Laboratório 1 - Máquina de Estados Finita e Behavior Tree

Marcelo Buga Martins da Silva

CT-213 - Professor Marcos Ricardo Omena de Albuquerque Máximo

15/03/2021

1 Implementação da Máquina de Estados Finita

Para a implementação da Máquina de Estados Finita, foi necessário alterar 3 funções de cada Estado, o construtor, a função *check_transition* e a função *execute*. Uma breve explicação do que foi feito em cada uma dessas funções nos distintos estados:

1.1 Construtor

Para o construtor de cada estado, apenas foi definida uma variável de tempo com valor inicial 0. Essa variável tem o propósito de definir quando deve acontecer uma mudança de estado baseada no tempo passado.

1.2 *check_transition*

Essa função converte o tempo "contado" na função *execute* para segundos e realiza transições de estado baseadas na passagem de tempo. Por exemplo, quando passado o tempo de girar em espiral, essa função faz o *roomba* voltar a andar reto, como mostra a Figura 1

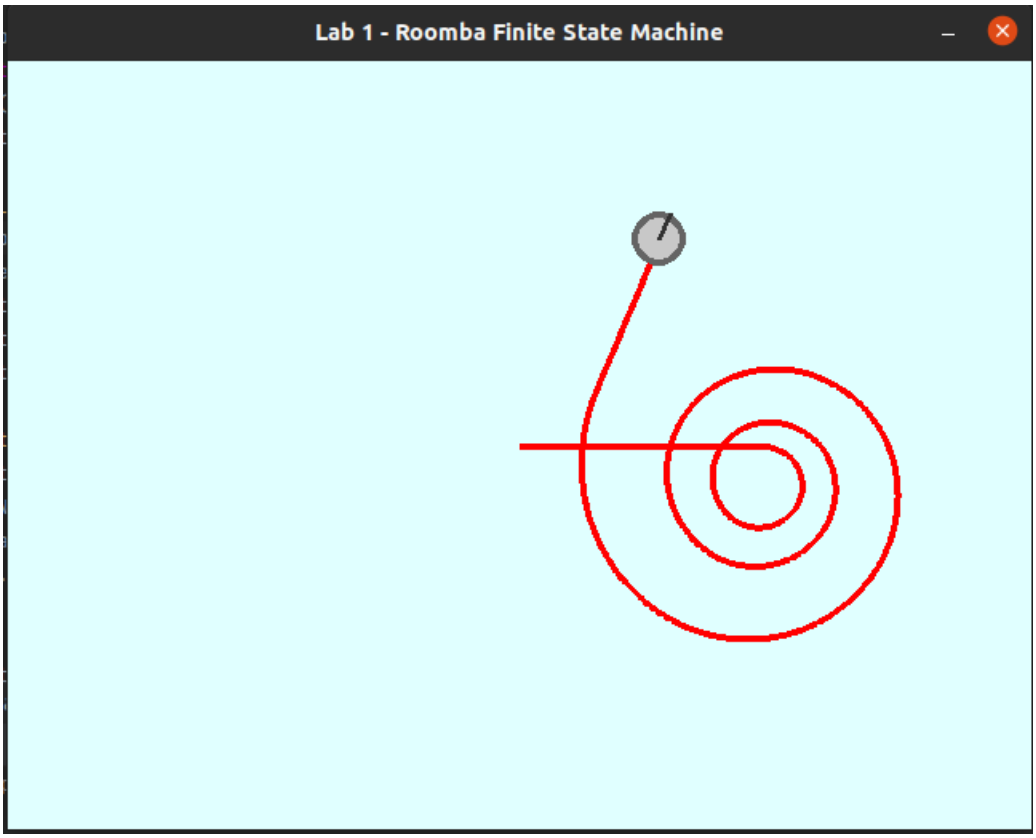

Figura 1: Troca de comportamento de espiral para andar reto

Além disso, nos estados *MoveForwrdState* e *MoveInSpiralState*, essa função é responsável por checar se houve colisão, mudando estados para o robô voltar um pouco para trás e girar em um ângulo aleatório para sair de perto da "parede". Um exemplo desse comportamento é representado pela Figura 2.

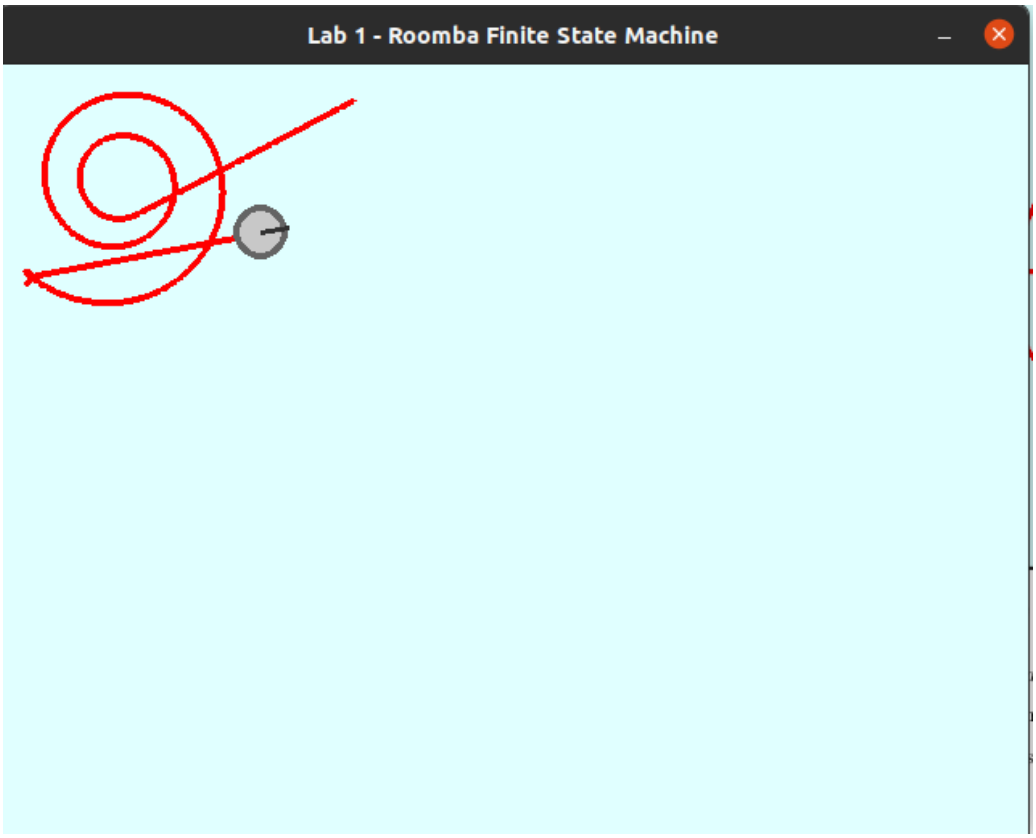

Figura 2: Comportamento após batida na “parede”

1.3 *execute*

Essa função é chamada a todo instante e é a responsável por atualizar o tempo definido no construtor. Além disso, *execute* manda também define a velocidade linear e angular do robô. No estado *MoveInSpiralState*, essa função também calcula o raio da espiral conforme a passagem de tempo para definir o valor adequado da velocidade angular do robô para o movimento em espiral.

2 Implementação da Behavior Tree

Para a implementação da *Behavior Tree*, foi necessário, primeiro, montar a árvore representada pela Figura 3, colocando um nó *Selector* como raiz e dois nós filhos do tipo *Sequence*. As folhas são, então, os nós *MoveForwardNode* e *MoveInSpiralNode* para o primeiro *Selector* e *GoBackNode* e *RotateNode* para o segundo *Selector*.

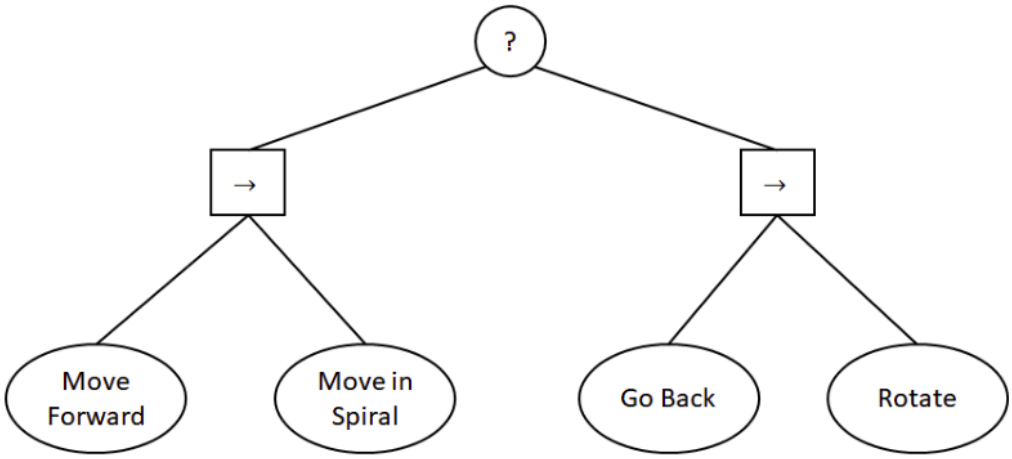

Figura 3: Representação esquemática da *Behavior Tree* do *roomba*

Assim, foi necessário programar o funcionamento das folhas da árvore, sendo utilizadas três funções para isso: o construtor, a função *enter* e a função *execute*.

2.1 Construtor

O construtor da *Behavior Tree* simplesmente cria uma variável de tempo. No caso do *RotateNode* ele também cria uma variável de ângulo aleatório.

2.2 *enter*

Diferentemente do caso da máquina de estados, o nó é criado apenas uma vez. Assim, zerar o valor da variável tempo não pode ser feito pelo construtor. Portanto a função *enter* define zero como o valor da variável tempo. Além disso, essa função foi utilizada para definir a velocidade do robô de um nó quando ela é constante.

2.3 *execute*

A função *execute* faz a passagem de tempo e avalia situações para definir seu retorno. Caso o tempo de execução de um determinado *behavior* tenha passado, a função retornará SUCCESS. Caso ele não tenha passado ainda e o robô não tenha colidido, ela retornará RUNNING. Em casos de colisão, ela retornará FAILURE. Isso permite que o robô funcione de forma análoga ao que ocorre com a máquina de estados. A Figura 4 mostra a rotação do robô após colisão.

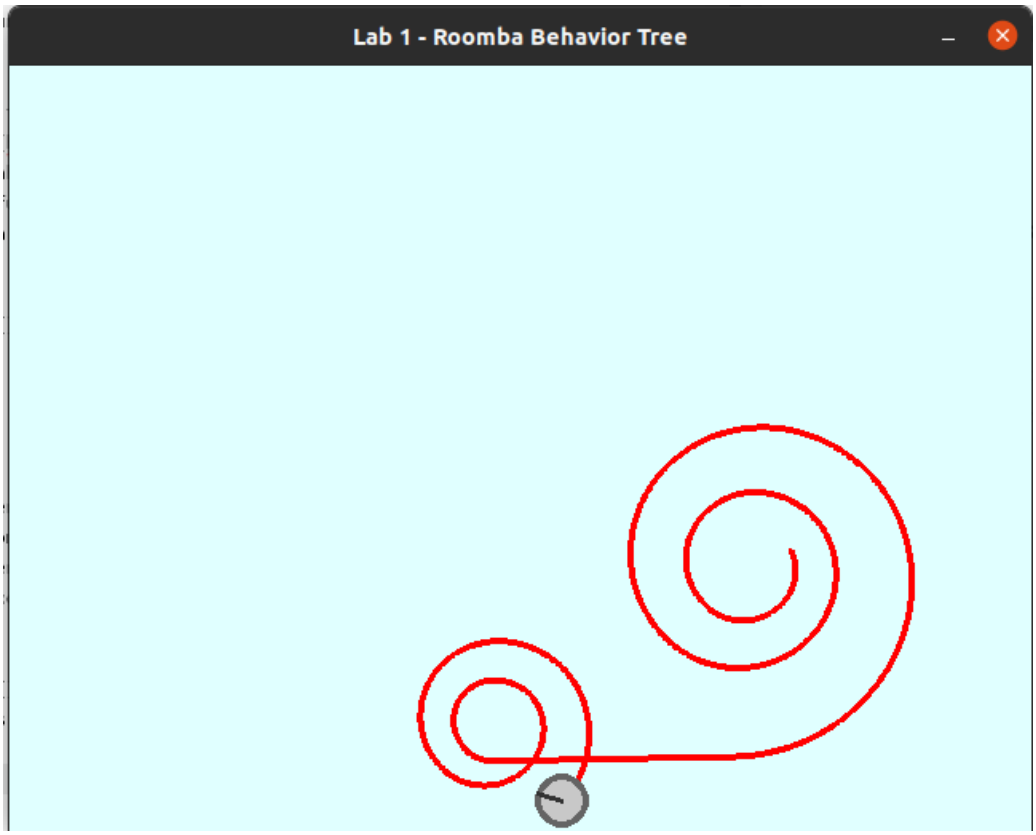

Figura 4: Rotação do *roomba* após colisão

3 Comparação

Nenhuma diferença significativa foi notada entre as duas implementações. Rodando ambas simulações lado a lado, pode-se perceber que são idênticas, como mostra a Figura 5. Porém, caso mais comportamentos fossem adicionados, a implementação via *behavior tree* provavelmente seria mais simples e exigiria menor mudança no código como um todo.

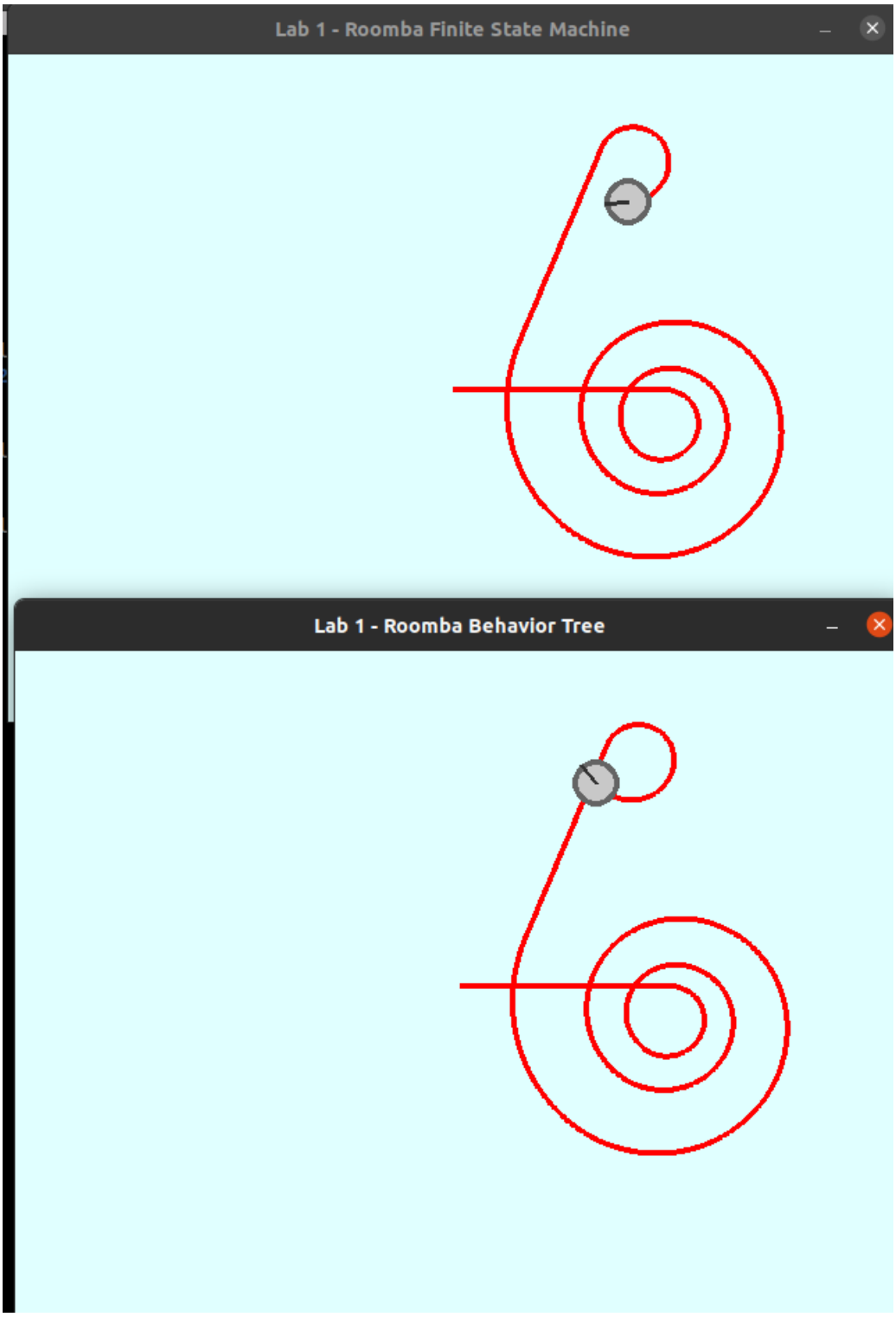

Figura 5: Comparação lado a lado entre as implementações