3. MACCHINE DI TURING

La ***Macchina di Turing*** è simile ad un ***automa finito***, ma con una memoria illimitata e senza restrizioni. Tuttavia, anche essa non è in grado di risolvere alcuni problemi.

Funzionamento della macchina:

Utilizza un nastro infinito come propria memoria illimitata. Ha una testina che è in grado di leggere e scrivere simboli ed è libera di muoversi lunga il nastro. Inizialmente il nastro contiene solo la stringa di input e tutto il resto è vuoto. Se la macchina deve memorizzare una informazione la può scrivere sul nastro. Per leggere l'informazione che ha scritto la macchina può spostare la testina su di essa.

La macchina continua a computare finchè non decide di produrre un output. Gli output ***accetta*** e ***rifiuta*** sono ottenuti occupando appositi ***stati di accettazione*** e ***rifiuto***. Se non raggiunge uno di questi stati la macchina andrà avanti per sempre.

Differenza tra un automa finito e macchina di Turing:

1. Una macchina di Turing può sia scrivere che leggere sul nastro;
2. La testina di lettura-scrittura può muoversi sia verso sinistra che verso destra;
3. Il nastro è infinito;
4. Gli stati speciali di accettazione e rifiuto sono immediati.

Definizione formale di macchine di Turing:

Una ***macchina di Turing*** è una 7-tupla (Q, $\sum$, Γ, f, q_0, q_{accept}, q_{reject}), dove Q, $\sum$, Γ sono tutti insiemi finiti e:

1. ***Q*** è ***l'insieme degli stati***;
2. $\sum$ è ***l'alfabeto di input*** non contenente il ***simbolo blank '_'***;
3. Γ è ***l'alfabeto del nastro*** con _ $\in \Gamma$ e $\sum \subseteq \Gamma$ contenente tutti i simboli che possono essere scritti all'interno di una cella di memoria (tecnicamente, l'unione tra l'alfabeto di lavoro e l'insieme dei simboli non processabili dalla macchina, come _);
4. ***F: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$*** è la ***funzione di transizione***, dove le lettere lette sono all'interno del nastro (in ogni istante si hanno dei simboli nelle varie celle, e la cella a cui punta la testina rappresenta lo stato q in cui si trova la macchina).;
5. $q_0 \in Q$ è lo ***stato iniziale***;
6. $q_{accept} \in Q$ è lo ***stato di accettazione***;
7. $q_{reject} \in Q$ è lo ***stato di rifiuto***, con $q_{reject} \neq q_{accept}$.

Nota: q_{accept} e q_{reject} sono stati di arresto.

Sia M una Macchina di Turing definita da (Q, Σ, Γ, δ, q_0, q_{accept}, q_{reject}); allora:
- ad ogni istante, M occupa uno degli stati in Q;
- la testina si trova in un quadrato del nastro contenente un qualche simbolo $\gamma \in \Gamma$;
- la funzione di transizione δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ dipende dallo stato q e dal simbolo di nastro γ.

Il *range* della funzione di transizione sono triple (q', γ', d), con:
- $q' \in Q$;
- $\gamma' \in \Gamma$ è il simbolo scritto dalla testina sulla cella del nastro su cui la testina si trova *all'inizio* della transizione;
- $d \in \{L, R\}$ è la direzione in cui la testina muove un passo, con L = *left* ed R = *right*.

Esempio1:

Dato l'esempio posto a destra, si ha che i passi 1, 2, 3, per passare dal tempo 0 al tempo 1, sono rappresentati da una funzione di transizione $\delta(q, a) = (r, k, L)$, con q ed r stati generici per supposizione.

Esempio2:

Dato l'esempio posto a destra, se la testina puntasse alla seconda cella, scrivesse *f* e si muovesse a destra, allora avremmo $\delta(r, b) = (r', f, R)$, con r ed r' stati generici per supposizione.

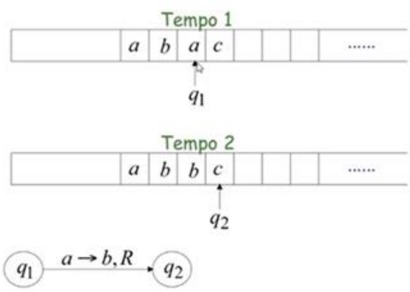
La computazione parte sempre da uno stato iniziale q_0, con l'input scritto sul nastro: quest'ultimo inizialmente contiene l'input, ed in particolare l'inizio dell'input è scritto all'inizio del nastro. La testina si trova nella prima cella a sinistra del nastro (cella 0). A questo punto, la macchina è pronta per l'esecuzione, ed effettuerà le transizioni in accordo alla funzione δ.

Esempio1:

Con questo esempio, vediamo come la funzione δ può essere rappresentata in modo compatto all'interno del diagramma degli stati.

La figura a destra dice che la testina, al tempo 1, si trova nello stato q_1 e sulla cella 2, dove è scritto il simbolo 'a'; al tempo 2, la testina passa allo stato q_2 e si trova sulla cella 3, dove è scritto il simbolo 'c', mentre nella cella dove essa puntava al tempo 1 è stato scritto il simbolo 'b'.

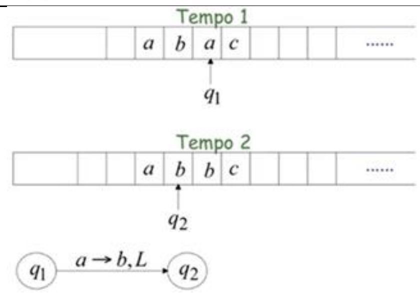
Ciò è sintetizzabile secondo la transizione in basso nella figura, che equivale a $\delta(q_1, a) = (q_2, b, R)$. Con $a \rightarrow b$ indichiamo che la nuova lettera scritta sarà *b*, ci si muove verso *R* e si passa da q_1 a q_2.



Esempio2:

La figura a destra dice che la testina, al tempo 1, si trova nello stato q_1 e sulla cella 2, dove è scritto il simbolo 'a'; al tempo 2, la testina passa allo stato q_2 e si trova sulla cella 1, dove è scritto il simbolo 'b', mentre nella cella dove essa puntava al tempo 1 è stato scritto il simbolo 'b'.

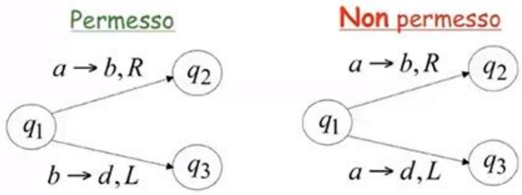
Ciò è sintetizzabile secondo la transizione in basso nella figura, che equivale a $\delta(q_1, a) = (q_2, b, L)$. Con $a \rightarrow b$ indichiamo che la nuova lettera scritta sarà *b*, ci si muove verso *L* e si passa da q_1 a q_2.



La computazione termina quando *M* raggiunge: uno stato accettazione q_{accept} (che implica una **Computazione Accept**), o uno stato rifiuto q_{rejeft} (che implica una **Computazione Reject**).

Macchina di Turing Deterministica:

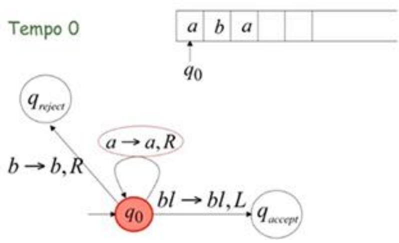
Parliamo di Macchina di Turing deterministica, in cui *non ci sono ε-transition*: nella funzione di transizione, *per ogni stato e per ogni lettera esiste uno ed un solo risutato*.
Nella figura a destra, notiamo che <u>non</u> è permesso che, nello stato q_1, leggendo *a* si possa scrivere o *b* o *d* e muoversi o verso *R* o verso *L*; ciò avviene in quanto abbiamo due possibili mosse per $\delta(q_1, a)$.



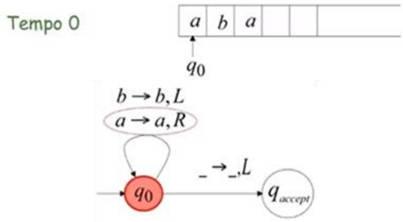
Gli **stati di arresto** non hanno archi uscenti: per definizione, in uno stato di arresto (accept o reject) la computazione termina.

Una macchina di Turing può accettare, non accettare o avere problemi con una stringa

Esempio1:
L'esempio di cui a destra mostra un caso in cui la stringa non viene accettata.
L'alfabeto di lavoro è Σ = {a, b}, mentre l'alfabeto di del nastro è Γ = {a, b, _}.
Possiamo denotare il blank sia con _ che con *bl*. Lo stato iniziale è q_0: se si trova in q_0 e legge 'a', allora non si cambia il contenuto della cella e la testina si sposta a R, e non si ha un cambio di stato; se si trova in q_0 e legge 'b', allora non si cambia il contenuto della cella e la testina si sposta a R, passando allo stato q_{reject}; se si trova in q_0 e legge '*bl*', allora non si cambia il contenuto della cella e la testina si sposta a L, passando allo stato q_{accept}. Si noti che la stringa, in questo esempio, viene rifiutata leggendo solo i primi due simboli, per cui non è necessario arrivare a fine stringa per determinare l'accettazione o il rifiuto. Se avessimo avuto in input la stringa aaa__, allora la stringa sarebbe stata accettata.



Esempio2:
Nell'esempio di cui a destra, l'alfabeto di lavoro è Σ = {a, b}, mentre l'alfabeto di del nastro è Γ = {a,b,_}.
Lo stato q_{accept} non è raggiungibile, in quanto si entra in un loop in cui si leggono sempre i primi due simboli. La computazione non termina, di conseguenza l'input non viene accettato.
Ciò mostra che la macchina di Turing può andare in loop: questo è uno dei grandi problemi che evidenzia che esistono problemi non risolvibili con un calcolatore.

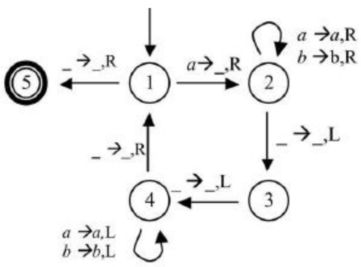


Esiste un'***eccezione*** nella funzione di transizione, secondo cui se ci troviamo all'inizio del nastro (prima cella) e leggiamo un simbolo qualsiasi per poi muoverci a sinistra, il risultato sarà sempre che la testina punterà alla prima cella. Nell'esempio precedente, se avessimo avuto in input la stringa bb_ avremmo ottenuto che la funzione $\delta(q_0, b) = (q_0, b, L)$ non avrebbe spostato la testina, che quindi punterà sempre alla cella corrente (e, in questo esempio, entrerebbe comunque in un loop). Questo non è considerato un errore.

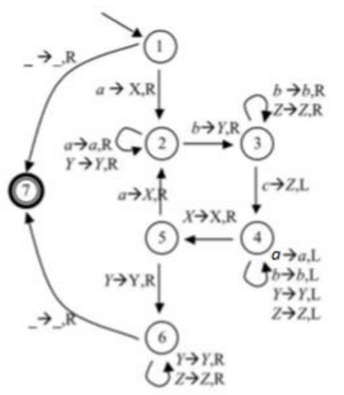
Esempio1:
Strategia per accettare $\{a^n b^n \mid n \geq 0\}$)**.** Si consideri l'alfabeto $\Sigma = \{a^n b^n \mid n \geq 0\}$. Possiamo *cancellare ripetutamente* la prima occorrenza di *a* e l'ultima occorrenza di *b*: se la stringa appartiene all'alfabeto Σ, allora non rimangono simboli. Tutto ciò lo possiamo formulare in cinque passi:

1. Se leggi _, vai al punto 5. Se leggi a, scrivi _ e vai al punto 2;
2. Spostati a destra (R) di tutti a e b. Al primo _, muovi a sinistra (L) e vai al punto 3;
3. Se leggi b, scrivi _ e vai al punto 4;
4. Spostati a sinistra (L) di tutti a e b. Leggendo _, muovi R e vai al punto 1;
5. Accept.

La figura a destra mostra una macchina di Turing che implementa tale algoritmo.
Lo stato *reject* manca: spesso, questo non viene spesso considerato per non avere un numero elevato di stati. Prima di costruire la MdT, si può assumere che *tutte le transizioni che non compaiono vanno implicitamente in uno stato, non inserito nel diagramma, di rifiuto*. Ad esempio, notiamo che se siamo nello stato 3 e leggiamo a, allora non c'è una transizione: in questo caso, si assume che la stringa viene rifiutata.



Esempio2:
La seguente è una MdT per stringhe $a^n b^n c^n$. Dunque, operiamo sull'alfabeto Σ = {a, b, c}. Inoltre, l'alfabeto del nastro è Γ = {_, a, b, c, X, Y, Z}.
Il "cammino" computazionale di tale macchina può riassumersi in 4 passi:
1) cancella la prima *a*. In questo esempio, sovrascriviamo le *a* con una *X*;
2) avanza (scorrendo la stringa), e cerca la prima occorrenza di *b*. Quando viene trovata la *b*, la sovrascriviamo con una *Y*;
3) avanza (scorrendo la stringa), e cerca la prima occorrenza di *c*. Quando viene trovata la *c*, la sovrascriviamo con una *Z*;
4) ritorna al passo 1).
Ad esempio, si riceve in input la stringa aabbcc. Al primo passo, si sostituisce la prima *a* con una *X*, e ci si sposta a destra; successivamente, si avanza fin quando non si trova una *b* (in questo caso, soltanto una volta). Trovata la *b*, la si sovrascrive con *Y* e ci si sposta a destra; successivamente, si avanza fin quando non si trova una *c* (in questo caso, soltanto una volta). Trovata la *c*, la si sovrascrive con *Z* e ci si sposta a sinistra. A questo punto, l'iterazione è finita e la testina può iniziare a spostarsi a sinistra per tornare all'inizio del nastro e ad effettuare una nuova iterazione (il passo 4 serve proprio a tornare indietro sul nastro, per qualsiasi simbolo letto a parte una *X*, che inizia a far muovere la testina verso destra).
Anche in questo esempio, ogni transizione da uno stato ad un altro non inserita, se si verifica porta la macchina in uno stato q_{reject} che rifiuta la stringa.



3.1 CONFIGURAZIONI MACCHINA DI TURING

Una **configurazione** di una Macchina di Turing è una descrizione concisa di stato e contenuto del nastro. Trattasi di una stringa C = *uqv*, dove:
- *q* è lo *stato* occupato dalla macchina M;
- *uv* è il *contenuto del nastro* (sinistra – destra);
- la testina punta sul primo (cioè, più a sinistra) simbolo di *v* (su primo blank _ se v = ε);
- dopo *v* sono presenti solo simboli blank _.

<u>*Esempio1*</u>:
Dato l'esempio precedente (MdT per stringhe $a^nb^nc^n$), al tempo in cui il nastro contiene la stringa XXYYZZ e punta alla seconda cella (cella 1, contenente l'ultimo X), la configurazione di MdT in questo tempo è
C = X4XYYZZ, dove *u* = X, *q* = 4, *v* = XYYZZ.
A partire da questa configurazione C, sapendo che δ(4, X) = (5, X, R), possiamo ottenere la configurazione successiva *C' = XX5YYZZ*, dove *u* = XX, *q* = 5, *v* = YYZZ.
In generale, la configurazione cambia ad ogni mossa. Si dice che C_1 *produce* C_2 (si indica con $C_1 \rightarrow C_2$) se una mossa della MdT può far andare la macchina da C_1 a C_2.

<u>*Esempio2*</u>:

Se *a*, *b*, *c* ∈ Γ (cioè, sono simboli),	*u*, *v* ∈ Γ* (cioè, sono stringhe),	q_i, q_j ∈ Q (cioè, sono stati),
allora $uaq_ibv \rightarrow uq_jacv$	se δ(q_i, b) = (q_j, c, L) è una *mossa a sinistra*,	
oppure $uaq_ibv \rightarrow uacq_jv$	se δ(q_i, b) = (q_j, c, R) è una *mossa a destra*.	

Casi particolari di configurazioni:
Data una configurazione $q_ibv \rightarrow q_jcv$, se la testina è ad inizio nastro e la prossima è una *mossa a sinistra*, allora <u>la posizione della testina non cambia</u> (senza che la Macchina di Turing se ne accorga).
La configurazione uaq_i è equivalente a uaq_i_, cioè la parte vuota del nastro viene riempita con simboli blank (_).
Una configurazione di M si dice **di start** su un input *w* (q_0w) se e solo se lo stato q_0 è lo stato iniziale, il nastro contiene *w* e la testina è posizionata sulla prima cella del nastro.
Una configurazione di M si dice **di accettazione** (Accept) se essa raggiunge uno stato q_{accept}.
Una configurazione di M si dice **di rifiuto** (Reject) se essa raggiunge uno stato q_{reject}.
Una configurazione di M si dice **di Halt** se essa raggiunge o uno stato q_{reject} o uno stato q_{accept} (cioè, una qualsiasi configurazione Accept o Reject).
Una Macchina di Turing M **accetta** una parola *w* se esiste una computazione (*sequenza di configurazioni*) di M del tipo $C_1, C_2, \dots, C_k$ tale che:
1. $C_1 = q_0w$ è la configurazione iniziale di M con input *w*;
2. $C_i \rightarrow C_{i+1}$ per ogni i = 1, 2, …, k - 1;
3. C_k è la configurazione di accettazione (Accept) di M.

Una Macchina di Turing M **rifiuta** una parola *w* se esiste una computazione (*sequenza di configurazioni*) di M del tipo $C_1, C_2, \dots, C_k$ tale che:
1. $C_1 = q_0w$ è la configurazione iniziale di M con input *w*;
2. $C_i \rightarrow C_{i+1}$ per ogni i = 1, 2, …, k - 1;
3. C_k è la configurazione di rifiuto (Reject) di M.

Una Macchina di Turing M può raggiungere tre *risultati computazionali*:
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 essa è in loop, in quanto si potrebbe fermare in seguito oppure no.

Una ***MdT M*** accetta una ***stringa w*** se esiste una ***computazione*** (***sequenza di configurazione***) di M: $C_1, \dots C_k$ tale che
1. $C_1 = q_0w$ è la configurazione iniziale di M con input w;
2. Ogni C_i produce C_{i+1} per ogni i=1, …, k-1;
3. C_k è una configurazione di accept.

Data una Macchina di Turing M, il **linguaggio di M** è l'insieme delle stringhe che M accetta, e viene denotato con L(M).

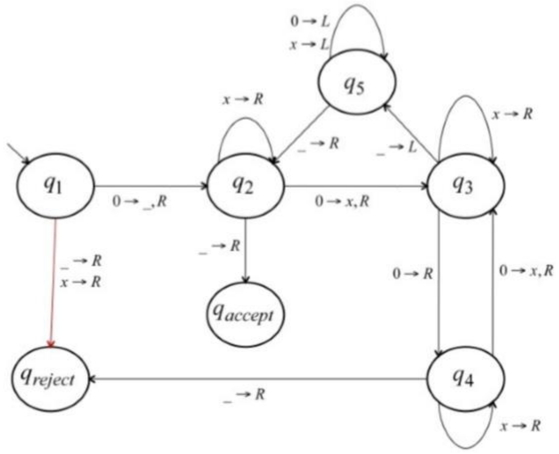
Un linguaggio si dice ***Turing-riconoscibile*** se esiste una macchina di Turing che lo riconosce.

Formalmente, un linguaggio L si dice Turing riconoscibile se esiste una Macchina di Turing M tale che L(M) = L. Di conseguenza, la macchina accetta tutte le stringhe del linguaggio. Se, invece, una stringa w ∉ L, allora la stringa può essere o *rifiutata* o mandare la macchina *in loop*.
Abbiamo visto che è difficile dire se una MdT è in loop. Possiamo evitare questo problema costruendo le macchine che si fermano (accettando o rifiutando) *su ogni input*: trattasi dei **deciders**.
Si dice che un decider *decide* il linguaggio L se esso riconosce L.
Un linguaggio si dice **Turing decidibile** se esiste una MdT che lo decide. Formalmente, un linguaggio L si dice Turing decidibile se esiste una Macchina di Turing M tale che L(M) = L e M è un decider.
La differenza tra un linguaggio L Turing riconoscibile e Turing decidibile risiede nel fatto che i primi possono mandare le MdT che li riconoscono in loop, mentre i secondi possono far sì che le MdT che li decidono o accettino o rifiutino le loro stringhe *su ogni input*.

Un linguaggio si dice ***Turing-decidibile*** o semplicemente ***decidibile*** se esiste una macchina di Turing che lo decide.

<u>*Esempio1*</u>:
Consideriamo il linguaggio L = {0^{2^n} | n > 0} dell'insieme di stringhe di 0 la cui lunghezza è potenza di 2. Vogliamo costruire una MdT M_2 che lo decide.
Nota. Il linguaggio non è regolare (*ciò è dimostrabile mediante Pumping Lemma*).
→ Si noti che *n* è potenza di 2 se e solo se ripetute divisioni per 2 danno resto 1.
Sia *w* l'input; allora, un algoritmo tale che M_2 decida tale linguaggio può essere il seguente:
1. Scorrere il nastro da sinistra a destra cancellando ogni SECONDO 0; *//divide per 2 il numero di 0*
2. Se rimane solo uno 0, allora ACCEPT;

3. Se rimane un numero di 0 dispari ≥ 3, allora REJECT; *//in quanto il numero di 0 non è potenza di 2*
4. Se rimane un numero pari di 0, allora riporta la testina all'inizio del nastro;
5. Ritorna al passo 1.



Allora, l'implementazione della MdT M_2 può essere la seguente.
Si noti che spesso si utilizza la forma contratta *x → R*, che equivale a *x → x, R* (cioè, quando si sovrascrive lo stesso carattere letto).
L'alfabeto della macchina è Σ = {0}. Inoltre, l'alfabeto del nastro è Γ = {0, x, _}.
Ad esempio, sia *w* = 0000. Quando ci si trova nello stato iniziale q_1, con un *passo tecnico* si sostituisce (*il primo*) 0 con _, per poi passare alla cella successiva (ed allo stato q_2). Passati allo stato q_2, si sostituisce (*il secondo*) 0 con x e si passa alla cella successiva (ed allo stato q_3). Nello stato q_3, si lascia (*il terzo*) 0 invariato e si passa alla cella successiva (ed allo stato q_4). Nello stato q_4, si sostituisce (*il quarto*) 0 con x e si passa alla cella successiva (ed allo stato q_3). Ora, la testina punta su _, quindi abbiamo scandito tutta la stringa in input; dobbiamo iterare di nuovo: si sposta la testina all'indietro (passando allo stato q_5, dove per ogni simbolo letto si sposta la testina a sinistra, fino ad arrivare all'inizio) fin quando si riconosce il simbolo _ (questo è il motivo per cui abbiamo utilizzato un carattere diverso, per simboleggiare l'inizio della stringa), tornando allo stato q_2: qui siamo pronti ad una nuova iterazione.
La testina si sposta fin quando non trova uno 0 (in questo caso, *il terzo*), che viene sostituito con x e si muove la testina a destra (e si passa allo stato q_3). Nello stato q_3, con le x si avanza: si arriva alla fine del nastro, quindi si legge il simbolo _ e si torna in q_5, dove si torna all'inizio della stringa e si passa quindi a q_2 (muovendo la testina a destra, quindi al secondo simbolo della stringa, cioè la prima x); tornati in q_2, infine, si scandisce ogni simbolo x sul nastro e si arriva alla fine della stringa: si legge, quindi, _ e si passa allo stato q_{accept}, e quindi la stringa 0000 viene *accettata*.
Esempio2:
Consideriamo il linguaggio L = {w#w | w ∈ {0, 1}*}. Di seguito un'idea per verificare se una stringa s ∈ L:
1. Leggi il primo carattere;
2. Memorizzalo e cancellalo;
3. Cerca # e guarda il carattere successivo;
4. Se esso è uguale al carattere memorizzato, allora cancellalo;
5. Ritorna all'inizio al primo carattere non cancellato;
6. Ripeti 1-5 fino a considerare tutta la stringa in input: se si trova qualcosa di "inatteso", allora REJECT; altrimenti, ACCEPT.
Vediamo, ora, come costruire una MdT M_1 per il linguggio L. Sia definita in input una stringa w; allora, M_1 deve:
1. Memorizzare il simbolo più a sinistra e cancellarlo (scriviamo x);
2. Avanzare sul nastro fino a superare #, se non si trova allora REJECT;
3. Confronta il primo simbolo diverso da x con il simbolo memorizzato, se è diverso allora REJECT;
4. Se è uguale, cancella il simbolo confrontato e ritorna a inizio nastro;
5. Vai al punto 1.
Vediamo, ora, la funzione di transizione per M_1. Si ha che Σ = {0, 1, #}, e Γ = {0, 1, #, x, _}. Nella descrizione, lo stato q_{reject} e tutte le transizioni in ingresso sono state omesse. Ovunque vi sia una transizione mancante, va in q_{reject}.
Nota. Abbiamo utilizzato il termine "memorizzare": le MdT non possono memorizzare simboli in quanto non hanno memoria, ma può essere ottenuta una memorizzazione progettando il diagramma degli stati in un modo preciso: in particolare, si useranno stati diversi a seconda che l'input sia un 1 o uno 0. A tal scopo, gli stati q_2 e q_3 memorizzano il bit 0, mentre gli stati q_4 e q_5 memorizzano il bit 1.
In altre parole, questi due segmenti sono identici, e in ogni segmento si utilizza il valore memorizzato.
Esempio del funzionamento alle slide 116 – 148.

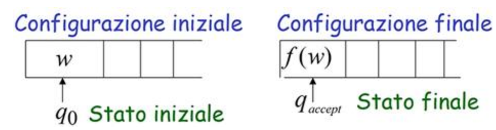
Le Macchine di Turing possono avere le proprie funzioni di transizione rappresentate utilizzando una **tabella**. Data una funzione di transizione generica δ: Q × Γ → Q × Γ × {*L, R*}, è possibile costruire una tabella di 5 colonne ed n righe (le n righe rappresentano le etichette complessive della funzione), dove:
- la prima colonna rappresenta lo stato attuale (q_i ∈ Q);
- la seconda colonna rappresenta il simbolo letto (s ∈ Γ);
- la terza colonna rappresenta il nuovo stato (q_j ∈ Q);
- la quarta colonna rappresenta il simbolo da scrivere (t ∈ Γ);
- la quinta colonna rappresenta il movimento della testina ({L, R}).

Ad esempio, data la tabella a lato otteniamo che:
- la riga 4 rappresenta la funzione δ(q_1, 0) = (q_1, 0, L);
- la riga 6 rappresenta la funzione δ(q_1, _) = (q_2, _, R);
- …

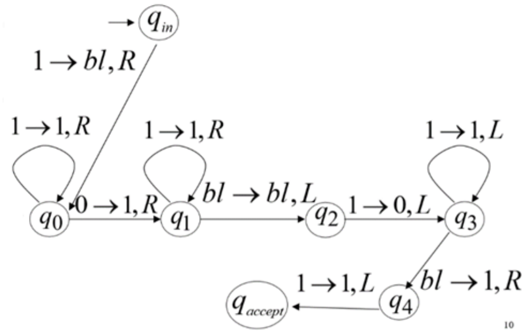
stato	simbolo	stato	simbolo	movimento
q0	0	q0	1	R
q0	1	q0	0	R
q0	_	q1	_	L
q1	0	q1	0	L
q1	1	q1	1	L
q1	_	q2	_	R

3.2 CALCOLO FUNZIONI CON MACCHINA DI TURING

Sia definita una funzione f: D → S che opera su stringhe; cioè, data una stringa w ∈ D si ottiene un'altra stringa f(w) ∈ S.
Una funzione può dipendere da più variabili. Ad esempio, la funzione addizione f(x, y) = x + y è una funzione a due variabili.
Consideriamo come l'insieme degli interi. Se usiamo la notazione decimale e vogliamo dare in output 5, allora il simbolo della MdT sarà 5 (con alfabeto Σ = {0, 1, …, 9}. Se usiamo la notazione binaria e vogliamo dare in output 5, allora il simbolo della MdT sarà 101 (con alfabeto Σ = {0, 1}). Noi useremo la *notazione unaria*, avente come alfabeto Σ = {1}, che dà in output tanti 1 quant'è il valore del numero in input (es. 5 → 11111): questa notazione è più semplice, anche se non è molto efficace.
Idea:
Una funzione *f* si dice *calcolabile* se esiste una macchina di Turing *M* tale che, per ogni w ∈ D, essa parte con un input w in uno stato iniziale q_0 e termina in uno stato finale q_{accept} avendo sul nastro il valore f(w).



Esempio1:



La funzione $f(x, y) = x + y$ è calcolabile. Infatti, dati x, y interi, allora possiamo costruire una Macchina di Turing che calcola questa funzione: ad una stringa in input *x0y* (unario) è associata una stringa in output *xy0* (unario). Lo 0 viene usato all'inizio semplicemente per separare le due stringhe, mentre viene usato al termine per eseguire eventualmente altre operazioni.

Ad esempio, dato l'input 2 + 2 (11011) ricaviamo l'output 4 (1111).

A destra è posta la MdT che esegue l'operazione appena vista.

È importante che, al termine del calcolo dell'output, la testina che si trova alla fine torni all'inizio del nastro per andare nella situazione finale (q_{accept}).

Esempio del funzionamento alle slide 11 – 24.

Esempio2:

La funzione $f(x) = 2x$ è calcolabile. Infatti, dato un numero x intero, allora possiamo costruire una Macchina di Turing che calcola questa funzione: ad una stringa in input *x* (unario) è associata una stringa in output *xx* (unario).

Si scriva, per esercizio, la Macchina di Turing corrispondente.

Esempio3:

La funzione $f(x, y) = \begin{cases} 1 \; se \; x > y \\ 0 \; se \; x \leq y \end{cases}$ è calcolabile. La MdT, ad esempio, parte con un input 111011_ e dà in output 1_ (in quanto 111 = 3 > 2 = 11).

Dunque, bisogna vedere se il numero di 1 prima dello 0 è maggiore rispetto al numero di 1 dopo lo 0.

Si scriva, per esercizio, la Macchina di Turing corrispondente.