Progetto di Linguaggi e Compilatori 1 – Parte 2
A.A. 2015/16

Gruppo 14

Marco Bucchiarone
Emanuele Tonetti
Francesco Zilli

Esercizio 1

(a)

Dato un testo formattato come

cognome nome/nomi data(gg/mm/aa) matricola altro-testo

con i campi separati da un numero arbitrario di spazi, le espressioni regolari, nella sintassi di flex, componenti l'espressione regolare e_{in} per eseguire la riformattazione del testo sono:

cognome	`([a-zA-Z\-\']+)`$_1$
nome/nomi	`(([a-zA-Z\-]){+([ ]+[a-zA-Z\-]+)*)`$_2$
gg	`((0[1-9])|([12][0-9])|3[01])`$_3$
mm	`((0[1-9])|(1[0-2]))`$_4$
aa	`([0-9]{2})`$_5$
matricola	`([0-9]{6})`$_6$
separatore	`("/")`
spazi	`([␣]+)`
altro-testo	`(.)`

dove, per semplicità di notazione, sono state numerate solo le parentesi contenenti le *regexp* facenti il match dei campi che si vuole siano presenti nell'espressione e_{out}.

Quindi la *regexp* e_{in} assumerà forma

$$e_{in} = \texttt{\{cognome\}\{spazi\}\{nome/nomi\}\{spazi\}\{gg\}\{separatore\}\{mm\}\{separatore\}\{aa\}\{spazi\}\{matricola\}\{spazi\}\{altro-testo\}}.$$

Volendo e_{out} della forma

matricola nome/nomi cognome data(aaaa-mm-gg)

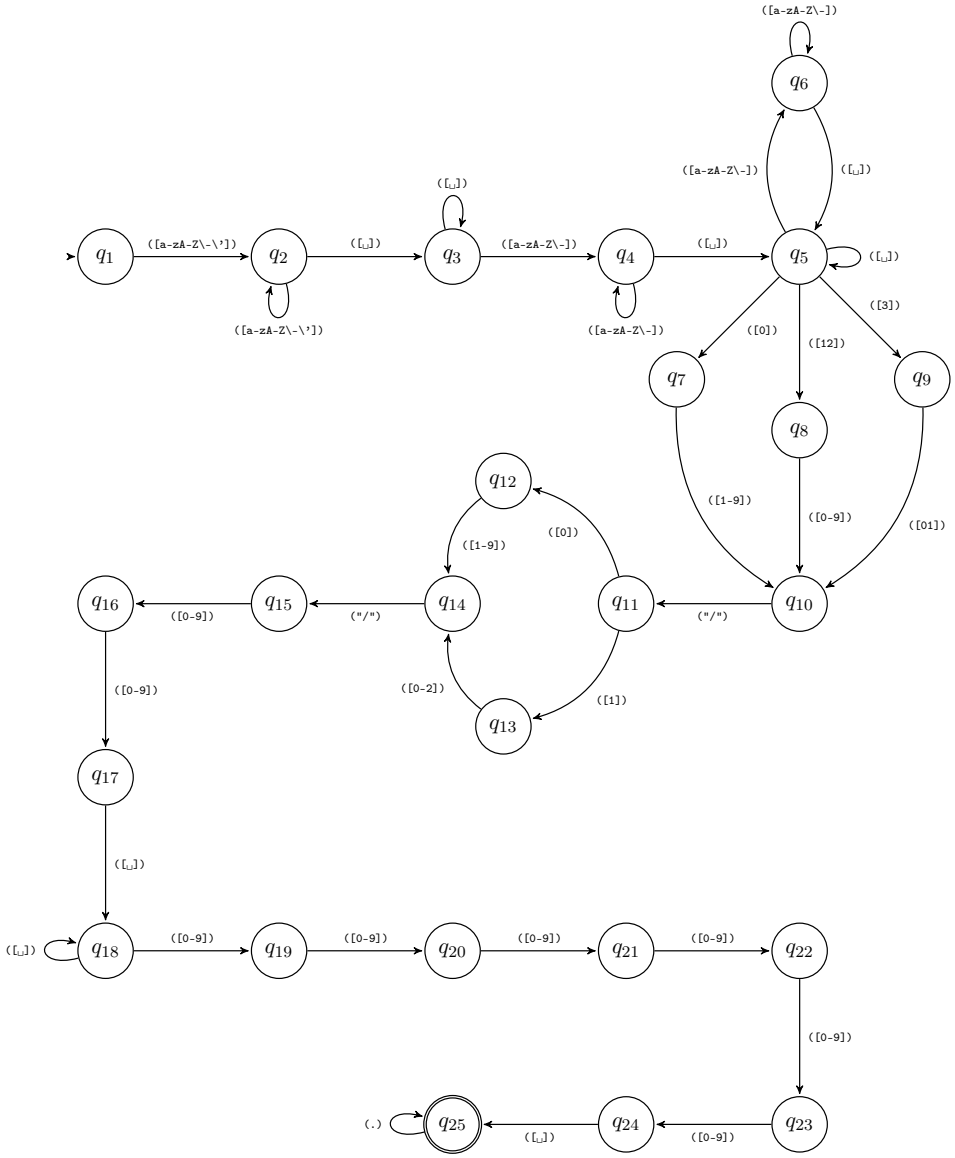
con i campi separati con un tabulatore ed assumendo che, tutte le date successive al 2000, non abbiano singole cifre non precedute da 0, si avrà

$$e_{out} = \texttt{\textbackslash 6\textbackslash t\textbackslash 2\textbackslash t\textbackslash 1\textbackslash t20\textbackslash 5-\textbackslash 4-\textbackslash 3}$$

dove `\t` indica il carattere di tabulazione.

(b)

Preso l'alfabeto Σ contenente i caratteri ASCII, il DFA minimo per e_{in} è:



dove, per chiarezza illustrativa, si sono indicati sugli archi il range di caratteri che, ricevuti dallo stato q_i causano la transizione allo stato q_j. Per la stessa ragione è stata omessa la rappresentazione dello stato "pozzo" a cui puntano tutti gli stati qualora ricevano un carattere non accettato dalle transizioni esplicitate.

Esercizio 2

```
init:
            ENT 2     -- variabile u e variabile temporanea
                      -- necessaria all'impl. del for
            LDA 0 9   -- carico la variabile temporanea
            MST 1
            LDA 1 4   -- IMAX
            IND
            LDA 0 5      -- j di init
            IND
            CUP 2 min   -- min(imax,j)
            STO         -- temp := min(IMAX, j)
            LDA 0 8        -- carico u
            LDC int 0
            STO         -- u := 0
            UJP guard-init  -- while u <= temp do S; u += 1;
body-init:
            MST 1
            LDA 0 8
            IND         -- u
            LDA 0 4
            IND         -- i
            MUL int     -- u * i
            LDA 0 7
            IND         --
            CUP 2 p        -- p(u*i , z)
            LDA 0 6     -- h : history
            IND         -- history := ^harray
            LDA 0 4     -- index i
            IND
            IXA 10      -- h[i][] pointer
            LDA 0 8     -- index u
            IND
            IXA 1       -- h[i][u] pointer
            LDA 1 8        -- mainharray(offset 8)
            IND
            LDA 0 8     -- index u
            IND
            IXA 10      -- [u] pointer
            LDA 0 5     -- index j
            IND
            IXA 1       -- [u][j] pointer
```

```
                IND             -- mainharray[u,j]
                STO
                LDA 0 8
                LDA 0 8         -- duplico la u
                IND
                LDC int 1
                SUM int
                STO
guard-init:
                LDA 0 8         -- u
                IND
                LDA 0 9         -- temp
                IND
                GTR
                FJP body-init
                RETP
f:
                ENT 0
                ...
                RETF
p:
                ENT 0
                LDA 0 5
                IND
                MST 1           -- pre-chiamata al primo f
                MST 1           -- pre-chiamata al secondo f
                LDA 0 4
                IND
                FLT             -- f() si aspetta real
                CUP 1 f
                CUP 1 f
                STO             -- y:= f(f(x))
                UJP guard-p
body-p:
                MST 1           -- preparo per chiamata ricorsiva di p(y,y)
                LDA 0 5
                IND
                IND
                LDA 0 5
                IND
                CUP 2 p
guard-p:
                LDA 0 5
                IND
```

```
                IND
                LDC int 0
                NEQ
                LDA 0 4
                IND
                LDA 1 6
                IND
                LEQ
                OR
                FJP body-p
                RETP
min:
                ENT 0
                ...
                RETF
f:                      --interna ad alt
                ENT 0
                LDA 0 0
                LDC real 1
                LDC real 1
                LDA 0 4
                IND
                DIV real
                SUM real
                STO
                RETF
alt:
                ENT 0
                LDA 0 4
                IND             -- carico i
                ODD
                FJP else
then:
                LDA 0 0
                MST 1           -- preparo chiamata ricorsiva alt
                LDA 0 4
                IND
                LDC int 1
                SUB int
                MST 0           -- preparo chiamata alla funzione interna f;
                LDA 0 5
                IND
                CUP 1 f
                CUP 2 alt
```

```
                STO
                RETF
else:
                LDA 0 0
                MST 1
                LDA 0 4
                IND
                LDC int 1
                SUB int
                LDA 0 5
                IND
                CUP 2 alt
                STO
                RETF
main:
                ...
                MST 0
                LDA 0 6         -- target
                IND
                LDA 0 7         -- aim
                CUP 2 p
                MST 0
                LDC int 20
                LDC int 30
                LDA 0 8         -- mainharray
                LDA 0 6
                CUP 4 init
                ...
```

Esercizio 3

compilato con bison 3.0.4 flex 2.6.0