LAB1 - Máquina de Estados Finita e *Behavior Tree*

CT-213 Aprendizado de Máquina para Robótica Móvel
Prof. Marcos Maximo

Rafael Mello Celente

March 22, 2022

1 Implementação

1.1 Máquina de Estados Finita

A implementação do código utilizando uma Máquina de Estados Finita (MEF) se baseou na lógica da figura 1. Cada um dos estados da lógica foi representado por uma classe que possuía 2 funções além da construtora: execute e check_transition.

A função execute executa as atividades de cada estado. Para o estado de andar para frente, por exemplo, a função assinala a velocidade linear do Roomba como uma constante definida e a velocidade angular como nula. Além disso, uma vez que essa função é executada todo passo de simulação, ela também mantém a contagem de vezes que ela mesma foi chamada para calcular o tempo em que o estado está ativo.

Já a função check_transition é responsável por, dependendo do estado em que a simulação está no momento, checar se alguma condição de transição é satisfeita. Caso verdadeiro, é ativada a função change_state da MEF, que

passa o sistema para o próximo estado em que a condição de transição foi acionada.

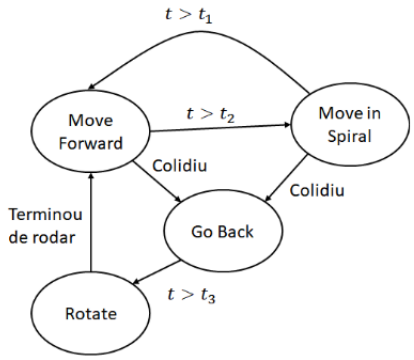

Figure 1: Lógica da Máquina de Estados Finita do comportamento do Roomba.

1.2 *Behavior Tree*

A implementação do código utilizando uma *Behavior Tree* se baseou na lógica da figura 2. Cada um dos nós da árvore foi representado por uma classe que possuía 2 funções além da construtora: execute e enter. Antes da definição dos comportamentos do Roomba, foi necessário construir a árvore a partir de um *root node* que é representado por um *selector node*, dois *sequence nodes* e 4 *leaf nodes*: MoveForwardNode, MoveInSpiralNode, GoBackNode e RotateNode.

Assim como na MEF, a função execute executa as atividades de cada estado. Entretanto, como essa função é executada todo *frame* enquanto o processo ainda não terminar ou falhar, também é responsável por checar as condições de sucesso, falha ou andamento do nó. No nó de andar para frente, por exemplo, a função é responsável por retornar um status de: sucesso, caso o tempo de execução da função ultrapassou um valor de tempo máximo definido a priori; falha, caso o Roomba bata na parede durante a execução do nó; andamento, caso nenhuma das condições anteriores foi satisfeita. Caso o nó retorne sucesso a árvore passa pro nó vizinho. Caso ele retorne falha, a árvore volta para o nó pai.

Já função enter é responsável por assinalar as condições iniciais do nó. Ela é executada apenas no momento que o sistema adentra o nó, *settando* as condições inicias e do sistema. No nó de rotacionar, por exemplo, a função assinala a velocidade linear do Roomba como nula e a velocidade angular

como uma constante definida, além de assinalar um valor de tempo aleatório que o sistema permanecerá rotacionando.

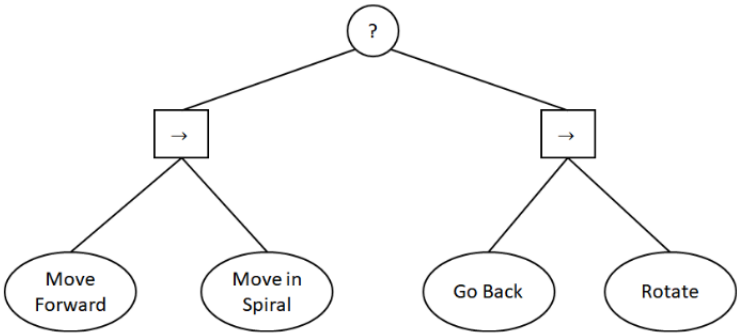

Figure 2: Lógica da *Behavior Tree* do comportamento do Roomba.

2 Figuras da simulação

2.1 Máquina de Estados Finita

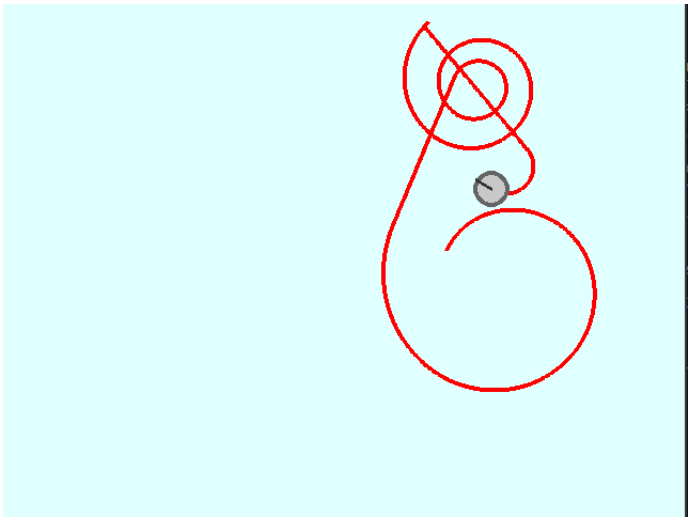

Figure 3: Simulação da trajetória do Roomba funcionando através de uma Máquina de Estados Finita.

2.2 *Behavior Tree*

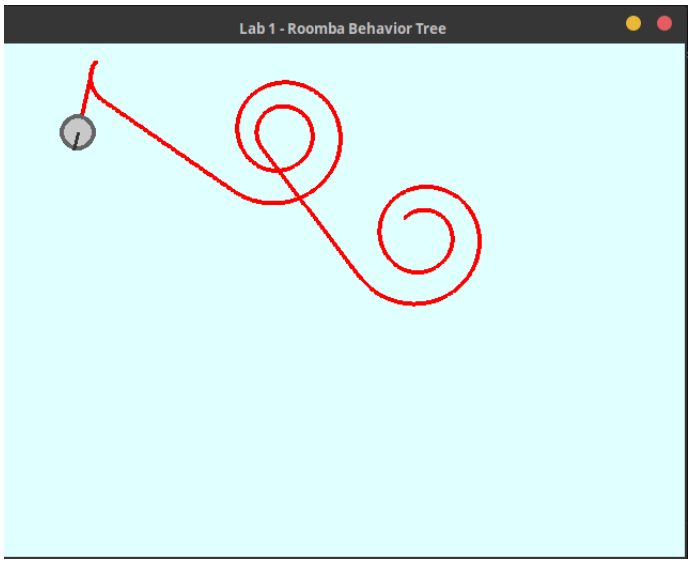

Figure 4: Simulação da trajetória do Roomba funcionando através de uma *Behavior Tree*.