TP4 - Exercício 1

Grupo 1

- Diogo Coelho da Silva A100092
- Pedro Miguel Ramôa Oliveira A97686

1. Definição do autómato híbrido

Começamos por modelar e definir o autómato híbrido que é responsável pelo comportamento do sistema de travagem ABS pedido. Do enunciado sabemos que:

- o sistema começa com uma velocidade inicial $V_0 > 0$, ou seja, o sistema só começa se o carro estiver em movimento e precisar de travar.
- As componentes `FREE` e `BLOCKED` são relativos às rodas e `STOPPING` e `STOPPED` relativos à velocidade do veículo.

title

2. Importar as bibliotecas importantes

In [13]:
```
from pysmt.shortcuts import *
import pysmt.typing as types
import random as rn
from pysmt.typing import BOOL, REAL, INT, BVType, STRING
from IPython.display import Latex
import itertools
```

- `pywraplp` : Importa o solver de programação linear da biblioteca OR-Tools para resolver o problema de otimização.
- `random` : Gera aleatoriedade para simular a disponibilidade de colaboradores e equipas dos projetos.
- `pandas` : Biblioteca usada para organizar os dados em formato de tabela (DataFrame), facilitando a manipulação e visualização.

3. Definição de constantes

Nesta parte, são definidos os valores fixos que representam características físicas e limites do sistema. Estes valores são utilizados como parâmetros no modelo de simulação do veículo para garantir que o comportamento do sistema seja realista e

consistente.

1. `atrito_restante_modos` e `atrito_modo_bloqueado`

- `atrito_restante_modos` aplica-se aos modos `FREE` e `STOPPING` (travagem).
- atrito_modo_bloqueado aplica-se ao modo `BLOCKED`, que simula o veículo travado.

2. `proporcionalidade_travagem_stopping` e proporcionalidade_travagem`

- `proporcionalidade_travagem_stopping` é utilizada no modo `STOPPING`.
- `proporcionalidade_travagem é aplicada` aos modos `FREE` e `BLOCKED`.

3. `atrito_corpo_ar`

- Representa o atrito devido à resistência do ar, que age como uma força de desaceleração para velocidade.

4. `peso_veiculo`

Define o peso do veículo.

5. `limite_timer` e `delta_tempo`

- `limite_timer`: Define o tempo máximo permitido para o timer nos modos `FREE` e `BLOCKED`.
- `delta_tempo`: Representa o incremento de tempo entre atualizações do sistema.

6. `velocidade_inicial`

Define a velocidade inicial do veículo.

7. `margem_erro`

Estabelece uma tolerância para considerar velocidades próximas de 0 ou 0.

In [14]:
```
# Constantes que representam características físicas e limites do sistema
atrito_restante_modos = 0.01
atrito_modo_bloqueado = 0.009
proporcionalidade_travagem_stopping = 10.0
proporcionalidade_travagem = 0.5
atrito_corpo_ar = 0.01
peso_veiculo = 1000.0
limite_timer = 0.1
delta_tempo = 0.1
```

```
velocidade_inicial = 20.0
margem_erro = 0.2
```

4 . Definição dos Modos do Sistema

- `MODO_INICIAL` : Representa o estado inicial

- `MODO_LIVRE` : Representa o estado em que o veículo está a andar - `MODO_TRAVAGEM` : Representa o estado em que o veículo está a travar - `MODO_BLOQUEADO` : Representa o estado em que o ABS está ligado para evitar bloqueio das rodas - `MODO_PARADO` : Representa o estado em que o veículo está parado

In [15]:
```
MODO_INICIAL = Int(0)
MODO_LIVRE = Int(1)
MODO_TRAVAGEM = Int(2)
MODO_BLOQUEADO = Int(3)
MODO_PARADO = Int(4)
```

5. Definição do FOTS

1. Declaração das Variáveis (Função declare)

Esta função cria um dicionário com as variáveis necessárias para representar o estado do sistema em num dado instante. As variáveis são:

- `tempo` : O tempo decorrido no sistema.
- `velocidade` : A velocidade do veículo.
- `velocidade_roda` : A velocidade da roda do veículo.
- `modo` : O estado discreto do veículo.
- `temporizador` : Um contador interno para medir o tempo decorrido dentro do modo atual

2. Estado Inicial (Função init)

Define as condições iniciais para o sistema:

- `tempo` é 0 no início.
- `velocidade` e velocidade_roda são iguais a uma velocidade inicial fornecida.
- O `modo` começa no estado `MODO_INICIAL` .

O `temporizador` é inicializado a 0.

Estas condições asseguram que o sistema começa num estado bem definido, com todas as variáveis ajustadas para o ponto de partida.

3. Transições:

Transições Discretas

1. MODO_INICIAL → MODO_LIVRE

Condições: Sempre que o sistema inicia.
Velocidades: $V' = V \land v' = v$
Temporizador: $timer' = 0$
O sistema transita automaticamente do estado inicial para o modo livre, mantendo as velocidades constantes e faz reset ao timer.

2. MODO_LIVRE → MODO_TRAVAGEM

Condição: $timer \geq \tau \lor V \leq erro \land v \leq erro$
Velocidades: $V' = V \land v' = v$
Temporizador: $timer' = 0$
O sistema entra em travagem quando o limite de tempo é atingido ou as velocidades são suficientemente pequenas, indicando que o veículo está quase parado.

3. MODO_TRAVAGEM → MODO_PARADO

Condição: $V \leq erro \land v \leq erro$
Velocidades: $V' = 0 \land v' = 0$
Temporizador: $timer' = 0$
Quando ambas as velocidades são suficientemente próximas de zero, o veículo transita para o estado parado.

4. MODO_TRAVAGEM → MODO_BLOQUEADO

Condição: $V - v \leq erro \land (V \geq erro \lor \geq erro)$
Velocidades: $V' = V \land v' = v$
Temporizador: $timer' = 0$
Se as velocidades do veículo e da roda estão próximas, mas ainda consideravelmente altas, ocorre o bloqueio, simulando a travagem das rodas.

5. MODO_BLOQUEADO → MODO_LIVRE

Condição: timer} \geq \tau \lor (V \leq erro \land v \leq erro)
Velocidades: $V' = V \land v' = v * *Temporizador$: **timer' = 0$
O sistema retorna ao estado livre após atingir o limite de tempo ou quando as velocidades ficam suficientemente pequenas.

Transições Contínuas

6. MODO_LIVRE → MODO_LIVRE

Equações de Atualização:

- $V' = V + (-ktravagem \cdot (V - v) - far) \cdot \Delta t$
- $v' = v + (-fmodos \cdot m + k_travagem \cdot (V - v)) \cdot \Delta t$
- $timer' = timer + \Delta t$

Atualiza as velocidades e o timer com base na dinâmica do sistema, incluindo os efeitos de travagem proporcional, atrito do ar e forças adicionais.

7. MODO_TRAVAGEM → MODO_TRAVAGEM

Equações de Atualização:

- $V' = V + (-ktravagemstop \cdot (V - v) - far) \cdot \Delta t$
- $v' = v + (-f_modos \cdot m + ktravagemstop \cdot (V - v)) \cdot \Delta t$
- $timer' = 0$

Ajusta as velocidades durante o modo de travagem, considerando uma dinâmica modificada para parar o veículo.

8. MODO_PARADO → MODO_PARADO

Condições: Sempre **Velocidades:** $V' = 0 \land v' = 0$
Temporizador:$timer' = 0$
No estado parado, as velocidades e o temporizador permanecem constantes em zero.

9. MODO_BLOQUEADO → MODO_BLOQUEADO

Equações de Atualização:

- $V' = V + (-fmodobloqueado \cdot m - far) \cdot \Delta t\)
- $ v' = V'
- $timer' = timer+\Deltat

As velocidades do veículo e da roda convergem, sendo ajustadas para permanecerem iguais, enquanto o temporizador continua a contar.

In [16]:
```
def declare(indice):
    estado = {}
    estado['tempo'] = Symbol(f'tempo{indice}', REAL)
    estado['velocidade'] = Symbol(f'velocidade{indice}', REAL)
    estado['velocidade_roda'] = Symbol(f'velocidade_roda{indice}', REAL)
    estado['modo'] = Symbol(f'modo{indice}', INT)
    estado['temporizador'] = Symbol(f'temporizador{indice}', REAL)
    return estado

def init(estado, velocidade_inicial):
```

```
    condicao_tempo = Equals(estado['tempo'], Real(0))
    condicao_velocidade = Equals(estado['velocidade'], Real(velocidade_in
    condicao_velocidade_roda = Equals(estado['velocidade_roda'], Real(vel
    condicao_modo = Equals(estado['modo'], MODO_INICIAL)
    condicao_temporizador = Equals(estado['temporizador'], Real(0))
    return And(condicao_tempo, condicao_velocidade, condicao_velocidade_r

def transitions(estado_atual, proximo_estado):
    transicao_inicial_livre = And(
        Equals(estado_atual['modo'], MODO_INICIAL),
        Equals(proximo_estado['modo'], MODO_LIVRE),
        Equals(proximo_estado['velocidade'], estado_atual['velocidade']),
        Equals(proximo_estado['velocidade_roda'], estado_atual['velocidad
        Equals(proximo_estado['tempo'], estado_atual['tempo']),
        Equals(proximo_estado['temporizador'], Real(0))
    )
    transicao_livre_travagem = And(
        Equals(estado_atual['modo'], MODO_LIVRE),
        Equals(proximo_estado['modo'], MODO_TRAVAGEM),
        Equals(proximo_estado['velocidade'], estado_atual['velocidade']),
        Equals(proximo_estado['velocidade_roda'], estado_atual['velocidad
        Equals(proximo_estado['tempo'], estado_atual['tempo']),
        Equals(proximo_estado['temporizador'], Real(0)),
        Or(estado_atual['temporizador'] >= limite_timer, And(estado_atual
    )
    transicao_travagem_parado = And(
        Equals(estado_atual['modo'], MODO_TRAVAGEM),
        Equals(proximo_estado['modo'], MODO_PARADO),
        Equals(proximo_estado['tempo'], estado_atual['tempo']),
        Equals(proximo_estado['temporizador'], Real(0)),
        And(estado_atual['velocidade_roda'] <= margem_erro, estado_atual[
            estado_atual['velocidade_roda'] >= -margem_erro, estado_atual
        Equals(proximo_estado['velocidade'], Real(0)),
        Equals(proximo_estado['velocidade_roda'], Real(0))
    )
    transicao_travagem_bloqueado = And(
        Equals(estado_atual['modo'], MODO_TRAVAGEM),
        Equals(proximo_estado['modo'], MODO_BLOQUEADO),
        Equals(proximo_estado['velocidade'], estado_atual['velocidade']),
        Equals(proximo_estado['tempo'], estado_atual['tempo']),
        Equals(proximo_estado['temporizador'], Real(0)),
        And(estado_atual['velocidade'] - estado_atual['velocidade_roda']
        Equals(proximo_estado['velocidade_roda'], proximo_estado['velocid
        Or(proximo_estado['velocidade'] > margem_erro, proximo_estado['ve
    )
    transicao_bloqueado_livre = And(
        Equals(estado_atual['modo'], MODO_BLOQUEADO),
        Equals(proximo_estado['modo'], MODO_LIVRE),
        Equals(proximo_estado['velocidade'], estado_atual['velocidade']),
        Equals(proximo_estado['velocidade_roda'], estado_atual['velocidad
        Equals(proximo_estado['tempo'], estado_atual['tempo']),
        Equals(proximo_estado['temporizador'], Real(0)),
        Or(estado_atual['temporizador'] >= limite_timer, estado_atual['ve
```

```
    )
    transicao_livre_livre = And(
        Equals(estado_atual['modo'], MODO_LIVRE),
        Equals(proximo_estado['modo'], MODO_LIVRE),
        Equals(proximo_estado['velocidade'], estado_atual['velocidade'] +
        Equals(proximo_estado['velocidade_roda'], estado_atual['velocidad
        Equals(proximo_estado['temporizador'], estado_atual['temporizador
        Equals(proximo_estado['tempo'], estado_atual['tempo'] + delta_tem
        proximo_estado['temporizador'] <= limite_timer,
        Or(proximo_estado['velocidade'] > margem_erro, proximo_estado['ve
    )
    transicao_travagem_travagem = And(
        Equals(estado_atual['modo'], MODO_TRAVAGEM),
        Equals(proximo_estado['modo'], MODO_TRAVAGEM),
        Equals(proximo_estado['temporizador'], Real(0)),
        Equals(proximo_estado['tempo'], estado_atual['tempo'] + delta_tem
        Equals(proximo_estado['velocidade'], estado_atual['velocidade'] +
        Equals(proximo_estado['velocidade_roda'], estado_atual['velocidad
        estado_atual['velocidade'] - estado_atual['velocidade_roda'] > ma
        proximo_estado['velocidade'] <= estado_atual['velocidade'],
        Or(estado_atual['velocidade'] > margem_erro, estado_atual['veloci
    )
    transicao_parado_parado = And(
        Equals(estado_atual['modo'], MODO_PARADO),
        Equals(proximo_estado['modo'], MODO_PARADO),
        Equals(proximo_estado['temporizador'], Real(0)),
        Equals(proximo_estado['tempo'], estado_atual['tempo'] + delta_tem
        Equals(proximo_estado['velocidade'], Real(0)),
        Equals(proximo_estado['velocidade_roda'], Real(0))
    )

    transicao_bloqueado_bloqueado = And(
        Equals(estado_atual['modo'], MODO_BLOQUEADO),
        Equals(proximo_estado['modo'], MODO_BLOQUEADO),
        Equals(proximo_estado['tempo'], estado_atual['tempo'] + delta_tem
        Equals(proximo_estado['temporizador'], estado_atual['temporizador
        And(estado_atual['velocidade'] - estado_atual['velocidade_roda']
        Equals(proximo_estado['velocidade'], estado_atual['velocidade'] +
        Equals(proximo_estado['velocidade_roda'], proximo_estado['velocid
        proximo_estado['temporizador'] <= limite_timer,
        proximo_estado['velocidade'] > 0
    )
    return Or(transicao_inicial_livre, transicao_livre_livre, transicao_l
              transicao_travagem_parado, transicao_travagem_bloqueado, tr
              transicao_bloqueado_livre, transicao_parado_parado)
```

6. Impressão das variáveis e satistabilidade do sistema

In [17]:
```
def imprimir_variaveis(estado, solvedor):
    print("MODO: ", end="")
    modo_valor = solvedor.get_py_value(estado['modo'])
    if modo_valor == 0:
        print("INICIAL")
```

```
    elif modo_valor == 1:
        print("LIVRE")
    elif modo_valor == 2:
        print("TRAVAGEM")
    elif modo_valor == 3:
        print("BLOQUEADO")
    elif modo_valor == 4:
        print("PARADO")
    print("Velocidade: ", end="")
    print(float(solvedor.get_py_value(estado['velocidade'])))
    print("Velocidade da Roda: ", end="")
    print(float(solvedor.get_py_value(estado['velocidade_roda'])))
    print("Tempo: ", end="")
    print(float(solvedor.get_py_value(estado['tempo'])))
    print("Temporizador: ", end="")
    print(float(solvedor.get_py_value(estado['temporizador'])))
    print("")
    print("")

def gerar_traco(declare, init, transitions, k, velocidade_inicial):
    with Solver(name="z3") as solver:
        traco = [declare(i) for i in range(k)]
        solver.add_assertion(init(traco[0], velocidade_inicial))
        for i in range(k - 1):
            solver.add_assertion(transitions(traco[i], traco[i + 1]))

        if solver.solve():
            i = 0
            print("is sat")
            for estado in traco:
                print("ESTADO: ", end="")
                print(i)
                imprimir_variaveis(estado, solver)
                i = i + 1
        else:
            print("unsat")

gerar_traco(declare, init, transitions, 30, 10)
```

```
is sat
ESTADO: 0
MODO: INICIAL
Velocidade: 10.0
Velocidade da Roda: 10.0
Tempo: 0.0
Temporizador: 0.0


ESTADO: 1
MODO: LIVRE
Velocidade: 10.0
Velocidade da Roda: 10.0
Tempo: 0.0
Temporizador: 0.0
```

```
ESTADO: 2
MODO: LIVRE
Velocidade: 9.999
Velocidade da Roda: 9.0
Tempo: 0.1
Temporizador: 0.1


ESTADO: 3
MODO: TRAVAGEM
Velocidade: 9.999
Velocidade da Roda: 9.0
Tempo: 0.1
Temporizador: 0.0


ESTADO: 4
MODO: TRAVAGEM
Velocidade: 8.999
Velocidade da Roda: 8.999
Tempo: 0.2
Temporizador: 0.0


ESTADO: 5
MODO: BLOQUEADO
Velocidade: 8.999
Velocidade da Roda: 8.999
Tempo: 0.2
Temporizador: 0.0


ESTADO: 6
MODO: BLOQUEADO
Velocidade: 8.098
Velocidade da Roda: 8.098
Tempo: 0.30000000000000004
Temporizador: 0.1


ESTADO: 7
MODO: LIVRE
Velocidade: 8.098
Velocidade da Roda: 8.098
Tempo: 0.30000000000000004
Temporizador: 0.0


ESTADO: 8
MODO: LIVRE
Velocidade: 8.097
```

```
Velocidade da Roda: 7.098
Tempo: 0.4
Temporizador: 0.1


ESTADO: 9
MODO: TRAVAGEM
Velocidade: 8.097
Velocidade da Roda: 7.098
Tempo: 0.4
Temporizador: 0.0


ESTADO: 10
MODO: TRAVAGEM
Velocidade: 7.0969999999999995
Velocidade da Roda: 7.0969999999999995
Tempo: 0.5
Temporizador: 0.0


ESTADO: 11
MODO: BLOQUEADO
Velocidade: 7.0969999999999995
Velocidade da Roda: 7.0969999999999995
Tempo: 0.5
Temporizador: 0.0


ESTADO: 12
MODO: BLOQUEADO
Velocidade: 6.196
Velocidade da Roda: 6.196
Tempo: 0.6000000000000001
Temporizador: 0.1


ESTADO: 13
MODO: LIVRE
Velocidade: 6.196
Velocidade da Roda: 6.196
Tempo: 0.6000000000000001
Temporizador: 0.0


ESTADO: 14
MODO: LIVRE
Velocidade: 6.194999999999999
Velocidade da Roda: 5.196
Tempo: 0.7000000000000001
Temporizador: 0.1
```

ESTADO: 15
MODO: TRAVAGEM
Velocidade: 6.194999999999999
Velocidade da Roda: 5.196
Tempo: 0.7000000000000001
Temporizador: 0.0

ESTADO: 16
MODO: TRAVAGEM
Velocidade: 5.194999999999999
Velocidade da Roda: 5.194999999999999
Tempo: 0.8
Temporizador: 0.0

ESTADO: 17
MODO: BLOQUEADO
Velocidade: 5.194999999999999
Velocidade da Roda: 5.194999999999999
Tempo: 0.8
Temporizador: 0.0

ESTADO: 18
MODO: BLOQUEADO
Velocidade: 4.294
Velocidade da Roda: 4.294
Tempo: 0.9
Temporizador: 0.1

ESTADO: 19
MODO: LIVRE
Velocidade: 4.294
Velocidade da Roda: 4.294
Tempo: 0.9
Temporizador: 0.0

ESTADO: 20
MODO: LIVRE
Velocidade: 4.292999999999999
Velocidade da Roda: 3.2939999999999996
Tempo: 1.0
Temporizador: 0.1

ESTADO: 21
MODO: TRAVAGEM
Velocidade: 4.292999999999999
Velocidade da Roda: 3.2939999999999996
Tempo: 1.0

Temporizador: 0.0

ESTADO: 22
MODO: TRAVAGEM
Velocidade: 3.2929999999999997
Velocidade da Roda: 3.2929999999999997
Tempo: 1.1
Temporizador: 0.0

ESTADO: 23
MODO: BLOQUEADO
Velocidade: 3.2929999999999997
Velocidade da Roda: 3.2929999999999997
Tempo: 1.1
Temporizador: 0.0

ESTADO: 24
MODO: BLOQUEADO
Velocidade: 2.3919999999999995
Velocidade da Roda: 2.3919999999999995
Tempo: 1.2000000000000002
Temporizador: 0.1

ESTADO: 25
MODO: LIVRE
Velocidade: 2.3919999999999995
Velocidade da Roda: 2.3919999999999995
Tempo: 1.2000000000000002
Temporizador: 0.0

ESTADO: 26
MODO: LIVRE
Velocidade: 2.3909999999999996
Velocidade da Roda: 1.3919999999999995
Tempo: 1.3
Temporizador: 0.1

ESTADO: 27
MODO: TRAVAGEM
Velocidade: 2.3909999999999996
Velocidade da Roda: 1.3919999999999995
Tempo: 1.3
Temporizador: 0.0

ESTADO: 28
MODO: TRAVAGEM

```
Velocidade: 1.3909999999999993
Velocidade da Roda: 1.3909999999999996
Tempo: 1.4000000000000001
Temporizador: 0.0


ESTADO: 29
MODO: BLOQUEADO
Velocidade: 1.3909999999999993
Velocidade da Roda: 1.3909999999999993
Tempo: 1.4000000000000001
Temporizador: 0.0
```

8. Modelação Lógica Temporal (LT) e provas por indução

Queremos traduzir as seguintes propriedades:

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

- Modelação ponto 1: $F(V = 0)$ após t segundos, ou seja, após t segundos, a velocidade do veículo vai ser igual a 0(parado).

-Modelação ponto 2: $V' < V$

A função `verificacao_bmc_eventualmente_com_tempo` verifica se a propriedade de um veículo estar parado dentro de um limite de tempo é verificada, juntamente com as restrições temporais para garantir que o comportamento ocorre dentro do período estipulado.

Por fim, a função `verificar_velocidade_a_diminuir` utiliza a técnica BMC para garantir que a velocidade de um veículo está sempre a diminuir ao longo de uma sequência de estados, verificando se, em algum ponto, a velocidade aumenta, o que invalidaria a propriedade de diminuição constante de velocidade. Se a velocidade aumentar em algum estado, o erro é identificado e reportado, caso contrário, é confirmada a propriedade para o número de estados verificados.

In [18]:
```
def propriedade_limite_temporizador(estado):
    return Implies(Or(Equals(estado['modo'], MODO_LIVRE), Equals(estado['

def propriedade_paragem(estado):
    return Equals(estado['modo'], MODO_PARADO)

def propriedade_velocidade_diminui_sempre(estado_atual, proximo_estado):

    return Implies(estado_atual['tempo'] <= proximo_estado['tempo'], prox
```

```
def propriedade_paragem_tempo_limite(estado, limite_tempo):
    """
    Define a propriedade de que o veículo está parado dentro de um limite
    """
    return And(Equals(estado['modo'], MODO_PARADO), estado['tempo'] <= Re

def verificacao_bmc_com_tempo(declarar_variaveis, inicializar_estado, def
    with Solver(name="z3") as solver:
        estados = [declarar_variaveis(i) for i in range(limite)]
        solver.add_assertion(inicializar_estado(estados[0], velocidade_in

        for i in range(limite - 1):
            solver.add_assertion(definir_transicao(estados[i], estados[i

        for i in range(limite):
            solver.push()
            solver.add_assertion(Not(propriedade_paragem_tempo_limite(est

            if solver.solve():
                solver.pop()
            else:
                print(f"> Veículo imobiliza-se em menos de {limite_tempo}
                return
        print(f"> Veículo não imobiliza-se em menos de {limite_tempo} seg
        return

def verificar_velocidade_diminui_sempre(declarar_variaveis, inicializar_e
    with Solver(name="z3") as solver:
        estados = [declarar_variaveis(i) for i in range(limite)]
        solver.add_assertion(inicializar_estado(estados[0], velocidade_in

        for i in range(limite - 1):
            solver.add_assertion(definir_transicao(estados[i], estados[i

        for i in range(limite - 1):
            solver.add_assertion(Not(propriedade_velocidade_diminui_sempr
            if solver.solve():
                print(f"Contradição! A propriedade de diminuição constant
                return
        print(f"A propriedade de diminuição constante de velocidade verif


verificacao_bmc_com_tempo(declare, init, transitions, propriedade_paragem
verificar_velocidade_diminui_sempre(declare, init, transitions, 50, 10)
```

```
> Veículo imobiliza-se em menos de 10.0 segundos dentro de 50 estados
A propriedade de diminuição constante de velocidade verifica-se em 50 esta
dos
```

In [ ]: