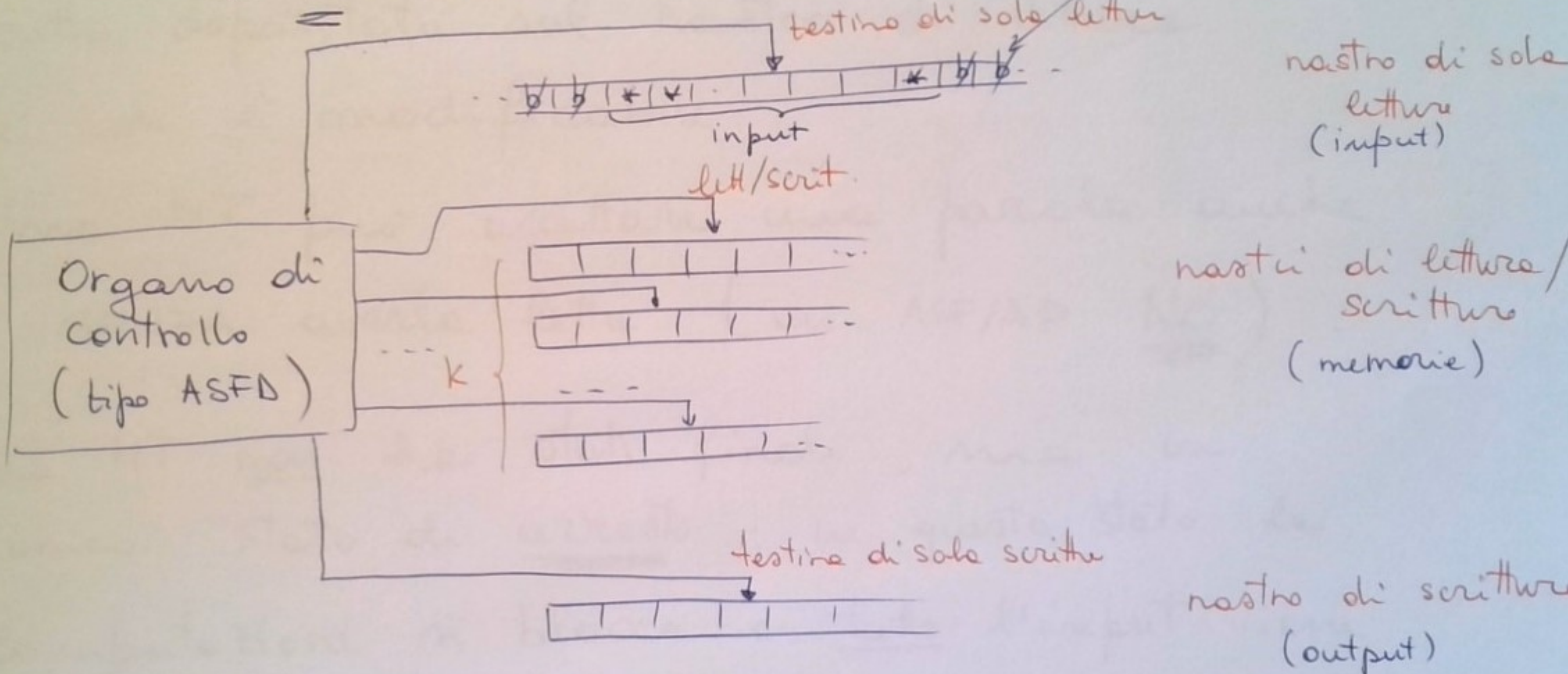
— Macchine di Turing —

"a k nastri" (con $k \in \mathbb{N}$)

uso "$\not b$" per "blank"
cioè: spazio libero



Differenza fondamentale tra ASF/AP e MT:

- all'inizio della computazione, l'input è tutto depositato sul nastro di lettura e non è modificabile.

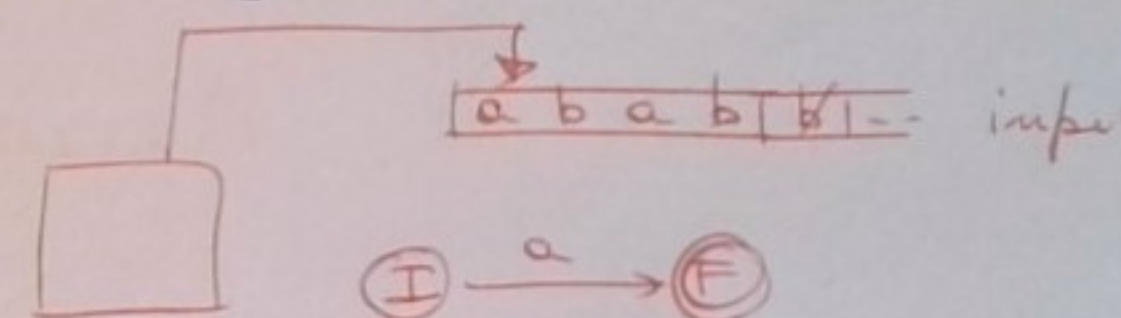
  Una MT può accettare una parola anche SENZA averla letta (un ASF/AP <u>NO</u>)

- una MT <u>non</u> ha stati finali, ma un unico stato di <u>arresto</u>: in questo stato la computazione si blocca e <u>tutto</u> l'input viene accettato.

Es Sia $L = a(a+b)^*$ (su $\Sigma = \{a,b\}$), progettare/disegnare una MT con k nastri (NB: decidete voi chi è k) che riconosca L.

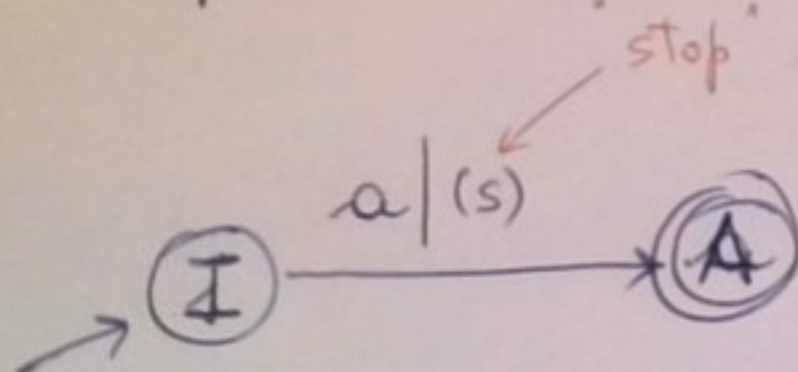
Serve proprio una MT? È meglio una MT o un ASF?

no, L è regolare, basta un ASF



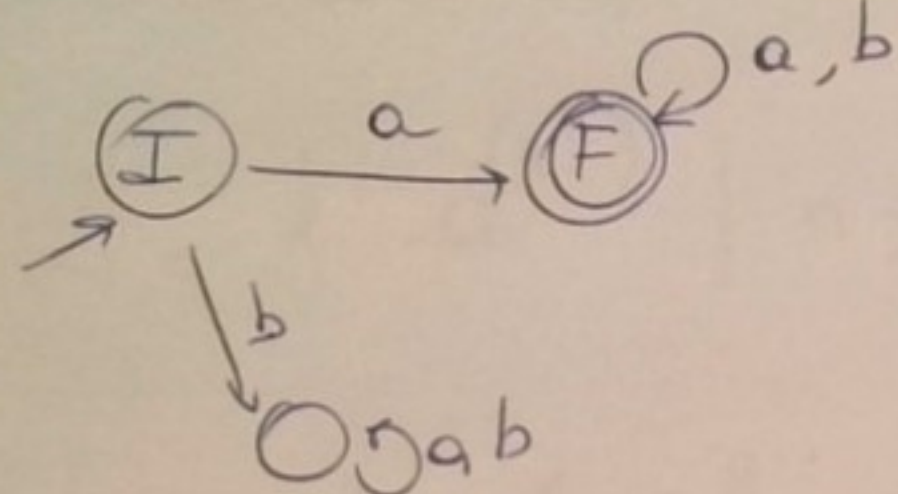
Possiamo prendere $k = 0$

MT:



leggo il primo carattere del nastro di input, se è 'a', non sposto la testina di lettura, vado in Ⓐ e accetto qualunque cosa c'è in input.

Con un ASF faccio così



leggo il primo carattere,
se 'a' vado in F e continuo
a leggere finché ho finito,
poi accetto.

<u>Es</u> Costruire un MT a k nastri (parte dell'es. è trovare k) che riconosca

$$L = \{ a^{(2^n)} \mid n \geq 0 \} \qquad (\text{alfabeto } \Sigma = \{a\}).$$

$$L = \{ a, a^2, a^4, a^8, a^{16}, a^{32}, \dots \}$$

Praticamente la macchina deve saper riconoscere le potenze di 2 scritte in base 1.

<u>Oss</u> Sia $A \subseteq a^*$: esiste di sicuro un MT che riconosca A? Non è detto.

Le MT sono tante quante i <u>naturali</u>; i sottoinsiemi di a^* sono tanti quanti i <u>reali</u>.

Per capire se esiste un MT che riconosce L, vediamo prima se esiste un algoritmo che distingua gli elementi di L da quelli che non stanno in L.

aaaa ⤳ 'sì'

aaaaa ⤳ 'no'

Ci sono due modi

① divido la lunghezza dell'input per 2 e ripeto: se resta 1 (senza resti) ok

② converto l'input da base 1 a base 2

a a a a $\longrightarrow$ 1 0 0 $\longrightarrow$ sì

4 in base 1   4 in base 2

a a a a a $\longrightarrow$ 1 0 1 $\longrightarrow$ no

5 base 1   5 base 2

qui serve una MT

Accetto a a … a sse nel convertirlo in base 2 trovo una stringa del tipo 10 … 0, e questo è facile: si fa con un ASF.

Voglio progettare una MT che trasformi una stringa da base 1 a base 2.

Basta avere una MT che "sappia" eseguire l'incremento di un'unità in base 2.

Esempio in: aaaaa

$0 \xrightarrow[a]{+1} 1 \xrightarrow[a]{+1} 10 \xrightarrow[a]{+1} 11 \xrightarrow[a]{+1} 100 \xrightarrow[a]{+1} 101.$

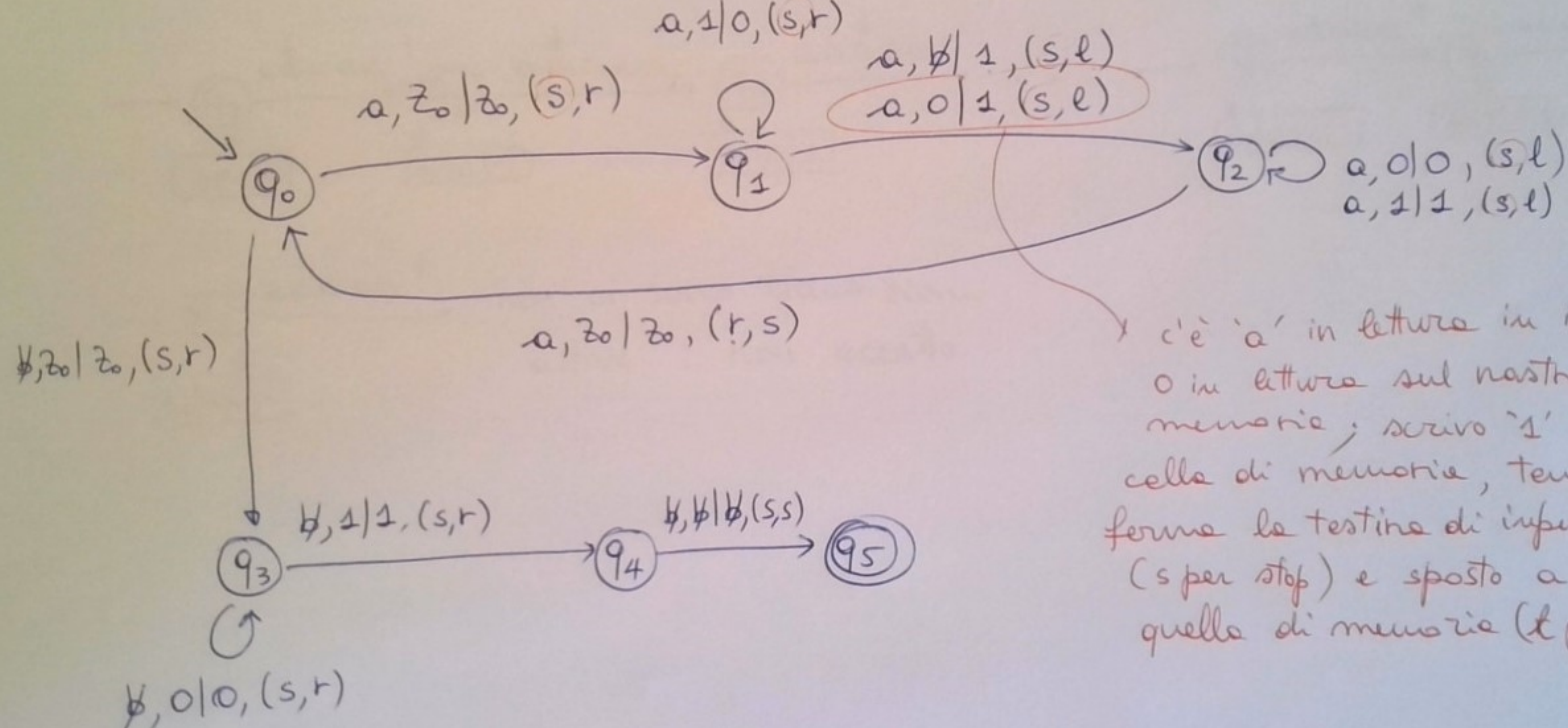
Per incrementare di 1 devo:

- posizionarmi sulla cifra <u>meno</u> significativa
- se è 0, diventa 1, fine.
- se è 1, diventa 0 e passo alla successiva
- se è 'b̸', diventa 1, fine

basta un solo nastro di memoria

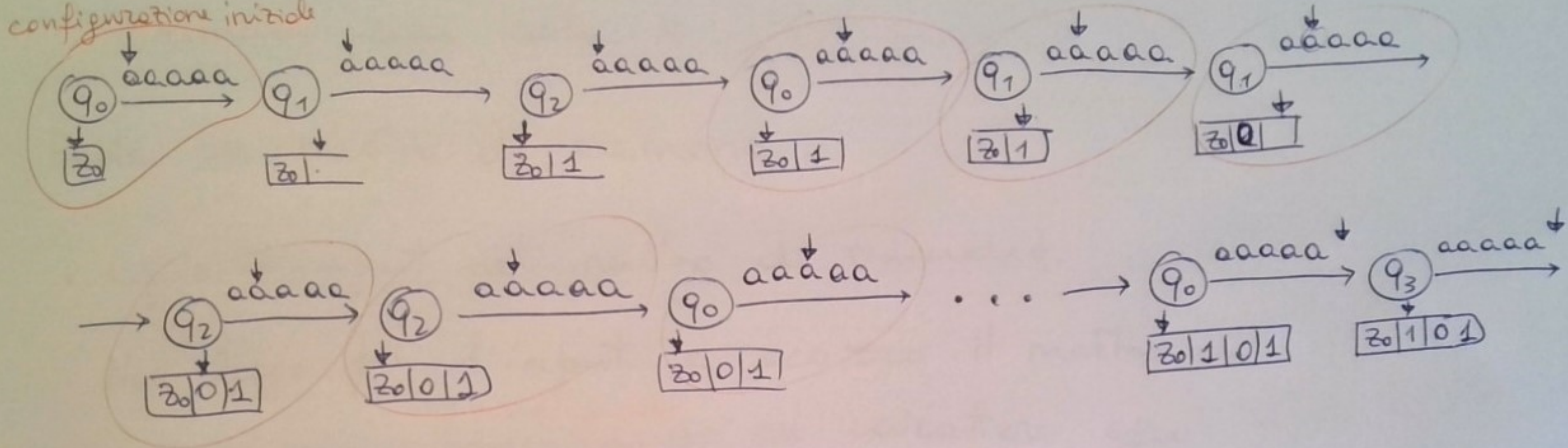
Problema: le stringhe binarie si "allungano" a sx, mentre i nastri di memoria crescono a dx;

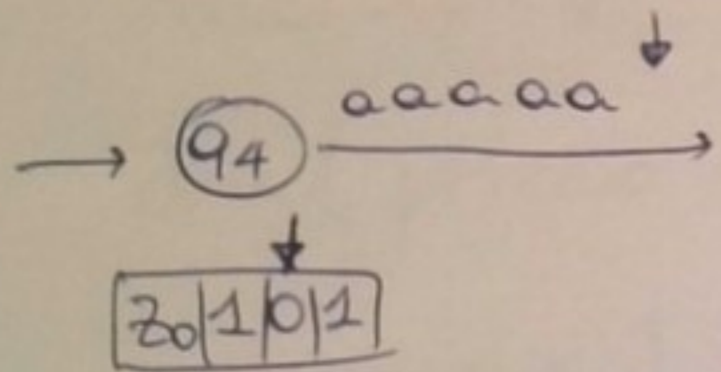
soluzione: scrivo tutto a rovescio.



c'è 'a' in lettura in input, 0 in lettura sul nastro di memoria; scrivo '1' nella cella di memoria, tengo ferma la testina di input (s per stop) e sposto a sx quella di memoria (l per ~~right~~ left)

Funzionamento : input aaaaa





non ci sono transizioni attive : non accetto

Si può fare in altro modo: "divido" per 2 finché rimane una sola a.

Basta <u>un</u> nastro di memoria.

- Copio l'input nel nastro di memoria
- Non leggo più l'input e scorro il nastro di
  modificando
  memoria "cancellando" un carattere ogni due (primo lo tengo, il secondo cancello):

se termino il nastro senza cancellare, non va bene.

Se l'ultima operazione è una cancellatura, ripeto. Eccezione: sul nastro resta una sola 'a'! allora accetto.

Inizio in : a a a a a a

mem Z0

⇝ a a a a a a

Z0 a a a a a a

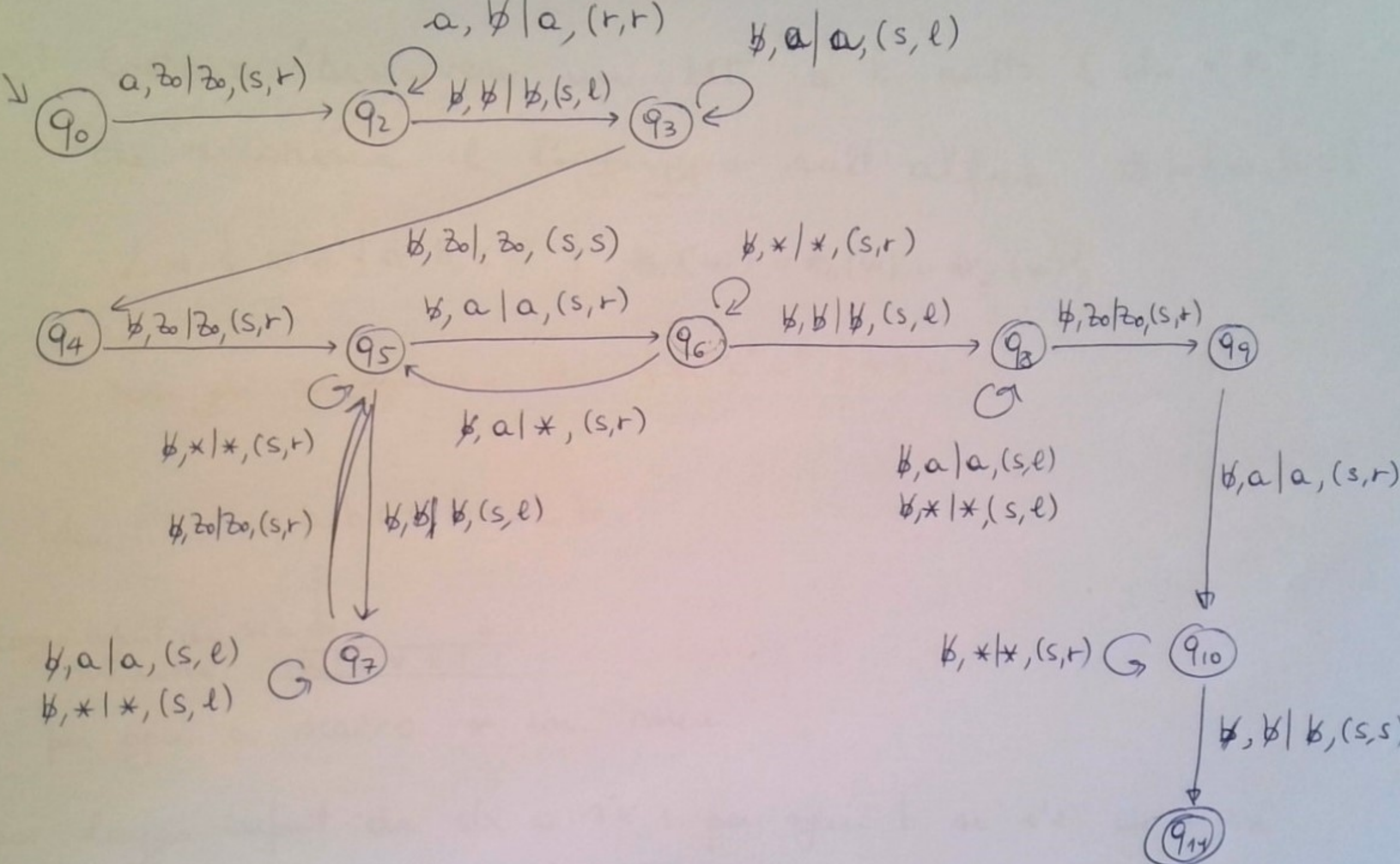
⇝ in : idem

mem Z0 a * a * a *

⇝ in : idem

mem : Z0 a * * * a * ○ rifiuta

non c'è una 'a' da cancellare


a, Z0 | Z0, (s, r)
a, b̸ | a, (r, r)
b̸, b̸ | b̸, (s, l)
b̸, a | a, (s, l)
b̸, Z0 |, Z0, (s, s)
b̸, * | *, (s, r)
b̸, Z0 | Z0, (s, r)
b̸, a | a, (s, r)
b̸, b̸ | b̸, (s, l)
b̸, Z0 | Z0, (s, r)
b̸, a | *, (s, r)
b̸, * | *, (s, r)
b̸, Z0 | Z0, (s, r)
b̸, b̸ | b̸, (s, l)
b̸, a | a, (s, l)
b̸, * | *, (s, l)
b̸, a | a, (s, r)
b̸, a | a, (s, l)
b̸, * | *, (s, l)
b̸, * | *, (s, r)
b̸, b̸ | b̸, (s, s)

Es) Costruire / Descrivere una MT a k nastri (chi è k?)

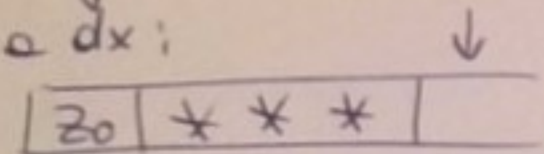
che riconosca il linguaggio sull'alfab. $\Sigma = \{a, b, c\}$

$$L = \{ w \in \{a, b, c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w) \}$$

sono gli anagrammi di $\{a^n b^n c^n \mid n \geq 0\}$

Idea: aabccbacb

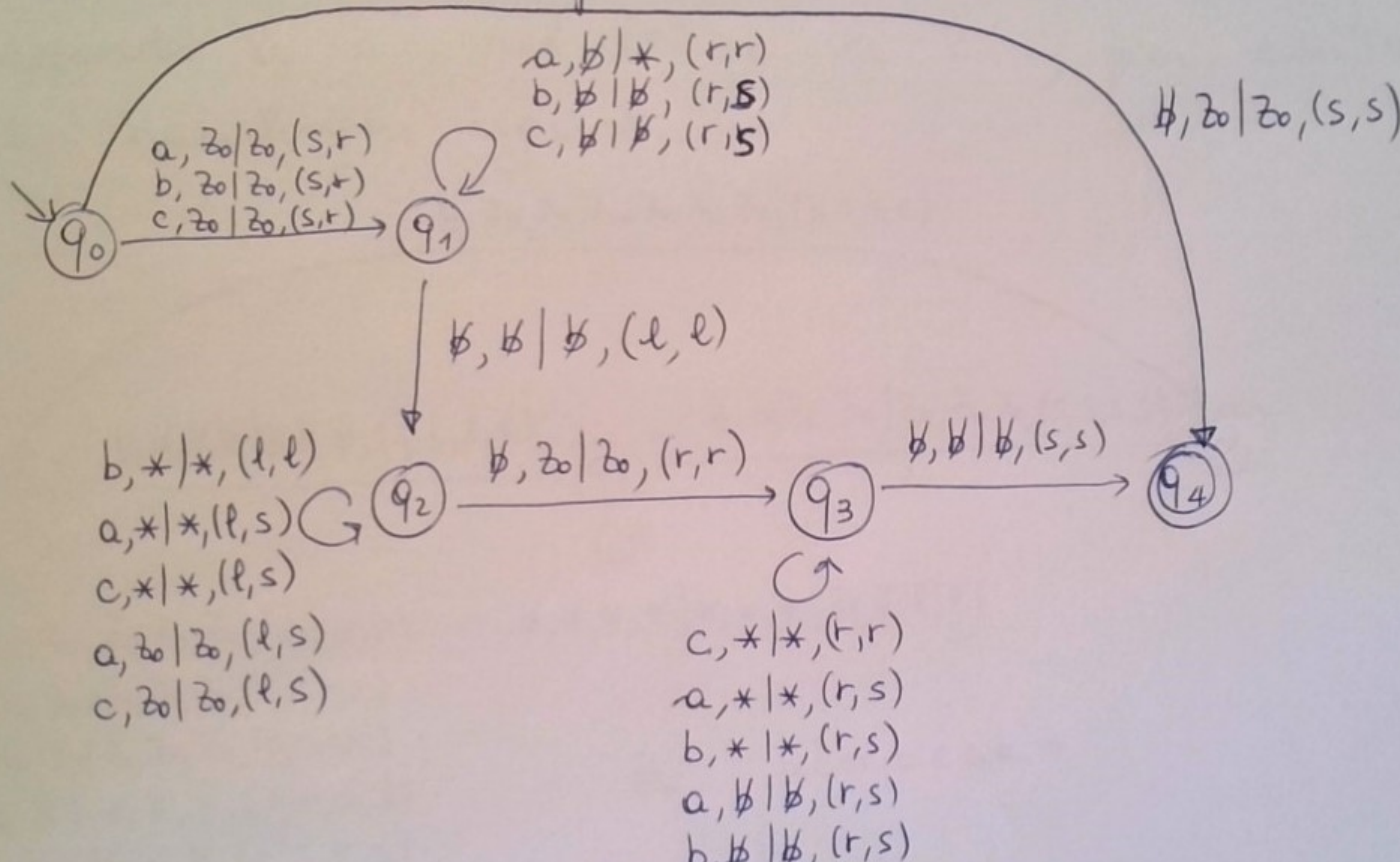
leggo input da sx a dx:
memoria



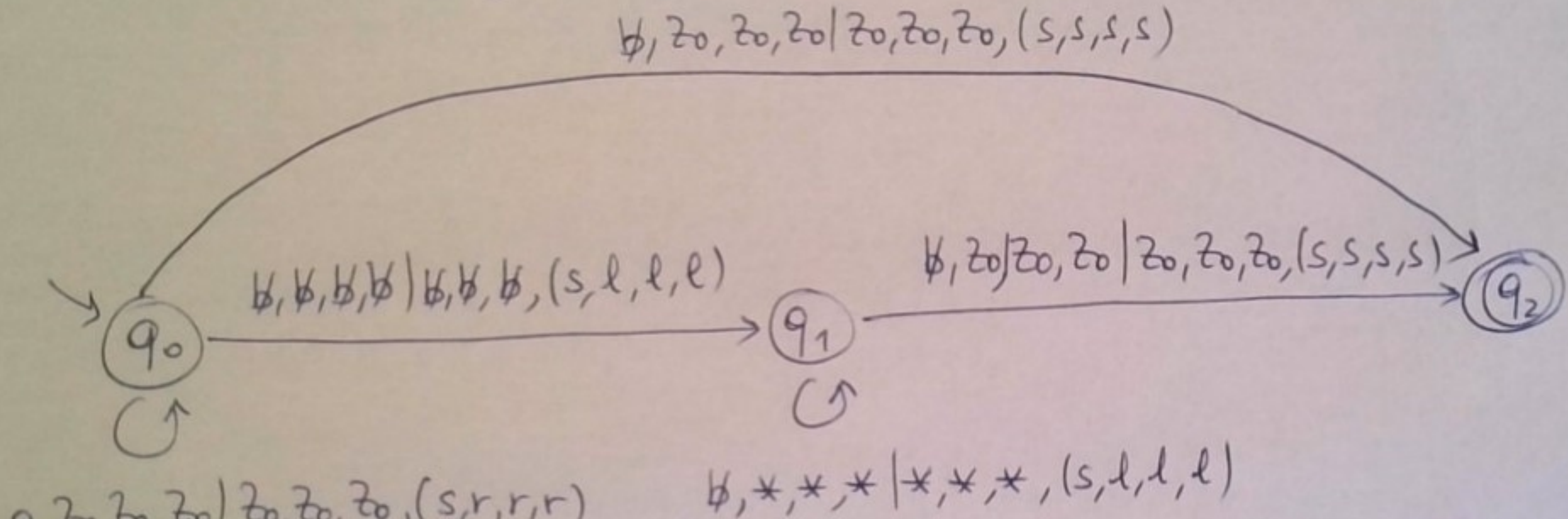
per ogni a scrivo * in mem.

poi leggo input da dx a sx: per ogni b se c'è un * in memoria sposto a sx input e memoria, se arrivo con $\not{b}$ a sx in ~~mem.~~ input e Z_0 in memoria continuo, se no rifiuto

Idem con le c spostandomi a dx: accetto se trovo ɓ in input e memoria.



Oppure si possono usare 3 nastri: nel primo "carico" il numero di 'a' (in unario), nel secondo le 'b', nel terzo le 'c', poi controllo che siano lunghi uguali.



Es abccbcaab