Capitolo 1

Esercizi

1.1 Espressioni e Comandi (Cap 2)

Esercizio 1 *Valutare le seguenti espressioni:*

1. $[\lambda x, y, z.\langle z, x\rangle](3, 1, 2)$
2. $[I_{3,2}; C_5](3, 1, 2)$
3. $[\lambda x, y.x^2 + 1 > y](2, 5)$
4. $[\lambda x.\sqrt{x} \wedge (x - 1)](4)$
5. $[I_{3,2} \wedge \lambda x, y, z.x + 1](3, 2, 1)$
6. $[I \times \lambda y.y^2](3, 2)$
7. $[\lambda x, y.x > y \to x, \sqrt{x + 1}](3, 4)$
8. $[\lambda x.\langle\sqrt{x}, \frac{x}{2}\rangle](4)$
9. $[I_{3,2} \wedge I_{3,1}](\langle 1, 3, 2\rangle)$

Soluzione:

1. $[\lambda x, y, z.\langle z, x\rangle](3, 1, 2) = \langle 2, 3\rangle$
2. $[I_{3,2}; C_5](3, 1, 2) = [C_5](I_{3,2}(3, 1, 2)) = [C_5](1) = 5$
3. $[\lambda x, y.x^2 + 1 > y](2, 5) = 2^2 + 1 > 5 = 5 > 5 \Rightarrow 0$
4. $[\lambda x.\sqrt{x} \wedge (x - 1)](4) = [\lambda x.\langle\sqrt{x}, (x - 1)\rangle](4) = \langle 2, 3\rangle$

5. $[I_{3,2} \wedge \lambda x, y, z.x + 1](3, 2, 1) = [\lambda x, y, z.y \wedge \lambda x, y, z.x + 1](3, 2, 1) = \langle 2, 4 \rangle$
6. $[I \times \lambda y.y^2](3, 2) = \langle \lambda x.x(3), \lambda y.y^2(2) \rangle = \langle 3, 4 \rangle$
7. $[\lambda x, y.x > y \rightarrow x, \sqrt{x+1}](3, 4) = 3 > 4 \rightarrow 3, \sqrt{4} = \sqrt{4} = 2$
8. $[\lambda x.\langle \sqrt{x}, \frac{x}{2} \rangle](4) = \langle \sqrt{4}, \frac{4}{2} \rangle = \langle 2, 2 \rangle$
9. $[I_{3,2} \wedge I_{3,1}](\langle 1, 3, 2 \rangle) = \langle I_{3,2}, I_{3,1} \rangle (\langle 1, 3, 2 \rangle) = \langle 3, 1 \rangle$

Esercizio 2 *Sia* $f^n = [\lambda x, y.(n = 0) \rightarrow x, f^{n-1} + (x \cdot y)]$, *calcolare* $f^3(4, 5)$.

Soluzione:

$$\begin{aligned} f^3(4,5) &= [\lambda x, y.(3 = 0) \rightarrow x, f^2 + (x \cdot y)](4, 5) \\ &= f^2 + 4 \cdot 5 \\ &= [\lambda x, y.(2 = 0) \rightarrow x, f^1 + (x \cdot y)](4, 5) + 4 \cdot 5 \\ &= f^1 + 4 \cdot 5 + 4 \cdot 5 \\ &= [\lambda x, y.(1 = 0) \rightarrow x, f^0 + (x \cdot y)](4, 5) + 20 + 20 \\ &= f^0 + 4 \cdot 5 + 40 \\ &= [\lambda x, y.(0 = 0) \rightarrow x, f^{-1} + (x \cdot y)](4, 5) + 60 \\ &= 4 + 60 = 64 \end{aligned}$$

Esercizio 3 *Sia* $f^n = [\lambda x.(n = 0) \rightarrow x, f^{n-1}(x) + f^{n-1}(x)]$, *calcolare* $f^3(2)$.

Soluzione:

$$\begin{aligned} f^3(2) &= [\lambda x.(3 = 0) \rightarrow x, f^2(x) + f^2(x)](2) \\ &= f^2(2) + f^2(2) = 2f^2(2) \\ &= 2[\lambda x.(2 = 0) \rightarrow x, f^1 + (x) + f^1(x)](2) \\ &= 2(f^1(2) + f^1(2)) = 4f^1(2) \\ &= 4[\lambda x.(1 = 0) \rightarrow x, f^0(x) + f^0(x)](2) \\ &= 4(f^0(2) + f^0(2)) = 8f^0(2) \\ &= 8[\lambda x.(0 = 0) \rightarrow x, f^{-1}(x) + f^{-1}(x)](2) \\ &= 8 * 2 = 16 \end{aligned}$$

Esercizio 4 *Dato* n, *sia* P *il seguente predicato:*

$$P(x) = \lambda x.(x^2 < n)((x + 1)^2 \geq n) \rightarrow 0, 1$$

Si dica qual è il valore di $succ^P(0)$, *dove* $succ^P = [\lambda x.P(x) \rightarrow succ^P(succ(x)), x]$.

Soluzione: Sia $n = 0$

$$\begin{aligned} P(0) &= [\lambda x.(x^2 < 0)((x+1)^2 \geq 0) \rightarrow 0, 1](0) \\ &= \lambda x.(0 < 0)(1^2 \geq 0) \rightarrow 0, 1 = 1 \end{aligned}$$

Sia $n = 1$

$$\begin{aligned} P(0) &= [\lambda x.(x^2 < 1)((x+1)^2 \geq 1) \rightarrow 0, 1](0) \\ &= \lambda x.(0 < 1)(1^2 \geq 1) \rightarrow 0, 1 = 0 \end{aligned}$$

Si dimostra facilmente che:

$$P(0) = \begin{cases} 0 & \text{se } n = 1 \\ 1 & \text{altrimenti} \end{cases}$$

Valutiamo $succ^P(0) = [\lambda x.P(x) \rightarrow succ^P(succ(x)), x](0)$
Se $n = 1$ allora $P(0) = 0$ da cui $succ^P(0) = 0$.
Se $n = 0$ allora $P(0) = 1$ da cui $succ^P(0) = succ^P(succ(0)) = succ^P(1) = succ^P(succ(1)) = succ^P(2) \ldots$ generiamo un calcolo infinito (questo vale per ogni $n \neq 1$).

Esercizio 5 *Rappresentare $\lambda x.\langle x, x\rangle$ tramite identità e accoppiamento congiunto.*

Soluzione: $\lambda x.\langle x, x\rangle = I \wedge I$

Esercizio 6 *Valutare le seguenti espressioni (indicando i vari passaggi necessari):*

1. $[(I_{3,2} \wedge I_{3,3}); (\lambda x.x^2 \times C_7)](3, 2, 1)$
2. $[I \wedge \lambda x.x + 1](7)$
3. $[\lambda x, y.x > y \rightarrow x, x + 1](3, 4)$

Soluzione:

1. $[(I_{3,2} \wedge I_{3,3}); (\lambda x.x^2 \times C_7)](3, 2, 1) = [\lambda x.x^2 \times C_7]\langle 2, 1\rangle = \langle 4, 7\rangle$
2. $[I \wedge \lambda x.x + 1](7) = \langle I, \lambda x.x + 1\rangle(7) = \langle 7, 8\rangle$
3. $[\lambda x, y.x > y \rightarrow x, x + 1](3, 4) = 3 > 4 \rightarrow 3, 3 + 1 = 4$

1.2 Istruzioni e Programmi (Cap 3)

Esercizio 7 *Scrivere un programma a registri che calcola l'operazione:*

$$\lambda x, y.x = 1 \rightarrow x + 1, y$$

Soluzione:

1 input_1 2	4 inc_3 5	7 output_1 9
2 input_2 3	5 $\text{test}_{1,3}$ 6,8	8 output_2 9
3 zero_3 4	6 inc_1 7	9 stop

Script di shell che corrisponde alla precedente macchina a registri:

```
#!/bin/bash
counter=1
while [ 1 ]
do
case $counter in
 1 ) r1=$1; counter=2;;                  # input_1 2
 2 ) r2=$2; counter=3;;                  # input_2 3
 3 ) r3=0; counter=4;;                   # zero_3 4
 4 ) r3=`expr $r3 + 1`; counter=5;;      # inc_3 5
 5 ) if [ $r1 -eq $r3 ]                  #
        then            #
        counter=6       # test_{1,3} 6 8
        else            #
        counter=8       #
        fi              #
        ;;
 6 ) r1=`expr $r1 + 1`;counter=7;;       # inc_1 7
 7 ) echo $r1;counter=9;;                # output_1 9
 8 ) echo $r2;counter=9;;                # output_2 9
 9 ) exit;;                              # stop
esac
done
```

Esercizio 8 *Scrivere un programma a registri che calcola l'operazione:*

$$\lambda x, y.x > y$$

Soluzione:

1 $input_1$ 2	6 $test_{1,2}$ 11,7	11 $output_3$ 13
2 $input_2$ 3	7 $test_{1,3}$ 11,8	12 $output_4$ 13
3 $zero_3$ 4	8 $test_{2,3}$ 12,9	13 stop
4 $zero_4$ 5	9 dec_1 10	
5 inc_4 6	10 dec_2 7	

Script di shell che corrisponde alla precedente macchina a registri:

```
#!/bin/bash
counter=1
while [ 1 ]
do
case $counter in
 1 ) r1=$1; counter=2;;                  # input1 2
 2 ) r2=$2; counter=3;;                  # input2 3
 3 ) r3=0; counter=4;;                   # zero3 4
 4 ) r4=0; counter=5;;                   # zero4 5
 5 ) r4=`expr $r4 + 1`; counter=6;;      # inc4 6
 6 ) if [ $r1 -eq $r2 ]                  #

        then            #
        counter=11      # test1,2 11 7
        else            #
        counter=7       #
        fi              #
        ;;

7 ) if [ $r1 -eq $r3 ] #

        then            #
        counter=11      # test1,3 11 8
        else            #
        counter=8       #
        fi              #
        ;;

8 ) if [ $r2 -eq $r3 ] #
```

```
            then            #
            counter=12      # test_{2,3} 12 9
            else            #
            counter=9       #
            fi              #
            ;;
 9 ) r1=‘expr $r1 - 1‘;counter=10;;   # dec_1 10
 10 ) r2=‘expr $r2 - 1‘;counter=7;;   # dec_2 7
 11 ) echo $r3; counter=13;;          # output_3 13
 12 ) echo $r4; counter=13;;          # output_4 13
 13 ) exit;;                          # stop
esac
done
```

Esercizio 9 *Scrivere un programma a registri che calcola la seguente operazione (nell'ipotesi in cui $x > y$):*

$$\lambda x, y.x - y$$

Soluzione:

1 input_1 2	4 $\text{test}_{2,3}$ 7,5	7 output_1 8
2 input_2 3	5 dec_2 6	8 stop
3 zero_3 4	6 dec_1 4	

Script di shell che corrisponde alla precedente macchina a registri:

```
# !/bin/bash
counter=1
while [ 1 ]
do
case $counter in
 1 ) r1=$1; counter=2;;   # input_1 2
 2 ) r2=$2; counter=3;;   # input_2 3
 3 ) r3=0; counter=4;;    # zero_3 4
 4 ) if [ $r2 -eq $r3 ]   #
            then          #
            counter=7     # test_{2,3} 7 5
            else          #
            counter=5     #
            fi            #
            ;;
```

```
  5 ) r2=`expr $r2 - 1`;counter=6;;   # dec_2 6
  6 ) r1=`expr $r1 - 1`;counter=3;;   # dec_1 3
  7 ) echo $r1;counter=8;;            # output_1 8
  8 ) exit;;                          # stop
esac
done
```

Esercizio 10 *Scrivere dei programmi a registri che calcolano il minimo e il massimo tra due numeri naturali non nulli.*

Soluzione:
Consideriamo il problema di calcolare il minimo tra due numeri naturali non nulli:

1 $input_1$ 2	7 $test_{2,4}$ 9,8	13 dec_1 14
2 $input_2$ 3	8 inc_4 7	14 dec_2 11
3 $zero_3$ 4	9 $zero_5$ 10	15 $output_3$ 17
4 $test_{1,3}$ 6,5	10 $test_{1,2}$ 15,11	16 $output_4$ 17
5 inc_3 4	11 $test_{1,5}$ 15,12	17 stop
6 $zero_4$ 7	12 $test_{2,5}$ 16,13	

In modo analogo si ottiene la macchina a registri che calcola il massimo.

Esercizio 11 *Scrivere un programma a registri che, presi in input due numeri interi non negativi posti rispettivamente nei registri 1 e 2, scambia il contenuto dei due registri.*

Soluzione:

1 $input_1$ 2	7 $test_{1,2}$ 9,8
2 $input_2$ 3	8 inc_1 7
3 $zero_3$ 4	9 $zero_2$ 10
4 $test_{1,3}$ 6,5	10 $test_{2,3}$ 11,12
5 inc_3 4	11 inc_2 10
6 $zero_1$ 7	12 stop

1.3 Linguaggi e Automi (Cap 7)

Nella risoluzione degli esercizi si suppongono le seguenti ipotesi (se non specificato diversamente).

Automi

- le transizioni non indicate conducono a stati pozzo non terminali;
- si accettano transizioni non deterministiche, cioè tali che da uno stato posso andare a due o più diversi stati leggendo uno stesso simbolo. Diremo che un automa nondeterministico M accetta (o riconosce) una stringa α se $\exists$ almeno una sequenza di transizioni che conducono $M : \alpha$ in uno stato finale, dove con $M : \alpha$ si indica l'automa M al quale è passata α come stringa di input.
- si accettano λ-transizioni, ovvero transizioni da uno stato all'altro che non ricevono nessun simbolo in input.

MdT

- il carattere B è il simbolo speciale "blank" o "bianco";
- si suppone che la testina della MdT all'inizio si trovi nello stato q_0 e sia posizionata sul primo carattere a sinistra diverso da B;

Esercizio 12 *Si sciva il programma di un MdT definita sull'alfabeto* $A = \{0, 1, B\}$ *la quale si arresta dopo aver letto il carattere 1 per quattro volte consecutive, o quando arriva alla fine della stringa.*

Soluzione:

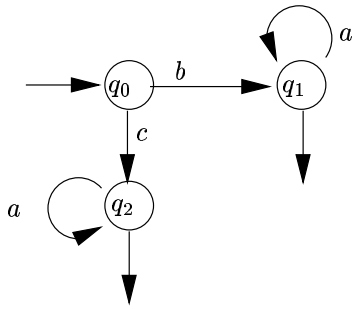
$(q_0, 0, 0, q_0, R)$
$(q_0, 1, 1, q_1, R)$
(q_0, B, B, q_4, R)
$(q_1, 0, 0, q_0, R)$
$(q_1, 1, 1, q_2, R)$
$(q_2, 0, 0, q_0, R)$
$(q_3, 0, 0, q_0, R)$
$(q_3, 0, 0, q_0, R)$
$(q_3, 1, 1, q_4, R)$

Quando la macchina si ferma nello stato q_4 significa che è arrivata alla fine della stringa senza trovare quattro 1 consecutivi, metre quando si ferma nello stato q_5 significa che ha trovato la prima occorrenza di quattro 1 consecutivi.

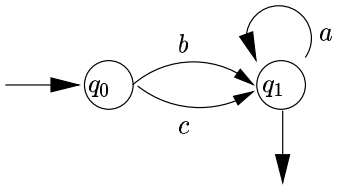
Esercizio 13 *Dare l'automa a stati finiti che produce il seguente insieme di stringhe sull'alfabeto* $A = \{a, b, c\}$*:*

$$\begin{array}{cccccc} b & ba & baa & baaa & \dots \\ c & ca & caa & caaa & \dots \end{array}$$

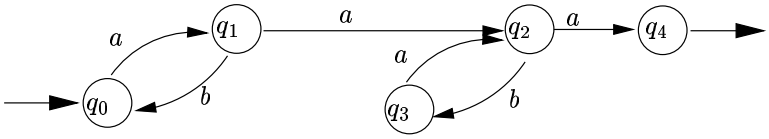
Soluzione:



tale automa può essere migliorato:



Esercizio 14 *Dare l'espressione regolare che produce l'insieme delle stringhe sull'alfabeto* $A = \{a, b\}$ *riconosciute dall'automa a stati finiti avente il diagramma seguente:*



Soluzione:

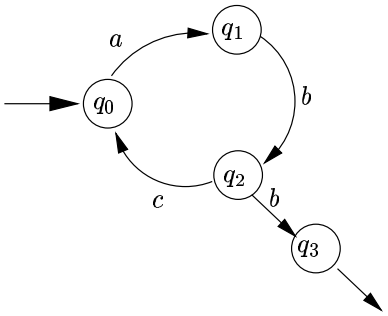
$$a(ab)^*a(ba)^*a$$

Esercizio 15 *Si scriva il programma di una MdT definita sull'alfabeto* $A = \{0, 1, 2\}$ *la quale, letto il primo carattere disponibile sul nastro:*

- *passa a uno stato terminale* q_F *se il carattere letto è uguale a 0;*
- *cicla indefinitamente restando nello stato* q_N *se il carattere letto è uguale a 1 oppure a 2.*

Soluzione:
$(q_0, 0, 0, q_F, R)$
$(q_0, 1, 1, q_N, R)$
$(q_0, 2, 2, q_N, R)$
$(q_N, 0, 0, q_N, R)$
$(q_N, 1, 1, q_N, R)$
$(q_N, 2, 2, q_N, R)$
(q_N, B, B, q_N, R)

Esercizio 16 *Dare l'espressione regolare che produce l'insieme delle stringhe sull'alfabeto* $A = \{a, b, c\}$ *riconosciute dall'automa a stati finiti avente il diagramma seguente:*



Soluzione:

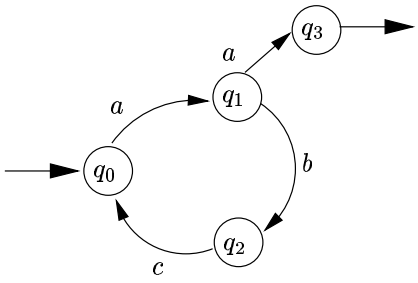
$$(abc)^*abb$$

Esercizio 17 *Definire una MdT sull'alfabeto* $\{0, 1, B\}$ *la quale, partendo da una posizione qualunque di un nastro interamente formato da simboli* B*, produce la sequenza:*

$$\ldots BBB011001BBB\ldots$$

Soluzione:
$(q_0, B, 0, q_1, R)$
$(q_1, B, 1, q_2, R)$
$(q_2, B, 1, q_3, R)$
$(q_3, B, 0, q_4, R)$
$(q_4, B, 0, q_5, R)$
$(q_5, B, 1, q_6, R)$

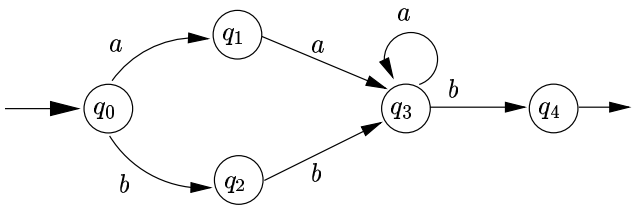
Esercizio 18 *Dare l'espressione regolare che produce l'insieme delle stringhe sull'alfabeto $A = \{a, b, c\}$ riconosciute dall'automa a stati finiti avente il diagramma seguente:*



Soluzione:

$$(abc)^*aa$$

Esercizio 19 *Dare l'espressione regolare che produce l'insieme delle stringhe sull'alfabeto $A = \{a, b\}$ riconosciute dall'automa a stati finiti avente il diagramma seguente:*



Soluzione:

$$aaa^*b + bba^*b$$

Esercizio 20 *È dato un alfabeto $A = \{0, 1, B\}$ per il quale si conviene che 0 e 1 siano simboli informativi mentre B non lo sia. Definire una MdT che partendo da un nastro contenente una stringa di simboli diversi da B, rimuove il contenuto informativo del nastro a partire dall'inizio della stringa fino ad arrivare al terzultimo simbolo della stringa (incluso).*

Soluzione:

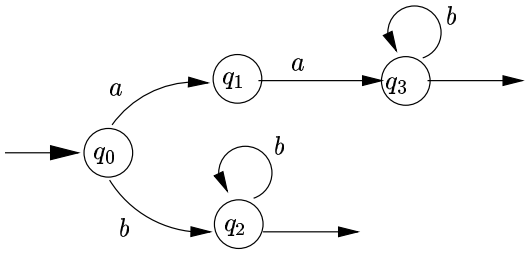
$(q_0, 0, 0, q_0, R)$	$(q_3, 0, 0, q_3, L)$	$(q_5, 1, B, q_5, R)$
$(q_0, 1, 1, q_0, R)$	$(q_3, 1, 1, q_3, L)$	$(q_5, B, 0, q_7, R)$
(q_0, B, B, q_1, L)	(q_3, B, B, q_5, L)	$(q_6, 1, B, q_6, R)$
$(q_1, 1, 1, q_2, L)$	$(q_4, 0, 0, q_4, L)$	$(q_6, 0, B, q_6, R)$
$(q_1, 0, 0, q_2, L)$	$(q_4, 1, 1, q_4, L)$	$(q_6, B, 1, q_7, R)$
$(q_2, 0, B, q_3, L)$	(q_4, B, B, q_6, L)	
$(q_2, 1, B, q_4, L)$	$(q_5, 0, B, q_5, R)$	

Esercizio 21 *Definire una MdT sui simboli* $0, 1, B$ *che restituisce il contenuto del nastro che precede la seconda occorrenza di due simboli consecutivi uguali a 1.*

Soluzione:

$(q_0, 0, 0, q_0, R)$	$(q_2, 1, 1, q_3, R)$	(q_4, B, B, q_5, R)
$(q_0, 1, 1, q_1, R)$	$(q_3, 0, 0, q_2, R)$	
$(q_1, 0, 0, q_0, R)$	$(q_3, 1, 1, q_4, L)$	
$(q_1, 1, 1, q_2, R)$	$(q_4, 1, B, q_4, R)$	
$(q_2, 0, 0, q_2, R)$	$(q_4, 0, B, q_4, R)$	

Esercizio 22 *Dare l'espressione regolare che produce l'insieme delle stringhe sull'alfabeto* $A = \{a, b\}$ *riconosciute dall'automa a stati finiti avente il diagramma seguente:*

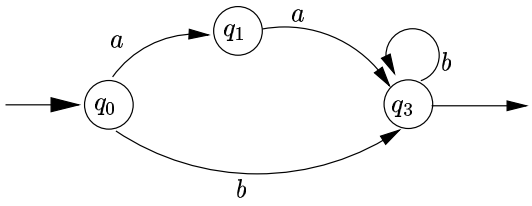


Esiste un automa più compatto (ovvero con un numero minore di stati) in grado di riconoscere lo stesso insieme di stringhe? Se si darne il diagramma corrispondente.

Soluzione:

$$aab^* + bb^*$$

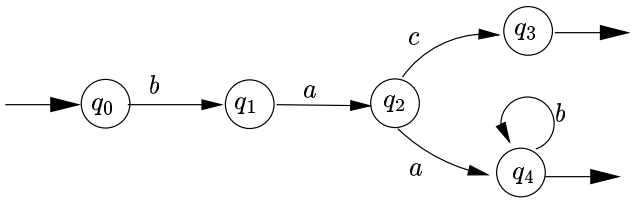
Automa più compatto:



Esercizio 23 *Dare il diagramma (oppure la tabella di transizione) dell'automa a stati finiti sull'alfabeto $A = \{a, b, c\}$ che riconosce stringhe del tipo:*

$$ba(ab^* + c)$$

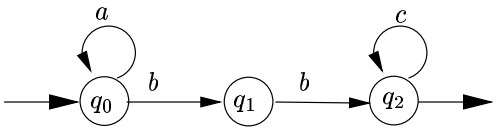
Soluzione:



Esercizio 24 *Dare il diagramma (oppure la tabella di transizione) dell'automa a stati finiti sull'alfabeto $A = \{a, b, c\}$ che riconosce stringhe del tipo:*

$$a^*bbc^*$$

Soluzione:



Esercizio 25 *Definire una MdT che preso in ingresso un numero naturale n, rappresentato come 1^{n+1}, verifica se il numero è pari (0 è pari) o dispari a seconda dello stato in cui essa termina.*

Soluzione:
$(q_0, 1, 1, q_1, R)$
$(q_1, 1, 1, q_0, R)$

Se la macchina termina nello stato q_0 significa che il numero è pari, mentre se termina nello stato q_1 significa che il numero è dispari.

Esercizio 26 *Definire una MdT che preso in ingresso un numero naturale non nullo, rappresentato come* 1^{2n}*, restituisce* $(10)^n$*.*

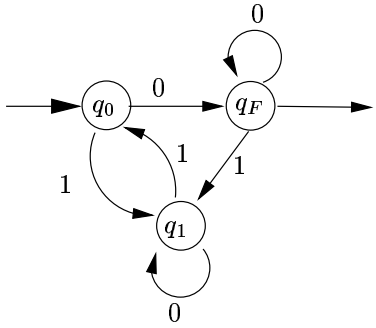
Soluzione:
$(q_0, 1, 1, q_1, R)$
$(q_1, 1, 0, q_0, R)$

Esercizio 27 *Determinare un automa a stati finiti e una MdT che riconoscono le stringhe binarie con un numero pari di 1 che terminano con 0.*

Soluzione:

La MdT è:
$(q_0, 0, 0, q_F, R)$
$(q_0, 1, 1, q_1, R)$
$(q_1, 0, 0, q_1, R)$
$(q_1, 1, 1, q_0, R)$
$(q_F, 0, 0, q_F, R)$
$(q_F, 1, 1, q_1, R)$

L'automa è:



Esercizio 28 *Si descriva il pattern, l'espressione regolare e l'automa a stati finiti che identificano il linguaggio sull'alfabeto* $\{a, b, c\}$*, dato dalle stringhe che cominciano per* ab *e non terminano per* b*.*

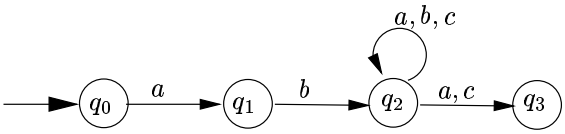
Soluzione:
Il pattern è:

$$abxy$$

dove $x \in \{a, b, c\}^*$ e $y \in \{a, c\}$.
L'espressione regolare è:

$$ab(a + b + c)^*(a + c)$$

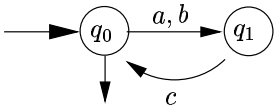
L'automa a stati finiti non deterministico è:



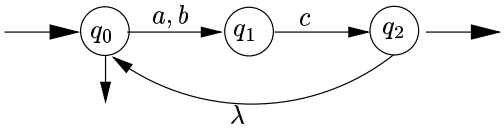
Esercizio 29 *Dare il diagramma dell'automa a stati finiti sull'alfabeto* $\{a, b, c\}$ *che riconosce stringhe del tipo:*

$$[(a + b)c]^*$$

Soluzione:
Automa deterministico:



Automa con λ-transizioni:



1.4 Script

Esercizio 30 *Si prepari uno script di shell il quale, acquisito dallo standard input un numero naturale n non negativo e minore di 20,visualizza sullo standard output la sequenza di valori separati da virgola che, a partire da n, arriva fino a 20:* $n, n+1, \ldots, 20$

Soluzione:

```
clear
echo "inserisci n:"
read n
out=$n
while [ $n -le 19 ]
do
n=`expr $n +1`
out=$out","$n
done
echo $out
```

Esercizio 31 *Si prepari uno script di shell il quale, acquisiti n numeri naturali non negativi* $a_1, a_2, \ldots, a_n$ *dallo standard input, visualizza sullo standard output la loro media geometrica M, definita come:*

$$M = \frac{1}{n}\prod_{i=1}^{n} a_i = \frac{a_1 \cdot a_2 \cdot \cdots \cdot a_n}{n}$$

Soluzione:

```
clear
echo "numero n:"
read n
i=1
m=1
while [ $i -le $n ]
do
echo "inserire numero " $i
read a
m=`expr $m  $a`
i=`expr $i + 1`
done
m=`expr $m / $n`
echo "media geometrica: " $m
```

Esercizio 32 *Si prepari uno script di shell il quale, acquisiti 2 numeri naturali non negativi a_1, a_2 dallo standard input, visualizza sullo standard output il numero $a_2 - a_1$ se $a_2 > a_1$, il numero $a_1 - a_2$ altrimenti.*

Soluzione:

```
clear
echo "Inserire il primo numero:"
read a1
echo "Inserire il secondo numero:"
read a2
if [ $a2 -gt $a1 ]
then
out=`expr $a2 - a1$`
else
out=`expr $a1 - a2$`
fi
echo "Risultato: "$out
```

Esercizio 33 *Si prepari uno script di shell il quale, acquisito un numero naturale non negativo n dallo standard input, produce in seuenza sullo standard output la lista di numeri naturali che parte dal valore $n - 8$ e arriva a $n + 8$,escludendo in questa i valori negativi.*

Soluzione:

```
clear
echo "Inserire il valore di n:"
read n
i=`expr $n - 8`
if [ $i -lt 0 ]
then
i=0
fi
m=`expr $n + 8`
out=" "
while [ $i -le $m ]
do
out=$out$i" "
i=`expr $i + 1`
done
echo $out
```

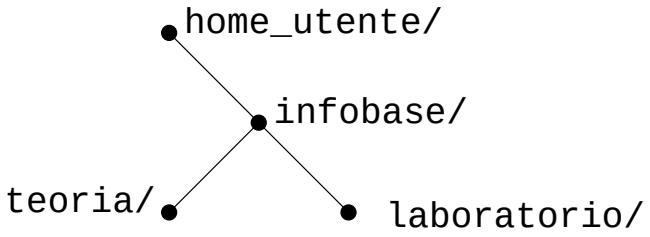
Esercizio 34 *Si prepari uno script di shell il quale, acquisito un numero naturale non negativo n dallo standard input, produce sullo standard output tutti i divisori dispari di n.*

Soluzione:

```
clear
echo "Inserire n:"
read n
i=1
out=""
while [ $i -le $n ]
do
resto=`expr $n % $i`
pari=`expr $i % 2`
if [ $resto -eq 0] && [ $pari -eq 1]
then
out=$out$i' '
fi
i=`expr $i + 1`
done
echo "Risultato: "$out
```

1.5 Laboratorio

Esercizio 35 *Si diano i comandi che a partire dalla propria home directory (denominata home_utente) creano il seguente albero di directory:*



Soluzione:

```
cd ~
mkdir infobase
```

cd infobase
mkdir laboratorio
mkdir teoria

Esercizio 36 *Si assumendo che un ipotetico file prova.txt sia contenuto nella directory* laboratorio *nella struttura di directory specificata all'esercizio precedente. Si impartiscano due comandi che:*

1. *copia nella directory* teoria *il file prova.txt.*
2. *sposta nella directory* teoria *il prova.txt, togliendolo dalla directory* laboratorio.

Soluzione:

cd ~

1. cp infobase/laboratorio/prova.txt infobase/teoria/prova.txt
2. mv infobase/laboratorio/prova.txt infobase/teoria/prova.txt

Esercizio 37 *Si tolgano i permessi di esecuzione e scrittura agli utenti* group *ed* others *del file prova.txt. Successivamente si dia lo stesso comando con i permessi esplicitati in forma numerica.*

Soluzione:

chmod go-xw prova.txt

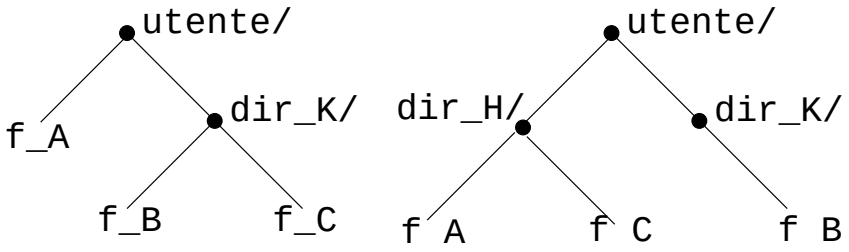
che in forma numerica equivale ad impartire il comando:

chmod 744 prova.txt

Esercizio 38 *Si supponga che la home dell'utente contenga i file* `f_A`, `f_B` *e* `f_C` *e la cartella* `dir_K` *come nella struttura 1 specificata a sinistra nella figura. Si dia la sequenza di comandi necessari a trasformare la struttura della home dalla configurazione 1 alla configurazione 2, illustrata a destra nella stessa figura.*

Soluzione:

mkdir dir_H
mv f_A dir_H
mv dir_K/f_C dir_H



Esercizio 39 *È dato il file* `clienti.txt` *seguente, ciascuna riga del quale contiene rispettivamente il cognome e nome del cliente, il luogo di residenza e il prodotto acquistato:*

Rossi Mario	*Vicenza*	*Scarpe*
Risi Franco	*Roma*	*Giacca*
Bianchi Paolo	*Roma*	*Ombrello*
Stocca Teresa	*Palermo*	*Calze*
De Paoli Augusto	*Roma*	*Bottoni*
Osio Sandra	*Aosta*	*Cappotto*

Scrivere un'istruzione di shell, la quale presenti in ordine alfabetico sullo standard output *i clienti residenti a Roma.*

Soluzione:

```
egrep 'Roma' clienti.txt|sort
```

Nel caso avessimo voluto esplicitare il buffer implicito nell'operazione di pipe "|" avremmo dovuto utilizzare la seguente sequenza di comandi:

```
egrep 'Roma' clienti.txt>buffer; sort <buffer; rm buffer
```

1.6 Macchina di Turing universale

A complemento dell'eserciziario si propone uno script di bash il quale implementa una macchina di Touring universale. Lo script prende in ingresso una macchina di Turing M specificata in un file testuale *m*acchina, il nastro iniziale contenuto nel file di testo *n*astro e lo stato iniziale sotto forma di stringa (es. q00) e restituisce le computazioni della macchina di Turing M sul nastro specificato in ingresso. L'esecuzione dello script avviene per mezzo del comando:

./mdt macchina nastro q00

Ecco il codice che realizza lo script che implementa la macchina universale:

```
#!/bin/bash

#si suppone che inizialmente la testina sia all'inizio del nastro
#e inoltre si ricordi che le istruzioni sono del tipo:
#q00 1 1 q00 R

#il nastro viene preso da un file passato come secondo input
nastro=‘cat $2‘

#il nastro viene preso da un file passato come secondo input
#il nastro viene preso da un file passato come secondo input
stato=$3

#testina:un intero maggiore o uguale a 0 che indica la posizione
#corrente della testina sul nastro.
testina=0

#nextins: la prossima istruzione applicabile in base al simbolo letto
#dal nastro e dallo stato corrente
nextins=""

#spost: il prossimo movimento della testina
spost=""

#symb: il simbolo che viene scritto sul nastro nella operazione in
#corso
symb=""

i=0
while [ 1 ]
do

#la variabile i contiene il progressivo del passo corrente
#per stampare il numero di passo eseguito assieme al nastro
let i=$i+1

#curr: il simbolo corrente sul nastro: ossia la sottostringa
```

```
#di lunghezza 1 della stringa nastro iniziante alla
#posizione $testina
curr=${nastro:$testina:1}

#nextins: la prossima istruzione applicabile viene
#calcolata cercando nella MdT le righe
#(la riga, la MdT \‘e deterministica!!!)
#che specificano l’azione nel caso
#la macchina sia nello stato corrente($stato)
#e legga il simbolo corrente ($curr)
nextins=‘egrep $stato’ ’$curr $1‘

if [ "$nextins" = "" ]
then
exit   #finito: nessuna regola applicabile: esci
else

#il simbolo da scrivere sul nastro \‘e
#il terzo elemento di ogni riga del
#file di input
symb=‘echo $nextins|awk ’ { print $3 }’‘

#sostituisce nel nastro
#il simbolo corrente con
#il nuovo simbolo.
nastro=${nastro:0:$testina}$symb${nastro:$testina+1}

#il prossimo stato \‘e il quarto elemento
#del file di input
stato=‘echo $nextins|awk ’{ print $4 }’‘

#lo spostamento \‘e il quinto elemento
#del file di input
spost=‘echo $nextins|awk ’{ print $5 }’‘

echo "Passo "$i": "$nastro   #stampa il valore del nastro al passo i.
if [ $spost = ’R’ ]
then
let testina=$testina+1
else
```

```
let testina=$testina-1
fi
#NB: il simbolo B \‘e il blank
if [ $testina -ge ${#nastro} ] || [ $testina -lt 0 ]
#la testina supera le dimensioni del nastro?
then
if [ $testina -lt 0 ]
then
testina=0
nastro='B'$nastro #estendo a sinistra il nastro con un blank
else
nastro=$nastro'B' #estendo a destra il nastro con un blank
fi
fi
fi
done
```

si noti l'utilizzo del linguaggio awk, per maggiori informazioni in merito si rimanda al manuale di Linux, consultabile col comando *m*an awk. Per comprendere lo script è sufficiente sapere che un comando awk ha la seguente forma:

*a*wk ' pattern azione istruzioni ' nomefile

dove *p*attern è facoltativo e può indicare una espressione regolare che indica su quali righe del file *n*omefile applicare il corpo del comando awk e cioè la parte *azione istruzioni*. Nello script l'unica azione utilizzata è *p*rint $i dove $i indica un valore numerico. L'uscita di un tale comando è di stampare a video il campo i-esimo di una stringa testuale in cui i vari campi sono separati da uno spazio.
Un esempio di specifica di macchina di Turing può essere il seguente file testuale che abbiamo chiamato *m*acchina:

```
q00 1 1 q00 R
q00 0 0 q01 L
q01 1 0 q02 R
```

mentre un verosimile nastro di input potrebbe contenere la stringa “1111110”. Se invochiamo lo script con il comando

./mdt m1.txt nastro q00

otteniamo come output della computazione il valore del nastro dopo ogni singolo passo di calcolo:

```
Passo 1: 1111110
Passo 2: 1111110
Passo 3: 1111110
Passo 4: 1111110
Passo 5: 1111110
Passo 6: 1111110
Passo 7: 1111110
Passo 8: 1111100
```