TP 4 EX 2

Grupo 5:

Breno Fernando Guerra Marrão A97768

Tales André Rovaris Machado A96314

In [1]:
```
from pysmt.shortcuts import *
from pysmt.typing import *
```

Primeiramente para a análise do problema tivemos que criar uma pré e pós condição para a futura correção do programa, e para isso fizemos uma alteração na inicialização do array que será ordenado, onde o seu ultimo elemento será o estado changed, para facilitar a criação da transição.

```
seq = [-2,1,2,-1,4,-4,-3,3]
changed = True
while changed:
    changed = False
    for i in range(len(seq) - 1):
        if seq[i] > seq[i+1]:
            seq[i], seq[i+1] = seq[i+1], seq[i]
            changed = True

pass
```

sabendo disso as condições sao as seguintes:

$$seq = init(seq, n)$$
$$\forall k.\, 0 \leq k < n \implies seq[k+1] \geq seq[k]$$

onde init é a seguinte função:

In [2]:
```
def init(seq,n,x):
    for i in range(n):
        seq = Store(seq,Int(i),Int(x[i]))
        aux = i
    else:
        seq = Store(seq,Int(aux+1),Int(1))
    return seq
```

Já a transição trans(seq,seq') foi feita de modo semelhante, mas ao invés de mudar a variavel changed quando um valor é trocado, trocamos somente se o resultado final for diferente do inicial, não mudando o comportamento do programa mas facilitando a criação da transição.

In [9]:
```
def trans(seq,n):

    seq2 = (Symbol('seq'+str(n),ArrayType(INT, INT)))

    for i in range(n+1):
        seq2 = Store(seq2,Int(i),Select(seq,Int(i)))
    else:
        seq2 = Store(seq2,Int(n+1),Int(1))
    for i in range(n):
        a = Ite(GT(Select(seq2,Int(i)),Select(seq2,Int(i)+Int(1))),
            Select(seq2,Int(i)+Int(1)),
            Select(seq2,Int(i)))
        b = Ite(GT(Select(seq2,Int(i)),Select(seq2,Int(i)+Int(1))),
            Select(seq2,Int(i)),
            Select(seq2,Int(i)+1))
        seq2 = Store(seq2,Int(i),a)
        seq2 = Store(seq2,Int(i+1),b)

    seq2 = Store(seq2,Int(n+1),Ite(Equals(seq,seq2),Int(0),Int(1)))

    return seq2
```

E finalmente para a correção do programa foi usada a abordagem "Single Assignment Unfold" para evitar a utilização do invariante. Para isso tivemos que criar a classe SAU para a verificação por passos do pragrama e tambem as suas condiçoes logicas do ciclo, pre e pos condição que sao as mesmas anteriores exceto a condição de clico que é a seguinte:

$$seq[n+1] == 1$$

E então é feita a correção do programa que calcula as diversas transições ate um número de passos N e verifica se a pos condição para aqueles dados inputs é verdadeira nesses N passos ou até menos.

In [4]:

```
# Auxiliares
def prime(v):
    return Symbol("next(%s)" % v.symbol_name(), v.symbol_type())
def fresh(v):
    return FreshSymbol(typename=v.symbol_type(),template=v.symbol_name()+"_%d")


class SAU(object):
    """Trivial representation of a while cycle and its unfolding."""
    def __init__(self, variables, pre , pos, control, trans, sname="z3"):

        self.variables = variables       # variables
        self.pre = pre                   # pre-condition as a predicate in "variables"
        self.pos = pos                   # pos-condition as a predicate in "variables"
        self.control = control           # cycle control as a predicate in "variables"
        self.trans = trans               # cycle body as a binary transition relation
                                         # in "variables" and "prime variables"

        self.prime_variables = [prime(v) for v in self.variables]
        self.frames = [And([Not(control),pos])]
                 # inializa com uma só frame: a da terminação do ciclo

        self.solver = Solver(name=sname)

    def new_frame(self):
        freshs = [fresh(v) for v in self.variables]
        b = self.control
        S = self.trans.substitute(dict(zip(self.prime_variables,freshs)))
        W = self.frames[-1].substitute(dict(zip(self.variables,freshs)))

        self.frames.append(And([b , ForAll(freshs, Implies(S, W))]))

    def unfold(self,bound=0):
        n = 0
        while True:
            if n > bound:
                print("falha: número de tentativas ultrapassa o limite %d "%bound)
                break

            f = Or(self.frames)
            if self.solver.solve([self.pre,Not(f)]):
                self.new_frame()
                n += 1
            else:
                print("sucesso na tentativa %d "%n)
                break
```

Segue exemplos de execução com diferentes arrays

In [5]:

```
seq = (Symbol('seq'+'0',ArrayType(INT, INT)))

k  = Symbol("k",INT)

variables = [seq]

pre  =  Equals(seq,init(seq,8,[-2,1,2,-1,4,-4,-3,3]))     # pré-condição
pos  =  ForAll([k],Implies(And(k>=Int(0),k<Int(7)),GE(Select(seq,k+Int(1)) , Select(seq,k))))  # pós-condição
cond =  Equals(Select(seq,Int(8)),Int(1))                 # condição de controlo do ciclo
trans = Equals(prime(seq),trans(seq,7))     # corpo do ciclo como uma relaçao de transição

W = SAU(variables, pre, pos, cond, trans)

W.unfold(10)
```

```
sucesso na tentativa 6
```

In [8]:

```
array = (Symbol('array'+'0',ArrayType(INT, INT)))

k1  = Symbol("k1",INT)

variables1 = [array]

pre1  =  Equals(array,init(array,13,[4,2,7,9,12,3,7,-1,66,0,23,7,9]))     # pré-condição
pos1  =  ForAll([k1],Implies(And(k1>=Int(0),k1<Int(12)),GE(Select(array,k1+Int(1)) , Select(array,k1))))  # pós-co
cond1 =  Equals(Select(array,Int(13)),Int(1))                # condição de controlo do ciclo
trans1 = Equals(prime(array),trans(array,12))     # corpo do ciclo como uma relaçao de transição

Z = SAU(variables1,pre1,pos1,cond1,trans1)

Z.unfold(18)
```

```
sucesso na tentativa 9
```

In [12]:

```
array2 = (Symbol('array'+'2',ArrayType(INT, INT)))

k2  = Symbol("k2",INT)

variables2 = [array2]

pre2  =  Equals(array2,init(array2,16,[22,33,44,55,66,77,999,888,777,666,0,1000,9999,10,11,1]))     # pré-condiçã
pos2  =  ForAll([k2],Implies(And(k2>=Int(0),k2<Int(15)),GE(Select(array2,k2+Int(1)) , Select(array2,k2))))  # pós-
cond2 =  Equals(Select(array2,Int(16)),Int(1))                # condição de controlo do ciclo
trans2 = Equals(prime(array2),trans(array2,15))     # corpo do ciclo como uma relaçao de transição

X = SAU(variables2,pre2,pos2,cond2,trans2)

X.unfold(20)
```

```
sucesso na tentativa 15
```