Trabalho 2 - >Bubble Sort

14 de dezembro 2022

André Oliveira Barbosa 91684

Francisco António Borges Paulino a91666

Caso de Estudo

O programa Python seguinte implementa o algoritmo de bubble sort para ordenação in situ de um array de inteiros seq.

In [75]:

```
!pip install z3-solver
#from z3 import *
from pysmt.shortcuts import *
from pysmt.typing import *
```

```
Requirement already satisfied: z3-solver in c:\users\andre\anaconda3\envs\logica\lib\site-packages (4.11.2.0)
```

Objetivos

1. Defina a pré-condição e a pós-condição que descrevem a especificação deste algoritmo.
2. O ciclo `for` pode ser descrito por uma transição $\texttt{seq} \leftarrow exp(\texttt{seq})$. Construa uma relação de transição $\text{trans}(\texttt{seq}, \texttt{seq}')$ que modele esta atribuição.
3. Usando a técnica que lhe parecer mais conveniente verifique a correção do algoritmo.

In [76]:

```
def bubblesort(seq):
    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
```

Testes:

In [77]:

```
seq = [-2,1,2,-1,4,-4,-3,3]
bubblesort(seq)
print(seq)
```

```
[-4, -3, -2, -1, 1, 2, 3, 4]
```

1- Pré e Pós Condição

1. Pré Condição: N>=0 && changed == TRUE

2. Pós Condição: $\forall_{0 \leq i \leq N-1} seq[i] \leq seq[i+1]. \; N = len(seq) \wedge changed = False$

In [78]:

```
N = Symbol("N", INT)
seq = Symbol("seq", ArrayType(INT, INT))
i = Symbol("i", INT)

changed_pre = Bool(True)
changed_pos = Bool(False)

pre_condicao = And( changed_pre, GE(N, Int(0)))

pos_condicao = And(ForAll([i],(LE(Select(seq, i), Select(seq, Plus(i, Int(1))))),changed_pos,LE(i, Minus(N, Int(1))))
```

2- Construção do sistema de transições

In [79]:

```
temp = Symbol("temp", INT)
changed = Symbol("changed", BOOL)

trans_for = And(
    ForAll([i],
           Implies(
               And(GE(i, Int(0)), LT(i, N-1)),
               Implies(
                   #if executado
                   GT(Select(seq, i), Select(seq, Plus(i, Int(1)))),
                   And(
                        Equals(temp, Select(seq, i)),
                        Equals(Select(seq, i), Select(seq, Plus(i, Int(1)))),
                        Equals(Select(seq, Plus(i, Int(1))), temp),
                        changed)
               )
           )
    )
)

trans_while = And(
    #inicialmente, changed==TRUE
    changed_pre,
    #for executado
    Implies(changed,
            And(
                changed,
                trans_for,
                Not(changed_pos)
            )
    ),
    #antes do ciclo for
    Implies(Not(changed),
            changed_pre
    )
)
```

Correção

Abordagem Single Assingnment Unfold (SAU)

In [80]:

```
# 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")

#classe Single Assignment Unfold
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
        self.pre = pre
        self.pos = pos
        self.control = control
        self.trans = trans


        self.prime_variables = [prime(v) for v in self.variables]
        self.frames = [And([Not(control),pos])]

        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
```

In [81]:

```
trans = Or(trans_for, trans_while)
cond = Not(changed_pos)
variables = [N, seq ,i ,temp ,changed]

W = SAU(variables,pre,pos,cond,trans)
W.unfold(6)
```

```
> Falha: número de tentativas ultrapassa o limite 6
```

In [ ]: