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

Prova scritta del 9 gennaio 2014

SOLUZIONI PROPOSTE

ESERCIZIO 1
Definire una grammatica libera che genera il seguente linguaggio sull'alfabeto $\Lambda = \{a, b\}$

$$\mathcal{L} = \{a^n \alpha b^{n+1} \mid \alpha \in \Lambda^*, n > 0\}$$

Soluzione

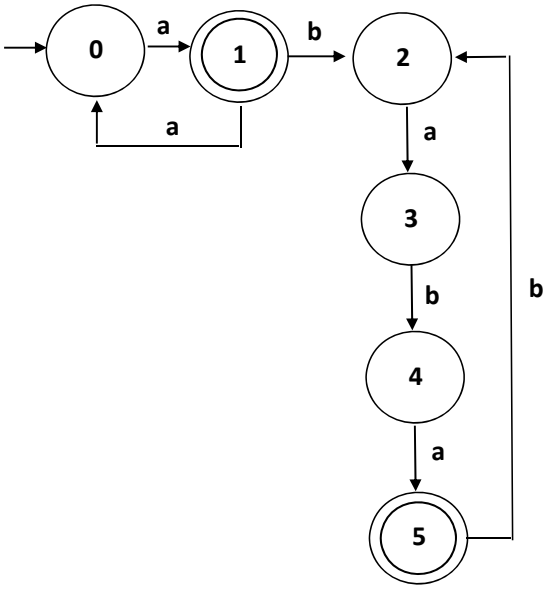
```
S ::= aSb | aAbb | abb
A ::= a | b | aA | bA
```

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

$$\mathcal{L} = \{a^{2n+1}(ba)^{2k} \mid n, k \geq 0\}$$

N.B.: la notazione $(ba)^x$ indica la stringa ba ripetuta x volte.

Soluzione



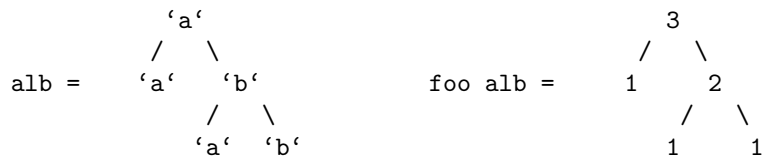
ESERCIZIO 3
Dato il tipo degli alberi binari

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

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

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

in modo che `(foo bt)` sia un albero di interi con la stessa struttura di `bt` in cui ogni nodo contiene il numero di occorrenze del corrispondente nodo in `bt` nel sottolabero di cui è radice in `bt`. Ad esempio:



Soluzione

```
let rec foo bt =
       let rec contaocc bt x = match bt with
           Void -> 0 |
           Node(y, lbt, rbt) -> (if y=x then 1 else 0) + (conta occ lbt x) + (conta occ rbt x)
       in
       match bt with
         Void -> Void |
         Node(x, lbt, rbt) -> Node (1+ (contaocc lbt x) + (contaocc rbt x),  foo lbt, foo rbt);
```

ESERCIZIO 4
Si definisca in `C` una funzione

`int check (int a[], int dima, int b[], int dimb)`

che restituisce il valore di verità della seguente formula:

$$\exists k \in [0, dima].\ \big((\forall i \in [0,k).\ \exists j \in [0, dimb).a[i] = b[j])\ \wedge\ (\forall i \in [k, dima).\ \forall j \in [0, dimb).a[i] \neq b[j])\big)$$

Soluzione

```
int member (int a[], int dim, int x)
  /* restituisce 1 se x occorre in a, 0 altrimenti */
  {
     int trovato=0;
     int i=0;
     while (i < dim && !trovato)
       if (a[i]==x) trovato = 1; else i++;
     retrun trovato;
  }

int check (int a[], int dima, int b[], int dimb)
{
  int ia;
  int trovato = 0;
  while (ia<dima && !trovato)
    {
      trovato = !member(b, dimb, a[i]);
      ia++;
    }

  trovato = 0;
  while (ia<dima && !trovato)
    {
      trovato = member(b, dimb, a[ia]);
      ia++;
    }
  return (!trovato);
}
```

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

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

in modo che `(foo lis x)` restituisca la coppia `(y, n)` dove `n` è il numero di elementi di `lis` uguali a `x` e `y` è l'ultimo elemento di `lis` diverso da `x`, se tale elemento esiste, `x` stesso altrimenti.

Soluzione

```
let foo lis x = let f z (y,n) = if z=x then (y,n+1) else ((if y=x then z else y), n)
                in
                    foldr f (x,0) lis ;;
```

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 ed un intero `x`, elimina dalla lista i primi `x` elementi maggiori di `0` (se la lista contiene meno di `x` elementi maggiori di `0` la procedura li deve eliminare tutti).

```
void canc_primi (ListaDiElementi *l, int x)
{
  if (*l != NULL)
    {
      ListaDiElementi prec=NULL, corr=*l;
      int ancora = (x>0);

      while (ancora)
       {
         if (corr->info > 0)
           { if (prec==NULL)
                {*l = *l->next; free(corr); corr=*l;}
             else
                {prec->next = corr->next; free(corr); corr=prec->next;}
             x=x-1;
           }
         else
           { prec=corr; corr=corr->next;}
         ancora = (corr != NULL) && (x>0);
       }
    }
 }
```