PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2013/14

Prova scritta del 3 febbraio 2014

SOLUZIONI PROPOSTE

ESERCIZIO 1
Si supponga di avere una grammatica $G = \langle \Lambda, V, S, P \rangle$, con $\Lambda = \{a, b, c\}$, che genera il linguaggio

$$\mathcal{L}_G = \{a^n b^n c^n \mid n > 0\}$$

Definire una grammatica

$$G' = \langle \Lambda, V \cup V', S', P \cup P' \rangle$$

con $V' \cap V = \{\}$ che genera il linguaggio

$$\mathcal{L}_{G'} = \{a^m b^n c^n \mid m > n > 0\} \cup \{a^n b^n c^m \mid m > n > 0\}$$

Soluzione

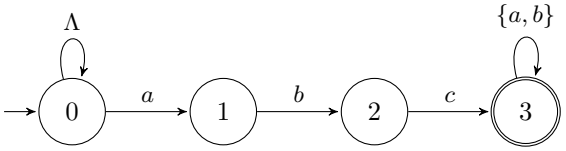
- $V' = \{S', A, C\}$ con $V' \cap V = \{\}$.
- P' è l'insieme che contiene le seguenti sei produzioni.
  $S' \ ::= \ AS \mid SC$
  $A \ ::= \ a \mid aA$
  $C \ ::= \ c \mid cC$

ESERCIZIO 2
Definire un automa **deterministico** che riconosce il seguente linguaggio sull'alfabeto $\Lambda = \{a, b, c\}$

$$\mathcal{L} = \left\{\alpha abc\beta \mid \alpha \in \Lambda^*, \beta \in \{a, b\}^*\right\}$$

Soluzione
Proponiamo una soluzione che consiste nel definire un automa **non** deterministico che riconosce il linguaggio dato, per poi trasformarlo in un automa deterministico equivalente utilizzando la tecnica della costruzione dei sottinsiemi.

Automa non deterministico

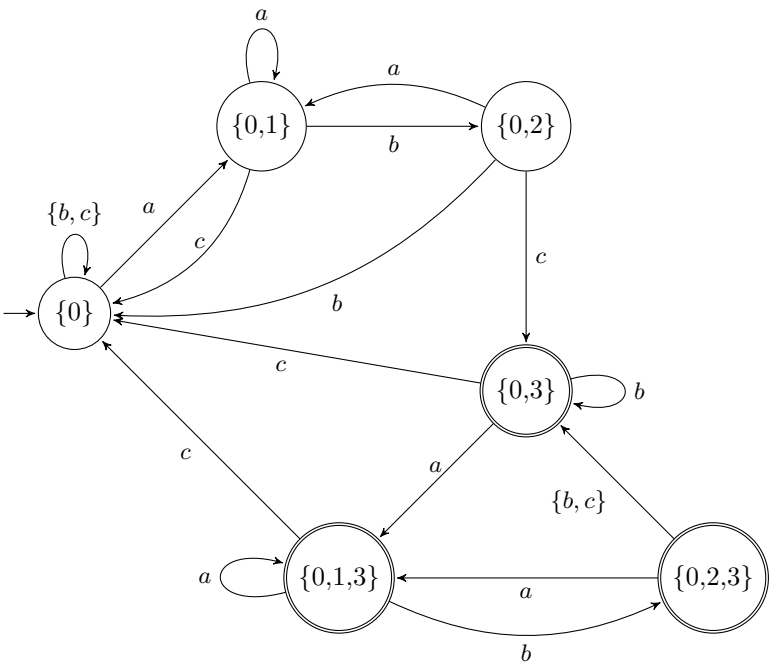


Applichiamo la tecnica della trasformazione dei sottinsiemi (gli stati finali sono asteriscati).

stato	a	b	c
$\{0\}$	$\{0,1\}$	$\{0\}$	$\{0\}$
$\{0,1\}$	$\{0,1\}$	$\{0,2\}$	$\{0\}$
$\{0,2\}$	$\{0,1\}$	$\{0\}$	$\{0,3\}$
* $\{0,3\}$	$\{0,1,3\}$	$\{0,3\}$	$\{0\}$
* $\{0,1,3\}$	$\{0,1,3\}$	$\{0,2,3\}$	$\{0\}$
* $\{0,2,3\}$	$\{0,1,3\}$	$\{0,3\}$	$\{0,3\}$

Di seguito la rappresentazione grafica dell'automa deterministico ottenuto.

Automa deterministico



ESERCIZIO 3
Dato il tipo degli alberi binari

```
type 'a btree = Void  | Node of 'a * 'a btree * 'a btree
```

si definisca in `CAML` una funzione `foo` con tipo

```
foo : 'a btree -> int * int
```

in modo che `(foo bt)` sia la coppia costituita dalla profondità dell'albero e dal numero di nodi dell'albero.

Soluzione

```
let rec foo bt = match bt with
    Void -> (0,0)   |
    Node(_,lt, rt) -> let (p1,n1) = foo lt in
                      let (p2, n2) = foo rt in
                      (if p1>p2 then p1+1 else p2+1), n1+n2+1;;
```

ESERCIZIO 4
Si definisca in `C` una funzione

```
int check (int a[], int dim)
```

che restituisce il valore di verità della seguente formula:

$$\exists j \in [1, dim-1).\ \#\{i \mid i \in [0, j) \wedge a[j] < a[i]\} = \#\{i \mid i \in [j+1, dim) \wedge a[j] > a[i]\}$$

N.B. : si ricorda che, dato un insieme A, la notazione $\#A$ indica la cardinalità (ovvero il numero di elementi) di A.

Soluzione

```
int check (int a[], int dim)
{
   int trovato, j, i, conta_prec, conta_succ;
   j=1;
   while (j<dim-1 && !trovato)
   {
     conta_prec=0;
     for (i=0; i<j; i++)
       if (a[j]<a[i]) conta_prec++;
     conta_succ=0;
     for (i=j+1; i<dim; i++)
       if (a[j]>a[i]) conta_succ++;
     trovato = conta_prec==conta_succ;
     j++;
   }
   return trovato;
}
```

ESERCIZIO 5
Senza utilizzare ricorsione esplicita, definire in CAML una funzione

```
foo : 'a list -> 'a * 'a
```

in modo che `(foo lis)` restituisca la coppia `(p, m)` dove `p` è il primo elemento della lista e `m` è il il valore massimo nella lista. La funzione non è definita su liste vuote.

Soluzione

```
let foo lis = let f x (p,m) = p, (if x>m then x else m) in
              match lis with
                x::xs -> foldr f (x,x) xs;;
```

ESERCIZIO 6
Date le seguenti definizioni:

```
struct el {int info; struct el *next;};
typedef struct el ElementoDiLista;
typedef ElementoDiLista *ListaDiElementi;
```

scrivere in C una procedura che, dati in ingresso attraverso opportuni parametri una lista di interi e due interi x e y, inserisce nella lista un elemento che contiene x come predecessore dell'ultimo elemento che contiene y. Se nessun elemento della lista contiene y, la procedura inserisce un elemento che contiene x in fondo alla lista.

Soluzione

```
void ins (ListaDiELementi *l, int x, int y)
{
   ListaDiElementi new= malloc(sizeof(ElementoDiLista));
   new -> info = x;
   ListaDiElementi prec=NULL, corr=*l, prec_ultimo_y=NULL;
   int trovato_y=0;

   while (corr != NULL)
      {
         if (corr->info==y) { trovato_y = 1; prec_ultimo_y=prec;}
         prec=corr;
         corr=corr->next;
      }
   if (prec_ultimo_y == NULL)
      if (trovato_y) {new_next=*l; *l=new;}
      else {new_next=NULL; prec->next=new;}
   else
      {
        new->next= prec_ultimo_y->next;
        prec_ultimo_y->next = new;
      }
}
```