TP4

Daniel Francisco Texeira Andrade - A100057

Pedro André Ferreira Malainho - A100050

Problema 1

Enunciado

No contexto do sistema de travagem ABS (“Anti-Lock Breaking System”), pretende-se construir um autómato híbrido que descreva o sistema e que possa ser usado para verificar as suas propriedades dinâmicas.

A componente discreta do autómato contém os modos: `Start`, `Free`, `Stopping`, `Blocked`, e `Stopped`. No modo `Free` não existe qualquer força de travagem; no modo `Stopping` aplica-se a força de travagem alta; no modo `Blocked` as rodas estão bloqueadas em relação ao corpo mas o veículo desloca-se; no modo `Stopped` o veículo está imobilizado.

A componente contínua do autómato usa variáveis contínuas V, v para descrever a `velocidade do corpo` do veículo em relação ao solo e a `velocidade linear das rodas` também em relação ao solo. Assume-se que o sistema de travagem exerce uma força de atrito nos travões proporcional à diferença das duas velocidades. A dinâmica contínua está descrita abaixo no bloco Equaçoes de Fluxo.

a) Defina um autómato híbrido que descreva a dinâmica do sistema segundo as notas abaixo indicadas e com os “switchs” por si escolhidos. Os “switchs” (“jumps”) são uma componente de projeto deste trabalho; cabe ao aluno definir quais devem ser estas condições de modo a que o sistema tenha um comportamento desejável: imobilize-se depressa e não “derrape” muito.
b) Modele em lógica temporal linear LT propriedades que caracterizam o comportamento desejável do sistema. Nomeadamente \

1. “o veículo imobiliza-se completamente em menos de t segundos”
2. “a velocidade V diminui sempre com o tempo”.

c) Construa o FOTS que que descreve a discretização do modelo que definiu em a. e codifique em SMT’s
d) Codifique a verificação das propriedades temporais que definiu em b.

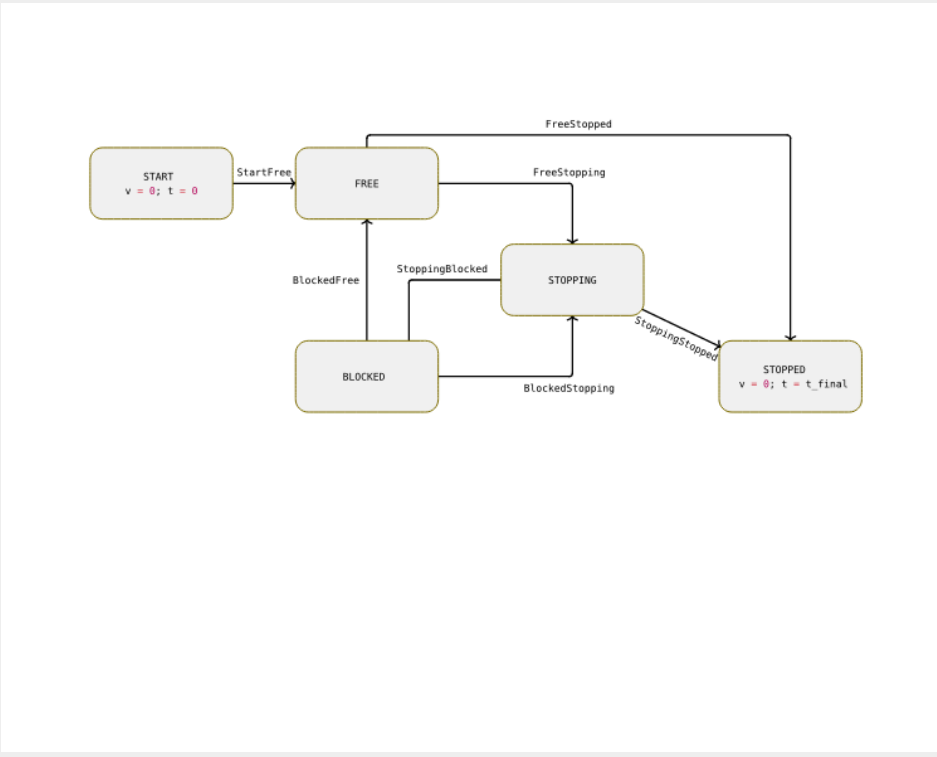
Implementação

Imports

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

Alinea a)

Autómato Híbrido



Alinea c)

In [2]:
```
START = Int(0)
FREE = Int(1) # Não existe força de travagem
STOPPING = Int(2)
BLOCKED = Int(3) # Rodas bloqueadas mas veículo desloca-se
STOPPED = Int(4) # Veículo imobilizado

MODE = {0 : 'START', 1 : 'FREE', 2 : 'STOPPING', 3 : 'BLOCKED', 4 : 'STOPPED'}

a = 0.02 # Atrito
P = 1000 # Peso
Vi = 20 # Velocidade Inicial
T = 0.2 # Max Segundos (Free ou Blocked)
```

Declararação das variáveis do FOTS associadas ao veículo, onde t representa o tempo, m representa o estado do veículo, V a velocidade do corpo em relação ao solo e V_r a velocidade linear das rodas em relação ao solo.

In [3]:
```
def declare(i):
    s = {}
    s['t'] = Symbol('t'+str(i), REAL)
    s['m'] = Symbol('m'+str(i), INT)
    s['V'] = Symbol('V'+str(i), REAL)
    s['v'] = Symbol('v'+str(i), REAL)
    s['cr'] = Symbol('cr'+str(i), REAL)
    return s
```

O estado inicial do FOTS é o seguinte :

$$m = \text{START} \wedge v = V_0 \wedge V_0 + 0.1 = V \wedge t = 0 \wedge V_0 >= 0 \wedge cr = 0$$

Iniciamos no estado `Start`, com $V = (V_0) + 0.1$. Isso é necessário para evitar erros em uma transição temporizada, já que a operação $V - v$ resulta em erro caso $V = v$. A variavél v representa a velocidade inicial e tem valor V_0, que pela Equação de Fluxo número 7 é o "input" do problema. Além disso, cr é usada para registrar a diferença de tempo.

In [4]:
```
def init(s):
    return And(
        Equals(s['m'],START),
        Equals(s['V'],Real(Vi)+0.1),
        s['V'] >= 0.0,
        Equals(Real(Vi),s['v']),
        Equals(s['t'], Real(0)),
        Equals(s['cr'], Real(0))
    )
```

In [5]:
```
def trans(curr, prox):
    # Untimed

    startFree = And(
        Equals(curr['m'], START),
        Equals(prox['m'], FREE),
        Equals(curr['t'], prox['t']),
        curr['t'] >= 0,
        prox['t'] >= 0,
        Equals(curr['v'], prox['v']),
        Equals(curr['V'], prox['V']),
        Equals(curr['cr'], prox['cr'])
    )
    freeStopping = And(
        Equals(curr['m'], FREE),
        Equals(prox['m'], STOPPING),
        Equals(curr['v'], prox['v']),
        Equals(prox['cr'], Real(0)),
        Equals(curr['V'], prox['V']),
        Equals(prox['t'], curr['t']),
        Equals(curr['cr'], Real(T)),
        curr['t'] >= 0,
        prox['t'] >= 0
    )
    freeStopping1 = And(
        Equals(curr['m'], FREE),
        Equals(prox['m'], STOPPING),
```

```
        Equals(prox['cr'], Real(0)),
        Equals(Real(0), prox['v']),
        Equals(Real(0), prox['V']),
        Equals(prox['t'], curr['t']),
        curr['V'] < 0.52
    )
    stoppingBlocked = And(
        Equals(curr['m'], STOPPING),
        Equals(prox['m'], BLOCKED),
        Equals(curr['v'], prox['v']),
        Equals(curr['t'], prox['t']),
        Equals(prox['cr'], Real(0)),
        Equals(curr['v'], Real(0)),
        Equals(curr['V'], prox['V'])
    )
    blockedFree = And(
        Equals(curr['m'], BLOCKED),
        Equals(prox['m'], FREE),
        Equals(curr['v'], prox['v']),
        Equals(curr['t'], prox['t']),
        Equals(prox['cr'], Real(0)),
        Equals(curr['V'] + 0.1, prox['V']),
        Equals(curr['cr'], Real(T)),
        curr['t'] >= 0,
        prox['t'] >= 0
    )
    stoppingStopped = And(
        Equals(curr['m'], STOPPING),
        Equals(prox['m'], STOPPED),
        Equals(curr['t'], prox['t']),
        Equals(curr['v'], prox['v']),
        Equals(curr['V'], prox['V']),
        Equals(curr['V'], Real(0)),
        Equals(curr['v'], Real(0)),
        Equals(prox['cr'], Real(0))
    )
    blockedStopping = And(
        Equals(curr['m'], BLOCKED),
        Equals(prox['m'], STOPPING),
        Equals(curr['t'], prox['t']),
        curr['V'] < 0.52,
        Equals(prox['cr'], Real(0)),
        (Equals(prox['V'], Real(0))),
        (Equals(prox['v'], Real(0)))
    )

    # Timed - Switchs
    freeFree = And(
        Equals(curr['m'], FREE),
        Equals(prox['m'], FREE),
        curr['t'] < prox['t'],
        curr['V'] > prox['V'],
        curr['v'] > prox['v'],
        prox['v'] >= 0,
        prox['V'] >= 0,
        Equals(Real(T), prox['t'] - curr['t']),
        Equals(curr['cr'], Real(0)),
        Equals(prox['cr'], Real(T)),
        cond(curr['V'], curr['v'], curr['t'], prox['V'], prox['v'], prox['t'], R
    )
```

```
    blockedBlocked = And(
        Equals(curr['m'], BLOCKED),
        Equals(prox['m'], BLOCKED),
        Equals(curr['cr'], Real(0)),
        Equals(Real(T),prox['t'] - curr['t']),
        curr['t'] < prox['t'],
        curr['V'] > prox['V'],
        prox['v'] >= 0,
        prox['V'] >= 0,
        Equals(prox['V'], prox['v']),
        Equals(prox['cr'], Real(T)),
        Equals(prox['V'] - curr['V'], (-a * P) * (prox['t'] - curr['t']))
    )
    stoppingStopping = And(
        Equals(curr['m'], STOPPING),
        Equals(prox['m'], STOPPING),
        curr['V'] > prox['V'],
        Equals(prox['v'], Real(0)),
        prox['V'] >= 0,
        curr['v'] > prox['v'],
        curr['t'] < prox['t'],
        cond(curr['V'], curr['v'], curr['t'], prox['V'], prox['v'], prox['t'], R
    )

    return Or(stoppingStopped, freeStopping1, startFree, freeStopping, freeFree,
              blockedBlocked, blockedFree)


def cond(V0, v0, t0, V, v, t, c):
    return And(Equals(V - V0, (-c * (V0 - v0)) * (t - t0)), Equals(v - v0, (-a *
```

In [6]:

```
def gera_traco(declare, init, trans, k):
    with Solver(name="z3") as s:

        trace = [declare(i) for i in range(k)]

        s.add_assertion(init(trace[0]))

        for i in range(k-1):
            s.add_assertion(trans(trace[i], trace[i+1]))

        if s.solve():
            '''
            m = s.get_model()
            for n, v in m:
                print(f'{n} = {v}')
            '''
            for i in range(k):
                print(f'Passo {i}')
                for v in trace[i]:
                    print(f'{v} = {s.get_value(trace[i][v])}')

            print(f'---------')

gera_traco(declare, init, trans, 16)
```

```
Passo 0
t = 0.0
m = 0
V = 724178820081175757/36028797018963968
v = 20.0
cr = 0.0
Passo 1
t = 0.0
m = 1
V = 724178820081175757/36028797018963968
v = 20.0
cr = 0.0
Passo 2
t = 3602879701896397/18014398509481984
m = 1
V = 469972158513223789722941500462478146902840015488811/233840261972944466912589573234605283144949206876160
v = 374191187209105730769017421842989987433192700344533/233840261972944466912589573234605283144949206876160
cr = 3602879701896397/18014398509481984
Passo 3
t = 3602879701896397/18014398509481984
m = 2
V = 469972158513223789722941500462478146902840015488811/233840261972944466912589573234605283144949206876160
v = 374191187209105730769017421842989987433192700344533/233840261972944466912589573234605283144949206876160
cr = 0.0
Passo 4
t = 19339127992847167345434764926020163216938811736142432440810720129451243527540022323212732066523618931435676649805138605617774218706944
m = 2
V = 22374396738401652293178604557040289069378616889058153149779648952297/2487055080044646425464133047237862871353299610277211235548437413888
v = 0.0
cr = 0.0
Passo 5
t = 19339127992847167345434764926020163216938811736142432440810720129451243527540022323212732066523618931435676649805138605617774218706944
m = 3
V = 22374396738401652293178604557040289069378616889058153149779648952297/2487055080044646425464133047237862871353299610277211235548437413888
v = 0.0
cr = 0.0
Passo 6
t = 109130915364459069544790943210422170234788029807683836758305246376162176377001116160636603326180946571783832490256930280888710935347 2
m = 3
V = 12426176418223066039084909692783205749303439987480813506378332369217/2487055080044646425464133047237862871353299610277211235548437413888
v = 12426176418223066039084909692783205749303439987480813506378332369217/2487055080044646425464133047237862871353299610277211235548437413888
cr = 3602879701896397/18014398509481984
Passo 7
t = 109130915364459069544790943210422170234788029807683836758305246376162176377001116160636603326180946571783832490256930280888710935347 2
m = 1
V = 63374409631137653477186260321948164161526597050101234987316826418971243527540022323212732066523618931435676649805138605617774218706944
v = 12426176418223066039084909692783205749303439987480813506378332369217/24870550
```

800446464254641330472378628713532996102772112355484374138888
cr = 0.0
Passo 8
t = 24313238152936460472481612358148704876976400186931102262511378420 99/124352754
002232321273206652361893143567664980513860561777421870 6944
m = 1
V = 82232403189116043321767953457032279122536033846868706957499214319829711507608
382053610077514847761 35/161419103488986273814291889815358237525030439100454580726
269052129077438611496834143963831227488665 6
v = 16115138400027261904356958896749207590313281775522902194675831826167218959082
204947798639854235405 37/161419103488986273814291889815358237525030439100454580726
269052129077438611496834143963831227488665 6
cr = 3602879701896397/18014398509481984
Passo 9
t = 24313238152936460472481612358148704876976400186931102262511378420 99/124352754
002232321273206652361893143567664980513860561777421870 6944
m = 2
V = 82232403189116043321767953457032279122536033846868706957499214319829711507608
382053610077514847761 35/161419103488986273814291889815358237525030439100454580726
269052129077438611496834143963831227488665 6
v = 16115138400027261904356958896749207590313281775522902194675831826167218959082
204947798639854235405 37/161419103488986273814291889815358237525030439100454580726
269052129077438611496834143963831227488665 6
cr = 0.0
Passo 10
t = 17509103379217141916269236986430725117258095664418122968748375497015716920792
481023320459966947179381522074745384651 1/858402650513162923455570714354514743181 1
922253733965967698743369283454877933249308110288419738973135334438338979430 4
m = 2
V = 75569050945176038870112253658354300152597118407437582224310110859514518437118
184372883562729147377082557486792026599 1/171680530102632584691114142870902948636 2
384450746793193539748673856690975586649861622057683947794627066887667795886 08
v = 0.0
cr = 0.0
Passo 11
t = 17509103379217141916269236986430725117258095664418122968748375497015716920792
481023320459966947179381522074745384651 1/858402650513162923455570714354514743181 1
922253733965967698743369283454877933249308110288419738973135334438338979430 4
m = 3
V = 75569050945176038870112253658354300152597118407437582224310110859514518437118
184372883562729147377082557486792026599 1/171680530102632584691114142870902948636 2
384450746793193539748673856690975586649861622057683947794627066887667795886 08
v = 0.0
cr = 0.0
Passo 12
t = 48064771700608669646205542787593117501253438425600205387739527999952959649562
001300893659284664077616853389950142117/214600662628290730863892678588628685795 29
805634334914919246858423208637194833123270275721049347432838336095847448576
m = 3
V = 68968389041230011815930485140934248228659301745592147704126827408958408758254
044914034128524093104737862374051387711/171680530102632584691114142870902948636 23
844507467931935397486738566909755866498616220576839477946270668876677958860 8
v = 68968389041230011815930485140934248228659301745592147704126827408958408758254
044914034128524093104737862374051387711/171680530102632584691114142870902948636 23
844507467931935397486738566909755866498616220576839477946270668876677958860 8
cr = 3602879701896397/18014398509481984
Passo 13
t = 48064771700608669646205542787593117501253438425600205387739527999952959649562
001300893659284664077616853389950142117/214600662628290730863892678588628685795 29
805634334914919246858423208637194833123270275721049347432838336095847448576

```
m = 1
V = 215341105128733178095150716067498667652730247009529348919002961096532695371443099711010414983194053475137965995274172/429201325256581461727785357172573715905961126866982983849371684641727438966624654055144209869486567667219169489715 2
v = 689683890412300118159304851409342482286593017455921477041268274089584087582540449140341285240931047378623740513877111/171680530102632584691114142870902948636238445074679319353974867385669097558664986162205768394779462706688766779588608
cr = 0.0
Passo 14
t = 480647717006086696462055427875931175012534384256002053877395279999529596495620013008936592846640776168533899501421172/214600662628290730863892678588628685795298056343349149192468584232086371948331232702757210493474328383360958474485762
m = 2
V = 0.0
v = 0.0
cr = 0.0
Passo 15
t = 480647717006086696462055427875931175012534384256002053877395279999529596495620013008936592846640776168533899501421172/214600662628290730863892678588628685795298056343349149192468584232086371948331232702757210493474328383360958474485762
m = 4
V = 0.0
v = 0.0
cr = 0.0
---------
```

1. ”O veículo imobiliza-se completamente em menos de t segundos”

Sendo T o número de segundos que queremos limitar sendo S o conjunto dos estados .

$$(s_t >= T \wedge s_V = 0) \vee (s_t < T)$$

In [7]:
```python
def termina_t(s, tempo):
    return Or(
        And(
            s['t'] >= tempo,
            Equals(s['V'], Real(0))
        ),
        s['t'] < tempo
    )

def bmc_always1(declare, init, trans, inv, t, K):

    for k in range(1, K + 1):
        with Solver(name="z3") as s:
            trace = [declare(i) for i in range(k)]

            s.add_assertion(init(trace[0]))

            for i in range(k-1):
                s.add_assertion(trans(trace[i], trace[i+1]))

            for i in range(k):
                s.add_assertion(Not(And(inv(trace[i], t))))
            if s.solve():
                for i in range(k):
```

```
                    print(f'Passo {i}')
                    for v in trace[i]:
                        print(f'{v} = {s.get_value(trace[i][v])}')
                    print(f'---------')
                print(f'Tem erro')
                return
    print(f'Erro não encontrado')

bmc_always1(declare, init, trans, termina_t, 5, 10)
```

```
Erro não encontrado
```

2.“A velocidade V diminui sempre com o tempo”

Sendo S o conjunto dos estados .

$$\forall_{i \in length(S)-1} \quad se \quad s_{i,t} < s_{i+1,t} \quad então \quad s_{i,V} > s_{i+1,V}$$

In [8]:

```
p = declare(1)

def veldiminui(c, p):
    return Or(
        And(
            c['t'] < p['t'],
            p['V'] < c['V']
        ),
        Equals(p['t'], c['t'])
    )

def bmc_always2(declare, init, trans, inv, K):

    for k in range(2, K + 1):
        with Solver(name="z3") as s:
            trace = [declare(i) for i in range(k)]
            s.add_assertion(init(trace[0]))

            for i in range(k-1):
                s.add_assertion(trans(trace[i], trace[i+1]))
                s.add_assertion(Not(inv(trace[i], trace[i+1])))

            if s.solve():
                for i in range(k):
                    print(f'Passo {i}')
                    for v in trace[i]:
                        print(f'{v} = {s.get_value(trace[i][v])}')
                    print(f'---------')
                print(f'Tem erro')
                return
    print(f'Erro não encontrado')

bmc_always2(declare, init, trans, veldiminui, 10)
```

```
Erro não encontrado
```