# TP4-Exercicio2

January 17, 2021

Trabalho Realizado Por:

Carlos Ferreira - a87953 Daniel Ribeiro - a87994

### 0.0.1 Exercício 2

2. Considere um sistema híbrido formado por 4 autómatos híbridos: três navios (análogos aos do trabalho TP3) e um controlador. Neste sistema cada autómato desconhece o estado dos restantes e comunica com eles exclusivamente através de eventos.
   A propriedade de segurança é a mesma da do trabalho TP2: ausência de colisões entre navios. Para isso a área de navegação é dividida em setores e o controlador assegura que, em qualquer instante, cada sector não contém mais do que um navio.
   Assim

   a. Cada navio está, em cada estado, numa de três velocidades $v$ possíveis: $10\,\mathtt{m/s}\,(\mathsf{high})$ , $1\,\mathtt{m/s}\,(\mathsf{low})$ e $0\,(\mathsf{stop})$. As transições $\mathsf{high} \leftrightarrow \mathsf{low}$ têm uma duração mínima de $500\,\mathsf{seg}$ ; as transições $\mathsf{low} \leftrightarrow \mathsf{stop}$ têm uma duração mínima de $50\,\mathsf{seg}$.

   b. Cada navio está, em cada estado, numa rota $r \in \{0..23\}$ ; cada valor de $r$ identifica um ângulo múltiplo de $15^o$ (também designado por `hora`).

   c. A área de navegação é dividida numa matriz $N \times N$ de setores quadrados com $1\,\mathtt{km}$ de lado. Cada setor setor é identificado por um par de índices $0 \leq \mathtt{linha}, \mathtt{coluna} < N$. Cada navio está, em cada estado, num único setor.

   d. O estado do controlador incluiu o seu setor e a sua velocidade.

A navegação é determinada pelas seguintes regras:

```
>a. Um navio só muda de rota ou velocidade quando muda de setor.   \
>b. Quando um navio entra ou sai de um setor emite um evento, que identifica o navio e os seto
>c. Se existir risco de dois navios estarem simultaneamente no mesmo setor, o controlador deve
>d. Dois navios em setores adjacentes estão ambos em velocidade $\,\mathsf{low}\,$ ou $\,\maths
```

Pretende-se verificar, dada uma determinada posição inicial dos três navios, a seguinte propriedade de segurança:
Em qualquer traço e em qualquer estado, nenhum setor contém mais do que um navio.

Explicação

Para a resolução do problema usamos um autómato híbrido com controlador.

Resolução

[83]:
```
from z3 import *
from random import choice,randint
from numpy import cos,sin,deg2rad


#declarar lista de angulos possiveis
l = [i for i in range(0,346,15)]
muda_rota = [ -15,15]

#declarar velocidades e estados do automato
Velocidade, (STOP,LOW,HIGHT) = EnumSort('Velocidade', ('STOP','LOW','HIGHT'))
Mode, (INIT,ON) = EnumSort('Mode', ('INIT','ON'))



def declare(i):
    s = {}
    s['bA'] = {}
    s['bB'] = {}
    s['bC'] = {}
    s['con'] = {}

    #modos
    s['bA']['m'] =␣
 ↪(Real('rota_barcoA_'+str(i)),Const('velocidade_barcoA_'+str(i),Velocidade))
    s['bB']['m'] =␣
 ↪(Real('rota_barcoB_'+str(i)),Const('velocidade_barcoB_'+str(i),Velocidade))
    s['bC']['m'] =␣
 ↪(Real('rota_barcoC_'+str(i)),Const('velocidade_barcoC_'+str(i),Velocidade))

    #setores
    s['bA']['s'] = (Int('setor_barcoA_x_'+str(i)),Int('setor_barcoA_y_'+str(i)))
    s['bB']['s'] = (Int('setor_barcoB_x_'+str(i)),Int('setor_barcoB_y_'+str(i)))
    s['bC']['s'] = (Int('setor_barcoC_x_'+str(i)),Int('setor_barcoC_y_'+str(i)))

    #posicoes
    s['bA']['p'] = (Real('x_barcoA_'+str(i)),Real('y_barcoA_'+str(i)))
    s['bB']['p'] = (Real('x_barcoB_'+str(i)),Real('y_barcoB_'+str(i)))
    s['bC']['p'] = (Real('x_barcoC_'+str(i)),Real('y_barcoC_'+str(i)))
```

```
    #tempos
    s['bA']['t'] = (Real('t_barco1_'+str(i)))
    s['bB']['t'] = (Real('t_barcoB_'+str(i)))
    s['bC']['t'] = (Real('t_barcoC_'+str(i)))

    #controlador A
    s['con']['bA'] = {}
    s['con']['bA']['m'] =␣
↪(Real('c_rota_barcoA_'+str(i)),Const('c_velocidade_barcoA_'+str(i),Velocidade))
    s['con']['bA']['s'] =␣
↪(Int('c_setor_bA_x_'+str(i)),Int('c_setor_bA_y_'+str(i)))
    s['con']['bA']['tempoTrans'] = (Real('c_tempoTrans_bA_x_'+str(i)))

    #controlador B
    s['con']['bB'] = {}
    s['con']['bB']['m'] =␣
↪(Real('c_rota_barcoB_'+str(i)),Const('c_velocidade_barcoB_'+str(i),Velocidade))
    s['con']['bB']['s'] =␣
↪(Int('c_setor_bB_x_'+str(i)),Int('c_setor_bB_y_'+str(i)))
    s['con']['bB']['tempoTrans'] = (Real('c_tempoTrans_bB_x_'+str(i)))

    #controlador C
    s['con']['bC'] = {}
    s['con']['bC']['m'] =␣
↪(Real('c_rota_barcoC_'+str(i)),Const('c_velocidade_barcoC_'+str(i),Velocidade))
    s['con']['bC']['s'] =␣
↪(Int('c_setor_bC_x_'+str(i)),Int('c_setor_bC_y_'+str(i)))
    s['con']['bC']['tempoTrans'] = (Real('c_tempoTrans_bC_x_'+str(i)))

    #tempos
    s['con']['t'] = (Real('t_con_'+str(i)))

    ######
    s['e'] = Const('m'+str(i),Mode)

    return s



def init(s):

    #modos
    modos = And(s['bA']['m'][0] == s['con']['bA']['m'][0],
                s['bA']['m'][1] == s['con']['bA']['m'][1],
                s['bB']['m'][0] == s['con']['bB']['m'][0],
```

```
            s['bB']['m'][1] == s['con']['bB']['m'][1],
            s['bC']['m'][0] == s['con']['bC']['m'][0],
            s['bC']['m'][1] == s['con']['bC']['m'][1])

#setores
setores = And(s['bA']['s'][0] == randint(0,1000),
              s['bA']['s'][1] == randint(0,1000),
              s['bB']['s'][0] == randint(0,1000),
              s['bB']['s'][1] == randint(0,1000),
              s['bC']['s'][0] == randint(0,1000),
              s['bC']['s'][1] == randint(0,1000),
             )

#posicao
posicao = And(s['bA']['p'][0] == s['bA']['s'][1] + randint(0,1000),
              s['bA']['p'][1] == s['bA']['s'][1] + randint(0,1000),
              s['bB']['p'][0] == s['bB']['s'][1] + randint(0,1000),
              s['bB']['p'][1] == s['bB']['s'][1] + randint(0,1000),
              s['bC']['p'][0] == s['bC']['s'][1] + randint(0,1000),
              s['bC']['p'][1] == s['bC']['s'][1] + randint(0,1000)
             )

#tempos
tempos = And (s['bA']['t'] == 0,
              s['bB']['t'] == 0,
              s['bC']['t'] == 0
             )


#controlador
controlador = And(s['con']['bA']['m'][0] == choice(l),
                  s['con']['bB']['m'][0] == choice(l),
                  s['con']['bC']['m'][0] == choice(l),

                  s['con']['bA']['m'][1] == HIGHT,
                  s['con']['bB']['m'][1] == HIGHT,
                  s['con']['bC']['m'][1] == HIGHT,

                  s['con']['bA']['s'][0] == s['bA']['s'][0],
                  s['con']['bA']['s'][1] == s['bA']['s'][1],

                  s['con']['bB']['s'][0] == s['bB']['s'][0],
                  s['con']['bB']['s'][1] == s['bB']['s'][1],

                  s['con']['bC']['s'][0] == s['bC']['s'][0],
                  s['con']['bC']['s'][1] == s['bC']['s'][1],
                  s['con']['t'] == 0,
```

```
                    s['con']['bA']['tempoTrans'] == 0,
                    s['con']['bB']['tempoTrans'] == 0,
                    s['con']['bC']['tempoTrans'] == 0
                   )



    return And(modos,setores,posicao,tempos,controlador,s['e'] == INIT)
```

[88]:

```
# Função que indica se existe perigo entre 2 barcos #

def danger(s,b1,b2,d):
    x=And(s['con'][b1]['s'][0] <= s['con'][b2]['s'][0] + d,␣
 ↪s['con'][b2]['s'][0] <= s['con'][b1]['s'][0] + d  )
    y=And(s['con'][b1]['s'][1] <= s['con'][b2]['s'][1] + d,␣
 ↪s['con'][b2]['s'][1] <= s['con'][b1]['s'][1] + d  )
    return And(x,y)


#funcao que dá-nos o proximo x do barco usando a formula x_prox = x_ant *␣
 ↪cos(angulo) * v
def proximaPosicaoX (x,y,a,v,sol):
    if sol.check() == sat:
        m = sol.model()
        return If( v == LOW , x + math.cos(m[a].as_long())*1 , If( v == HIGHT ,␣
 ↪x + math.cos(m[a].as_long())*10 , x))


#funcao que dá-nos o proximo y do barco usando a formula y_prox = y_ant *␣
 ↪sen(angulo) * v
def proximaPosicaoY (x,y,a,v,sol):
    if sol.check() == sat:
        m = sol.model()

        return If( v == LOW , y + math.sin(m[a].as_long())*1 , If( v == HIGHT ,␣
 ↪x + math.sin(m[a].as_long())*10 , x))


def mudaSetor (s,p,b,coord):

        return If(p[b]['p'][coord] >= s[b]['s'][coord] * 1000,
               If(p[b]['p'][coord] <= s[b]['s'][coord] * 1000, p[b]['s'][coord]␣
 ↪== s[b]['s'][coord],
```

```
                    p[b]['s'][coord] == s[b]['s'][coord] + 1) ,  p[b]['s'][coord]
↪== s[b]['s'][coord] - 1)


def trans(s,p,sol):

    r = 100
    # Transições auxiliares

    # Modo barcos igual
    M_bA=And(p['bA']['m'][0] == s['bA']['m'][0], p['bA']['m'][1] ==
↪s['bA']['m'][1])
    M_bB=And(p['bB']['m'][0] == s['bB']['m'][0], p['bB']['m'][1] ==
↪s['bB']['m'][1])
    M_bC=And(p['bC']['m'][0] == s['bC']['m'][0], p['bC']['m'][1] ==
↪s['bC']['m'][1])

    # Modo controlador igual
    M_con_bA=And(p['con']['bA']['m'][0] == s['con']['bA']['m'][0],
↪p['con']['bA']['m'][1]==s['con']['bA']['m'][1])
    M_con_bB=And(p['con']['bB']['m'][0] == s['con']['bB']['m'][0],
↪p['con']['bB']['m'][1]==s['con']['bB']['m'][1])
    M_con_bC=And(p['con']['bC']['m'][0] == s['con']['bC']['m'][0],
↪p['con']['bC']['m'][1]==s['con']['bC']['m'][1])

    # Perigo
    d_bA=Or(danger(s,'bA','bB',r),danger(s,'bA','bC',r))
    d_bB=Or(danger(s,'bB','bA',r),danger(s,'bB','bC',r))
    d_bC=Or(danger(s,'bC','bA',r),danger(s,'bC','bB',r))

    # Seguro
    safe_bA=Not(d_bA)
    safe_bB=Not(d_bB)
    safe_bC=Not(d_bC)

    # Posicao igual
    p_bA=And(p['bA']['p'][0] == s['bA']['p'][0],p['bA']['p'][1] ==
↪s['bA']['p'][1])
    p_bB=And(p['bB']['p'][0] == s['bB']['p'][0],p['bB']['p'][1] ==
↪s['bB']['p'][1])
    p_bC=And(p['bC']['p'][0] == s['bC']['p'][0],p['bC']['p'][1] ==
↪s['bC']['p'][1])

    # setor igual
    se_bA=And(p['bA']['s'][0] == s['bA']['s'][0],p['bA']['s'][1] ==
↪s['bA']['s'][1])
```

```
   se_bB=And(p['bB']['s'][0] == s['bB']['s'][0],p['bB']['s'][1] ==␣
↪s['bB']['s'][1])
   se_bC=And(p['bC']['s'][0] == s['bC']['s'][0],p['bC']['s'][1] ==␣
↪s['bC']['s'][1])

   # tempos igual
   t_bA=And(p['bA']['t'] == s['bA']['t'])
   t_bB=And(p['bB']['t'] == s['bB']['t'])
   t_bC=And(p['bC']['t'] == s['bC']['t'])

   # setor igual controlador
   s_con_bA=And(p['con']['bA']['s'][0] ==␣
↪s['con']['bA']['s'][0],p['con']['bA']['s'][1] == s['con']['bA']['s'][1])
   s_con_bB=And(p['con']['bB']['s'][0] ==␣
↪s['con']['bB']['s'][0],p['con']['bB']['s'][1] == s['con']['bB']['s'][1])
   s_con_bC=And(p['con']['bC']['s'][0] ==␣
↪s['con']['bC']['s'][0],p['con']['bC']['s'][1] == s['con']['bC']['s'][1])

   # igualar a variaçao de tempo
   vt=And(p['con']['t'] - s['con']['t'] == p['bA']['t'] - s['bA']['t'],
          p['con']['t'] - s['con']['t'] == p['bB']['t'] - s['bB']['t'],
          p['con']['t'] - s['con']['t'] == p['bC']['t'] - s['bC']['t'])

   # igualar a variaçao de tempo
   vt_vbA=(p['con']['t'] - s['con']['t'] == p['con']['bA']['tempoTrans'] -␣
↪s['con']['bA']['tempoTrans'])
   vt_vbB=(p['con']['t'] - s['con']['t'] == p['con']['bB']['tempoTrans'] -␣
↪s['con']['bB']['tempoTrans'])
   vt_vbC=(p['con']['t'] - s['con']['t'] == p['con']['bC']['tempoTrans'] -␣
↪s['con']['bC']['tempoTrans'])

   #transicao do estado inicial para "on" do automato
   init_on = And(
                  s['e'] == INIT,
                  p['e'] == ON,
                  M_bA, M_bB, M_bC, M_con_bA, M_con_bB, M_con_bC,
                  p_bA, p_bB, p_bC, se_bA, se_bB, se_bC,t_bA,t_bB,t_bC,
                  safe_bA, safe_bB, safe_bC, s_con_bA, s_con_bB, s_con_bC,
       vt ,vt_vbA , vt_vbB , vt_vbC
              )


   # timed aux

  ␣
↪sc_bAx=And(p['bA']['p'][0]>=s['bA']['s'][0]*1000,p['bA']['p'][0]<s['bA']['s'][0]*1000+1000,
```

```
            p['bA']['s'][0]==s['bA']['s'][0])
  ␣
↪sc_bBx=And(p['bB']['p'][0]>=s['bB']['s'][0]*1000,p['bB']['p'][0]<s['bB']['s'][0]*1000+1000,
            p['bB']['s'][0]==s['bB']['s'][0])
  ␣
↪sc_bCx=And(p['bC']['p'][0]>=s['bC']['s'][0]*1000,p['bC']['p'][0]<s['bC']['s'][0]*1000+1000,
            p['bC']['s'][0]==s['bC']['s'][0])
  ␣
↪sm_bAx=And(p['bA']['p'][0]<s['bA']['s'][0]*1000,p['bA']['s'][0]==s['bA']['s'][0]-1)
  ␣
↪sm_bBx=And(p['bB']['p'][0]<s['bB']['s'][0]*1000,p['bB']['s'][0]==s['bB']['s'][0]-1)
  ␣
↪sm_bCx=And(p['bC']['p'][0]<s['bC']['s'][0]*1000,p['bC']['s'][0]==s['bC']['s'][0]-1)
  ␣
↪sM_bAx=And(p['bA']['p'][0]>=s['bA']['s'][0]*1000+1000,p['bA']['s'][0]==s['bA']['s'][0]+1)
  ␣
↪sM_bBx=And(p['bB']['p'][0]>=s['bB']['s'][0]*1000+1000,p['bB']['s'][0]==s['bB']['s'][0]+1)
  ␣
↪sM_bCx=And(p['bB']['p'][0]>=s['bC']['s'][0]*1000+1000,p['bC']['s'][0]==s['bC']['s'][0]+1)
  ␣
↪sc_bAy=And(p['bA']['p'][1]>=s['bA']['s'][1]*1000,p['bA']['p'][0]<s['bA']['s'][1]*1000+1000,
            p['bA']['s'][1]==s['bA']['s'][1])
  ␣
↪sc_bBy=And(p['bB']['p'][1]>=s['bB']['s'][1]*1000,p['bB']['p'][0]<s['bB']['s'][1]*1000+1000,
            p['bB']['s'][1]==s['bB']['s'][1])
  ␣
↪sc_bCy=And(p['bC']['p'][1]>=s['bC']['s'][1]*1000,p['bC']['p'][0]<s['bC']['s'][1]*1000+1000,
            p['bC']['s'][1]==s['bC']['s'][1])
  ␣
↪sm_bAy=And(p['bA']['p'][1]<s['bA']['s'][1]*1000,p['bA']['s'][1]==s['bA']['s'][1]-1)
  ␣
↪sm_bBy=And(p['bB']['p'][1]<s['bB']['s'][1]*1000,p['bB']['s'][1]==s['bB']['s'][1]-1)
  ␣
↪sm_bCy=And(p['bC']['p'][1]<s['bC']['s'][1]*1000,p['bC']['s'][1]==s['bC']['s'][1]-1)
  ␣
↪sM_bAy=And(p['bA']['p'][1]>=s['bA']['s'][1]*1000+1000,p['bA']['s'][1]==s['bA']['s'][1]+1)
  ␣
↪sM_bBy=And(p['bB']['p'][1]>=s['bB']['s'][1]*1000+1000,p['bB']['s'][1]==s['bB']['s'][1]+1)
  ␣
↪sM_bCy=And(p['bB']['p'][1]>=s['bC']['s'][1]*1000+1000,p['bC']['s'][1]==s['bC']['s'][1]+1)

   s_bA=And(Or(sc_bAx,sm_bAx,sM_bAx),Or(sc_bAy,sm_bAy,sM_bAy))
   s_bB=And(Or(sc_bBx,sm_bBx,sM_bBx),Or(sc_bBy,sm_bBy,sM_bBy))
   s_bC=And(Or(sc_bCx,sm_bCx,sM_bCx),Or(sc_bCy,sm_bCy,sM_bCy))

   #caso nao se verifique perigo todos os barcos vao andar
```

```
    timed = And(
                     s['e'] == ON,
                     p['e'] == ON,
                     s_bA, s_bB, s_bC,
                     vt, vt_vbA, vt_vbB, vt_vbC,
                     safe_bA , safe_bB , safe_bC ,
                      M_bA, M_bB, M_bC, M_con_bA, M_con_bB, M_con_bC,
                    p['bA']['p'][0] ==␣
↪proximaPosicaoX(s['bA']['p'][0],s['bA']['p'][1],s['bA']['m'][0],s['bA']['m'][1],sol),
                    p['bA']['p'][1] ==␣
↪proximaPosicaoY(s['bA']['p'][0],s['bA']['p'][1],s['bA']['m'][0],s['bA']['m'][1],sol),
                    p['bA']['t'] == s['bA']['t'] + 1,
                    p['bB']['p'][0] ==␣
↪proximaPosicaoX(s['bB']['p'][0],s['bB']['p'][1],s['bB']['m'][0],s['bB']['m'][1],sol),
                    p['bB']['p'][1] ==␣
↪proximaPosicaoY(s['bB']['p'][0],s['bB']['p'][1],s['bB']['m'][0],s['bB']['m'][1],sol),
                    p['bB']['t'] == s['bA']['t'] + 1,
                    p['bC']['p'][0] ==␣
↪proximaPosicaoX(s['bC']['p'][0],s['bC']['p'][1],s['bC']['m'][0],s['bC']['m'][1],sol),
                    p['bC']['p'][1] ==␣
↪proximaPosicaoY(s['bC']['p'][0],s['bC']['p'][1],s['bC']['m'][0],s['bC']['m'][1],sol),
                    p['bC']['t'] == s['bA']['t'] + 1,p['con']['t'] ==␣
↪s['con']['t'] + 1 ,

                    s_con_bA, s_con_bB, s_con_bC)




    return Or(init_on,timed)
```

[89]:
```
#função para gerar o traco de execução
def gera_traco(declare,init,trans,k):
    s = Solver()
    state =[declare(i) for i in range(k)]
    s.add(init(state[0]))
    for i in range(k-1):
        s.add(trans(state[i],state[i+1],s))


    if s.check()==sat:
        m=s.model()
        for i in range(k):
            print("\n\n----------------------------\n\nEstado -> ",i, end =␣
↪"\n\n----------------------------\n\n")
            for x in ['bA','bB','bC','con']:
```

```
                if (x == 'bA'):
                    print("Barco A:")
                if (x == 'bB'):
                    print("\nBarco B:")
                if (x == 'bC'):
                    print("\nBarco C:")
                if (x == 'con'):
                    print("\nControlador:")
                    for y in state[i][x]:
                        if (y != 't'):
                            for z in state[i][x][y]:
                                if (z == 'm'):
                                    print("Modo do barco ",y )
                                    print("Rota ",m[state[i][x][y][z][0]])
                                    print("Velocidade ",m[state[i][x][y][z][1]])
                                if (z == 's'):
                                    print("Setor ",y )
                                    print("Linha ",m[state[i][x][y][z][0]])
                                    print("Coluna ",m[state[i][x][y][z][1]])
                                if (z == 'tempoTrans'):
                                    print("Tempo Transição␣
↪",m[state[i][x][y][z]] )
                        else:
                            print("Tempo ",m[state[i][x][y]])



                if ( x != 'con'):
                    for y in state[i][x]:
                        if (y == 'm'):
                            print("Modo: ")
                            print("Rota ",m[state[i][x][y][0]])
                            print("Velocidade ",m[state[i][x][y][1]])
                        if (y == 'p'):
                            print("Posição: ")
                            print("X ",m[state[i][x][y][0]].as_decimal(3))
                            print("Y ",m[state[i][x][y][1]].as_decimal(3))
                        if (y == 't'):
                            print("Tempo: ",m[state[i][x][y]])
                        if (y == 's'):
                            print("Setor: ")
                            print("Linha ",m[state[i][x][y][0]])
                            print("Coluna ",m[state[i][x][y][1]])
                        if (y == 'tempoTrans'):
                            print("Tempo de Transição: ",m[state[i][x][y]])
```

```
gera_traco(declare,init,trans,3)
```

```
----------------------------

Estado ->  0

----------------------------

Barco A:
Modo:
Rota  150
Velocidade  HIGHT
Setor:
Linha  286
Coluna  840
Posição:
X  1138
Y  1339
Tempo:  0

Barco B:
Modo:
Rota  150
Velocidade  HIGHT
Setor:
Linha  671
Coluna  191
Posição:
X  369
Y  216
Tempo:  0

Barco C:
Modo:
Rota  75
Velocidade  HIGHT
Setor:
Linha  918
Coluna  981
```

```
Posição:
X  1262
Y  1750
Tempo:  0

Controlador:
Modo do barco  bA
Rota  150
Velocidade  HIGHT
Setor  bA
Linha  286
Coluna  840
Tempo Transição  0
Modo do barco  bB
Rota  150
Velocidade  HIGHT
Setor  bB
Linha  671
Coluna  191
Tempo Transição  0
Modo do barco  bC
Rota  75
Velocidade  HIGHT
Setor  bC
Linha  918
Coluna  981
Tempo Transição  0
Tempo  0


----------------------------

Estado ->  1

----------------------------

Barco A:
Modo:
Rota  150
Velocidade  HIGHT
Setor:
Linha  286
Coluna  840
Posição:
X  1138
Y  1339
Tempo:  0
```

```
Barco B:
Modo:
Rota  150
Velocidade  HIGHT
Setor:
Linha  671
Coluna  191
Posição:
X  369
Y  216
Tempo:  0

Barco C:
Modo:
Rota  75
Velocidade  HIGHT
Setor:
Linha  918
Coluna  981
Posição:
X  1262
Y  1750
Tempo:  0

Controlador:
Modo do barco  bA
Rota  150
Velocidade  HIGHT
Setor  bA
Linha  286
Coluna  840
Tempo Transição  0
Modo do barco  bB
Rota  150
Velocidade  HIGHT
Setor  bB
Linha  671
Coluna  191
Tempo Transição  0
Modo do barco  bC
Rota  75
Velocidade  HIGHT
Setor  bC
Linha  918
Coluna  981
Tempo Transição  0
Tempo  0
```

```
----------------------------

Estado ->  2

----------------------------

Barco A:
Modo:
Rota  150
Velocidade  HIGHT
Setor:
Linha  285
Coluna  839
Posição:
X  1144.992?
Y  1130.851?
Tempo:  1

Barco B:
Modo:
Rota  150
Velocidade  HIGHT
Setor:
Linha  670
Coluna  190
Posição:
X  375.992?
Y  361.851?
Tempo:  1

Barco C:
Modo:
Rota  75
Velocidade  HIGHT
Setor:
Linha  917
Coluna  980
Posição:
X  1271.217?
Y  1258.122?
Tempo:  1

Controlador:
Modo do barco  bA
Rota  150
Velocidade  HIGHT
Setor  bA
```

```
Linha  286
Coluna  840
Tempo Transição  1
Modo do barco  bB
Rota  150
Velocidade  HIGHT
Setor  bB
Linha  671
Coluna  191
Tempo Transição  1
Modo do barco  bC
Rota  75
Velocidade  HIGHT
Setor  bC
Linha  918
Coluna  981
Tempo Transição  1
Tempo  1
```

Vamos agora tentar encontrar um caso em que houve colisão entre barcos, caso não exista uma contra-exemplo é possível afirmar que não ocorrem colisões entre barcos.

[90]:
```
def bmc_always(declare,init,trans,inv,K):
    for k in range(1,K+1):
        s = Solver()
        state =[declare(i) for i in range(k)]
        s.add(init(state[0]))
        for i in range(k-1):
            s.add(trans(state[i],state[i+1],s))
        s.add(inv(state[k-1]))
        if s.check()==sat:
            m=s.model()
            for i in range(k):
                print(i)
                for x in state[i]:
                    if (x == 'm'):
                        for xx in range(3):
                            print("Barco",xx)
                            for y in range(2):
                                print("Modo",xx,y,"=",m[state[i][x][xx][y]])

                    elif (x=='p'):

                        for xx in range(3):
                            print("Barco",xx)
                            for y in range(3):
                                print("Posição",xx,y,"=",m[state[i][x][xx][y]].
↪as_decimal(2))
```

```
                    else:
                        print("Estado=",m[state[i][x]],"\n")
        else:
            return print ("Property is valid up to traces of length "+str(K))
```

Vamos verificar que os barcos não estão nunca no mesmo setor.

[91]:
```
def colisao(s):
    d_bA=Or(danger(s,'bA','bB',0),danger(s,'bA','bC',0))
    d_bB=Or(danger(s,'bB','bA',0),danger(s,'bB','bC',0))
    d_bC=Or(danger(s,'bC','bA',0),danger(s,'bC','bB',0))
    return Or(d_bA , d_bB , d_bC)



bmc_always(declare,init,trans,colisao,10)
```

```
Property is valid up to traces of length 10
```

[ ]:
```
```