Máquina de estados finitos

Allan Ribeiro

22 de Março de 2023

1 Introdução

Seja bem vindo à Pato Bots. Somos uma equipe multidisciplinar que aplica conhecimentos técnicos, experiência e valores pessoais na construção de protótipos robóticos para competição.

Como parte da otimização do processo de desenvolvimento dos robôs, estamos interessados em repassar conhecimentos úteis, e noções facilitadoras, auxiliando assim o desenvolvimento dos indivíduos envolvidos no projeto. E um tópico interessantíssimo é o aqui apresentado.

Estejam atentos aos detalhes do tópico, entendam seus princípios e façam uso prático. FSM é um conceito poderoso, e permite o entendimento de outras abordagens futuras que encontrarão no universo dos bits

2 Usos e definições

Podemos imaginar a definição mais simples de programas como um conjunto de funções, que avalia uma entrada, e devolve o dado, operado, como saída. Esse princípio é válido para toda a computação.

Quando as tarefas são sequenciais, condicionais, e muito bem definidas, (e provavelmente serão repetitivas), diz-se que a sequência dessas tarefas são "estados". Os dados que condicionam a continuidade destas funções são subsequentes, e uma forma de imaginá-los é uma grande "string"que é inserida como entrada ao programa. Estamos, historicamente falando, ganhando uma noção semelhante à que Warren McCulloch e Walter Pitts tiveram em 1943.

Vamos definir.

Máquina Determinística de Estados Finitos, Aceitador Determinístico finito ou ainda Autômato determinístico Finito (FSM para facilitar), é uma máquina de estados finitos que aceita ou rejeita símbolos, através da evolução uma sequência finita de estados. O autômato, de maneira formal, é composto de 3 estruturas:

Um conjunto finito de estados

Um conjunto finito de símbolos

Um conjunto finito de transições

Os estados determinam pontos no qual o autômato possui um estado definido, que pode mudar, ou não, a depender das transições e dos símbolos recebidos. Temos um único estado inicial, e podemos ter um ou mais estados finais, sobre o qual o autômato pode concluir sua execução.

Os símbolos definem entradas desse autômato, e serão julgados a cada nova iteração.

As transições determinam qual será o novo estado de um autômato considerando o estado atual, e o símbolo recebido.

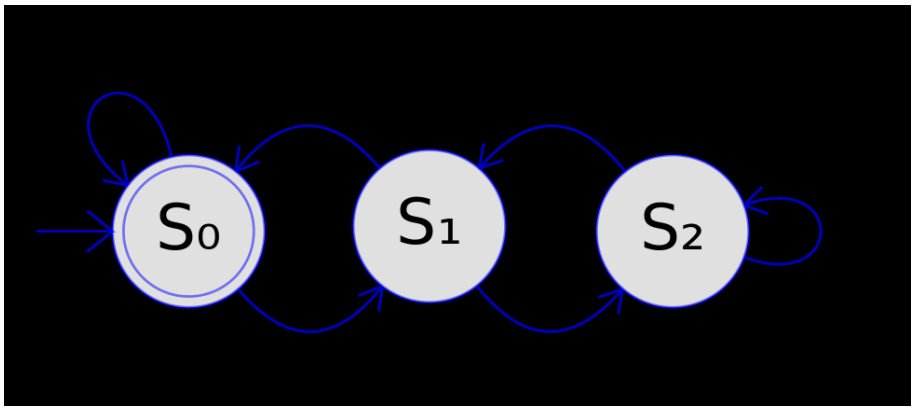


Figura 1: Exemplo de autômato

O algoritmo de um Autômato Finito Determinístico é simples, e definido a seguir:

1 - A cada instância, o autômato consome um, e somente um, símbolo.

2 - Dentre as funções de transição disponíveis, deve haver uma, e somente uma, transição correspondente a um símbolo.

3 - A cada instância, dentre as funções de transição disponíveis, das transições que partem do estado atual, se alguma está definida para o símbolo atual, o autômato transita para o novo estado determinado pela função.

4 - Isto ocorre até que o autômato chegue à um estado final, ou estado de aceitação, onde não existem mais símbolos para serem consumidos.

Esta definição pura compreende a grande maioria dos autômatos encontrados na computação. Mas existem agregadores que adicionam maior capabilidade à definição

2.1 Máquinas de Moore e de Mealy

Moore surgiu com um novo conceito em 1956, na publicação "Gedanken-experiments on Sequential Machines.".

O agregado são dois novos conjuntos: funções de saída e alfabeto de saída. Semanticamente, cada estado agora também é capaz de mapear, através das funções de saída, um conjunto de símbolos de saída.

Mas o que isso tudo significa?

Essa representação toda é uma definição formal do que conhecemos como condicionais em programas: o alfabeto é a entrada dos programas: a máquina os recebe, consumindo uma entrada por vez, a cada nova iteração. Seus estados são definidos por condicionais claros, e dado que a entrada esteja disponível, o mesmo evolui em estados, até uma possível condição final, onde o programa termina esta execução.

Da mesma maneira, um programa pode ainda devolver ações ao usuário, ou mesmo executar tarefas internas conforme cada estado que possui, definindo assim uma aplicação prática das funções e símbolos de saída.

Implementemos um na prática

3 Exemplos práticos

3.1 Seguidor de Linhas

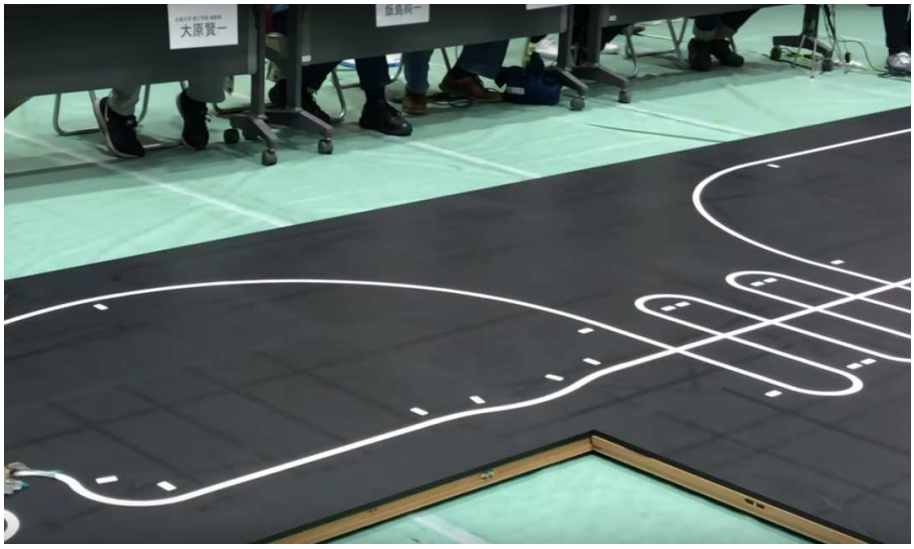


Figura 2: Exemplo de pista

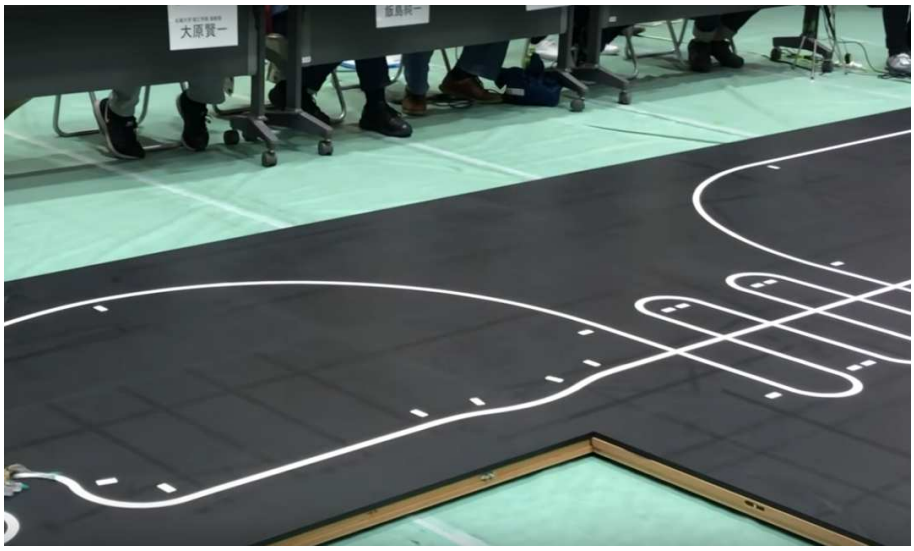


Figura 3: Exemplo de seguidor

Dado que tenhamos um seguidor de linhas, nas determinações atuais das regras de competição, uma função necessária, apenas:

Um seguidor corre, de forma a perseguir a linha.

Portanto, definimos seu primeiro estado, que chamemos de E1.

Enquanto este segue a linha, alguns eventos podem ocorrer, e definiremos símbolos para o autômato que representem este evento:

Um seguidor pode encontrar uma curva;

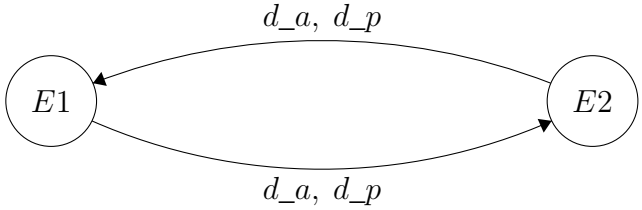
Um seguidor pode encontrar um cruzamento;

Um seguidor pode encontrar a linha de início da corrida;

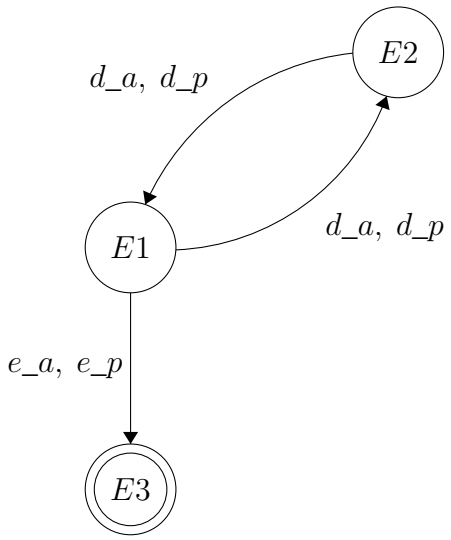
Um seguidor pode encontrar a linha de fim da corrida;

Uma vez que o seguidor esteja correndo, apenas 3 dessas ações são possíveis de ocorrer, sendo uma delas o objetivo que queremos atingir: correndo, ele encontrará curvas, e/ou cruzamentos, até que chegue ao final do percurso.

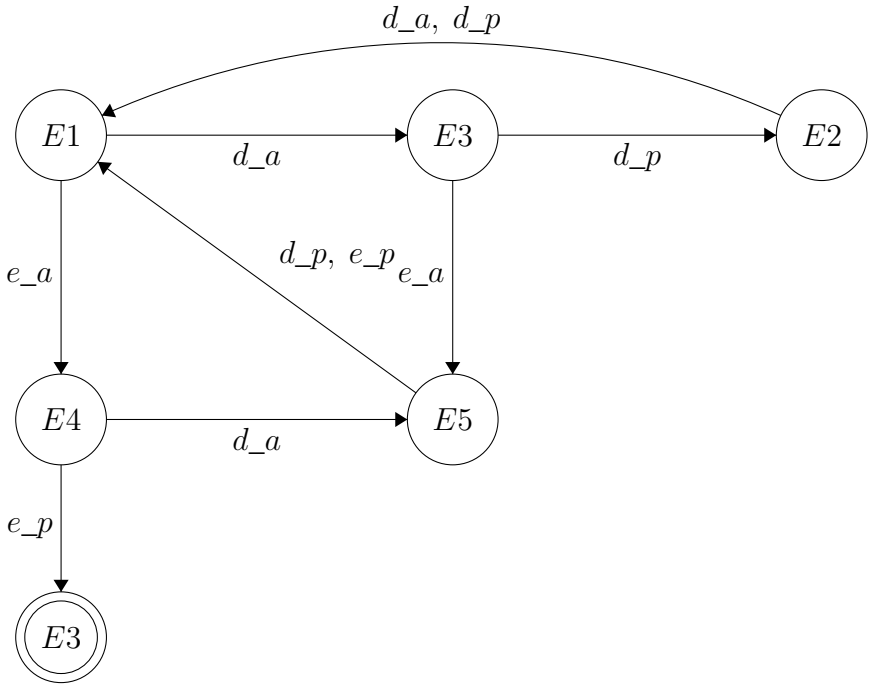
Chamemos o encontro de uma linha de curva de d_a, e a perda de uma linha de curva de d_p. Temos um novo estado (E2, ou "em curva"), e algumas transições.



Ele fará estas ações até encontre a linha de fim, portanto, mais um estado, e transição: o achado de uma linha de fim é e_a, e o perder é e_p. Temos, finalmente, um estado de aceitação E3.



Temos uma pergunta aqui: Como detectar cruzamentos? Precisamos pensar que cruzamento seja o evento e_a seguido de d_a antes que ocorra um e_p. Bem como pode ser d_a, e_a antes de d_p. Portanto, alteremos um pouco o que foi feito: E4 e E5 significam "possibilidade de cruzamento", e E6 significa "cruzamento".



Problema: Nosso autômato não leva em consideração cruzamentos dentro de curva. Fica para vocês resolverem.

4 Fontes

Não foi preciso.