Trabalho 3

Grupo 5
Filipe Barbosa a77252
Hugo Ferreira a78555

In [1]:
```python
from z3 import *
import math
```

Problema 2

Pretende-se construir um autómato híbrido que modele uma situação definida por 3 navios a navegar num lago infinito. Cada navio é caracterizado pela sua posição no plano (x,y) , a sua rota medida num ângulo com o eixo horizontal em unidades de 15^o, e uma velocidade que assume apenas 2 valores: $1\,\mathtt{m/s}$ ("low") e $10\,\mathtt{m/s}$ ("hight").

Criamos a matrix ang que contem os senos e cossenos predefinidos. Se seguida definimos o coeficiente de proximidade $r = 1$

In [2]:
```python
# Matriz com os angulos predifinidos
sen = []
cos = []
ang = []
for i in range(24):
    sen.append(math.sin(math.radians(15*i)))
    cos.append(math.cos(math.radians(15*i)))

ang.append(sen)
ang.append(cos)

# coeficiente de proximidade
r = 1
```

A função "declare" cria a i-ésima cópia das variáveis de estado, agrupadas num dicionário que nos permite aceder às mesmas pelo nome. Declaramos de seguida os vários modos, $INIT$, $HIGH$ e LOW.

In [3]:
```python
#modos init, e das velocidades
Mode, (INIT,HIGH,LOW) = EnumSort('Mode', ('INIT','HIGH','LOW'))

#declara a i-esima copia
def declare(i,x):
    s = {}
    s['t'] = Real('t'+str(i)+str(x))
    s['m'] = Const('m'+str(i)+str(x),Mode)
    s['x'] = Real('x'+str(i)+str(x))
    s['y'] = Real('y'+str(i)+str(x))
    s['v'] = Int('v'+str(i)+str(x))
    s['rota'] = Int('rota'+str(i)+str(x))
    return s
```

Define-se então a função $init$ que declara o estado inicial do programa. O tempo começa no 0, o modo inicial é o $INIT$ a velocidade e rota encontram-se a 0.

$$t = 0 \wedge m = INIT \wedge v = 0 \wedge rota = 0$$

In [4]:
```python
#declara estado inicial
def init(s):
    return And(s['t'] == 0, s['m'] == INIT, s['v'] == 0, s['rota'] == 0)
```

As transições untimed são caracterizadas pelo seguinte predicado:

$$(m = INIT \wedge m' = HIGH \wedge x' = x \wedge y' = y \wedge t' = t \wedge v' = 10 \wedge rota' = rota)$$
$$\vee$$
$$(m = HIGH \wedge m' = LOW \wedge x' = x \wedge y' = y \wedge t' = t \wedge v = 10 \wedge v' = 1$$
$$\wedge ((rota' = rota + 1 \wedge rota' \geq 0) \vee (rota' = rota - 1 \wedge rota' > 0)) \wedge x' \leq x + r \wedge x \leq x' + r \wedge y' \leq y + r \wedge y$$
$$\leq y' + r \wedge t' \leq t + \tfrac{r}{v} \wedge t \leq t' + \tfrac{r}{v})$$
$$\vee$$
$$(m = LOW \wedge m' = HIGH \wedge x' = x \wedge y' = y \wedge t' = t \wedge v = 10 \wedge v' = 1 \wedge rota' = rota \wedge x' > x + r \wedge x > x' + r$$
$$\wedge y' > y + r \wedge y > y' + r \wedge t' > t + \tfrac{r}{v} \wedge t > t' + \tfrac{r}{v})$$

As transições timed são caracterizadas pelo seguinte predicado:

$$(m = LOW \wedge m' = LOW \wedge x' = x + 1 * (t' - t) * cos \wedge y' = y + 1 * (t' - t) * sen \wedge t' > t \wedge \wedge v = 1 \wedge v' = v$$
$$\wedge rota' = rota)$$
$$\vee$$
$$(m = HIGH \wedge m' = HIGH \wedge x' = x + 10 * (t' - t) * cos \wedge y' = y + 10 * (t' - t) * sen \wedge t' > t \wedge v = 10 \wedge v' = v$$
$$\wedge rota' = rota)$$

In [5]:
```python
#funcao com as transicoes
def trans(curr,prox,sen,cos,x,y,t,v):
    # untimed
    t1_ih = And(curr['m'] == INIT, prox['m'] == HIGH, prox['x'] == curr['x'], prox['y'] == curr['y'], prox['t'] == curr['t'],                    prox['v'] == 10, prox['rota'] == curr['rota'])
    t2_hl = And(curr['m'] == HIGH, prox['m'] == LOW,  prox['x'] == curr['x'], prox['y'] == curr['y'], prox['t'] == curr['t'], curr['v'] == 10, prox['v'] == 1,  Or(And(prox['rota'] == curr['rota'] + 1, curr['rota'] >= 0), And(prox['rota'] == curr['rota'] - 1, curr['rota'] > 0)), prox['x'] <= x + r, x <= prox['x'] + r, prox['y'] <= y + r, y <= prox['y'] + r, prox['t'] <= t + r / v, t <= prox['t'] + r / v)
    t3_lh = And(curr['m'] == LOW,  prox['m'] == HIGH, prox['x'] == curr['x'], prox['y'] == curr['y'], prox['t'] == curr['t'], curr['v'] == 1,  prox['v'] == 10, prox['rota'] == curr['rota'],
       prox['x'] > x + r, x > prox['x'] + r, prox['y'] > y + r, y > prox['y'] + r, prox['t'] > r / v + t, t > prox['t'] + r / v)
    # timed
    t4_ll = And(curr['m'] == LOW,  prox['m'] == curr['m'], prox['t'] > curr['t'], curr['v'] == 1,  prox['v'] == curr['v'], prox['x'] == curr['x'] + 1 * (prox['t'] - curr['t']) * cos,  prox['y'] == curr['y'] + 1 * (prox['t'] - curr['t']) * sen,  prox['rota'] == curr['rota'])
    t5_hh = And(curr['m'] == HIGH, prox['m'] == curr['m'], prox['t'] > curr['t'], curr['v'] == 10, prox['v'] == curr['v'], prox['x'] == curr['x'] + 10 * (prox['t'] - curr['t']) * cos, prox['y'] == curr['y'] + 10 * (prox['t'] - curr['t']) * sen, prox['rota'] == curr['rota'])
    return Or(t1_ih,t2_hl,t3_lh,t4_ll,t5_hh)
```

Criamos agora a função de ordem superior $gera\_traco$ que, para três estados (um para cada navio), gera uma cópia das variáveis do estado, um predicado que define o estado inicial, outro que adiciona pares de transições entre pares de estados e um número n que define o tamanho do traço.

In [6]:
```python
#funcao que gera o traco
def gera_traco(declare,init,trans,k):
    s = Solver()
    state1 = [declare(i,1) for i in range(k)]
    state2 = [declare(i,2) for i in range(k)]
    state3 = [declare(i,3) for i in range(k)]
    s.add(init(state1[0]))
    s.add(init(state2[0]))
    s.add(init(state3[0]))
    for i in range(k-1):
        s.add(trans(state1[i],state1[i+1],ang[0][i],ang[1][i],state2[i]['x'],state2[i]['y'],state2[i]['t'],(state1[i]['v'] + state2[i]['v'])/2))
        s.add(trans(state2[i],state2[i+1],ang[0][i],ang[1][i],state3[i]['x'],state3[i]['y'],state3[i]['t'],(state2[i]['v'] + state3[i]['v'])/2))
        s.add(trans(state3[i],state3[i+1],ang[0][i],ang[1][i],state1[i]['x'],state1[i]['y'],state1[i]['t'],(state3[i]['v'] + state1[i]['v'])/2))
    if s.check() == sat:
        m = s.model()
        for i in range(k):
            print("Transicao",i+1)
            print("Navio1")
            for x in state1[i]:
                if state1[i][x].sort() != RealSort():
                    print(x,'=',m[state1[i][x]])
                else:
                    print(x,'=',float(m[state1[i][x]].numerator_as_long())/float(m[state1[i][x]].denominator_as_long()))
            print("Navio2")
            for x in state2[i]:
                if state2[i][x].sort() != RealSort():
                    print(x,'=',m[state2[i][x]])
                else:
                    print(x,'=',float(m[state2[i][x]].numerator_as_long())/float(m[state2[i][x]].denominator_as_long()))
            print("Navio3")
            for x in state3[i]:
                if state3[i][x].sort() != RealSort():
                    print(x,'=',m[state3[i][x]])
                else:
                    print(x,'=',float(m[state3[i][x]].numerator_as_long())/float(m[state3[i][x]].denominator_as_long()))
            print("\n")

gera_traco(declare,init,trans,10)
```

```
Transicao 1
Navio1
t = 0.0
m = INIT
x = -2.3147842970973125
y = -2.3781146681832848
v = 0
rota = 0
Navio2
t = 0.0
m = INIT
x = -3.2206984454883774
y = -3.605258007165
v = 0
rota = 0
Navio3
t = 0.0
m = INIT
x = -1.3147842970973125
y = -1.378114668183285
v = 0
rota = 0


Transicao 2
Navio1
t = 0.0
m = HIGH
x = -2.3147842970973125
y = -2.3781146681832848
v = 10
rota = 0
Navio2
t = 0.0
m = HIGH
x = -3.2206984454883774
y = -3.605258007165
v = 10
rota = 0
Navio3
t = 0.0
m = HIGH
x = -1.3147842970973125
y = -1.378114668183285
v = 10
rota = 0


Transicao 3
Navio1
t = 0.16710772781411617
m = HIGH
x = -0.7006475964159236
y = -1.94560804276227
v = 10
rota = 0
Navio2
t = 0.07292809448516281
m = HIGH
x = -2.5162671462356956
y = -3.416506209407037
v = 10
rota = 0
Navio3
t = 0.0
m = LOW
x = -1.3147842970973125
y = -1.378114668183285
v = 1
rota = 1


Transicao 4
Navio1
t = 0.24003582229927894
m = HIGH
x = -0.06907177167849542
y = -1.5809675703364563
v = 10
rota = 0
Navio2
t = 0.24003582229927894
m = HIGH
x = -1.0690717716784954
y = -2.5809675703364565
v = 10
rota = 0
Navio3
t = 0.24003582229927894
m = LOW
x = -1.1069071771678496
y = -1.2580967570336457
v = 1
rota = 1


Transicao 5
Navio1
t = 0.31296391678444174
m = HIGH
x = 0.44660772981622343
y = -1.0652880688417374
v = 10
rota = 0
Navio2
t = 0.31296391678444174
m = HIGH
x = -0.5533922701837766
y = -2.065288068841737
v = 10
rota = 0
Navio3
t = 0.31296391678444174
m = LOW
x = -1.0553392270183777
y = -1.2065288068841737
v = 1
rota = 1


Transicao 6
Navio1
t = 0.3858920112696046
m = HIGH
x = 0.8112482022420375
y = -0.4337122441043093
v = 10
rota = 0
Navio2
t = 0.3858920112696046
m = HIGH
x = -0.18875179775796253
y = -1.4337122441043093
v = 10
rota = 0
Navio3
t = 0.3858920112696046
m = LOW
x = -1.0188751797757964
y = -1.143371224410431
v = 1
rota = 1


Transicao 7
Navio1
t = 0.45882010575476734
m = HIGH
x = 1.0
y = 0.27071905514837197
v = 10
rota = 0
Navio2
t = 0.45882010575476734
m = HIGH
x = -4.465557873958519e-17
y = -0.7292809448516281
v = 10
rota = 0
Navio3
t = 0.45882010575476734
m = LOW
x = -1.0
y = -1.0729280944851627
v = 1
rota = 1


Transicao 8
Navio1
t = 0.45882010575476734
m = LOW
x = 1.0
y = 0.27071905514837197
v = 1
rota = 1
Navio2
t = 0.5317482002399301
m = HIGH
x = 0.0
y = 0.0
v = 10
rota = 0
Navio3
t = 0.5317482002399301
m = LOW
x = -1.0
y = -1.0
v = 1
rota = 1


Transicao 9
Navio1
t = 1.4588201057547674
m = LOW
x = 0.7411809548974791
y = 1.2366448814374402
v = 1
rota = 1
Navio2
t = 0.5317482002399301
m = LOW
x = 0.0
y = 0.0
v = 1
rota = 1
Navio3
t = 1.53174820023993
m = LOW
x = -1.2588190451025207
y = -0.0340741737109317
v = 1
rota = 1


Transicao 10
Navio1
t = 2.458820105754767
m = LOW
x = 0.24118095489747932
y = 2.102670285221879
v = 1
rota = 1
Navio2
t = 1.53174820023993
m = LOW
x = -0.4999999999999998
y = 0.8660254037844387
v = 1
rota = 1
Navio3
t = 2.53174820023993
m = LOW
x = -1.7588190451025205
y = 0.831951230073507
v = 1
rota = 1
```

Agora para testar se não há colisões de navios em k iterações criamos o bmc_always. Adicionamos a propriedade e verificamos se ela é válida.

In [9]:
```python
#funcao para testar invariantes
def bmc_always(declare,init,trans,inv,k):
    s = Solver()
    state1 = [declare(i,1) for i in range(k)]
    state2 = [declare(i,2) for i in range(k)]
    state3 = [declare(i,3) for i in range(k)]
    s.add(init(state1[0]))
    s.add(init(state2[0]))
    s.add(init(state3[0]))
    for i in range(k-1):
        s.add(trans(state1[i],state1[i+1],sen[i],cos[i],state2[i]['x'],state2[i]['y'],state2[i]['t'],(state1[i]['v'] + state2[i]['v'])/2))
        s.add(trans(state2[i],state2[i+1],sen[i],cos[i],state3[i]['x'],state3[i]['y'],state3[i]['t'],(state2[i]['v'] + state3[i]['v'])/2))
        s.add(trans(state3[i],state3[i+1],sen[i],cos[i],state1[i]['x'],state1[i]['y'],state1[i]['t'],(state3[i]['v'] + state1[i]['v'])/2))
    s.add(And([Not(inv(state1[i],state2[i])) for i in range(k)]))
    s.add(And([Not(inv(state2[i],state3[i])) for i in range(k)]))
    s.add(And([Not(inv(state3[i],state1[i])) for i in range(k)]))
    if s.check() == sat:
        m = s.model()
        for i in range(k):
            print("Transicao",i)
            print("Navio1")
            for x in state1[i]:
                if state1[i][x].sort() != RealSort():
                    print(x,'=',m[state1[i][x]])
                else:
                    print(x,'=',float(m[state1[i][x]].numerator_as_long())/float(m[state1[i][x]].denominator_as_long()))
            print("Navio2")
            for x in state2[i]:
                if state2[i][x].sort() != RealSort():
                    print(x,'=',m[state2[i][x]])
                else:
                    print(x,'=',float(m[state2[i][x]].numerator_as_long())/float(m[state2[i][x]].denominator_as_long()))
            print("Navio3")
            for x in state3[i]:
                if state3[i][x].sort() != RealSort():
                    print(x,'=',m[state3[i][x]])
                else:
                    print(x,'=',float(m[state3[i][x]].numerator_as_long())/float(m[state3[i][x]].denominator_as_long()))
            print("\n")
        print ("Nao há colisões para "+str(k)+" estados")
        return

    print ("Há colisões para "+str(k)+" estados")
```

Para não haver colisões entre navios as coordenadas x e y não podem ser iguais entre dois navios no mesmo tempo.

In [10]:
```python
#invariante
def colide(s1,s2):
    return Or(And(s1['x']==s2['x'],s1['t']==s2['t']),And(s1['y']==s2['y'],s1['t']==s2['t']))

bmc_always(declare,init,trans,colide,7)
```

```
Transicao 0
Navio1
t = 0.0
m = INIT
x = -0.8155575109965235
y = -0.4901674386892491
v = 0
rota = 0
Navio2
t = 0.0
m = INIT
x = 0.15100509197431786
y = -0.4901674386892491
v = 0
rota = 0
Navio3
t = 0.0
m = INIT
x = -0.8489949080256821
y = -1.4901674386892492
v = 0
rota = 0


Transicao 1
Navio1
t = 0.0
m = HIGH
x = -0.8155575109965235
y = -0.4901674386892491
v = 10
rota = 0
Navio2
t = 0.0
m = HIGH
x = 0.15100509197431786
y = -0.4901674386892491
v = 10
rota = 0
Navio3
t = 0.0
m = HIGH
x = -0.8489949080256821
y = -1.4901674386892492
v = 10
rota = 0


Transicao 2
Navio1
t = 0.03343739702915857
m = HIGH
x = -0.4925770574530673
y = -0.4036250869912422
v = 10
rota = 0
Navio2
t = 0.0
m = LOW
x = 0.15100509197431786
y = -0.4901674386892491
v = 1
rota = 1
Navio3
t = 0.03343739702915857
m = HIGH
x = -0.5260144544822258
y = -1.4036250869912423
v = 10
rota = 0


Transicao 3
Navio1
t = 0.06687479405831714
m = HIGH
x = -0.20300070481629087
y = -0.23643810184544944
v = 10
rota = 0
Navio2
t = 0.03343739702915857
m = LOW
x = 0.1799627272379955
y = -0.4734487401746698
v = 1
rota = 1
Navio3
t = 0.06687479405831714
m = HIGH
x = -0.23643810184544944
y = -1.2364381018454496
v = 10
rota = 0


Transicao 4
Navio1
t = 0.1003121910874757
m = HIGH
x = 0.03343739702915857
y = 0.0
v = 10
rota = 0
Navio2
t = 0.1396953626437567
m = LOW
x = 0.25509845527916486
y = -0.39831301213350045
v = 1
rota = 1
Navio3
t = 0.1003121910874757
m = HIGH
x = 0.0
y = -1.0
v = 10
rota = 0


Transicao 5
Navio1
t = 0.1731327596729153
m = HIGH
x = 0.3975402399563565
y = 0.6306446231301771
v = 10
rota = 0
Navio2
t = 0.1731327596729153
m = LOW
x = 0.27181715379374416
y = -0.36935537686982284
v = 1
rota = 1
Navio3
t = 0.1731327596729153
m = HIGH
x = 0.36410284292719797
y = -0.36935537686982284
v = 10
rota = 0


Transicao 6
Navio1
t = 0.1731327596729153
m = LOW
x = 0.3975402399563565
y = 0.6306446231301771
v = 1
rota = 1
Navio2
t = 0.5942933725051565
m = LOW
x = 0.38082154144177727
y = 0.03745453608057009
v = 1
rota = 1
Navio3
t = 0.1731327596729153
m = LOW
x = 0.36410284292719797
y = -0.36935537686982284
v = 1
rota = 1


Nao há colisões para 7 estados
```

In [ ]: