Relatório de CT-213

Laboratório 1 – Máquina de Estados Finita e Behavior Tree

Autor:
Otávio Henrique Ribas Guimarães

Turma 23

Professor
Marcos Ricardo Omena de Albuquerque Maximo

Data de submissão do relatório: 19/03/2020

Instituto Tecnológico de Aeronáutica - ITA
Departamento de Computação

Para o laboratório, foi implementado o comportamento de um robô *Roomba*, produzido pela empresa *iRobot* e destinado à limpeza de interiores. A implementação foi feita de duas formas distintas: a primeira foi uma Máquina de Estados Finita, enquanto a segunda foi uma *Behavior Tree*.

1 Máquina de Estados Finita

Para a implementação da Máquina de Estados Finita, simplesmente determinou-se as ações a serem executadas em cada um dos quatro estados possíveis: mover em linha reta, mover em espiral, mover para trás e girar. Além disso, foram implementadas as transições entre cada um desses estados, utilizando sempre o método change_state do behavior do agente. Para todos os estados, foi adicionado ao método _init_ um contador n, inicializado ao começo do estado como sendo 0. Esse contador será usado para determinar o tempo de permanência do robô em cada estado, multiplicando-o pela constante SAMPLE_TIME.

1.1 Move Forward State

Execução: Para o estado *Move Foward*, a execução consiste unicamente em alterar a velocidade do robô, usando o método do agente set_velocity. A velocidade utilizada é a constante FORWARD_SPEED, e a velocidade de rotação é escolhida como sendo zero. Ao fim da execução, se soma 1 ao contador n.
Transição: A transição é feita em duas situações. Se o metodo get_bumper_state do agente retorna *True*, então é feita a transição do estado para o Go Back State. Além disso, se o tempo obtido pelo contador n for igual ou superior à constante MOVE_FOWARD_TIME, então é feita a transição para do estado para o Move In Spiral State. Caso nenhum desses casos seja alcançado, a transição não ocorre.

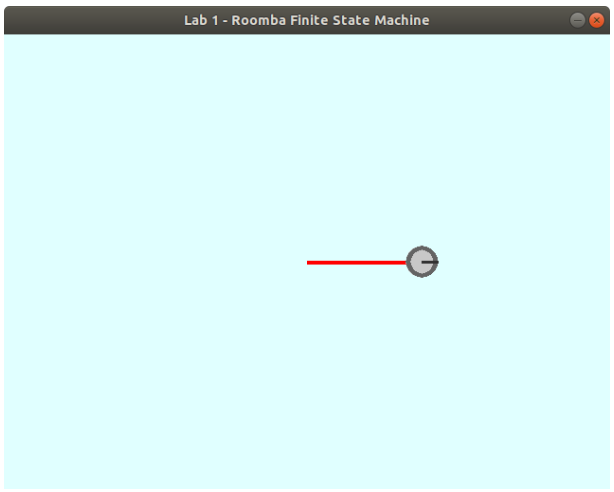

Figura 1: Funcionamento do Move Forward State para a Maquina de Estados Finitos

1.2 Move In Spiral State

Execução: Para o estado *Move In Spiral*, a execução consiste em alterar a velocidade de rotação e de translação do robô, usando o método do agente set_velocity. A velocidade utilizada para a rotação é a constante ANGULAR_SPEED, e a velocidade de translação é determinada a partir do tempo decorrido no estado, a partir da relação $v = \omega \cdot (r_0 + a \cdot t)$, sendo ω=ANGULAR_SPEED, r_0=INITIAL_RADIUS_SPIRAL e a=SPIRAL_FACTOR . Ao fim da execução, se soma 1 ao contador n.
Transição: A transição é feita em duas situações. Se o metodo get_bumper_state do agente retorna *True*, então é feita a transição do estado para o Go Back State. Além disso, se o tempo obtido pelo contador n for igual ou superior à constante MOVE_IN_SPIRAL_TIME, então é feita a transição para do estado para o Move Forward State. Caso nenhum desses casos seja alcançado, a transição não ocorre.

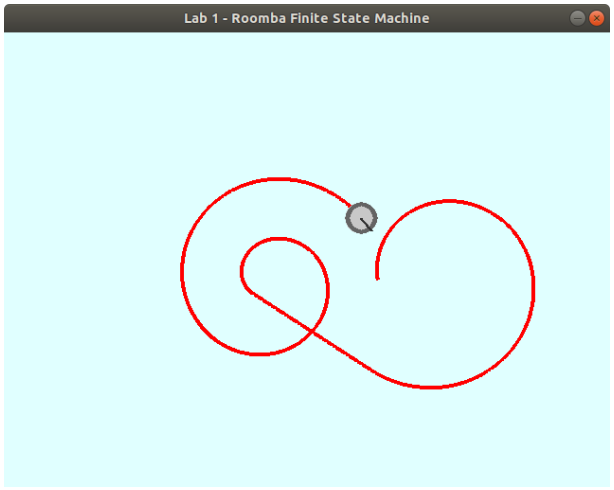

Figura 2: Funcionamento do Move In Spiral State para a Maquina de Estados Finitos

1.3 Go Back State

Execução: Para o estado *Go Back*, a execução consiste unicamente em alterar a velocidade do robô, usando o método do agente set_velocity. A velocidade utilizada é a constante BACKWARD_SPEED, e a velocidade de rotação é escolhida como sendo zero. Ao fim da execução, se soma 1 ao contador n.
Transição: A transição é feita em apenas uma situação. Se o tempo obtido pelo contador n for igual ou superior à constante GO_BACK_TIME, então é feita a transição para do estado para o Rotate State. Caso contrario, a transição não ocorre.

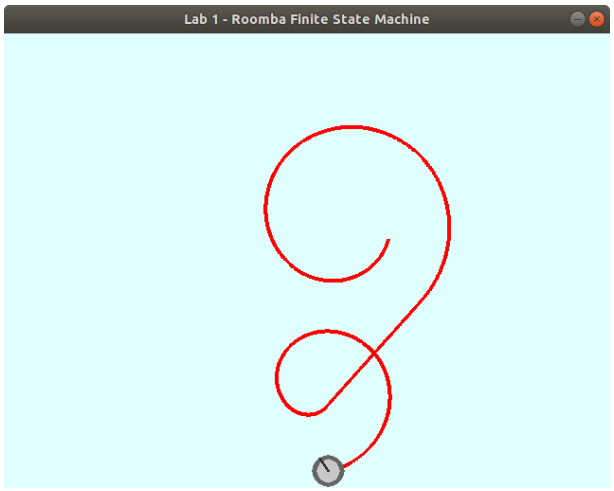

Figura 3: Funcionamento do Go Back State para a Maquina de Estados Finitos

1.4 Rotate State

Inicialização: Ao se iniciar o estado, determina-se como *angle* um valor de ângulo aleatório entre 0 e 2π com o método Uniform da biblioteca Random, além do contador n.
Execução: Para o estado *Go Back*, a execução consiste unicamente em alterar a velocidade angular do robô, usando o método do agente set_velocity. A velocidade utilizada é a constante ANGULAR_SPEED, e a velocidade de translação é escolhida como sendo zero. Ao fim da execução, se soma 1 ao contador n.
Transição: A transição é feita em apenas uma situação. Utilizando-se o contador n, determina-se o tempo de permanência no estado e, assim, multiplicando esse valor pela velocidade de rotação, determina-se o ângulo total de rotação. Se esse valor for igual ou superior ao valor de *angle*, então é feita a transição para do estado para o Move Forward State. Caso contrario, a transição não ocorre.

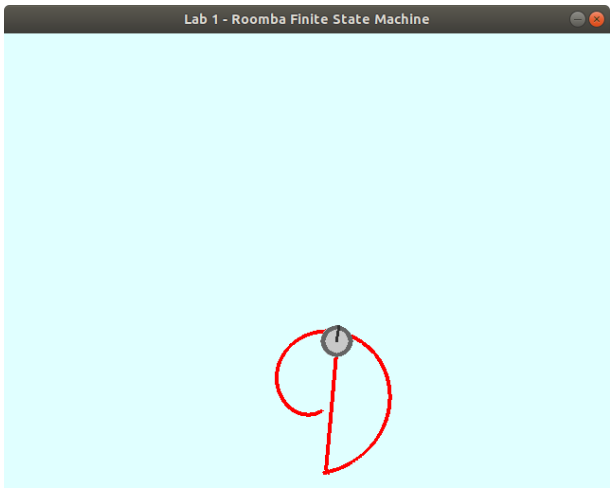

Figura 4: Funcionamento do Rotate State para a Maquina de Estados Finitos

2 Behavior Tree

Para a implementação da Behavior Tree, determinou-se as ações a serem executadas em cada uma das quatro folhas possíveis: mover em linha reta, mover em espiral, mover para trás e girar. Além disso, foram implementadas as condições que fariam cada uma dessas folhas retornar valores de SUCCESS, FAILURE ou RUNNING. Para todas elas, foi adicionado aos métodos _init_ e enter um contador n, inicializado ao início de cada tarefa como sendo 0. Assim como na Maquina de Estados Finita, esse contador será usado para determinar o tempo de permanência do robô em cada tarefa, multiplicando-o pela constante SAMPLE_TIME. Enfim, construi-se a Behavior Tree para o Roomba, utilizando as folhas implementadas e os nós compostos.

2.1 Move Forward Node

A execução do nó consiste unicamente em alterar a velocidade do robô, usando o método do agente set_velocity. A velocidade utilizada é a constante FORWARD_SPEED, e a velocidade de rotação é escolhida como sendo zero. Ao fim da execução, se soma 1 ao contador n.
Em seguida, checamos se o metodo get_bumper_state do agente retorna *True*: nesse caso, o método retorna FAILURE. Checamos, então, se o tempo obtido pelo contador n for igual ou superior à constante MOVE_FOWARD_TIME: nesse caso, retorna-se SUCCESS. Caso nenhum desses casos seja alcançado, retorna-se RUNNING.

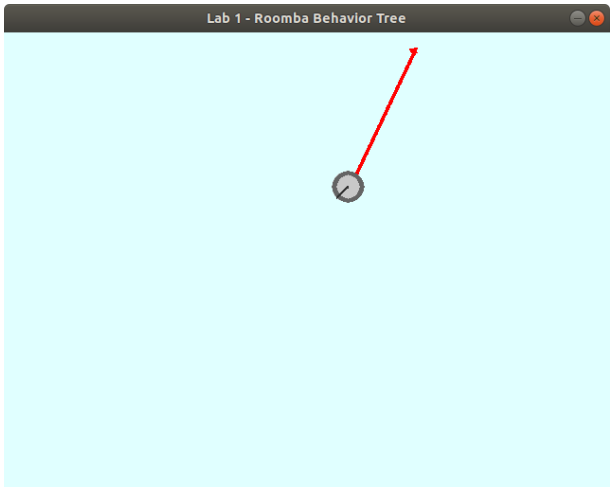

Figura 5: Funcionamento do Move Forward Node para a Behavior Tree

2.2 Move In Spiral Node

A execução do nó consiste em alterar a velocidade de rotação e de translação do robô, usando o método do agente set_velocity. A velocidade utilizada para a rotação é a constante ANGULAR_SPEED, e a velocidade de translação é determinada a partir do tempo decorrido no estado, a partir da relação $v = \omega \cdot (r_0 + a \cdot t)$, sendo ω=ANGULAR_SPEED, r_0=INITIAL_RADIUS_SPIRAL e a=SPIRAL_FACTOR . Ao fim da execução, se soma 1 ao contador n.

Em seguida, checamos se o metodo get_bumper_state do agente retorna *True*: nesse caso, o método retorna FAILURE. Checamos, então, se o tempo obtido pelo contador n for igual ou superior à constante MOVE_IN_SPIRAL_TIME: nesse caso, retorna-se SUCCESS. Caso nenhum desses casos seja alcançado, retorna-se RUNNING.

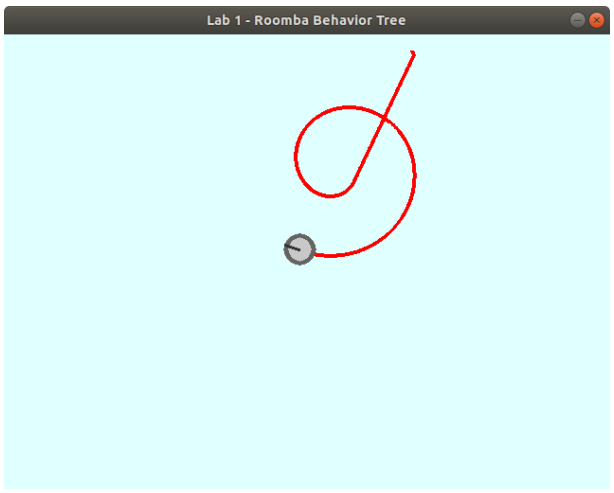

Figura 6: Funcionamento do Move In Spiral Node para a Behavior Tree

2.3 Go Back Node

A execução consiste em alterar a velocidade do robô, usando o método do agente set_velocity. A velocidade utilizada é a constante BACKWARD_SPEED, e a velocidade de rotação é escolhida como sendo zero. Ao fim da execução, se soma 1 ao contador n.
Em seguida, checamos se o tempo obtido pelo contador n for igual ou superior à constante GO_BACK_TIME: nesse caso, o método retorna FAILURE. Caso contrario, retorna-se RUNNING.

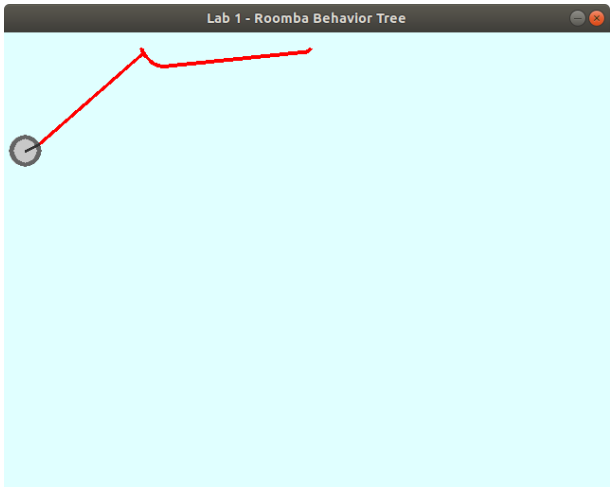

Figura 7: Funcionamento do Go Back Node para a Behavior Tree

2.4 Rotate Node

No método _init_, cria-se, além do contador n, uma variável *angle*. Ao se iniciar a tarefa, inicializa-se *angle* com um valor de ângulo aleatório entre 0 e 2π com o método Uniform da biblioteca Random.
A execução do nó consiste unicamente em alterar a velocidade angular do robô, usando o método do agente set_velocity. A velocidade utilizada é a constante ANGULAR_SPEED, e a velocidade de translação é escolhida como sendo zero. Ao fim da execução, se soma 1 ao contador n.
Em seguida, utilizando o contador n, determina-se o tempo de permanência no estado e, assim, multiplicando esse valor pela velocidade de rotação, determina-se o ângulo total de rotação. Se esse valor for igual ou superior ao valor de *angle*, então o método retorna SUCCESS. Caso contrario, retorna-se RUNNING.

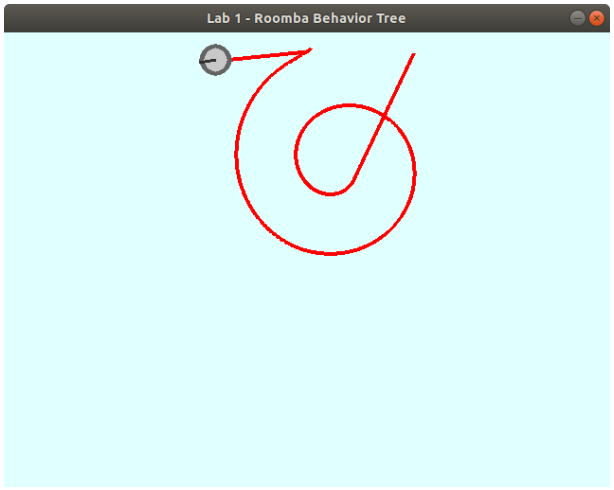

Figura 8: Funcionamento do Rotate Node para a Behavior Tree

2.5 Roomba Behavior Tree

Enfim, construi-se a arquitetura da Behavior Tree para o Roomba, implementada no seu método _init_. A raiz da árvore foi inicializada como um nó composto do tipo Selector. Depois, foram inicializados dois nós compostos do tipo sequence, denotados por $s1$ e $s2$. Utilizando-se o método add_child atribui-se a $s1$ como filhos uma folha do tipo MoveFowardNode e outra do tipo MoveInSpiralNode. Da mesma forma, atribui-se a $s2$ como filhos uma folha do tipo GoBackNode e outra do tipo RotateNode. Enfim, atribui-se à raiz da árvore $s1$ e $s2$ como filhos.