Trabalho 2 - SFOTS

```
14 de dezembro 2022
```

André Oliveira Barbosa 91684 Francisco António Borges Paulino

Caso de estudo

O seguinte sistema dinâmico denota 4 inversores (A, B, C, D) que lêm um bit num canal input e escrevem num canal output uma transformação desse bit.

Condições

1. Cada inversor tem um bit s de estado, inicializado com um valor aleatório.
2. Cada inversor é regido pelas seguintes transformações $\textbf{invert}(in, out)$ $x \leftarrow \mathsf{read}(\texttt{in})$
   $s \leftarrow \neg x \;\; \| \;\; s \leftarrow s \oplus x$
   $\mathsf{write}(\texttt{out}, s)$
3. A escolha neste comando é sempre determinística; isto é, em cada inversor a escolha do comando a executar é sempre a mesma. Porém qual é essa escolha é determinada aleatoriamente na inicializarão do sistema.
4. O estado do sistema é um duplo definido pelos 4 bits s, e é inicializado com um vetor aleatório em $\{0,1\}^4$.
5. O sistema termina em ERRO quando o estado do sistema for $(0,0,0,0)$.

Objetivos

1. Construa um SFOTS que descreva este sistema e implemente este sistema, numa abordagem BMC (“bouded model checker”) num traço com n estados.
2. Verifique se o sistema é seguro usando BMC, k-indução ou model checking com interpolantes.

In [16]:
```
from pysmt.shortcuts import *
from pysmt.typing import INT
from pysmt.typing import ArrayType
from pysmt.shortcuts import Symbol, Store, Select, Int, get_model
from random import randint
import random
import itertools
```

In [17]:
```
n=4
x=1
t=random.randint(0,1)
y=random.randint(0,1)
g=random.randint(0,1)
h=random.randint(0,1)
```

SFOTS

In [18]:
```
def genState(vars,s,i):
    state = {}
    for v in vars:
        '''
        if v == 'pc':
            state[v] = Symbol(v+'!'+s+str(i),INT)
        else:
        '''
        state[v] = Symbol(v+'!'+s+str(i),BVType(n))
    return state
```

In [19]:
```
def init1(state):
    return And(Equals(state['pc'], BV(0,n)),
               Equals(state['a'], BV(t,n)),
               Equals(state['op1'], BV((random.randint(1,2)),n)),
               Equals(state['b'], BV(y,n)),
               Equals(state['op2'], BV((random.randint(1,2)),n)),
               Equals(state['c'], BV(g,n)),
               Equals(state['op3'], BV((random.randint(1,2)),n)),
               Equals(state['d'], BV(h,n)),
               Equals(state['op4'], BV((random.randint(1,2)),n)),
               Equals(state['x'], BV(x,n)))

def error1(state):
    return Equals(state['pc'], BV(13,n))

def trans1(curr, prox):

    t01 = And(Equals(curr['pc'], BV(0,n)),
              Equals(prox['pc'], BV(1,n)),
              Equals(prox['a'], curr['a']),
              Equals(prox['op1'], curr['op1']),
              Equals(prox['b'], curr['b']),
              Equals(prox['op2'], curr['op2']),
              Equals(prox['c'], curr['c']),
              Equals(prox['op3'], curr['op3']),
              Equals(prox['d'], curr['d']),
              Equals(prox['op4'], curr['op4']),
              Equals(prox['x'], curr['x']))
    #A

    t12 = And(Equals(curr['pc'], BV(1,n)),
              NotEquals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
              Equals(prox['op1'],BV(1,n)),
              Equals(prox['pc'], BV(2,n)),
              Equals(prox['a'], curr['a']),
              Equals(prox['op1'], curr['op1']),
              Equals(prox['b'], curr['b']),
              Equals(prox['op2'], curr['op2']),
              Equals(prox['c'], curr['c']),
              Equals(prox['op3'], curr['op3']),
              Equals(prox['d'], curr['d']),
              Equals(prox['op4'], curr['op4']),
              Equals(prox['x'], curr['x']))



    t24 = And(Equals(curr['pc'], BV(2,n)),
              Equals(prox['pc'], BV(4,n)),
              Equals(prox['a'], ~curr['x']+2),
              Equals(prox['op1'], curr['op1']),
              Equals(prox['b'], curr['b']),
              Equals(prox['op2'], curr['op2']),
              Equals(prox['c'], curr['c']),
              Equals(prox['op3'], curr['op3']),
              Equals(prox['d'], curr['d']),
              Equals(prox['op4'], curr['op4']),
```

```
            Equals(prox['x'], curr['x']))

t13 = And(Equals(curr['pc'], BV(1,n)),
          NotEquals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
          Equals(prox['op1'],BV(2,n)),
          Equals(prox['pc'], BV(3,n)),
          Equals(prox['a'], curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))


t34 = And(Equals(curr['pc'], BV(3,n)),
          Equals(prox['pc'], BV(4,n)),
          Equals(prox['a'], curr['a']^curr['x']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))
#B
t45 = And(Equals(curr['pc'], BV(4,n)),
          NotEquals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
          Equals(prox['op2'],BV(1,n)),
          Equals(prox['pc'], BV(5,n)),
          Equals(prox['a'], curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))



t57 = And(Equals(curr['pc'], BV(5,n)),
          Equals(prox['pc'], BV(7,n)),
          Equals(prox['b'], ~curr['a']+2),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['a'], curr['a']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))
```

```
t46 = And(Equals(curr['pc'], BV(4,n)),
          NotEquals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
          Equals(prox['op2'],BV(2,n)),
          Equals(prox['pc'], BV(6,n)),
          Equals(prox['a'], curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))


t67 = And(Equals(curr['pc'], BV(6,n)),
          Equals(prox['pc'], BV(7,n)),
          Equals(prox['b'], curr['b']^curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['a'], curr['a']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))

#C

t78 = And(Equals(curr['pc'], BV(7,n)),
          NotEquals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
          Equals(prox['op3'],BV(1,n)),
          Equals(prox['pc'], BV(8,n)),
          Equals(prox['a'], curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))



t810 = And(Equals(curr['pc'], BV(8,n)),
          Equals(prox['pc'], BV(10,n)),
          Equals(prox['c'], ~curr['b']+2),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['a'], curr['a']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))
```

```
t79 = And(Equals(curr['pc'], BV(7,n)),
          NotEquals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
          Equals(prox['op3'],BV(2,n)),
          Equals(prox['pc'], BV(9,n)),
          Equals(prox['a'], curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))


t910 = And(Equals(curr['pc'], BV(9,n)),
          Equals(prox['pc'], BV(10,n)),
          Equals(prox['c'], curr['c']^curr['b']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['a'], curr['a']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))

#D

t1011 = And(Equals(curr['pc'], BV(10,n)),
          NotEquals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
          Equals(prox['op4'],BV(1,n)),
          Equals(prox['pc'], BV(11,n)),
          Equals(prox['a'], curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))



t110 = And(Equals(curr['pc'], BV(11,n)),
          Equals(prox['pc'], BV(0,n)),
          Equals(prox['d'], ~curr['c']+2),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['a'], curr['a']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['d']))
```

```
t1012 = And(Equals(curr['pc'], BV(10,n)),
          NotEquals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
          Equals(prox['op4'],BV(2,n)),
          Equals(prox['pc'], BV(12,n)),
          Equals(prox['a'], curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))


t120 = And(Equals(curr['pc'], BV(12,n)),
          Equals(prox['pc'], BV(0,n)),
          Equals(prox['d'], curr['d']^curr['c']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['a'], curr['a']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['d']))

#Error

t013 = And(Equals(curr['pc'], BV(0,n)),
          Equals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
          Equals(prox['pc'], BV(13,n)),
          Equals(prox['a'], curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))

t413 = And(Equals(curr['pc'], BV(4,n)),
          Equals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
          Equals(prox['pc'], BV(13,n)),
          Equals(prox['a'], curr['a']),
          Equals(prox['op1'], curr['op1']),
          Equals(prox['b'], curr['b']),
          Equals(prox['op2'], curr['op2']),
          Equals(prox['c'], curr['c']),
          Equals(prox['op3'], curr['op3']),
          Equals(prox['d'], curr['d']),
          Equals(prox['op4'], curr['op4']),
          Equals(prox['x'], curr['x']))

t713 = And(Equals(curr['pc'], BV(7,n)),
          Equals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
```

```
            Equals(prox['pc'], BV(13,n)),
            Equals(prox['a'], curr['a']),
            Equals(prox['op1'], curr['op1']),
            Equals(prox['b'], curr['b']),
            Equals(prox['op2'], curr['op2']),
            Equals(prox['c'], curr['c']),
            Equals(prox['op3'], curr['op3']),
            Equals(prox['d'], curr['d']),
            Equals(prox['op4'], curr['op4']),
            Equals(prox['x'], curr['x']))

    t1013 = And(Equals(curr['pc'], BV(10,n)),
            Equals((curr['a']+curr['b']+curr['c']+curr['d']),BV(0,n)),
            Equals(prox['pc'], BV(13,n)),
            Equals(prox['a'], curr['a']),
            Equals(prox['op1'], curr['op1']),
            Equals(prox['b'], curr['b']),
            Equals(prox['op2'], curr['op2']),
            Equals(prox['c'], curr['c']),
            Equals(prox['op3'], curr['op3']),
            Equals(prox['d'], curr['d']),
            Equals(prox['op4'], curr['op4']),
            Equals(prox['x'], curr['x']))

    t1313 = And(Equals(curr['pc'], BV(13,n)),
            Equals(prox['pc'], BV(13,n)),
            Equals(prox['a'], curr['a']),
            Equals(prox['op1'], curr['op1']),
            Equals(prox['b'], curr['b']),
            Equals(prox['op2'], curr['op2']),
            Equals(prox['c'], curr['c']),
            Equals(prox['op3'], curr['op3']),
            Equals(prox['d'], curr['d']),
            Equals(prox['op4'], curr['op4']),
            Equals(prox['x'], curr['x']))


    return Or(t01, t12, t24, t13, t34, t45, t57, t46, t67, t78, t810, t79, t910,
```

In [20]:
```
def genTrace(vars,init,trans,n):
    with Solver(name="z3") as s:

        X = [genState(vars,'X',i) for i in range(n+1)]   # cria n+1 estados (com
        #print(X)
        I = init(X[0])
        print(I)
        Tks = [ trans(X[i],X[i+1]) for i in range(n) ]
        #print(Tks)

        if s.solve([I,And(Tks)]):      # testa se I /\ T^n  é satisfazível
            for i in range(n):
                print("Estado:",i)
                for v in X[i]:
                    print("          ",v,'=',s.get_value(X[i][v]))
```

In [21]:
```
genTrace(['a','b','c','d','x','op1','op2','op3','op4','pc'],init1,trans1,10)
```

```
           a = 0_4
           b = 1_4
           c = 0_4

           d = 1_4
           x = 1_4
           op1 = 2_4
           op2 = 1_4
           op3 = 1_4
           op4 = 2_4
           pc = 12_4
Estado: 9
           a = 0_4
           b = 1_4
           c = 0_4
           d = 1_4
           x = 1_4
           op1 = 2_4
           op2 = 1_4
           op3 = 1_4
```

Verificação do sistema

In [22]:
```
def baseName(s):
    return ''.join(list(itertools.takewhile(lambda x: x!='!', s)))

def rename(form,state):
    vs = get_free_variables(form)
    pairs = [ (x,state[baseName(x.symbol_name())]) for x in vs ]
    return form.substitute(dict(pairs))

def same(state1,state2):
    return And([Equals(state1[x],state2[x]) for x in state1])
```

In [23]:
```
def invert(trans1):
    return (lambda u, v : trans1(v,u))
```

In [24]:
```
def model_checking(vars,init,trans,error,N,M):
    Iflag=1
    while (Iflag==1): #ciclo de verificação do interpolante
        with Solver(name="z3") as s:

            # Criar todos os estados que poderão vir a ser necessários.
            X = [genState(vars,'X',i) for i in range(N+1)]
            #print(X)
            Y = [genState(vars,'Y',i) for i in range(M+1)]
            #print(Y)


            # Estabelecer a ordem pela qual os pares (n,m) vão surgir. Por exempl
            order = sorted([(a,b) for a in range(1,N+1) for b in range(1,M+1)],ke

            #print(order)
            #print(len(order))

            for (n,m) in order:

                I = init(X[0])
                #print(I)

                Tn = And([trans(X[i],X[i+1]) for i in range(n)])
                #print(Tn)

                Rn = And(I,Tn)
                #print(Rn)

                E = error(Y[0])
                #print(E)

                Bm = And([invert(trans)(X[i],X[i+1]) for i in range(n)])
                #print(Bm)

                Um = And(E,Bm)
                #print(Um)


                Vnm = And(Rn,same(X[n],Y[m]),Um)

                #print(Vnm)

                if s.solve([Vnm]):
                    print('unsafe')
                    return

                C = binary_interpolant(And(Rn,same(X[n],Y[m])),Um)
                if C is None:                          # Vnm insatisfazível
                    print('interpolante None\n')
                    while TRUE: #não aceita outros indices
                        escolha = input("Deseja incrementar o índice n ou m?: ")
                        if (escolha=="n"):
                            n=n+1
                            print("> Valor de n alterado para: n=",n)
                            break
```

```
                        elif(escolha== "m"):
                            m=m+1
                            print("> Valor de m alterado para: m=",m)
                            break
                else: Iflag=0
                # fim do ciclo de verificação do interpolante

                C0 = rename(C,X[0])
                C1 = rename(C,X[1])

                T = trans(X[0], X[1])

                if not s.solve([C0,T,Not(C1)]):         # C é invariante de T
                    print('safe')
                    return
                else:                                   # tenta gerar o majorante S
                    S = rename(C,X[n])
                    while True:
                        A = And(S,trans(X[n],Y[n]))
                        if s.solve([A,Um]):
                            print('Não foi possivel encontrar majorante.')
                            break
                        else:
                            Cnew = binary_interpolant(A,Um)
                            Cn = rename(Cnew,X[n])
                            if s.solve([Cn,Not(S)]):     # se Cn->S não é tautologia
                                S = Or(S,Cn)
                            else:
                                print('safe')
                                return

        print('unknown')

model_checking(['a','b','c','d','x','op1','op2','op3','op4','pc'], init1, trans1,
```

```
safe
```

In [ ]:

In [ ]: