25 giugno 2014

Titolo nota 24/09/2015

Dato l'alfabeto $\Lambda = \{0, 1, 2\}$ si definisce una grammatica che genera

$$L = \{ \alpha \mid \alpha \in \Lambda^+ \wedge \Sigma\alpha \bmod 2 = 0 \wedge \alpha \bmod 10 = 0 \}$$

dove $\Sigma\alpha$ è la somma di tutte le cifre di α.

$121 \notin L$ $\quad 10 \notin L$ $\quad 2110 \in L$

α deve terminare con 0

Bisogna sommare le cifre di α e la somma deve essere pari

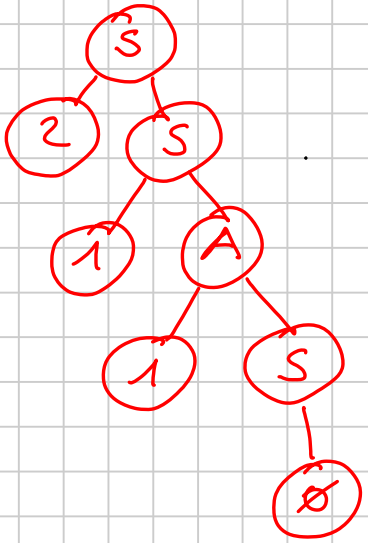
Titolo nota 24/09/2015

$S \rightarrow \emptyset \mid \emptyset S \mid 2S \mid 1A$

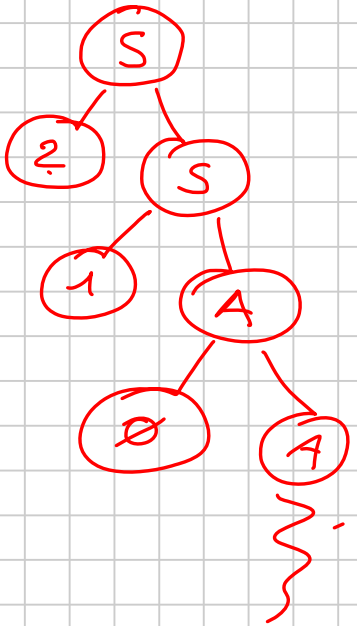
$A \rightarrow \emptyset A \mid 2A \mid 1S$

$\emptyset \in L$

$211\emptyset \in L$



$210 \notin L$

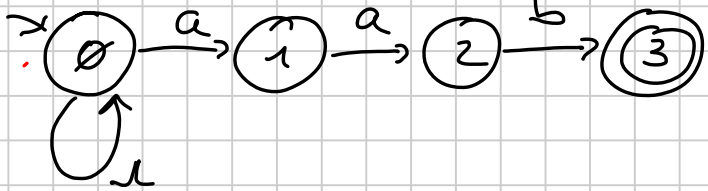


Titolo nota 24/09/2015

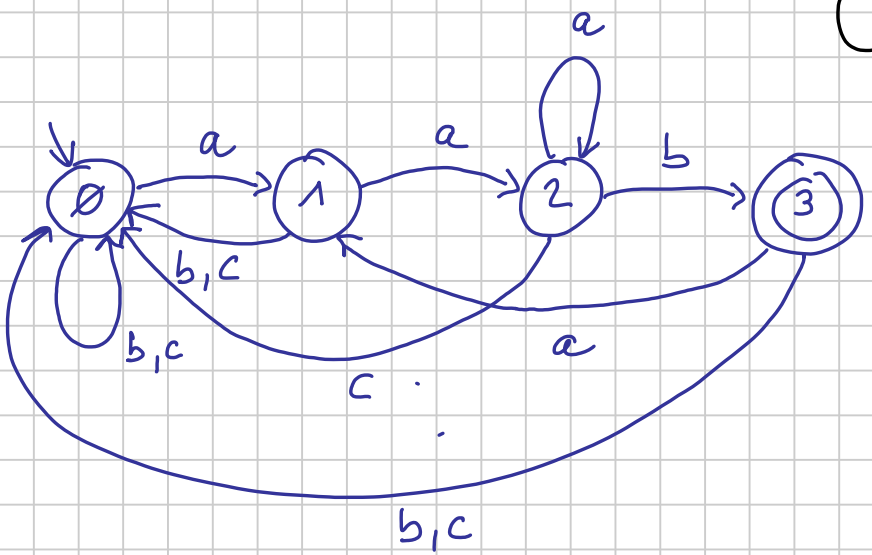
Definire un automa deterministico che riconosce nell'alfabeto $\Lambda = \{a, b, c\}$

$$L = \{ \alpha\, aab \mid \alpha \in \Lambda^* \}$$

$aab \in L$



N. Det.



Titolo nota 24/09/2015

Data la seguente definizione di tipo:

```
type 'a bilist = Tail of 'a * 'a | Cons of 'a * 'a * 'a bilist ;;
```

Si definisca in CAML una funzione

```
check : 'a bilist → bool
```

che restituisca true se tutti gli elementi di tipo 'a dell'argomento sono distinti tra loro.

Suggerimento: definire una funzione ausiliaria analoga alla funzione member su liste.

Valore di tipo bilist

```
check (Cons (5, 6, Tail (5, 3))) = false
```

Titolo nota 24/09/2015

```
let rec member bl n = match bl with
    Tail (x,y) -> n = x or n = y
  | Cons (x, y, bl1) -> n = x or n = y or
                        member bl1 n ;;
```

if n = x then true
else n = y

if n = x or n = y
then true
else member bl1 n ;;

```
  | Cons (x, y, bl1) when n = x or n = y -> true
  | Cons (x, y, bl1) when n <> x & n <> y -> member bl1 n ;;
```

Titolo nota 24/09/2015

```
let rec check bl = match bl with
    Tail (x,y) ->  x <> y
  | Cons (x,y, bl1) -> if x = y then false
                       else if member bl1 x then false
                            else if member bl1 y then false
                                 else check bl1 ;;
```

Titolo nota 24/09/2015

Si definisce in C una funzione

int check (int a[], int dim)

che restituisce il valore di verità della seguente formula

$$\exists j \in (0, dim-1).$$
$$\left(\#\{k \mid k \in [0, j) \wedge a[k] = a[j]\} = 1\right) \wedge$$
$$\left(\#\{k \mid k \in (j, dim) \wedge a[k] = a[j]\} = 1\right)$$

Titolo nota 24/09/2015

```
int check (int a[], int dim)
{ int trovato = 0; int j = 1;
  while (j < dim - 1 && !trovato)
  { int k = 0;
    int conta = 0;
    while (k < j && conta < 2)
    { if (a[k] == a[j]) conta = conta + 1;
      k = k + 1;
    }
    if (conta == 1) { conta = 0;
                      k = j + 1;
                      while (k < dim && conta < 2)
                      { if (a[k] == a[j]) conta = conta + 1;
                        k = k + 1;
                      }
                      if (conta == 1) trovato = 1;
                      else j = j + 1; }
    else j = j + 1;
  }
  return trovato;
}
```

Titolo nota 24/09/2015

```
int conta (int n, int a[], int ini, int fin)
{ int cont = 0;
  int i = ini;
  while (i <= fin && cont < 2)
     { if (a[i] == n) cont = cont + 1;
       i = i + 1;
     }
  return cont;
}
```

Titolo nota 24/09/2015

```
int check (int a[], int dim)
{ int trovato = 0; int j = 1;
  while (j < dim-1 && !trovato)
  { int c = conta(a[j], a, 0, j-1);
    if (c == 1)
      { c = conta(a[j], a, j+1, dim-1);
        if (c==1) trovato = 1;
          else j = j+1;
      }
    else j = j+1;
  }
  return trovato;
}
```

Titolo nota 24/09/2015

Senza utilizzare "ricorsione esplicita", definire una funzione CAML

subst : ('a * 'a) list → bool list

che, data una lista di coppie, restituisca la lista di booleani nella quale il valore true (risp. false) compare in posizione i sse nella stessa posizione della lista originaria compare una coppia di elementi uguali (risp. diversi).

subst [(2,3); (3,3); (4,4); (5,6)] = [false; true; true; false]

Titolo nota 24/09/2015

Utilizzando map

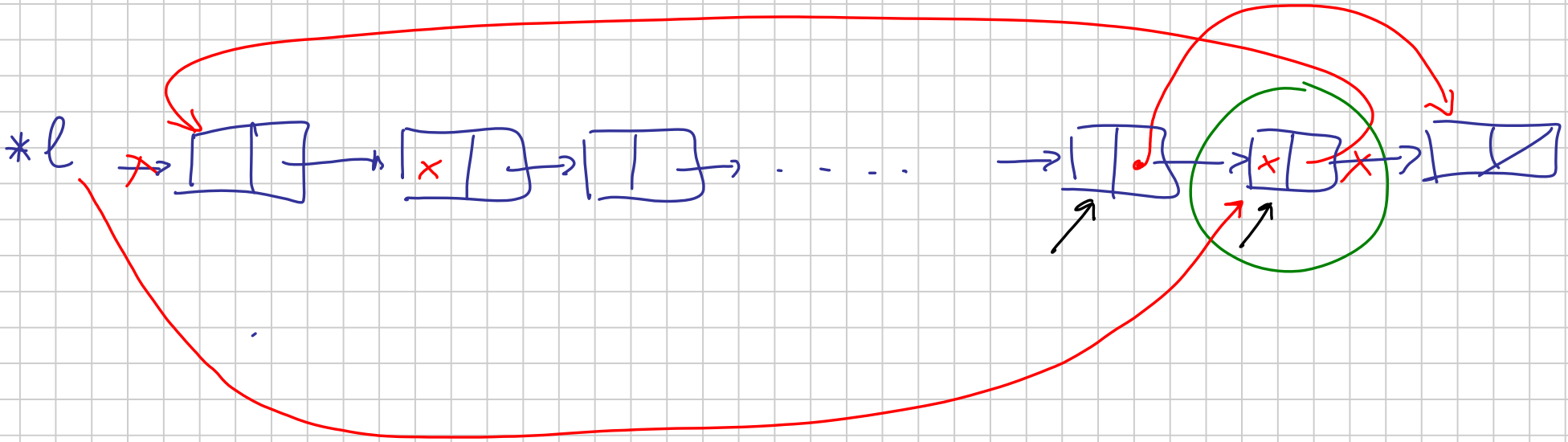
```
let subst l =
    let f (x,y) = x=y
in
    map f l;;
```

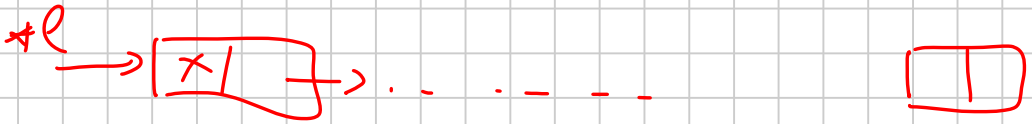
Utilizzando foldr

```
let subst l =
    let f (x,y) z =
        (x=y) :: z
in foldr f [] l;;
```

Titolo nota 24/09/2015

Scrivere in C una procedura che, dati in ingresso, attraverso opportuni parametri, una lista di interi e un intero x, sposta in prima posizione l'ultima occorrenza dell'elemento x.





Titolo nota 24/09/2015

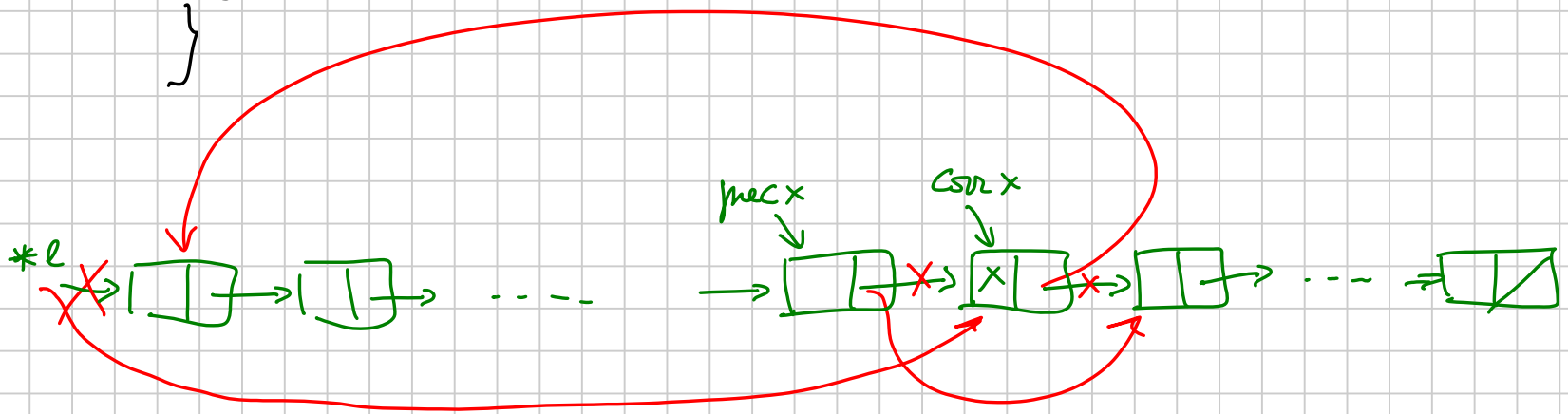
```
void sposte (ListaDiElementi * l, int x)
{ if (*l != NULL)
    if ((*l)->next != NULL)
    { ListaDiElement precx = NULL, corrx = NULL;
      ListaDiElement prec = *l, corr = (*l)->next;
      while (corr != NULL)
        { if (corr->info == x) { precx = prec;
                                 corrx = corr;
                               }

          prec = corr;
          corr = corr->next;
        }
```

Titolo nota 24/09/2015

dopo il comando while

```
if (curx != NULL)
{
    precx -> next = curx -> next;
    curx -> next = *l;
    *l = curx;
}
}
```



mediante foldr

Titolo nota 24/09/2015

pari : 'a list → bool

in modo che pari l sia true sse l è di lunghezza pari.

pari [2;3;4] = false pari [2;3] = true pari [] = true

```
let pari l =
  let f x y = not y

in foldr f true l;;
```