Titolo nota 29/10/2015

(1) Progettare un automa DETERMINISTICO che riconosce il seguente linguaggio sull'alfabeto $\Lambda = \{a, b, c\}$

$L = \{\alpha \mid \alpha \in \Lambda^* \text{ e } \alpha \text{ contiene la sottosequenza } ab \text{ oppure la sottosequenza } ac\}$

$L = \{\alpha ab \beta \mid \alpha, \beta \in \Lambda^*\} \cup \{\alpha ac \beta \mid \alpha, \beta \in \Lambda^*\}$

Definire una grammatica REGOLARE che genera lo stesso linguaggio.

(2) Scrivere una grammatica libera che genera il seguente linguaggio su $\Lambda = \{a, b\}$

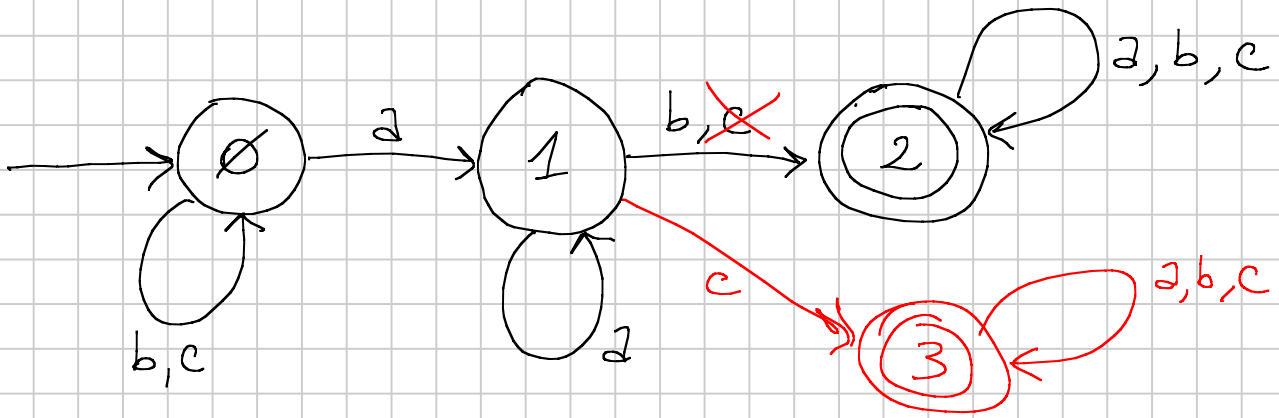
$L = \{b a^n b^m \mid n > m \geq 0\}$

e dimostrare che il linguaggio NON è regolare

(3) Scrivere in C una funzione che restituisce vero (1) se in un dato array a di dimensione $dima$, il valore minimo occorre nell'array esattamente una volta; restituisce falso ($\emptyset$) altrimenti.

(4) Scrivere una funzione C che, dati due array di interi a e b e le loro dimensioni $dima$ e $dimb$ restituisce vero (1) se ogni elemento di a compare anche in b, falso ($\emptyset$) altrimenti.

1



$$\emptyset \rightarrow b\emptyset \mid c\emptyset \mid a1$$

$$1 \rightarrow a1 \mid b2 \mid b \mid c2 \mid c$$

$$2 \rightarrow a2 \mid a \mid b2 \mid b \mid c2 \mid c$$

(2) $L = \{ b a^n b^m \mid n > m \geq 0 \}$

$S \to bA$

$A \to aAb \mid a \mid aA$

Dimostriamo che non è regolare usando il Pumping Lemma.

Sia $k \in \mathbb{N}$. Scelgo una stringa $w \in L$, con $|w| \geq k$

$w = b a^{k+1} b^k \qquad |w| = 1 + k + 1 + k = 2k + 2 \geq k$

(2.1) Sia $w = xyz$ con $x = b a^t$ $y = a^s$ con $s \geq 1$

$z = a^{k+1-t-s} b^k \qquad 1 + t + s \leq k$

$x y^0 z = b a^t a^{k+1-t-s} b^k$

$= b a^{k+1-s} b^k \notin L$ poiché $s \geq 1 \Rightarrow k + 1 - s \neq k$

(2.2) $w = ba^{k+1}b^k$ $\quad x = \varepsilon \quad y = ba^s \quad s \geq 0$

$z = a^{k+1-s}b^k$

$xy^0z = a^{k+1-s}b^k \notin L$ perché non inizia con b

```
int check (int a[], int dim)
{
    int i; int min; int conta = 0;
    min = a[0];
    for (i = 1; i < dim; i++)
        if (a[i] < min) min = a[i];

    for (i = 0; i < dim; i++)
        if (a[i] == min) conta++;

    return (conta == 1);
}
```

```
int check (int a[], int dim)
  { int i; int min; int contamin;
    min = a[0]; contamin = 1;
    for (i=1; i<dim; i++)
      if (a[i] < min)
            { min = a[i]; contamin = 1;}
        else if (a[i] == min) contamin = contamin + 1;
    return (contamin == 1);
  }
```

④

```
int check (int a[], int b[], int dima, int dimb)
{
    int i; int trovato;
    i = 0; trovato = 0;
    while (i < dima && !trovato)
        if (a[i] ∉ b)
            trovato = 1;
        else i = i + 1;
    return(!trovato)
}
```

non posso scriverla in C

```
int check (int a[], int b[], int dima, int dimb)
{ int i; int trovato;
  i=0; trovato=0;        ok=1
  while (i<dima && !trovato)   ok
  {  int conta=0; int J=0;
     while (J<dimb && conta==0)
        if a[i]==b[J] conta++;
        else J=J+1;          ok=0
     if (conta==0) trovato = 1;
        else i=i+1;
  return(!trovato);
                      ok
}
```