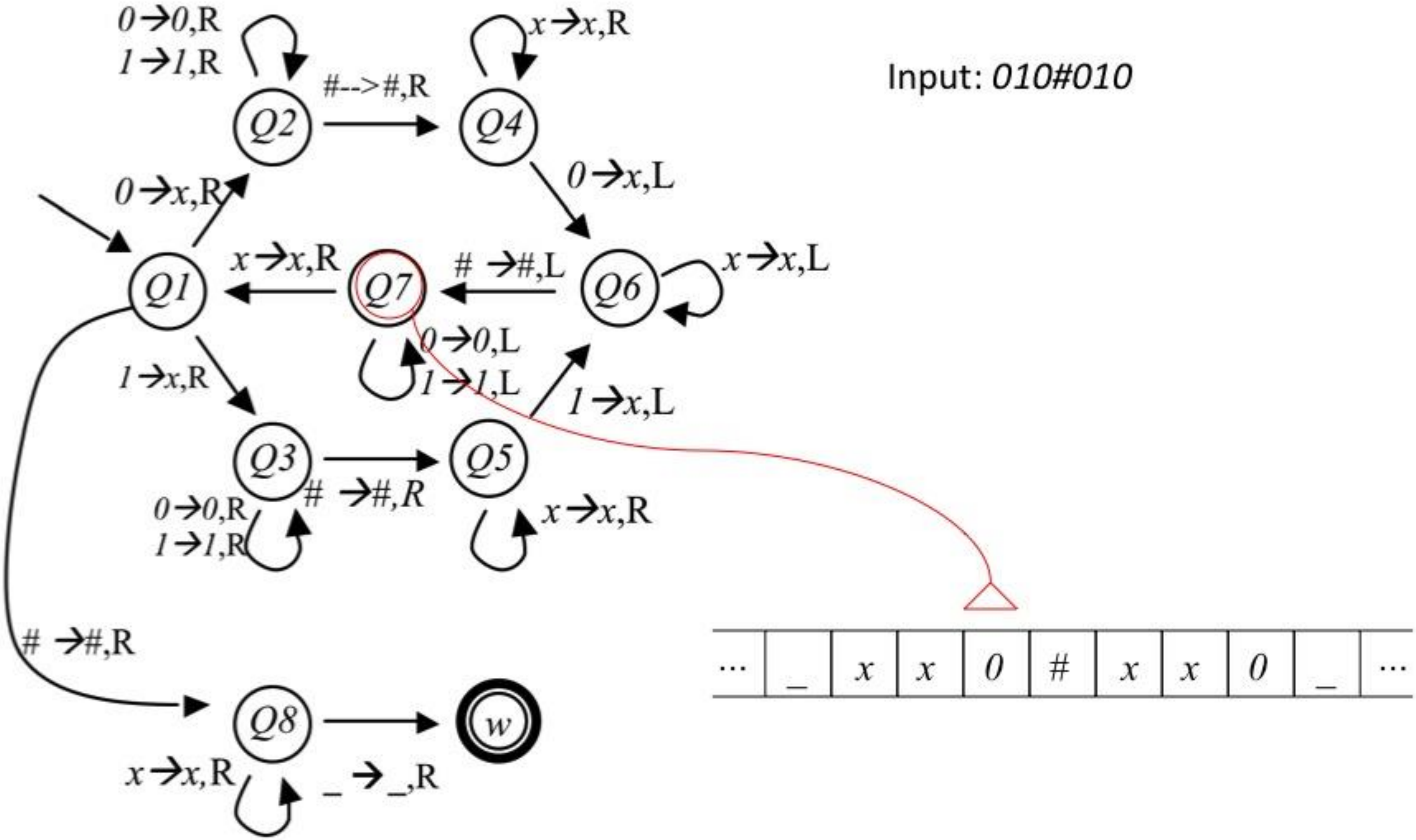


Input: *010#010*

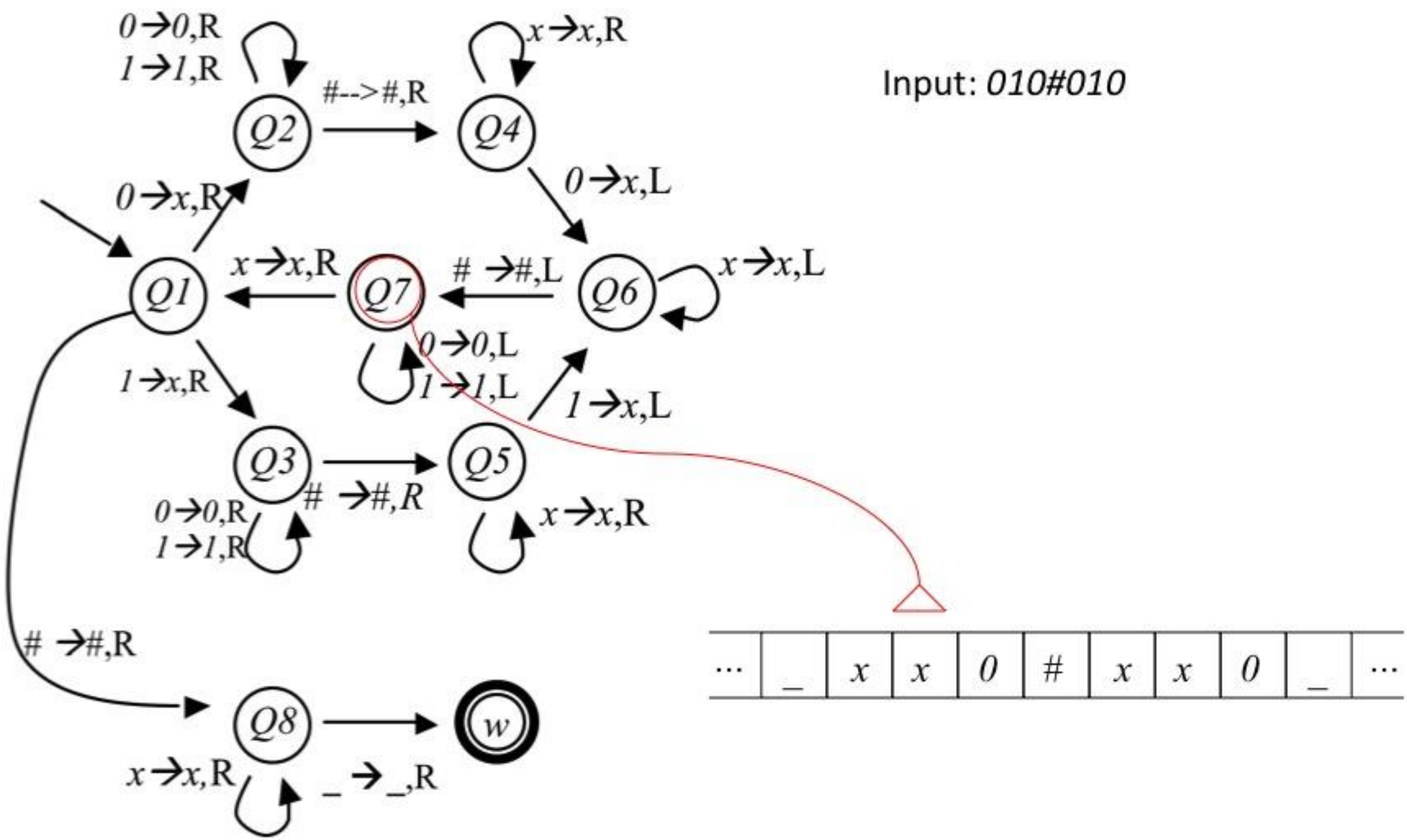



Input: 010#010
0→0,R
1→1,R
#-->#,R
x→x,R
0→x,R
0→x,L
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
w
x→x,R
→#,L
x→x,L
0→0,L
1→1,L
1→x,R
1→x,L
→#,R
0→0,R
1→1,R
x→x,R
→#,R
x→x,R
_ →_,R
... _ x x 0 # x x 0 _ ...

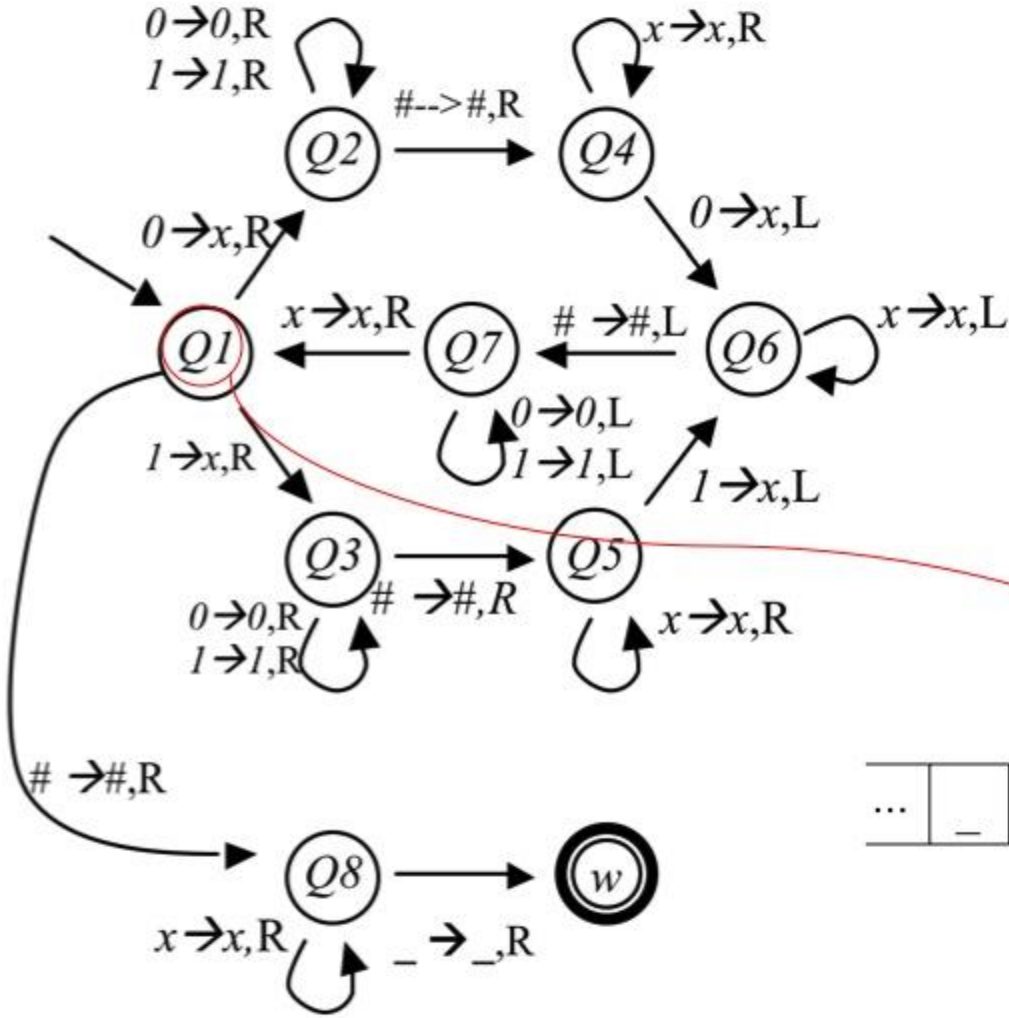
Input: *010#010*

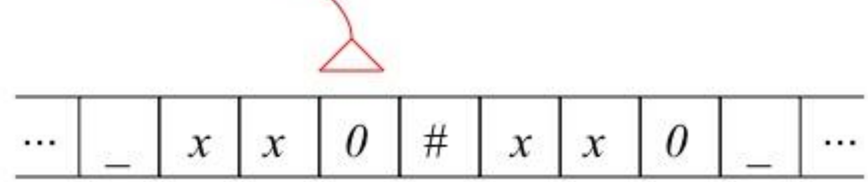


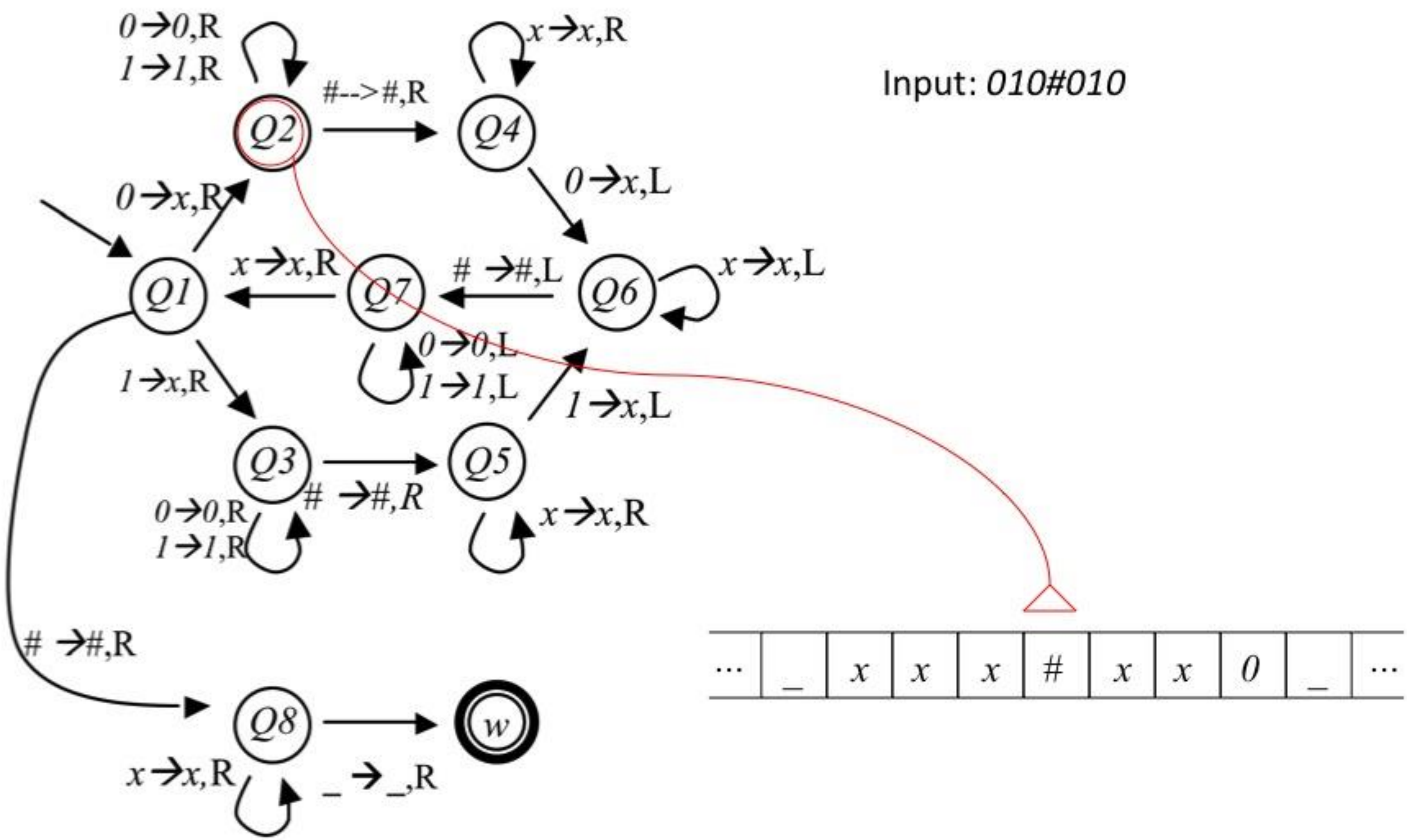
Input: *010#010*



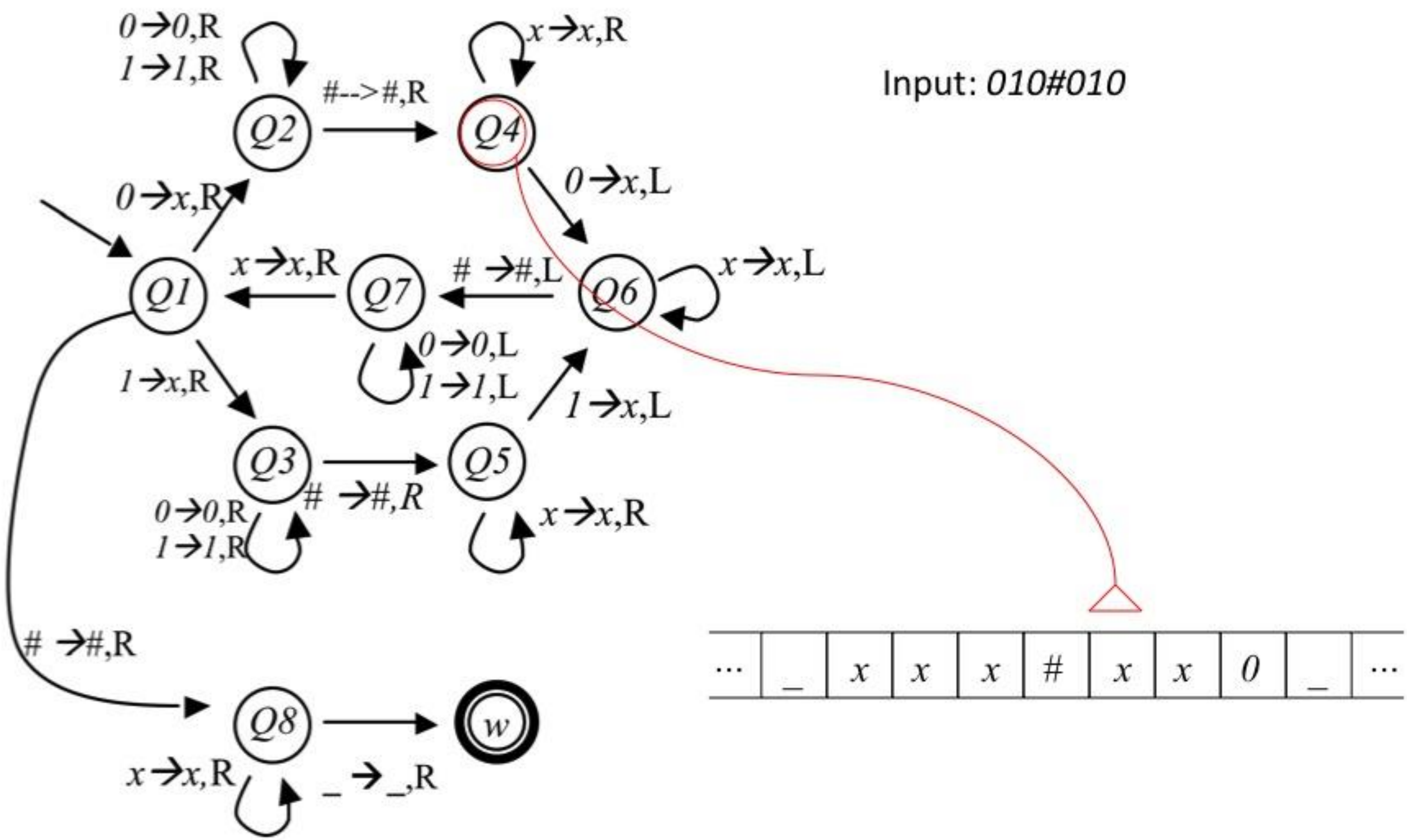
Input: *010#010*

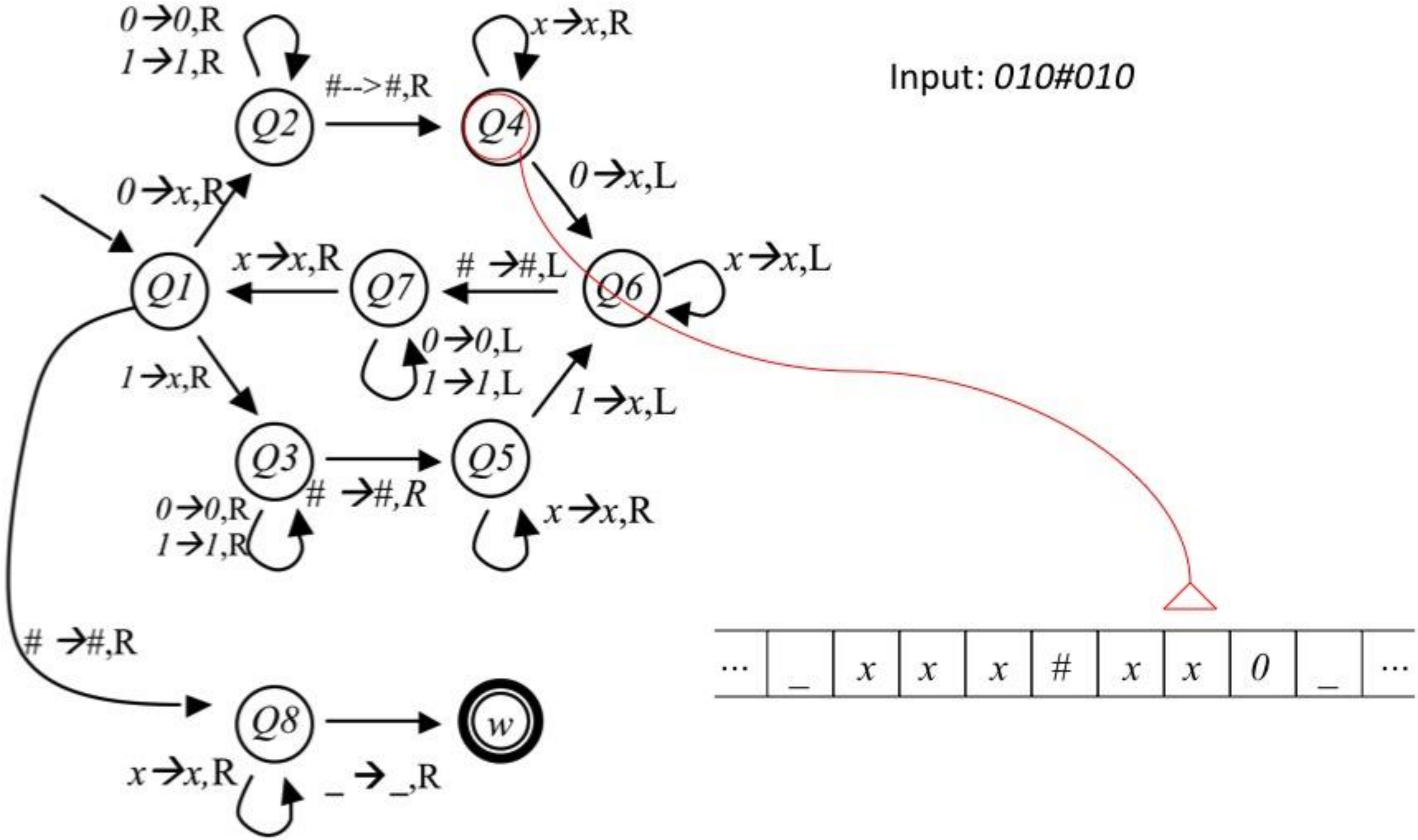




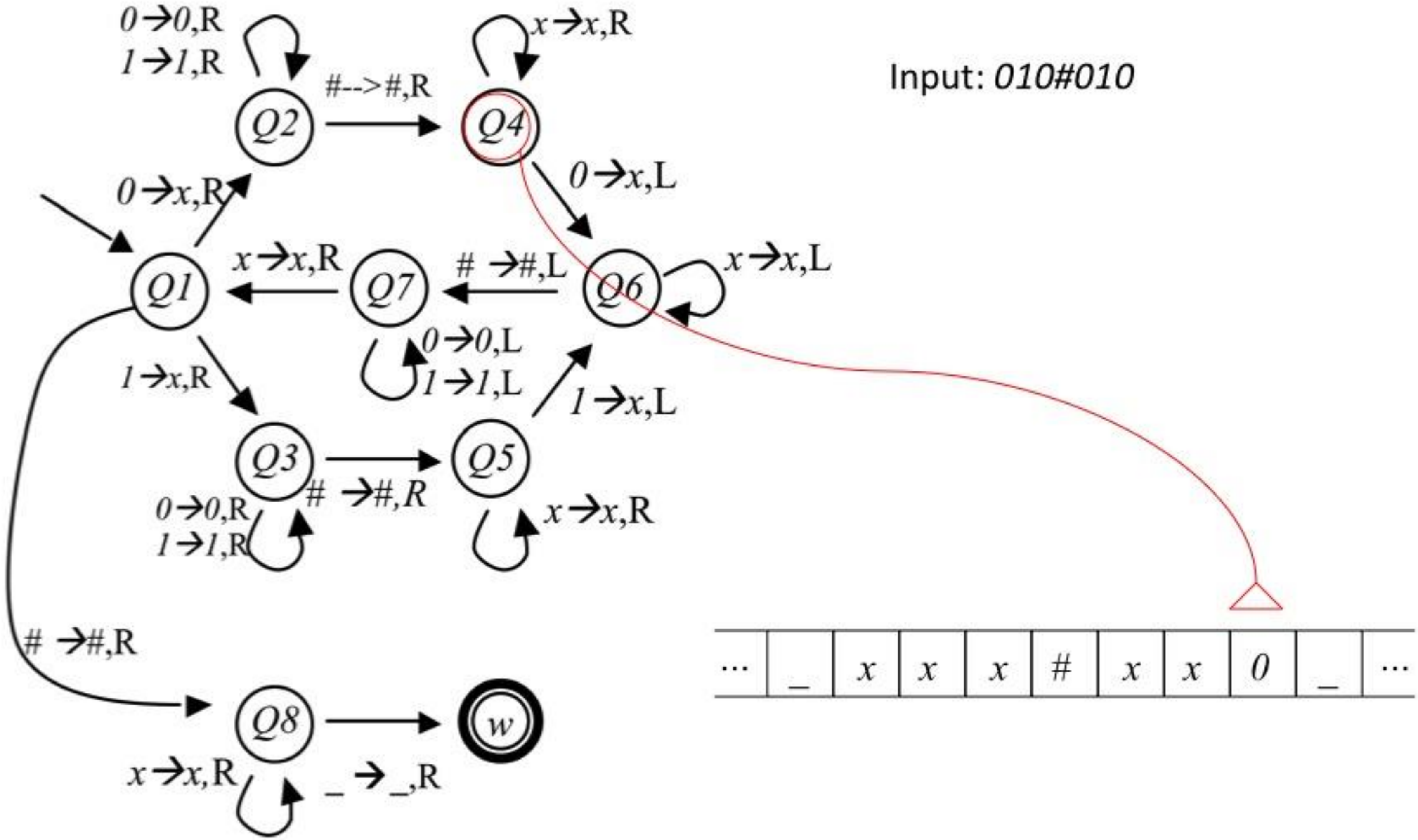

Input: *010#010*
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
w
0→0,R
1→1,R
#-->#,R
x→x,R
0→x,R
0→x,L
x→x,R
→#,L
x→x,L
0→0,L
1→1,L
1→x,R
1→x,L
→#,R
0→0,R
1→1,R
x→x,R
→#,R
x→x,R
_ →_,R
... _ x x x # x x 0 _ ...

Input: *010#010*

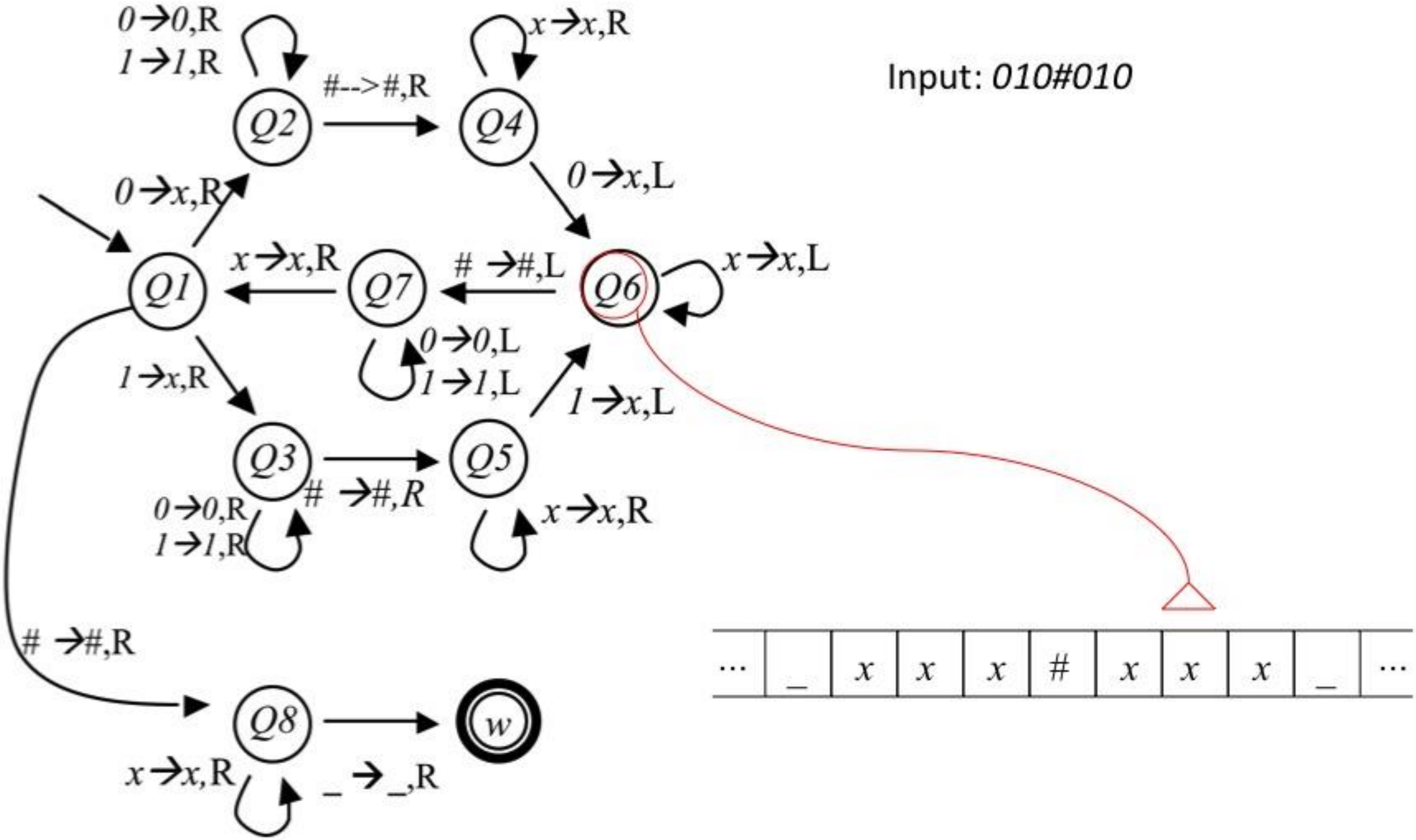


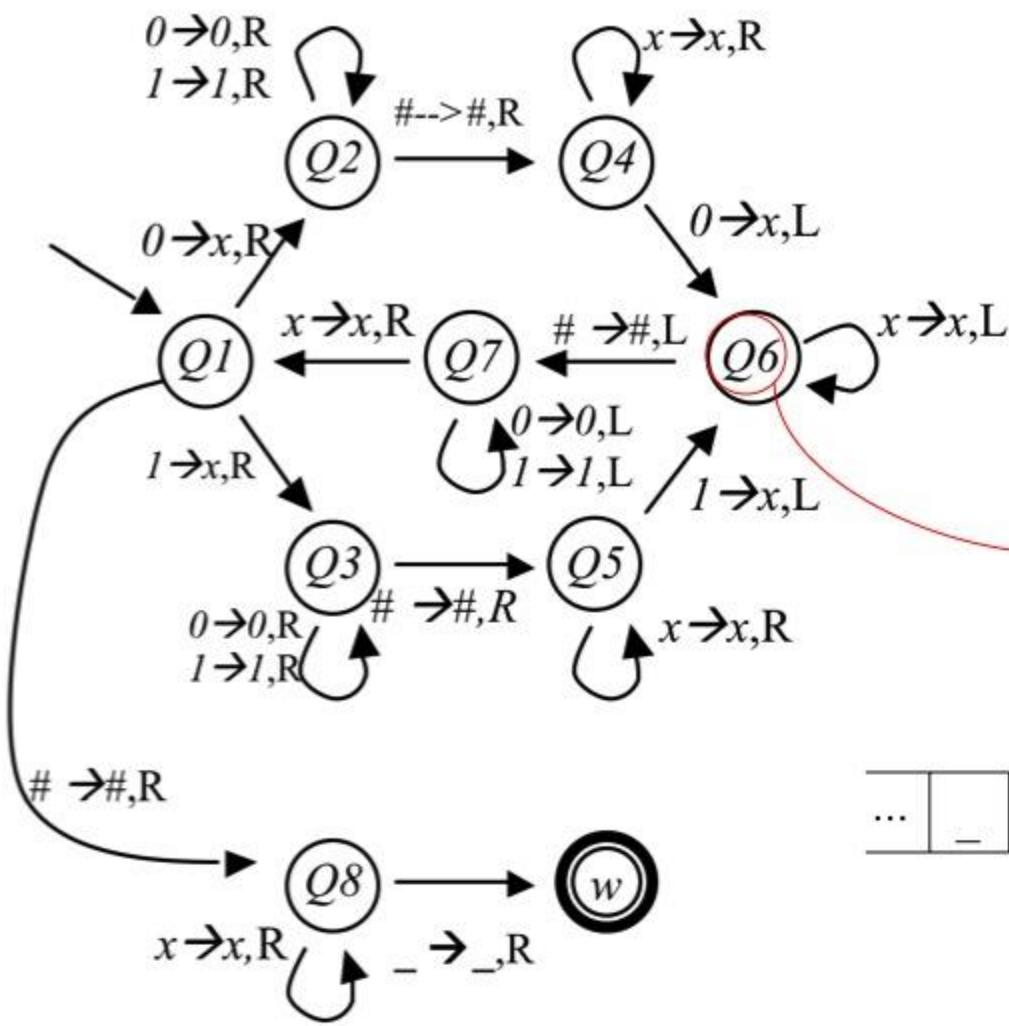

Input: 010#010
0→0,R
1→1,R
#-->#,R
x→x,R
Q2
Q4
0→x,L
0→x,R
x→x,R
→#,L
x→x,L
Q1
Q7
Q6
0→0,L
1→1,L
1→x,R
1→x,L
Q3
Q5
→#,R
0→0,R
1→1,R
x→x,R
→#,R
Q8
w
x→x,R
_ →_,R
... _ x x x # x x 0 _ ...

Input: *010#010*

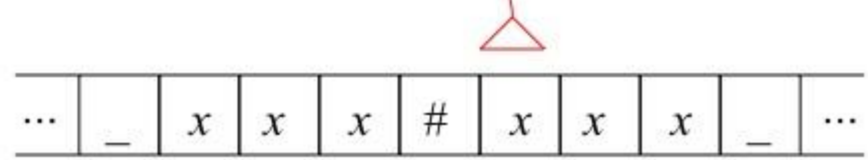


Input: *010#010*

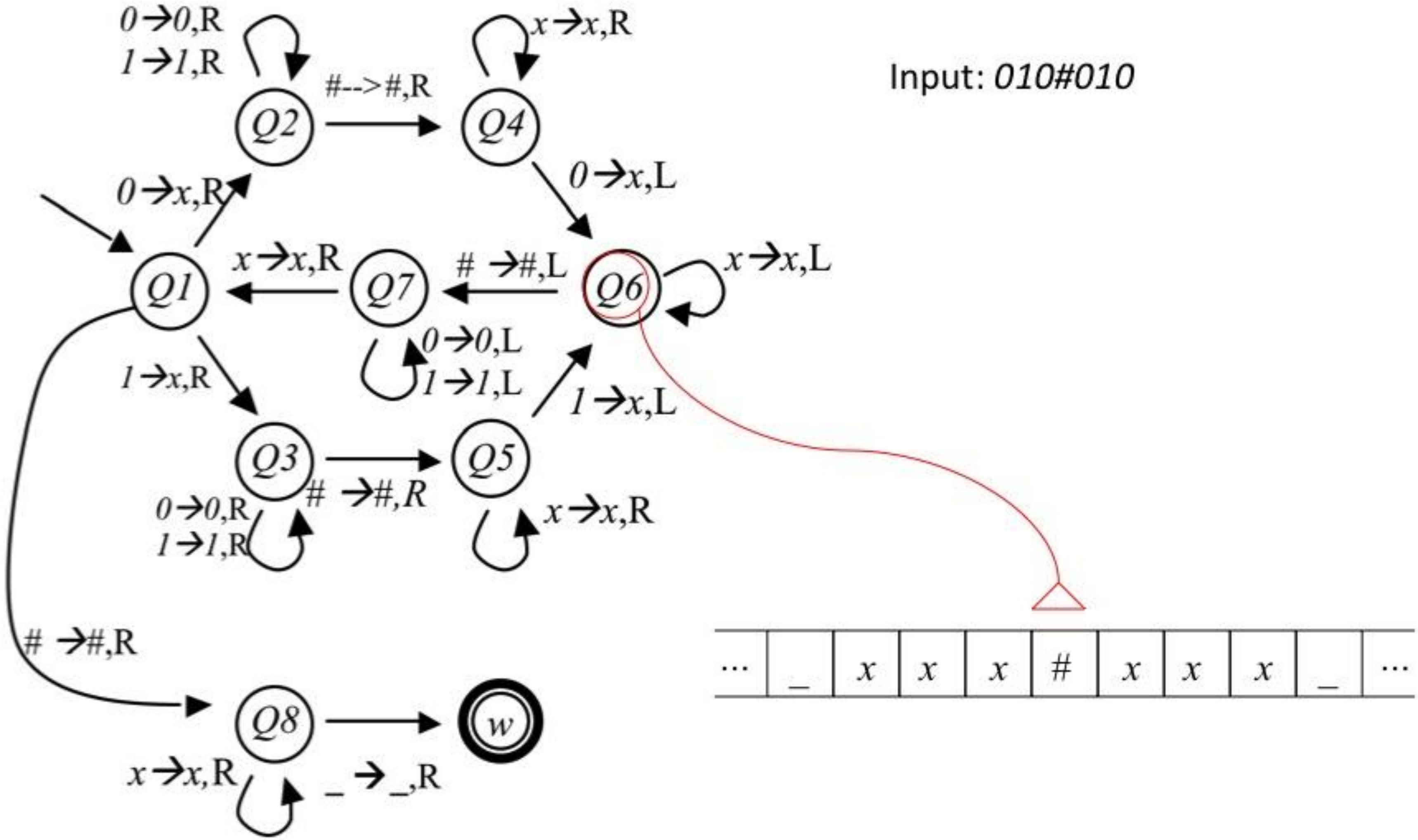




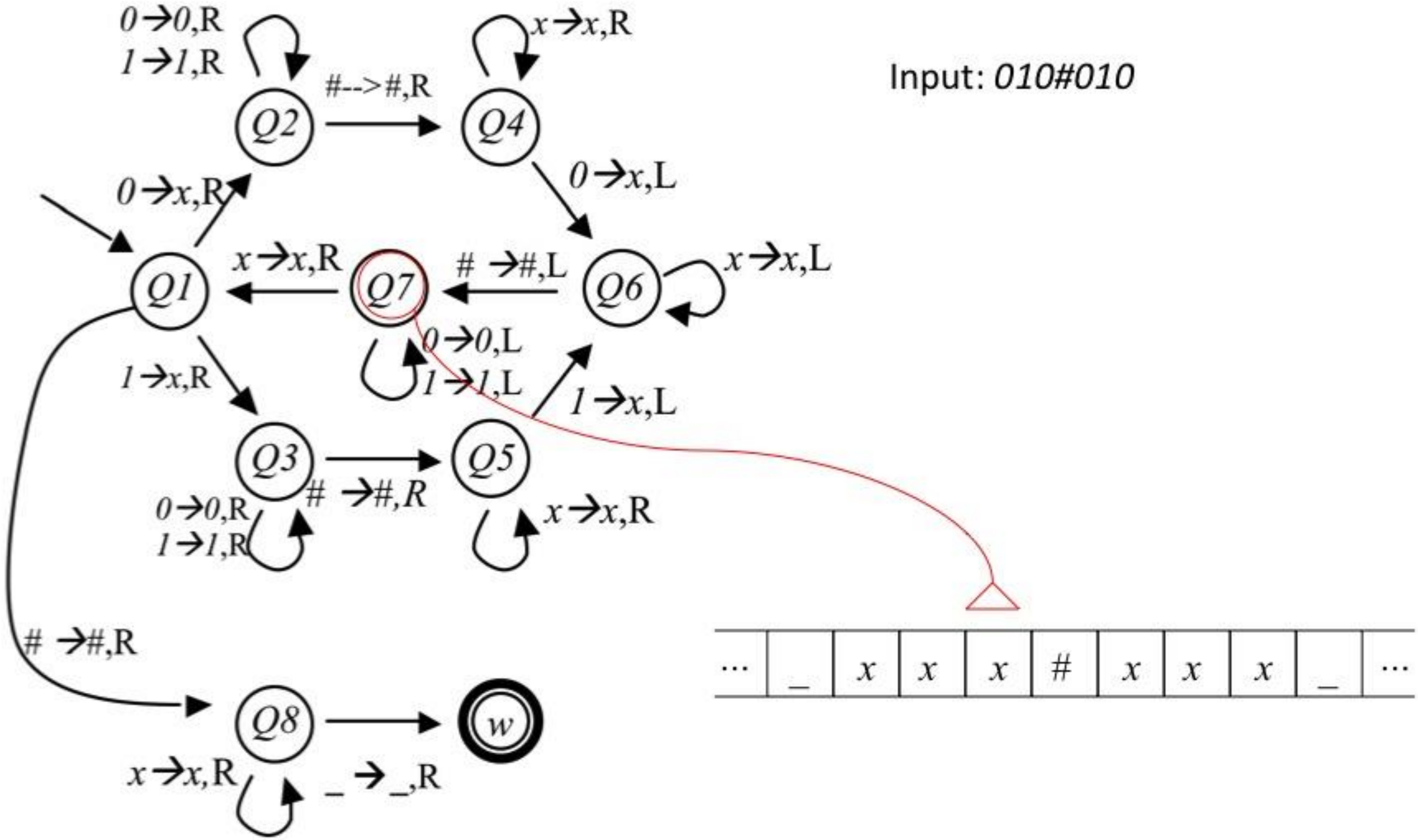
Input: *010#010*

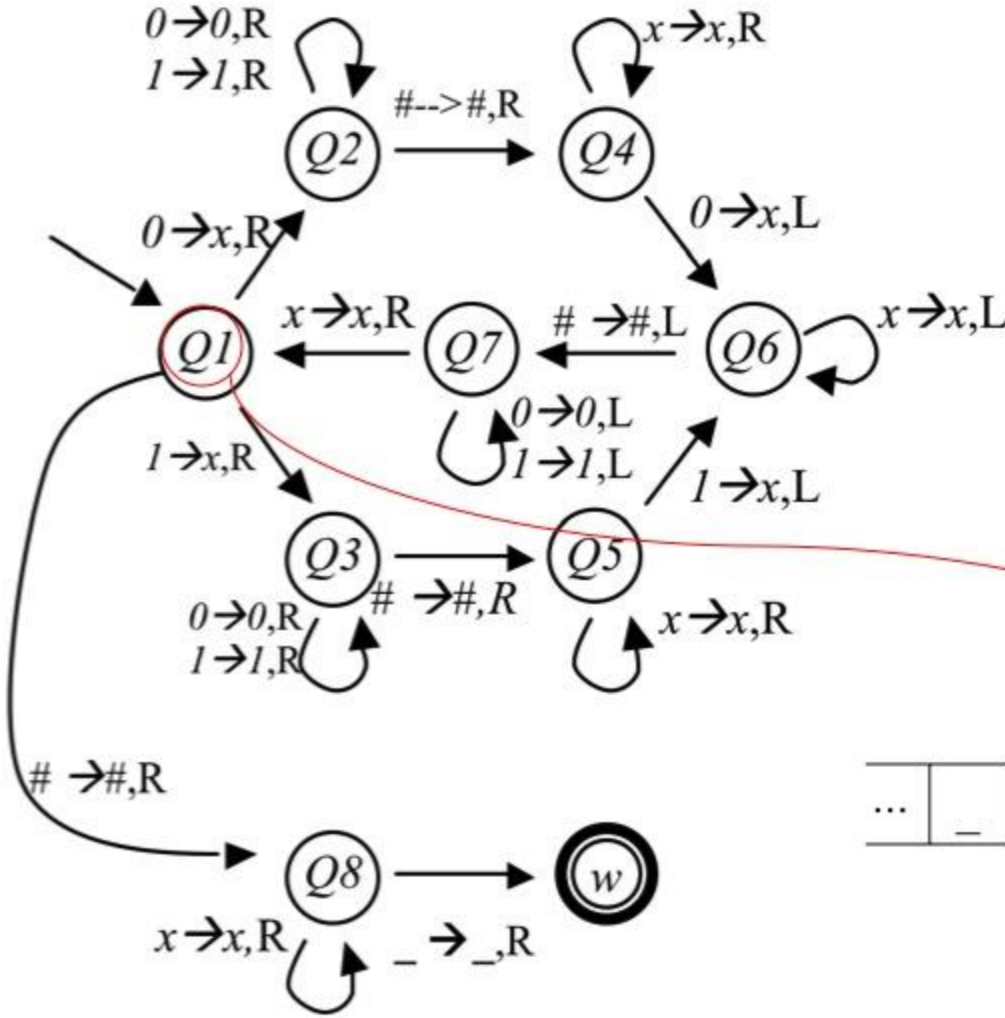


Input: *010#010*

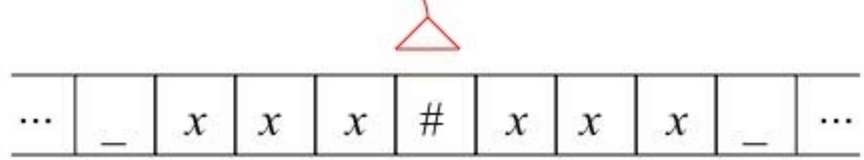


Input: *010#010*

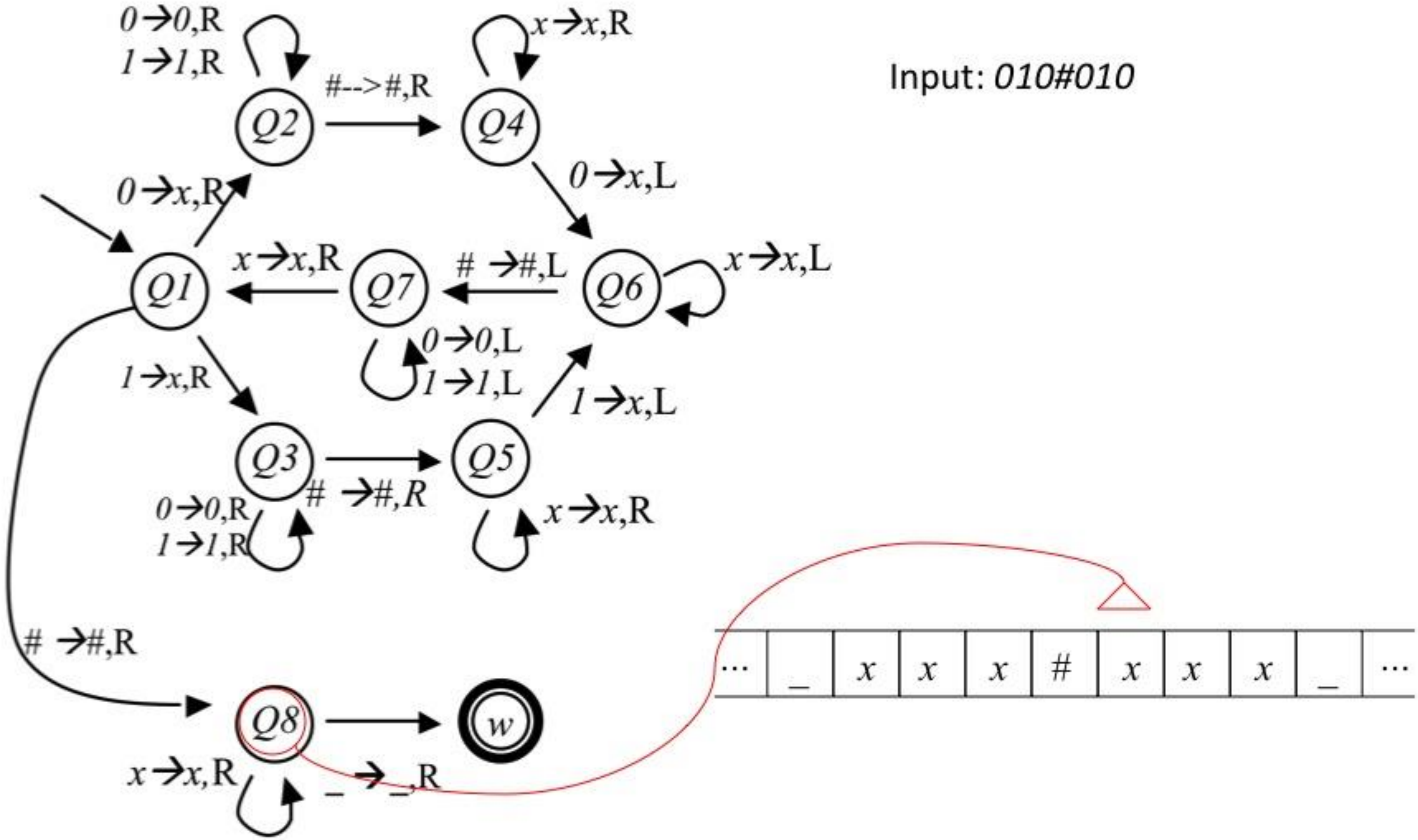




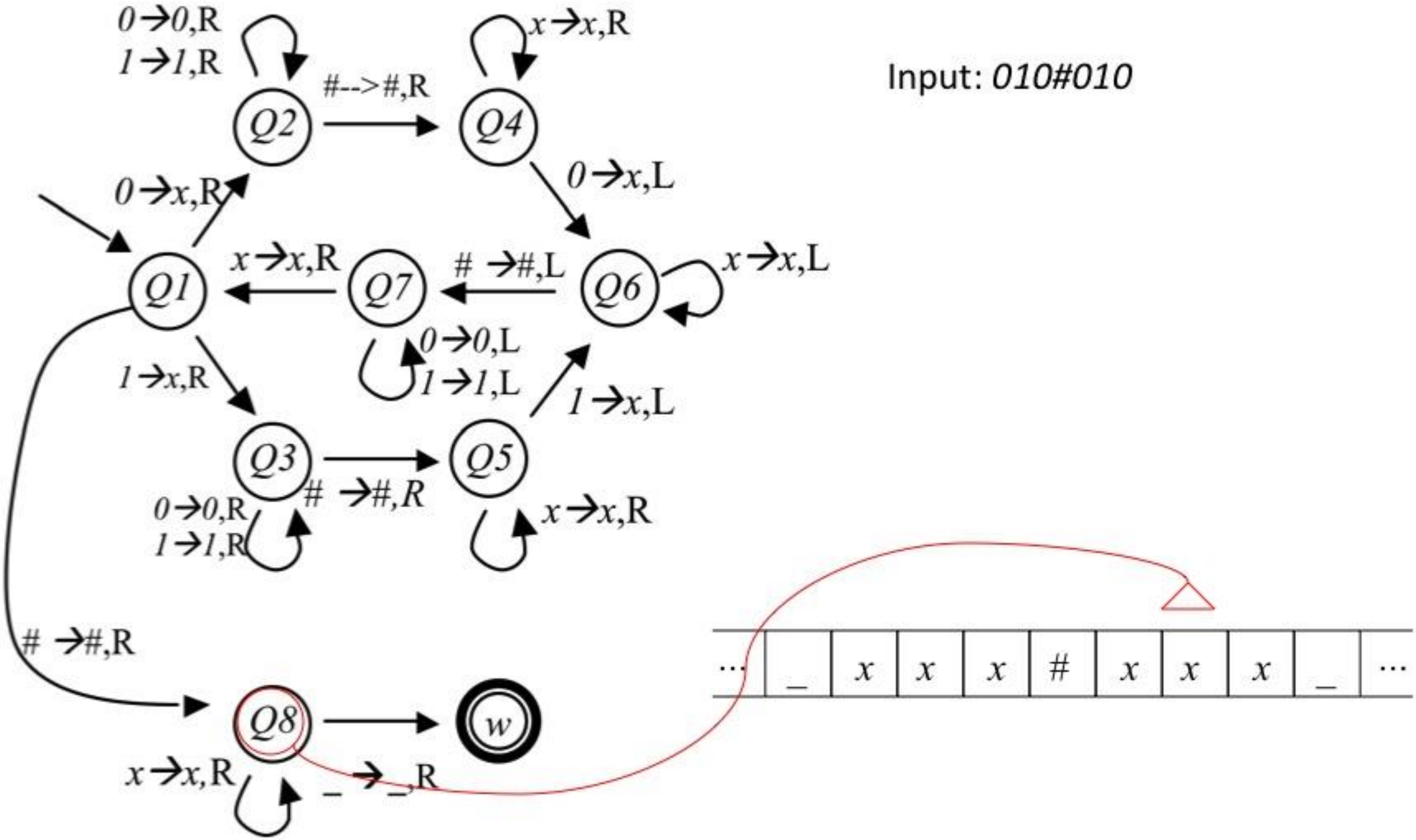
Input: *010#010*



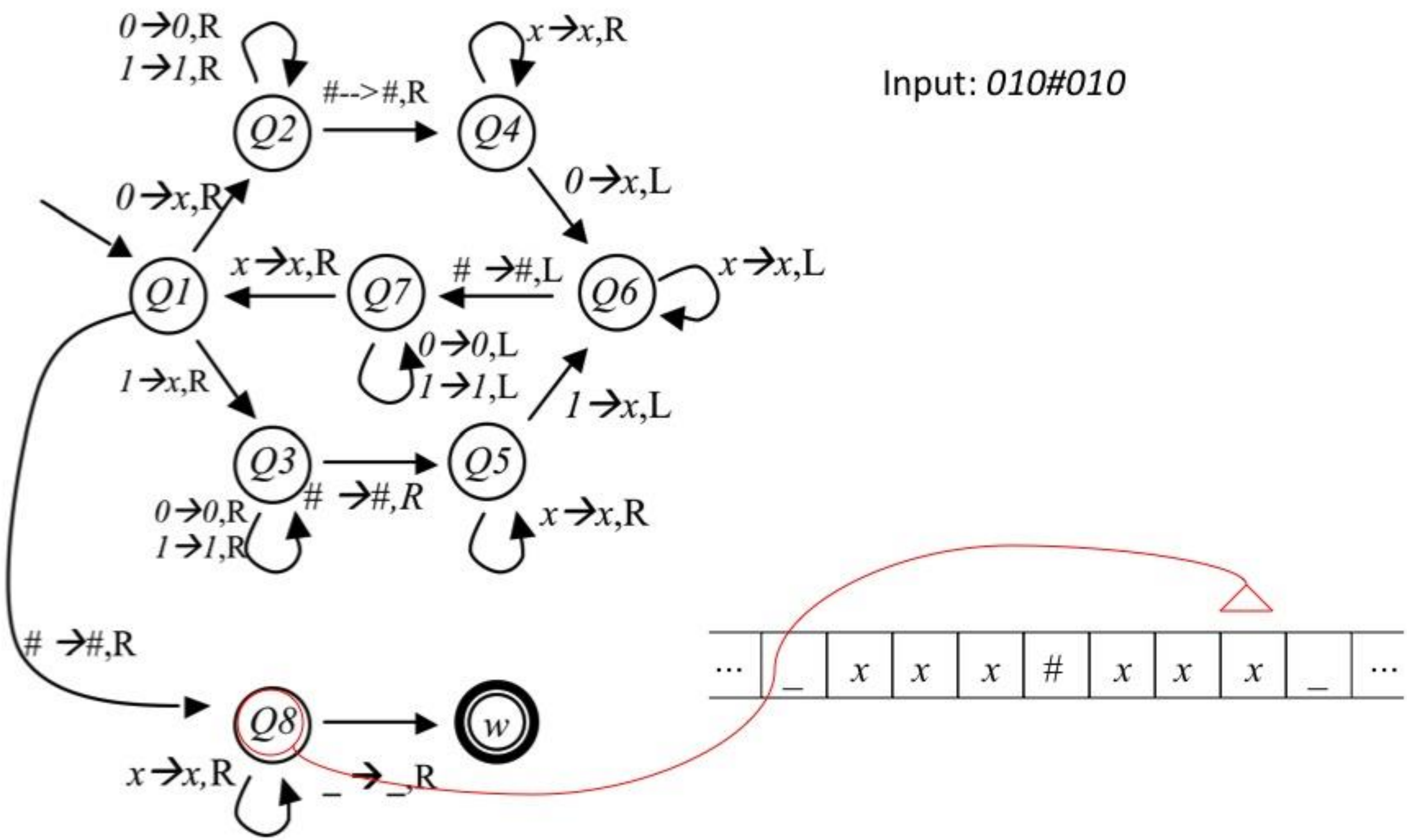
Input: *010#010*



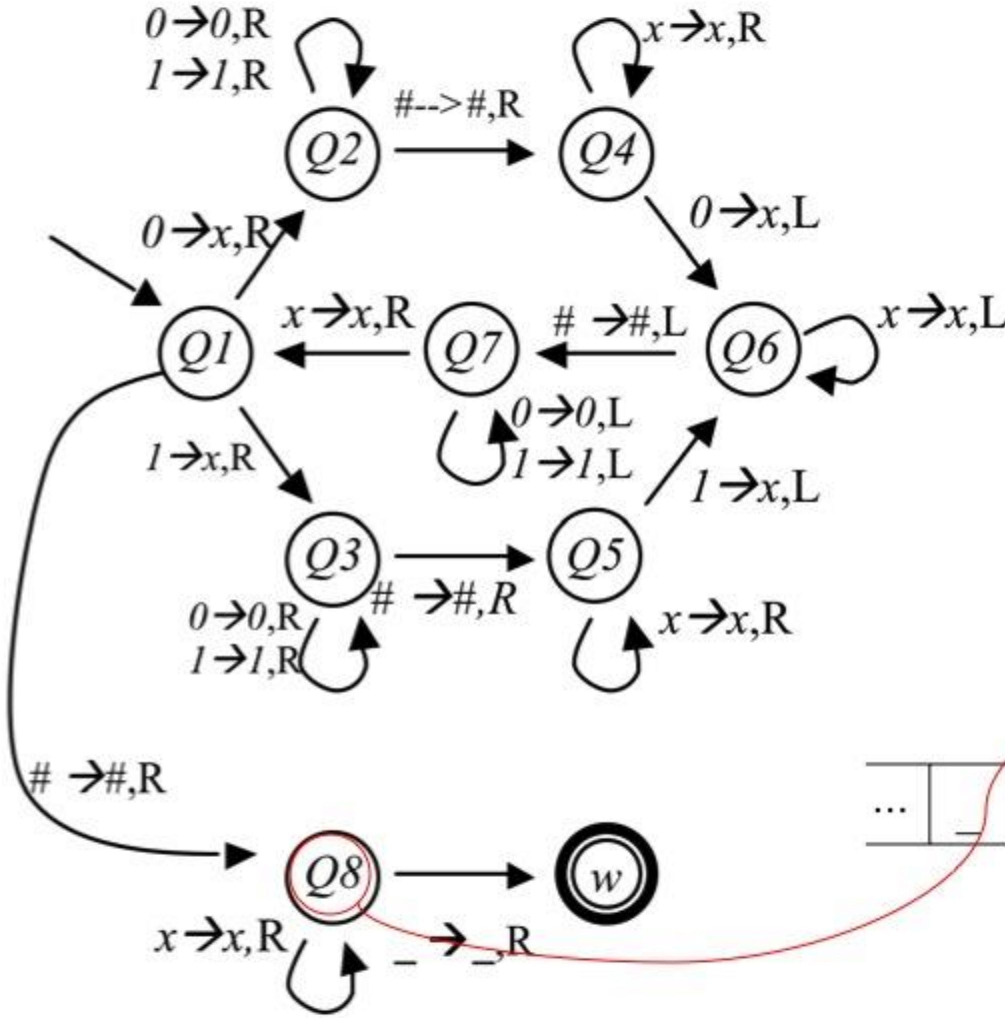
Input: *010#010*

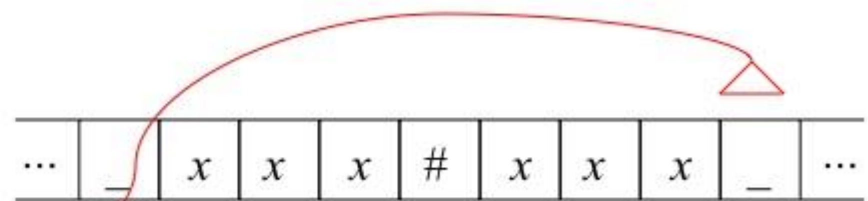


Input: *010#010*



Input: *010#010*





[A]**3.5** Examine the formal definition of a Turing machine to answer the following questions, and explain your reasoning.

a. Can a Turing machine ever write the blank symbol ␣ on its tape?

b. Can the tape alphabet Γ be the same as the input alphabet Σ?

c. Can a Turing machine's head *ever* be in the same location in two successive steps?

d. Can a Turing machine contain just a single state?

(svolto in aula)

Siete invitati a svolgere gli esercizi nelle prossime slides, che verranno discussi nelle prossime lezioni.

3.8 Give implementation-level descriptions of Turing machines that decide the following languages over the alphabet $\{0,1\}$.

[A] **a.** $\{w \mid w \text{ contains an equal number of 0s and 1s}\}$

b. $\{w \mid w \text{ contains twice as many 0s as 1s}\}$

c. $\{w \mid w \text{ does not contain twice as many 0s as 1s}\}$

Esercizio

Progettare una MdT che decida il linguaggio

$$L = \{ 0^{2^n} \mid n \geq 0\}$$

e che sia **concettualmente diversa** da quella vista nella scorsa lezione.

Esercizio

Progettare una MdT che calcoli la funzione f(x,y), differenza intera di due interi positivi x e y

$$f(x, y) = x - y \qquad \text{se } x \geq y$$
$$f(x, y) = 0 \qquad \text{altrimenti.}$$

Si supponga che l'input sia $< x > 0 < y >$, dove $< n > = 1^n$, è la rappresentazione unaria dell'intero positivo n.

1. Progettare una macchina di Turing che sposta l'input a destra di una casella.

2. Progettare una macchina di Turing che calcola il successore in binario.

3. Esercizi_ETC_lez2_definizioni.pdf