Lezione 3 – esercizi su macchine di Turing

Lezione del 07/03/2025

Esercizi sulle macchine di Turing

- In questa esercitazione taluni esercizi verranno utilizzati per introdurre il concetto di simulazione (a scatola chiusa)
- Gli esercizi che vedremo sono:
  - 1) Palindromia
  - 2) Somma di k (non costante) interi in riga
  - 3) Decidere se l'input è una parola nella forma xx (esempio abbaabba)
- La lezione terminerà con una breve discussione sul concetto di simulazione

Esercizi sulle macchine di Turing

- **ESERCIZIO 1**: vogliamo progettare una macchina di Turing T_{PAL} che, con una parola x ∈ {a, b}* scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è palindroma o meno
  - una parola è palindroma se rimane invariata quando letta da destra a sinistra o d sinistra a destra
  - esempio: abba, aababaa, ...
  - il primo carattere è uguale all'ultimo carattere, il secondo carattere è uguale al penultimo carattere, il terzo carattere è uguale al terzultimo carattere, ...
- **1.1) Cominciamo a considerare un riconoscitore T_{PPAL} che termina in uno stato che attesta se x è una parola palindroma *di lunghezza pari* o meno**
  - quindi con input x=aababaa T_{PPAL} attesta che x non è una parola palindroma di lunghezza pari
  - Indichiamo gli stati di T_{PPAL} con una P in apice
    - ad esempio: q_0^P , q_P^P , q_{NPP}^P ...
    - dove q_0^P è lo stato iniziale e q_P^P e q_{NPP}^P sono gli stati finali che attestano, rispettivamente, che x è una palindromia di lunghezza pari o no

1.1) Palindromia e lunghezza pari

- Descriviamo ora *ad alto livello* come lavora T_{PPAL} :
  - T_{PPAL} esegue una serie di scansioni del nastro da-sinistra-a-destra e da-destra-a-sinistra
  - quando inizia una scansione da-sinistra-a-destra memorizza il simbolo letto nella prima cella, lo cancella e si porta nell'ultima cella a destra occupata da un carattere di x
  - quando inizia una scansione da-destra-a-sinistra confronta il simbolo letto nella cella più a destra con il simbolo memorizzato e:
    - se sono diversi, la parola x non è palindroma (se legge a o b) oppure non ha lunghezza pari (se legge blank): pertanto, entra nello stato q^P_{NPP} e termina;
    - se sono uguali cancella il carattere letto e si porta sulla cella più a sinistra occupata da un carattere di x non ancora cancellato: se a sinistra del carattere appena cancellato si trova una cella vuota (blank) allora significa che tutti i caratteri di x sono stati cancellati (ossia, si sonno accoppiati specularmente) e quindi T_{PPAL} entra nello stato q^P_P e termina
  - Sì, ma che vuol dire che T_{PPAL} memorizza un carattere? Dove lo memorizza?
    - Beh, questo ormai lo sappiamo: lo memorizza nello stato interno!
    - Quindi, l'insieme Q_{PPAL} degli stati di T_{PPAL} contiene gli stati q^P_a e q^P_b oltre agli stati iniziale e finali q^P_0, q^P_{PP} e q^P_{NPP}

1.1) Palindromia e lunghezza pari

- Descriviamo ora *a basso livello* come lavora T_{PPAL} : le sue quintuple sono
  - per la scansione da-sinistra-a-destra, a partire dallo stato iniziale q_0^P
    - legge e cancella il carattere più a sinistra sul primo nastro e lo memorizza nello stato interno: $\langle q_0^P$,a, ␣, q_a^P , D$\rangle$, $\langle q_0^P$, b, ␣, q_b^P , D$\rangle$,
    - ricordando il carattere letto, sposta la testina sul ␣ a destra del carattere più a destra di x: $\langle q_a^P$, a, a, q_a^P , D$\rangle$, $\langle q_a^P$, b, b, q_a^P , D$\rangle$,
      $\langle q_b^P$, a, a, q_b , D$\rangle$, $\langle q_b^P$, b, b, q_b^P , D$\rangle$,
    - cambiando stato (e sempre ricordando il carattere letto all'inizio dello spostamento da-sinistra-a destra, ossia, entrando nello stato q_{a1}^P o q_{b1}^P), sposta la testina a sinistra per posizionarla sul carattere più a destra di x:
      $\langle q_a^P$, ␣, ␣, q_{a1}^P , S$\rangle$, $\langle q_b^P$, ␣, ␣, q_{b1}^P , S$\rangle$,

1.1) Palindromia e lunghezza pari

- Descriviamo ora *a basso livello* come lavora T_{PPAL} : le sue quintuple sono
  - per la scansione da-destra-a-sinistra, a partite dallo stato q^P_{a1} o q^P_{b1}
    - se il carattere memorizzato nello stato interno è uguale a quello letto dalla testina, lo cancella e, entrando nello stato q^P_2, sposta la testina sul ␣ a sinistra del carattere più a sinistra di x: $\langle q^P_{a1}$, a, ␣, q^P_2 , S$\rangle$, $\langle q^P_{b1}$, b, ␣, q^P_2 , S$\rangle$,
      $\langle q^P_2$, a, a, q^P_2 , S$\rangle$, $\langle q^P_2$, b, b, q^P_2 , S$\rangle$,
      - dopodiché, raggiunto il ␣, entra nuovamente nello stato q^P_0 per iniziare una nuova fase da-sinistra-a-destra: $\langle q^P_2$, ␣, ␣, q^P_0 , D$\rangle$,
      - (*) e, se nello stato q^P_0, la testina legge ␣ allora tutti i simboli sono stati cancellati (e, poiché vengono cancellati a coppie, questo significa che x aveva lunghezza pari) e dunque T_{PPAL} può concludere che x era palindroma:
        $\langle q^P_0$, ␣, ␣, q^P_{PP} , F$\rangle$
    - (**) se il carattere memorizzato nello stato interno è diverso da quello che viene letto dalla testina, può concludere che x non era palindroma oppure aveva lunghezza dispari: $\langle q^P_{a1}$, b, ␣, q^P_{NPP}, F$\rangle$, $\langle q^P_{b1}$, a, ␣, q^P_{NPP}, F$\rangle$,
      $\langle q^P_{a1}$, ␣, ␣, q^P_{NPP}, F$\rangle$, $\langle q^P_{b1}$, ␣, ␣, q^P_{NPP}, F $\rangle$

Esercizi sulle macchine di Turing

- **1) *palindromia***: vogliamo progettare una macchina di Turing T_{PAL} che, con una parola x ∈ {a, b}* scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è palindroma o meno
- **1.2) Consideriamo ora una macchina T_{DPAL} che termina in uno stato che attesta se x è una parola palindroma *di lunghezza dispari* o meno**
  - quindi con input x=aababaa T_{DPAL} attesta che x è palindroma di lunghezza dispari
- Descriviamo ora *ad alto livello* come lavora T_{DPAL} :
  - Indichiamo gli stati di T_{DPAL} con una D in apice
    - ad esempio: q_0^D , q_{PD}^D , q_{NPD}^D…
  - Anche T_{DPAL} esegue una serie di scansioni di x da-sinistra-a-destra e da-destra-a-sinistra
  - e le fasi di scansione da-sinistra-a-destra sono identiche a quelle di T_{PPAL}
  - le fasi di scansione da-destra-a-sinistra di T_{DPAL} sono molto simili a quelle di T_{PPAL}
    - solo i passi che in T_{PPAL} sono stati contrassegnati con (*) e (**) sono diversi in T_{DPAL}

1.2) Palindromia e lunghezza dispari

- Descriviamo ora *ad alto livello* come lavora T_{DPAL} : le sue quintuple sono
  - [scansione da-sinistra-a-destra identica a T_{PPAL} con, ora, stati interni q_0^D, q_a^D, q_b^D, q_{a1}^D e q_{b1}^D]
  - per la scansione da-destra-a-sinistra, a partire dallo stato q_{a1}^D o q_{b1}^D
    - se il carattere memorizzato nello stato interno è uguale a quello letto dalla testina, lo cancella e, entrando nello stato q_2^D, sposta la testina sul ␣ a sinistra del carattere più a sinistra di x: $\langle q_{a1}^D$, a, ␣, q_2^D , S$\rangle$, $\langle q_{b1}^D$, b, ␣, q_2^D , S$\rangle$,
      $\langle q_2^D$, a, a, q_2^D , S$\rangle$, $\langle q_2^D$, b, b, q_2^D , S$\rangle$,
      - dopodiché, raggiunto il ␣, entra nuovamente nello stato q_0^D per iniziare una nuova fase da-sinistra-a-destra: $\langle q_2^D$, ␣, ␣, q_0^D , D$\rangle$,
      - (*) e, se nello stato q_0^D, la testina legge ␣ allora tutti i simboli sono stati cancellati e, poiché vengono cancellati a coppie, questo significa che x aveva lunghezza pari e dunque può concludere che x era palindroma ma di lunghezza pari: $\langle q_0^D$, ␣, n, q_{NPD}^D , F$\rangle$

[... continua ...]

1.2) Palindromia e lunghezza dispari

- Descriviamo ora *ad alto livello* come lavora T_{DPAL} : le sue quintuple sono
  - per la scansione da-destra-a-sinistra
    - [... segue ...]
    - (**) se il carattere letto dalla testina è diverso da ␣ e da quello memorizzato nello stato interno, x non era palindroma:
      $\langle q^{D}_{a1}$, b, ␣, q^{D}_{NPD} , S$\rangle$, $\langle q^{D}_{b1}$, a, ␣, q^{D}_{NPD} , S$\rangle$,
    - (**) se il carattere letto dalla testina è uguale a ␣ può concludere che x ha lunghezza dispari e, poiché non ha mai concluso che *non* è palindroma allora *deve essere* palindroma (con il carattere a centrale se si trova nello stato q^{D}_{a1}, con il carattere b centrale se si trova nello stato q^{D}_{b1}):
      $\langle q^{D}_{a1}$, ␣, p, q^{D}_{PD} , F$\rangle$, $\langle q^{D}_{b1}$, ␣, p, q^{D}_{PD} , F $\rangle$

Esercizi sulle macchine di Turing

- **1) *palindromia***: vogliamo progettare una macchina di Turing T_{PAL} che, con una parola x ∈ {a, b}* scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è palindroma o meno
- **1.3) Consideriamo ora una macchina T_{PAL} che termina in uno stato che attesta se x è una parola palindroma, *di lunghezza pari o dispari che sia***
  - quindi con input x=aababaa T_{PAL} attesta che x è palindroma, con input y=abba T_{PAL} attesta che y è palindroma
- Descriviamo ora *ad alto livello* come lavora T_{PAL} :
  - Indichiamo gli stati di T_{PAL} senza apice
    - ad esempio: q_0 , q_P , q_{NP} ...
  - Anche T_{PAL} esegue una serie di scansioni di x da-sinistra-a-destra e da-destra-a-sinistra
  - e le fasi di scansioni da-sinistra-a-destra sono identiche a quelle di T_{PPAL} e di T_{DPAL}
  - le fasi di scansione da-destra-a-sinistra di T_{PAL} sono molto simili a quelle delle altre due macchine
    - come nel caso precedente, solo i passi che in T_{PPAL} e in T_{DPAL} sono stati contrassegnati con (*) e (**) sono diversi in T_{PAL}

1.3) Palindromia

- Descriviamo ora *ad alto livello* come lavora T_{PAL} : le sue quintuple sono
  - [scansione da-sinistra-a-destra identica a T_{PPAL} e T_{DPAL} con, ora, stati interni q_0, q_a, q_b, q_{a1} e q_{b1}]
  - per la scansione da-destra-a-sinistra, a partire dallo stato q_{a1} o q_{b1}
    - se il carattere memorizzato nello stato interno è uguale a quello letto dalla testina, lo cancella e, entrando nello stato q_2, sposta la testina sul ␣ a sinistra del carattere più a sinistra di x: $\langle q_{a1}$, a, ␣, q_2 , S$\rangle$, $\langle q_{b1}$, b, ␣, q_2 , S$\rangle$,
      $\langle q_2$, a, a, q_2 , S$\rangle$, $\langle q_2$, b, b, q_2 , S$\rangle$,
      - dopodiché, raggiunto il ␣, entra nuovamente nello stato q_0 per iniziare una nuova fase da-sinistra-a-destra: $\langle q_2$, ␣, ␣, q_0 , D$\rangle$,
      - (*) e, se nello stato q_0, la testina legge ␣ allora tutti i simboli sono stati cancellati e, poiché vengono cancellati a coppie, questo significa che x aveva lunghezza pari e dunque può concludere che x era palindroma (e di lunghezza pari): $\langle q_0$, ␣, ␣, q_P , F$\rangle$

    [... continua ...]

1.3) Palindromia

- Descriviamo ora *ad alto livello* come lavora T_{PAL} : le sue quintuple sono
  - per la scansione da-destra-a-sinistra
    - [... segue ...]
    - (**) se il carattere letto dalla testina è diverso da ␣ e da quello memorizzato nello stato interno, x non era palindroma:
      $\langle q_{a1}$, b, ␣, q_{NP} , F$\rangle$, $\langle q_{b1}$, a, ␣, q_{NP}, F$\rangle$,
    - (**) se il carattere letto dalla testina è uguale a ␣ può concludere che x ha lunghezza dispari e, poiché non ha mai concluso che *non* è palindroma allora *deve essere* palindroma (con il carattere a centrale se si trova nello stato q_{a1}, con il carattere b centrale se si trova nello stato q_{b1}):
      $\langle q_{a1}$, ␣, ␣, q_P , F$\rangle$, $\langle q_{b1}$, ␣, ␣, q_P , F $\rangle$

1.4) Palindromia – con simulazione

- Ma non finisce qui...
- In effetti, una volta che abbiamo (faticosamente!) progettato le macchine T_{PPAL} e T_{DPAL}, possiamo costruire la macchina senza ridefinire tutte le sue quintuple, ma utilizzando le due macchine T_{PPAL} e T_{DPAL}
- Come? Invocandole come se fossero due funzioni di un linguaggio di programmazione
- Ossia, come si dice nella terminologia delle macchine di Turing, ***simulando*** T_{PPAL} e T_{DPAL}
- Descriviamo ora *con questa nuova tecnica* la macchina T_{PAL}

1.4) Palindromia – con simulazione

- Descriviamo ora *con questa nuova tecnica* la macchina T_{PAL} :
  - T_{PAL} è una macchina a 3 nastri: sul primo nastro è scritto l'input, il secondo nastro serve alla simulazione di T_{PPAL}, il terzo nastro serve alla simulazione di T_{DPAL}
  - T_{PAL} opera in tre fasi:
    - FASE 1) copia l'input x sul secondo e terzo nastro;
    - FASE 2) simula T_{PPAL} sul secondo nastro: se $T_{PPAL}(x)$ termina in q_{PP}^{P} allora x è una parola palindroma di lunghezza pari e T_{PAL} entra nello stato finale q_P, altrimenti (l'input ha lunghezza dispari o non è palindroma) passa alla fase 3)
    - FASE 3) simula T_{DPAL} sul secondo nastro: se $T_{DPAL}(x)$ termina in q_{PD}^{D} allora x è una parola palindroma di lunghezza dispari e T_{PAL} entra nello stato finale q_P, altrimenti (l'input ha lunghezza dispari ma non è palindroma), grazie all'esito della fase 2), può concludere che x non è palindroma e, dunque, entra nello stato finale q_{NP}

1.4) Palindromia – con simulazione

- Ma cosa significa esattamente simulare una macchina di Turing così come lo abbiamo descritto alla pagina precedente?
- Esplicitiamo le tre fasi:
  - FASE 1) copia di x sul secondo e terzo nastro (e riposizionamento delle testine sui primi caratteri a sinistra): $\langle q_0$, (a, ␣, ␣), (a, a, a), q_0 , (D,D,D)$\rangle$,
    $\langle q_0$, (b, ␣, ␣), (b, b, b), q_0 , (D,D,D)$\rangle$,
    $\langle q_0$, (␣, ␣, ␣), (␣, ␣, ␣), q_1 , (S,S,S)$\rangle$,
    $\langle q_1$, (a, a, a), (a, a, a), q_1 , (S,S,S)$\rangle$,
    $\langle q_1$, (b, b, b), (b, b, b), q_1 , (S,S,S)$\rangle$,
    $\langle q_1$, (␣, ␣, ␣), (␣, ␣, ␣), $\mathbf{q_0^P}$, (D,D,D)$\rangle$,
    - osserviamo che l'ultima quintupla, che riporta le tre testine sul carattere più a sinistra di x, *fa entrare T_{PAL} nello stato interno iniziale $\mathbf{q_0^P}$ di* T_{PPAL}: ossia, predispone la macchina T_{PAL} alla simulazione di T_{PPAL}

1.4) Palindromia – con simulazione

- Ma cosa significa esattamente simulare una macchina di Turing così come lo abbiamo descritto alla pagina precedente?
- Esplicitiamo le tre fasi:
  - FASE 2) **simula T_{PPAL}** sul secondo nastro: ora, è necessario copiare le quintuple di T_{PPAL} facendola lavorare sul secondo nastro (senza muovere le testine sugli altri nastri):
    - ad esempio le due quintuple $\langle q_0^P$,a, $\square$, q_a^P , D$\rangle$, $\langle q_0^P$, b, $\square$, q_b^P , D$\rangle$ di T_{PPAL} diventano
      $\langle q_0^P$,(a, a, a), (a, $\square$, a), q_a^P , (F, D, F)$\rangle$,
      $\langle q_0^P$, (b, b, b), (b, $\square$, b), q_b^P , (F, D , F)$\rangle$
    - e così via...
    - osserviamo che tutti gli stati interni di T_{PPAL} sono anche stati interni di T_{PAL}
    - pertanto la simulazione di T_{PPAL} terminerà con la macchina T_{PAL} che si trova nello stato interno q_{PP}^P o q_{NPP}^P
      - con la testina sul primo e terzo nastro posizionate sui caratteri più a sinistra

1.4) Palindromia – con simulazione

- Ma cosa significa esattamente simulare una macchina di Turing?
- Esplicitiamo le tre fasi:
  - FASE 2) simula T_{PPAL} sul secondo nastro: ora, sarebbe necessario copiare le quintuple di T_{PPAL} facendola lavorare sul secondo nastro:
    - la fase 2) termina allora con le quintuple che portano T_{PAL} ad accettare:
      $\langle q_{PP}^{P}$, (a, ␣, a), (a, ␣, a), q_P , (F, F, F)$\rangle$,
      $\langle q_{PP}^{P}$, (b, ␣, b), (b, ␣, b), q_P , (F, F, F)$\rangle$,
    - [.. continua...]

1.4) Palindromia – con simulazione

- Ma cosa significa esattamente simulare una macchina di Turing?
- Esplicitiamo le tre fasi:
  - FASE 2) simula T_{PPAL} sul secondo nastro: ora, sarebbe necessario copiare le quintuple di T_{PPAL} facendola lavorare sul secondo nastro:
    - [... segue ...]
    - oppure, la fase 2) termina facendo entrare T_{PAL} nello stato $\mathbf{q_0^D}$ (che predispone alla simulazione di T_{DPAL} della fase 3)):

      $\langle\, \mathbf{q_{NPP}^P}, (a, \square, a), (a, \square, a), \mathbf{q_0^D}, (F, F, F)\rangle$,
      $\langle\, \mathbf{q_{NPP}^P}, (b, \square, b), (b, \square, b), \mathbf{q_0^D}, (S, S, S)\rangle$,

      - OSSERVAZIONE: in T_{PAL} lo stato finale $\mathbf{q_{NPP}^P}$ di T_{PPAL} non è uno stato finale!

1.4) Palindromia – con simulazione

- Ma cosa significa esattamente simulare una macchina di Turing?
- Esplicitiamo le tre fasi:
  - FASE 3) simula T_{DPAL} sul terzo nastro: ora, è necessario copiare le quintuple di T_{DPAL} facendola lavorare sul terzo nastro:
    - ad esempio le due quintuple $\langle q_0^D, a, \square, q_a^D, D\rangle$, $\langle q_0^D, b, \square, q_b^D, D\rangle$ di T_{DPAL} diventano
      $\langle q_0^D, (a, a, a), (a, a, \square), q_a^D, (F, F, D)\rangle$,
      $\langle q_0^D, (b, b, b), (b, b, \square), q_b^D, (F, F, D)\rangle$
    - e così via...
    - osserviamo che tutti gli stati interni di T_{DPAL} sono anche stati interni di T_{PAL}
    - pertanto la simulazione di T_{DPAL} terminerà con la macchina T_{PAL} che si trova nello stato interno q_{PD}^D o q_{NPD}^P

1.4) Palindromia – con simulazione

- Ma cosa significa esattamente simulare una macchina di Turing?
- Esplicitiamo le tre fasi:
  - FASE 3) simula T_{DPAL} sul terzo nastro: ora, sarebbe necessario copiare le quintuple di T_{DPAL} facendola lavorare sul terzo nastro:
    - la fase 3) termina allora con le quintuple che portano T_{PAL} ad accettare o a rigettare:

$\langle q^{D}_{PD}, (x, y, z), (x, y, z), q_{P}, (F, F, F)\rangle \quad \forall x, y, z \in \{a, b, \square\}$,

$\langle q^{D}_{NPD}, (x, y, z), (x, y, z), q_{NP}, (F, F, F)\rangle \quad \forall x, y, z \in \{a, b, \square\}$

1.4) Palindromia – con simulazione

- Ma il gioco vale la candela?! Cioè: sembra che abbiamo fatto più fatica a simulare che a costruire ex-novo
- In effetti, però, quella che abbiamo mostrato è una simulazione *dettagliata*
  - che abbiamo utilizzato solo per capire bene come funziona la simulazione
- Ora che l'abbiamo capito, però, possiamo descrivere in modo molto più semplice una macchina che simula un'altra macchina
- esattamente allo stesso modo in cui descriveremmo una invocazione di funzione in un linguaggio di programmazione
- utilizzando una notazione che ci... aiuterà a esprimere le quintuple della simulazione in modo sintetico

1.4) Palindromia – con simulazione

- Possiamo descrivere in modo molto più semplice una macchina che simula un'altra macchina
  - esattamente allo stesso modo in cui descriveremmo una invocazione di funzione in un linguaggio di programmazione
- Ossia, possiamo descrivere T_{PAL} nel modo seguente: T_{PAL} è una macchina a 3 nastri che opera in tre fasi: sul nastro N_1 è scritto l'input, il nastro N_2 serve alla simulazione di T_{PPAL}, il nastro N_3 serve alla simulazione di T_{DPAL}
  - FASE 1) copia l'input x su N_2 e N_3 ; *e le quintuple le abbiamo viste*

$\langle q_0$, (a, ▢, ▢), (a, a, a), q_0 , (D,D,D)$\rangle$,
$\langle q_0$, (b, ▢, ▢), (b, b, b), q_0 , (D,D,D)$\rangle$,
$\langle q_0$, (▢, ▢, ▢), (▢, ▢, ▢), q_1 , (S,S,S)$\rangle$,
$\langle q_1$, (a, a, a), (a, a, a), q_1 , (S,S,S)$\rangle$,
$\langle q_1$, (b, b, b), (b, b, b), q_1 , (S,S,S)$\rangle$,
$\langle q_1$, (▢, ▢, ▢), (▢, ▢, ▢), q_0^P, (D,D,D)$\rangle$,

1.4) Palindromia – con simulazione

- Possiamo descrivere in modo molto più semplice una macchina che simula un'altra macchina
  - esattamente allo stesso modo in cui descriveremmo una invocazione di funzione in un linguaggio di programmazione
- Ossia, possiamo descrivere T_{PAL} nel modo seguente: T_{PAL} è una macchina a 3 nastri che opera in tre fasi: sul nastro N_1 è scritto l'input, il nastro N_2 serve alla simulazione di T_{PPAL}, il nastro N_3 serve alla simulazione di T_{DPAL}
  - FASE 2) simula T_{PPAL} sul secondo nastro e, quando termina, decide se x è palindroma oppure deve procedere con la fase 3):
    - per indicare la simulazione invece di scrivere esplicitamente le quintuple che avevamo indicato in rosso utilizziamo la notazione $\langle q_0^P, N_2, N_2, T_{PPAL}, (F, F, F)\rangle$,
    - al termine della precedente ''istruzione'' assumiamo che le testine su N_1 e N_3 sono posizionate sui caratteri più a sinistra, la testina su N_2 legge ␣ e lo stato interno di T_{PAL} è quello nel quale termina T_{PPAL}
    - cui seguono, pertanto, le quintuple $\langle q_{PP}^P, (a, ␣, a), (a, ␣, a), q_P, (F, F, F)\rangle$,
      $\langle q_{PP}^P, (b, ␣, b), (b, ␣, b), q_P, (F, F, F)\rangle$,
      $\langle q_{NPP}^P, (a, ␣, a), (a, ␣, a), q_0^D, (F, F, F)\rangle$,
      $\langle q_{NPP}^P, (b, ␣, b), (b, ␣, b), q_0^D, (F, F, F)\rangle$,

1.4) Palindromia – con simulazione

- Possiamo descrivere in modo molto più semplice una macchina che simula un'altra macchina
  - esattamente allo stesso modo in cui descriveremmo una invocazione di funzione in un linguaggio di programmazione
- Ossia, possiamo descrivere T_{PAL} nel modo seguente: T_{PAL} è una macchina a 3 nastri che opera in tre fasi: sul nastro N_1 è scritto l'input, il nastro N_2 serve alla simulazione di T_{PPAL}, il nastro N_3 serve alla simulazione di T_{DPAL}
  - FASE 3) simula T_{DPAL} sul terzo nastro e, quando termina, decide definitivamente se x è palindroma o no:
    - per indicare la simulazione invece di scrivere esplicitamente le quintuple che avevamo indicato in verde utilizziamo la notazione $\langle q_0^D, N_3, N_3, T_{DPAL}, (F, F, F)\rangle$,
    - al termine della precedente ''istruzione'' assumiamo che le testina su N_1 è posizionata sul caratteei più a sinistra, le testine su N_2 e N_3 leggono $\square$ e lo stato interno di T_{PAL} è quello nel quale termina T_{DPAL}
    - cui seguono le quintuple $\langle q_{PP}^P, (a, \square, \square), (a, \square, \square), q_P, (F, F, F)\rangle$,
      $\langle q_{PP}^P, (b, \square, \square), (b, \square, \square), q_P, (F, F, F)\rangle$,
      $\langle q_{NPP}^P, (a, \square, \square), (a, \square, \square), q_{NP}, (S, S, S)\rangle$,
      $\langle q_{NPP}^P, (b, \square, \square), (b, \square, \square), q_{NP}, (S, S, S)\rangle$,

1.4) Palindromia – con simulazione

- **OSSERVAZIONE**: quelle che abbiamo indicato con $\langle \mathbf{q_0^P}, \mathbf{N_2}, \mathbf{N_2}, \mathbf{T_{PPAL}}, \mathbf{(F, F, F)} \rangle$ e $\langle \mathbf{q_0^D}, \mathbf{N_3}, \mathbf{N_3}, \mathbf{T_{DPAL}}, \mathbf{(F, F, F)} \rangle$ non sono quintuple di una macchina di Turing!

- Sono solo una *notazione*, ossia, una scorciatoia: siccome le due macchine T_{PPAL} e T_{DPAL} le avevamo già progettate, utilizzarle all'interno delle quintuple di T_{PAL} sarebbe stato nient'altro che un (noioso) esercizio di riscrittura...

- ... e, siccome non amiamo perder tempo, abbiamo trovato un modo (una notazione) che ci permetta di esprimere in modo sintetico tutte le quintuple rosse e verdi che avremmo dovuto replicare

1.4) Palindromia – con simulazione

- **OSSERVAZIONE**: quelle che abbiamo indicato con $\langle \mathbf{q_0^P}, \mathbf{N_2}, \mathbf{N_2}, \mathbf{T_{PPAL}}, \mathbf{(F, F, F)} \rangle$ e $\langle \mathbf{q_0^D}, \mathbf{N_3}, \mathbf{N_3}, \mathbf{T_{DPAL}}, \mathbf{(F, F, F)} \rangle$ Sono solo una *notazione*, non sono quintuple di una macchina di Turing!
- In effetti, poiché le operazioni che T_{PAL} deve eseguire oltre alle due simulazioni sono operazioni di copiatura, di riavvolgimento e di test dello stato interno (ossia, operazioni ***molto*** *semplici*), una volta compresa la notazione appena introdotta possiamo descrivere T_{PAL} ad alto livello come avevamo fatto inizialmente:
  - T_{PAL} utilizza tre nastri: con input x sul primo nastro
    - FASE 1) copia l'input x sul secondo e terzo nastro;
    - FASE 2) simula T_{PPAL} sul secondo nastro: se T_{PPAL}(x) termina in q_{PP}^P allora x è una parola palindroma di lunghezza pari e T_{PAL} entra nello stato finale q_P, altrimenti (l'input ha lunghezza dispari o non è palindroma) passa alla fase 3)
    - FASE 3) simula T_{DPAL} sul secondo nastro: se T_{DPAL}(x) termina in q_{PD}^D allora x è una parola palindroma di lunghezza dispari e T_{PAL} entra nello stato finale q_P, altrimenti (l'input ha lunghezza dispari ma non è palindroma), grazie all'esito della fase 2), può concludere che x non è palindroma e, dunque, entra nello stato finale q_{NP}

2) Somma di k (non costante) interi

- **ESERCIZIO 2**: Progettare una macchina di Turing a due nastri che, avendo sul primo nastro una serie di numeri interi separati dal carattere '+', calcola il valore della loro somma scrivendo il risultato sul secondo nastro – ossia, si richiede di progettare una macchina di Turing che esegua la somma "in riga" di una quantità imprecisata di numeri
- A lezione abbiamo visto una macchina che esegue questo compito nel caso particolare in cui i numeri da sommare sono 2 e tali numeri hanno lo stesso numero di cifre
- e, per esercizio, vi avevo chiesto di generalizzare al caso in cui i due numeri non hanno lo stesso numero di cifre
  - chiamiamo questa macchina T_{A+B}
- Vediamo ora come utilizzare T_{A+B} per risolvere il nostro esercizio
- Ossia, progettiamo una macchina di Turing che risolve il nostro esercizio simulando T_{A+B}

2) Somma di k (non costante) interi2)

- **ESERCIZIO 2**: Progettare una macchina di Turing a due nastri che, avendo sul primo nastro una serie di numeri interi separati dal carattere '+', calcola il valore della loro somma scrivendo il risultato sul secondo nastro
  - ossia, si richiede di progettare una macchina di Turing che esegua la somma "in riga" di una quantità imprecisata di numeri
- **SOLUZIONE**: Progettiamo la macchina $T_{SOMMATORIA}$ dotandola di 3 nastri: il nastro N_1 che conterrà l'input, il nastro N_2 che sarà il nastro di input e di lavoro di T_{A+B}, e il nastro N_3 che sarà il nastro di output di T_{A+B}; N_2 è il nastro di output di
- Descriviamo $T_{SOMMATORIA}$ ad alto livello:
  - $T_{SOMMATORIA}$: con input $x_1+x_2+ \dots + x_n$ su N_1 opera come segue
  - FASE 1) scrive '0+' su N_2
  - FASE 2) se N_1 non è vuoto copia il primo addendo (ossia, tutti i caratteri a sinistra del primo '+') da N_1 a N_2 a destra del '+', cancellandolo da N_1, altrimenti cancella il '+' da N_2 e termina
  - FASE 3) simula T_{A+B} su N_2 scrivendo il risultato su N_3
  - FASE 4) cancella il contenuto di N_2 e, cancellandolo da N_3, copia il contenuto di N_3 su N_2;aggiunge il carattere '+' come ultimo carattere a destra di N_2 e torna alla fase 2)

2) Somma di k (non costante) interi2)

- **ESERCIZIO 2**: Progettare una macchina di Turing a due nastri che, avendo sul primo nastro una serie di numeri interi separati dal carattere '+', calcola il valore della loro somma scrivendo il risultato sul secondo nastro
  - ossia, si richiede di progettare una macchina di Turing che esegua la somma "in riga" di una quantità imprecisata di numeri
- **SOLUZIONE**: Descriviamo $T_{SOMMATORIA}$ a basso livello:
  - FASE 1) scrive '0+' su N_2 : per ogni a ∈ {0, 1, ..., 9} $\langle q_0, (a, \square, \square), (a, 0, \square), q_1, (F, D, F)\rangle$
    $\langle q_1, (a, \square, \square), (a, +, \square), q_2, (F, D, F)\rangle$
  - FASE 2) se N_1 non è vuoto copia il primo addendo (ossia, tutti i caratteri a sinistra del primo '+') da N_1 a N_2 a destra del '+' su N_2, cancellandolo da N_1:
    per ogni a ∈ {0, 1, ..., 9} $\langle q_2, (a, \square, \square), (\square, a, \square), q_2, (D, D, F)\rangle$
    $\langle q_2, (+, \square, \square), (\square, \square, \square), q_3, (D, F, F)\rangle$

    altrimenti cancella il '+' da N_2 e termina
    $\langle q_2, (\square, \square, \square), (\square, \square, \square), q_4, (F, S, F)\rangle$
    $\langle q_4, (\square, +, \square), (\square, \square, \square), q_F, (F, F, F)\rangle$

2) Somma di k (non costante) interi2)

- **ESERCIZIO 2**: Progettare una macchina di Turing a due nastri che, avendo sul primo nastro una serie di numeri interi separati dal carattere '+', calcola il valore della loro somma scrivendo il risultato sul secondo nastro
  - ossia, si richiede di progettare una macchina di Turing che esegua la somma "in riga" di una quantità imprecisata di numeri
- **SOLUZIONE**: Descriviamo $T_{SOMMATORIA}$ a basso livello:
  - FASE 3) simula T_{A+B} su N_2 scrivendo il risultato su N_3 :

    $\langle q_3, N_2, N_3, T_{A+B}, (F, F, F)\rangle$

    il cui significato è:

    esegui le quintuple della macchina T_{A+B} utilizzando N_2 come nastro di input e N_3 come nastro di output
    - e così abbiamo introdotto la notazione per simulare una macchina di Turing che scrive un risultato su un nastro di output
  - **indichiamo con q_F^{A+B} lo stato finale di T_{A+B} e assumiamo che, al termine della simulazione la macchina $T_{SOMMATORIA}$ si trovi nello stato q_F^{A+B}**
  - **e supponiamo che al termine della simulazione la testina su N_3 sia posizionata sul carattere più a sinistra**

2) Somma di k (non costante) interi2)

- **ESERCIZIO 2**: Progettare una macchina di Turing a due nastri che, avendo sul primo nastro una serie di numeri interi separati dal carattere '+', calcola il valore della loro somma scrivendo il risultato sul secondo nastro
  - ossia, si richiede di progettare una macchina di Turing che esegua la somma "in riga" di una quantità imprecisata di numeri
- **SOLUZIONE**: Descriviamo $T_{SOMMATORIA}$ a basso livello:
  - FASE 4) cancella il contenuto di N_2 :
    $\forall$ a, c $\in$ {0, 1, ..., 9, +, ␣} e $\forall$ b $\in$ {0, 1, ..., 9, +} $\langle q_F^{A+B}, (a, b, c), (a, ␣, c), q_F^{A+B}, (F, S, F)\rangle$
    $\langle q_F^{A+B}, (a, ␣, c), (a, ␣, c), q_4, (F, S, F)\rangle$
    e, cancellandolo da N_3, copia il contenuto di N_3 su N_2 , infine aggiunge il carattere '+' come ultimo carattere a destra di N_2 e torna alla fase 2)
    $\forall$ a $\in$ {0, 1, ..., 9, +, ␣} e $\forall$ c $\in$ {0, 1, ..., 9} $\langle q_4, (a, ␣, c), (a, c, c), q_4, (F, D, D)\rangle$
    $\forall$ a $\in$ {0, 1, ..., 9, +, ␣} $\langle q_4, (a, ␣, ␣), (a, +, ␣), q_2, (F, D, F)\rangle$

Note sulla simulazione

- La tecnica della simulazione permette in molti casi di semplificare il progetto di nuove macchine di Turing
- In particolare, quando possiamo utilizzare una certa macchina di Turing data così com'è
  - ossia, senza che sia necessario modificarla
  - come abbiamo fatto con gli esercizi 1) e 2)
- questo tipo di simulazione possiamo chiamarla ''***a scatola chiusa***''
- e possiamo utilizzarla anche quando, per farlo occorre modificare l'input
- **Esempio**:
  - la macchina di Turing T_2^1 termina nello stato $q_{sì}^{2,1}$ se il suo input contiene un numero pari di '1' e nello stato $q_{no}^{2,1}$ altrimenti, mentre la macchina di Turing T_3^0 termina nello stato $q_{sì}^{3,0}$ se il suo input contiene un numero di '0' che è un multiplo di 3 e nello stato $q_{no}^{3,0}$ altrimenti
  - abbiamo bisogno di progettare una macchina di Turing T_6^1 termina nello stato $q_{sì}^{6,1}$ se il suo input contiene un numero di '1' che è un multiplo di 3 e nello stato $q_{no}^{2,1}$ altrimenti
  - Possiamo utilizzare a scatola chiusa T_2^1 e T_3^0 per costruire T_6^1?

3) Utilizzare T_2^1 e T_3^0 per costruire T_6^1

- Possiamo utilizzare a scatola chiusa T_2^1 e T_3^0 per costruire T_6^1?
- Naturalmente, data una parola $x \in \{0,1\}^*$, il numero di '1' che essa contiene è multiplo di 6 se e soltanto se tale numero è pari e multiplo di 3
  - Nessun dubbio, dunque, che T_2^1 potrebbe esserci utile!
  - D'altra parte, anche T_3^0 sembra vicina alle nostre esigenze
  - se non fosse (mannaggia!) che conta il numero di '0' invece che di '1'
- E allora? Allora facciamo così:
- Progettiamo T_6^1 con 3 nastri che con input $x \in \{0,1\}^*$ opera come segue:
  - FASE 1) copia x sul secondo nastro e, su di esso, simula a scatola nera T_2^1: se $T_2^1(x)$ termina in $q_{no}^{2,1}$ allora T_6^1 termina in $q_{no}^{6,1}$ (perché il numero di '1' in x è dispari); se $T_2^1(x)$ termina in $q_{no}^{2,1}$ allora ha inizio la fase 2)
  - FASE 2) scrive x^c sul terzo nastro (ossia, per ogni carattere x_i di x scrive 0 se x_i=1, scrive 1 se x_i=0) e, su di esso, simula a scatola nera T_3^0: se $T_3^0(x)$ termina in $q_{no}^{3,0}$ allora T_6^1 termina in $q_{no}^{6,1}$; se $T_2^1(x)$ termina in $q_{no}^{2,1}$ allora T_6^1 termina in $q_{sì}^{6,1}$

Note sulla simulazione

- Osserviamo che per simulare ***a scatola chiusa*** non è necessario conoscere le quintuple della macchina che stiamo simulando
  - esattamente come accade quando invochiamo una funzione C o utilizziamo una classe Java
  - conosciamo le loro specifiche, mica il loro codice!
- Esiste, nel mondo delle macchine di Turing, un'altra tecnica di simulazione che prevede, invece, di simulare esplicitamente una quintupla alla volta
- e, per poterlo fare, è necessario (figurativamente parlando) *aprire* la macchina di Turing che intendiamo simulare
  - ossia, abbiamo bisogno di conoscere non solo le sue specifiche, ma *tutte le quintuple che la definiscono*
  - nella terminologia dei linguaggi di programmazione, abbiamo bisogno di conoscere il codice sorgente
- per tale ragione chiameremo ''***a scatola aperta***'' questa seconda tecnica di simulazione

Note sulla simulazione

- Naturalmente, quando è possibile si deve utilizzare la tecnica a scatola chiusa
- e riservarsi di utilizzare la tecnica a scatola aperta solo quando è strettamente necessario
- E quando è *strettamente necessario* utilizzare la tecnica a scatola aperta?
  - Sostanzialmente, quando dobbiamo simulare un certo numero di macchine che, su alcuni input (a noi sconosciuti), potrebbero andare in loop
    - ossia, non raggiungere mai lo stato finale ma continuare ad eseguire quintuple ''all'infinito''
  - mentre a noi serve che almeno una delle macchine arrivi al termine
- Questa tecnica la utilizzeremo in ambiti più teorici e ogni volta sottolineeremo che stiamo simulando a scatola aperta

Esercizi sulle macchine di Turing

- Torniamo al progetto ''classico'' di macchine di Turing
  - ossia, senza l'utilizzo della simulazione
- **ESERCIZIO 4**: vogliamo progettare una macchina di Turing T_{DOPPIA} che, con una parola $x \in \{a, b\}^*$ scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è nella forma yy o meno, con $y \in \{a, b\}^*$
  - esempio: sono in questa forma le parole abbaabba e babbab
- **SOLUZIONE**:
  - progettiamo T_{DOPPIA} ad un solo nastro, le cui computazioni terminano nello stato q_{doppia} quando l'input x ha la forma yy, terminano nello stato $q_{\text{non doppia}}$ altrimenti
  - intanto, osserviamo che affinché sia x = yy la lunghezza di x deve essere necessariamente pari
  - allora, la prima cosa che T_{DOPPIA} deve fare è individuare due parole, y_1 e y_2, della stessa lunghezza tali che $x = y_1\ y_2$, (FASE 1)
  - una volta individuate y_1 e y_2, non resterà che dimostrare se $y_1 = y_2$ (FASE 2)

4) Parola doppia

- **ESERCIZIO 4**: vogliamo progettare una macchina di Turing T_{DOPPIA} che, con una parola $x \in \{a, b\}^*$ scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è nella forma yy o meno, con $y \in \{a, b\}^*$
- **SOLUZIONE**: descriviamo le due fasi ad alto livello
  - FASE 1) è questa una fase di marcatura dei caratteri di x come appartenenti a x_1 oppure a x_2: marchiamo 1 il carattere più a sinistra, 2 il carattere più a destra, 1 il secondo carattere più a sinistra, 2 il penultimo carattere più a destra, ...
    - inizialmente nessun carattere è marcato e la testina è posizionata sul carattere più a sinistra
    - (1.1) se la testina è posizionata su un carattere non marcato allora lo marchiamo come appartenente a x_1 (ossia, sovrascriviamo a con a_1 e b con b_1), ci spostiamo sul carattere più a destra non marcato (se esiste) e passiamo a (1.2); se la testina legge un carattere marcato passiamo alla fase 2,
    - (1.2) se la testina è posizionata su un carattere non marcato allora lo marchiamo come appartenente a x_2 (ossia, sovrascriviamo a con a_2 e b con b_2), ci spostiamo sul carattere a destra del carattere marcato 1 più a sinistra e torniamo a (1.1); altrimenti concludiamo che y contiene un numero dispari di caratteri e quindi non può essere della forma yy

4) Parola doppia

- **ESERCIZIO 4**: vogliamo progettare una macchina di Turing T_{DOPPIA} che, con una parola $x \in \{a, b\}^*$ scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è nella forma yy o meno, con $y \in \{a, b\}^*$
- **SOLUZIONE**: descriviamo le due fasi ad alto livello
  - FASE 2: utilizziamo la marcatura eseguita nella fase 1)
    - ora sappiamo che y ha la forma x_1x_2, quindi ad ogni carattere marcato 1 corrisponde un carattere marcato 2
    - riportiamo la testina sul carattere più a sinistra
    - se il carattere letto è a_1 o b_1 lo memorizziamo nello stato interno, lo cancelliamo e ci spostiamo verso destra sul primo carattere che incontriamo che è marcato 2 (ossia, un carattere a_2 o b_2), altrimenti se il carattere letto è '*' allora abbiamo controllato tutte le coppie di caratteri e possiamo concludere che x = yy
    - se il carattere letto è uguale a quello memorizzato lo sostituiamo con '*' e ci spostiamo sul carattere a destra del primo blank a sinistra, altrimenti se il carattere letto è diverso da quello memorizzato e possiamo concludere che $x \neq yy$

4) Parola doppia

- **ESERCIZIO 4**: vogliamo progettare una macchina di Turing T_{DOPPIA} che, con una parola x ∈ {a, b}* scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è nella forma yy o meno, con y ∈ {a, b}*
- **SOLUZIONE**: ora possiamo descrivere le due fasi a basso livello
  - FASE 1)
    - (1.1) se la testina è posizionata su un carattere non marcato allora lo marchiamo come appartenente a x_1 (ossia, sovrascriviamo a con a_1 e b con b_1), ci spostiamo sul carattere a sinistra del ␣ o del primo carattere marcato 2 e passiamo a (1.2); altrimenti se la testina legge un carattere marcato passiamo alla fase 2:

$$\langle q_0, a, a_1, q_1, D\rangle \ , \ \langle q_0, b, b_1, q_1, D\rangle \ ,$$
$$\langle q_1, a, a, q_1, D\rangle \ , \ \langle q_1, b, b, q_1, D\rangle \ ,$$
$$\langle q_1, ␣, ␣, q_{sin}, S\rangle \ , \ \langle q_1, a_2, a_2, q_{sin}, S\rangle \ , \ \langle q_1, b_2, b_2, q_{sin}, S\rangle$$
$$\langle q_0, a_1, a_1, q_{fase\ 2}, F\rangle \ , \ \langle q_0, b_1, b_1, q_{fase\ 2}, F\rangle \ ,$$

4) Parola doppia

- **ESERCIZIO 4**: vogliamo progettare una macchina di Turing T_{DOPPIA} che, con una parola x ∈ {a, b}* scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è nella forma yy o meno, con y ∈ {a, b}*
- **SOLUZIONE**: ora possiamo descrivere le due fasi a basso livello
  - FASE 1)
    - (1.2) se la testina è posizionata su un carattere non marcato allora lo marchiamo come appartenente a x_2 (ossia, sovrascriviamo a con a_2 e b con b_2), ci spostiamo sul carattere a destra del carattere marcato 1 più a sinistra e torniamo a (1.1); altrimenti concludiamo che y contiene un numero dispari di caratteri e quindi non può essere della forma yy $\langle q_{sin}, a, a_2, q_2, S\rangle$, $\langle q_{sin}, b, b_2, q_2, S\rangle$ $\langle q_2, a, a, q_2, S\rangle$, $\langle q_2, b, b, q_2, S\rangle$ $\langle q_2, a_1, a_1, q_0, D\rangle$, $\langle q_2, b_1, b_1, q_0, D\rangle$ $\langle q_{sin}, a_1, a_1, q_{non\ doppia}, S\rangle$, $\langle q_{sin}, b_1, b_1, q_{non\ doppia}, S\rangle$

4) Parola doppia

- **ESERCIZIO 4**: vogliamo progettare una macchina di Turing T_{DOPPIA} che, con una parola x ∈ {a, b}* scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è nella forma yy o meno, con y ∈ {a, b}*
- **SOLUZIONE**: ora possiamo descrivere le due fasi a basso livello
  - FASE 2)
    - riportiamo la testina sul carattere più a sinistra
      $\langle q_{\text{fase 2}}, a_1, a_1, q_{\text{fase 2}}, S\rangle$, $\langle q_{\text{fase 2}}, b_1, b_1, q_{\text{fase 2}}, S\rangle$, $\langle q_{\text{fase 2}}, \square, \square, q_{\text{des}}, D\rangle$
    - se il carattere letto è a_1 o b_1 lo memorizziamo nello stato interno, lo cancelliamo e ci spostiamo verso destra sul primo carattere che incontriamo che è marcato 2 (ossia, un carattere a_2 o b_2), altrimenti se il carattere letto è '*' allora abbiamo controllato tutte le coppie di caratteri e possiamo concludere che x = yy
      $\langle q_{\text{des}}, a_1, \square, q_a^1, D\rangle$, $\langle q_{\text{des}}, b_1, \square, q_b^1, D\rangle$,
      $\langle q_a^1, a_1, a_1, q_a^1, D\rangle$, $\langle q_a^1, b_1, b_1, q_a^1, D\rangle$, $\langle q_a^1, *, *, q_a^1, D\rangle$
      $\langle q_b^1, a_1, a_1, q_b^1, D\rangle$, $\langle q_b^1, b_1, b_1, q_b^1, D\rangle$, $\langle q_b^1, *, *, q_b^1, D\rangle$,
      $\langle q_{\text{des}}, *, *, q_{\text{doppia}}, F\rangle$,
    - [... continua ...]

4) Parola doppia

- **ESERCIZIO 4**: vogliamo progettare una macchina di Turing T_{DOPPIA} che, con una parola x ∈ {a, b}* scritta sul nastro, esegue una serie di passi che terminano in uno stato che attesta se x è nella forma yy o meno, con y ∈ {a, b}*
- **SOLUZIONE**: ora possiamo descrivere le due fasi a basso livello
  - FASE 2)
    - [... segue ...]
    - se il carattere letto è uguale a quello memorizzato lo sostituiamo con '*' e ci spostiamo sul carattere a destra del primo blank a sinistra, se il carattere letto è diverso da quello memorizzato possiamo concludere che x ≠ yy

$$\langle q_a^1, a_2, *, q_3, S\rangle \ , \ \langle q_b^1, b_2, *, q_3, S\rangle \ ,$$
$$\langle q_a^1, b_2, *, q_{\text{non doppia}}, F\rangle \ , \ \langle q_b^1, a_2, *, q_{\text{non doppia}}, F\rangle \ ,$$
$$\langle q_3, a_1, a_1, q_3, S\rangle \ , \ \langle q_3, b_1, b_1, q_3, S\rangle \ , \ \langle q_3, *, *, q_3, S\rangle \ ,$$
$$\langle q_3, \square, \square, q_{\text{fase 2}}, D\rangle$$